

# Simulationstest

Die beiden Simulationstests umfassen je 72 Fragen und sind den realen Tests nachempfunden. Der erste Test ist in Deutsch (Fragen und Code), der zweite in Englisch gehalten, so dass Sie sich auch für die englischsprachige Prüfung vorbereiten können. In der realen Prüfung ist nur die Fragestellung eingedeutscht, der Code ist immer in Englisch. Die Fragen der beiden Serien sind voneinander unabhängig und nach Prüfungsziel geordnet. Es empfiehlt sich unbedingt, beide Serien durchzuarbeiten.

Diese Simulationsprüfung (*Mock-Exam*) hat folgende Eigenschaften:

- Die Fragen sind in dieser Prüfung nach Prüfungszielen geordnet, damit auch nur einzelne Prüfungsziele geübt werden können. In der realen Prüfung sind die Prüfungsziele gemischt. Die Prüfungsziele sind jeweils nur in der Lösung angegeben.
- Im Code finden sich in der realen Prüfung keine deutschsprachigen Bezeichner, da der Code von der englischsprachigen Prüfung stammt. Die englischsprachigen Bezeichner sind nur Objekt- oder Variablennamen wie zum Beispiel »Foo« oder »Bar« und behindern das Verständnis nicht.
- Wir haben hier wie im ganzen Buch das Programm komplett dargestellt und nicht nur als Ausschnitt (Snippet). Dies erleichtert das Verständnis. In der realen Prüfung werden oft nur Programmausschnitte gezeigt.
- Diese Prüfungsfragen lesen Sie auf Papier anstatt auf dem Bildschirm. Sich dreieinhalb Stunden lang auf einen Bildschirm zu konzentrieren, ist anstrengender.
- Sie müssen 49 Antworten korrekt gelöst haben, um zu bestehen. Es bedeutet nicht, dass Sie das auch an der Prüfung schaffen, falls Sie hier 49 Antworten richtig haben.
- Bitte melden Sie sich nur dann zur Prüfung an, falls Sie alle Themen der folgenden Prüfung verstehen, weil die Variationsbreite der Fragestellungen doch sehr groß ist.

Der Code-Name von Java 6 ist *Mustang*. Wir wünschen Ihnen also Hals- und Beinbruch beim Einreiten!

## 10.1 Aufgabenstellung Simulationstest 1

### Aufgabe 1

Gegeben ist folgender Code:

```
enum Wochentage{Mon,Tue,Wed,Thu,Fri,Sat,Sun}
public class Woche{
    enum Wochentage{MO,DI,MI,DO,FR,SA,SO}
    public static void main(String[] args){
        enum Wochentage{MO,DI,MI,DO,FR,SA,SO}
        System.out.println(Wochentage.MO);
    }
}
```

- [A] Die Ausgabe ist MO.
- [B] Das Programm kompiliert und läuft, gibt aber nichts aus.
- [C] Das Programm kompiliert, aber es ereignet sich ein Laufzeitfehler.
- [D] Das Programm erzeugt einen Kompilierfehler.

Wählen Sie eine Antwort.

### Aufgabe 2

Welche Variablennamen sind gültig? Wählen Sie drei richtige Antworten.

- [A] \$\_hallo
- [B] ämter3
- [C] 3er
- [D] \_test
- [E] #wert
- [F] Wert-o

### Aufgabe 3

Gegeben sind zwei Interfaces:

```
interface Fooable{void fooMethod();}
```

```
interface Barable{void barMethod();}
```

Welche Codes kompilieren? (3 korrekte Antworten)

□ [A]

```
class Foo implements Fooable{public void fooMethod(){}  
class Bar extends Foo implements Barable{public void barMethod(){}  
class Baz extends Bar implements Fooable {}
```

□ [B]

```
class Foo {public void fooMethod(){}  
class Bar extends Foo implements Barable {public void barMethod(){}  
class Baz extends Bar implements Fooable {}
```

□ [C]

```
class Foo {public void fooMethod(){}  
class Bar extends Foo implements Barable {public void barMethod();}  
class Baz extends Bar implements Fooable {public void barMethod(){}}
```

□ [D]

```
class Foo {public void fooMethod(){}  
class Bar extends Foo implements Barable, Fooable {}  
class Baz extends Bar implements Fooable {}
```

□ [E]

```
class Foo {public void fooMethod(){}  
class Bar implements Barable {void barMethod(){};}  
class Baz implements Fooable {}
```

□ [F]

```
class Foo {public void fooMethod(){}  
abstract class Bar extends Foo implements Barable {public abstract  
void barMethod();}  
class Baz extends Bar implements Fooable {public void barMethod(){}}
```

#### Aufgabe 4

```
class Vater{static void sagWas(){  
System.out.print("Hallo ");}  
void sagWasAnderes(){System.out.print("Hi ");}  
}  
public class Kind extends Vater{  
static void sagWas(){  
System.out.print("Tschuess ");  
}  
void sagWasAnderes(){
```

```
super.sagWasAnderes();
System.out.print("Bye ");
}
public static void main(String... args){
Kind k=new Kind();
k.sagWas();
k.sagWasAnderes();
Kind.k.sagWas();
Vater.sagWas();
}
}
```

Was gibt der obige Code aus? Wählen Sie eine Antwort.

- [A] Kompiliert nicht, Fehlermeldung: non-static variable super cannot be referenced from a static context
- [B] Tschuess Hi Bye Tschuess Hallo
- [C] Tschuess Bye Tschuess Hallo
- [D] Kompiliert nicht, Kind.sagWas() und Vater.sagWas() sind illegale Aufrufe.
- [E] Es ereignet sich ein Laufzeitfehler.

### Aufgabe 5

```
public class ArrayTest{
public void s(int... i) {//#1
int m=0;
for (int j:i)
if (j>m) m=j;
System.out.print(m);
}
public static void main(String... args){
ArrayTest at=new ArrayTest();
at.s(new int[] {10,60,3,77,88,65,44} ); //#2
}
}
```

Was ist das Resultat des obigen Codes? Wählen Sie eine Antwort.

- [A] Kompiliert, läuft, gibt 10 aus.
- [B] Kompilierfehler, Übergabe der Werte funktioniert nicht (##1).
- [C] Kompilierfehler, new int[] ... in #2 verursacht Fehler.

- [D] Laufzeitfehler ArrayIndexOutOfBoundsException
- [E] Kompiliert, läuft, gibt 88 aus.

### Aufgabe 6

```
public class [ ] {  
    public void message(){  
        System.out.print("von aussen ");  
    }  
    public class [ ] {  
        public void message(){  
            System.out.print("von innen ");  
        }  
    }  
    public static void main(String ... args){  
        [ ] a=new [ ];a.message();  
        [ ]. [ ] i=a.new [ ]; i.message();  
    }  
}
```

Setzen Sie folgende Elemente per Drag & Drop in die leeren Felder, sodass von aussen von innen ausgegeben wird.

### Aufgabe 7

```
new Thread(new Runnable() {  
    public void run() {  
        try {  
            while (true) {  
                sleep(1000); System.out.print(".");  
            }  
        }  
        catch(InterruptedException ex) {}  
    }  
}).start();
```

Welches Konstrukt liegt vor? Wählen Sie eine Antwort.

- [A] innere anonyme Klasse
- [B] innere lokale Klasse

- [C] innere Memberklasse
- [D] verschachtelte statische Klasse
- [E] verschachteltes Interface

### Aufgabe 8

```
//Hier einfügen
class Berechnung{
    public static void main(String... args){
        double alpha=30;
        double rad=alpha*2*PI/360;
        System.out.print(sin(rad));
    }
}
```

Welche Ausdrücke müssen Sie bei //Hier einfügen einsetzen, damit das Programm funktioniert. Wählen Sie zwei Antworten.

- [A]

```
import java.lang.Math.*;
```

- [B]

```
import static java.lang.Math.*;
```

- [C]

```
import static java.lang.Math;
```

- [D]

```
import static java.lang.Math.PI;
import static java.lang.Math.sin;
```

- [E]

```
import static Math.*;
```

### Aufgabe 9

Welche Array-Deklarationen kompilieren ohne Fehlermeldung? Wählen Sie zwei Lösungen.

- [A] `int [][]a=new int [][];`
- [B] `int [][]a=new int [][][5];`

- [C] `int [][]a=new int [10][];`
- [D] `int a[10]=new int [10][10];`
- [E] `int []a[]=new int [10][10];`

### Aufgabe 10

```
class Vater{};
class Sohn extends Vater{}
class A{
Vater meineMethode(Vater v){return v;}
}
public class B extends A{
//Hier einfügen
}
```

Welche Code-Stücke sind legal? Fügen Sie diese bei //Hier einfügen ein. Wählen Sie drei Antworten.

- [A] `Sohn meineMethode(Sohn s){return s};`
- [B] `private Vater meineMethode(Vater v){return v};`
- [C] `Vater meineMethode(Vater V){return V};`
- [D] `public Vater meineMethode(Vater V){return V};`
- [E] `void meineMethode(Vater V){};`

### Aufgabe 11

```
public class ArrayDrucken{
public static void main(String... args){
String[][]s={{"1","2","3","4"},"A","B","C"};
for( [ ] s1 [ ] [ ] )
for( [ ] s2 [ ] [ ] )
System.out.print([ ]);
}
}
```

Positionieren Sie per Drag & Drop die Elemente so, dass 1234ABC ausgegeben wird.

### Aufgabe 12

Gegeben ist folgender Code:

```
public class BreakFalle{
    public static void main(String[] args){
        Integer i=5;
        switch (i){
            case 2: System.out.print("2 ");
            default: System.out.print("Default ");
            case 3: System.out.print("3 ");
            break;
            case 4: System.out.print("4 ");
        }
    }
}
```

Was resultiert? Wählen Sie eine Antwort.

- [A] Default
- [B] 3
- [C] Kompilierfehler
- [D] 2
- [E] Default 3
- [F] Laufzeitfehler

### Aufgabe 13

Gegeben ist folgender Code:

```
public class TryTest{
    public static void main(String... args) throws ArithmeticException {
        throw new IndexOutOfBoundsException(); //#1
    }
    try{
        throw new ArithmeticException();
    }
    catch (ArithmeticException e){//#2
        System.out.println("ArithmeticException gefangen");
    }
    catch (IndexOutOfBoundsException e){//#3
        System.out.println("IndexOutOfBoundsException gefangen");
    }
}
}
```

Welche Aussagen treffen zu? Wählen Sie zwei Antworten.

- [A] Der Code kompiliert und läuft.
- [B] Der Code kompiliert und läuft, falls die mit #1 markierte Zeile gelöscht wird.
- [C] Die geworfene IndexOutOfBoundsException wird in #3 gefangen.
- [D] Der Code gibt bei der Kompilierung die Fehlermeldung »unreachable statement« (unerreichbares Statement) aus.
- [E] Der Code verursacht einen Laufzeitfehler.

#### Aufgabe 14

```
import java.io.*;
class A{
    void m1() throws ArithmeticException {
        throw new ArithmeticException();
    }
    void m2() throws IOException {
        throw new IOException();
    }
}
class B extends A{

    //Hier einfügen
}
public class Erbtest{
    public static void main(String... args) throws Exception {
        A a=new A();a.m1();a.m2();
        B b=new B();b.m1();b.m2();
    }
}
```

Welcher Codeblock, falls bei //Hier einfügen eingesetzt, erzeugt einen Kompilierfehler? Wählen Sie eine Antwort.

- [A]

```
void m1(){
    void m2() throws EOFException{
        throw new EOFException();
    }
}
```

[B]

```
void m1(){}  
void m2() throws Exception{  
    throw new Exception();  
}
```

[C]

```
void m1(){}  
void m2(){}
```

[D]

```
void m1(){}  
void m2() throws ArithmeticException{  
    throw new ArithmeticException();  
}
```

[E]

```
void m1() throws ArithmeticException{  
    throw new ArithmeticException();  
}  
void m2() throws ArithmeticException{  
    throw new ArithmeticException();  
}
```

### Aufgabe 15

Welche der folgenden Blöcke stellen einen angepassten Gebrauch von Assertions dar? Wählen Sie zwei Antworten.

[A]

```
public String test(int i){  
    assert(value>0 && value <10):"Wert muss zwischen 1 und 9 liegen";  
}
```

[B]

```
private String test(int i){  
    assert(value>0 && value <10): "Wert muss zwischen 1 und 9 liegen";  
}
```

[C]

```
public static void main (String ... args) {  
    assert(args[0]>0 && value <10):"Wert muss zwischen 1 und 9 liegen";  
}
```

[D]

```
switch(x) {default:assert false; }
```

### Aufgabe 16

Welche der folgenden Ausdrücke kompilieren korrekt *und* erzeugen eine Ausgabe? Wählen Sie zwei korrekte Lösungen.

- [A] `assert false : null;`
- [B] `assert true : return;`
- [C] `assert 10<1: Hallo;`
- [D] `assert false: 8 + 9;`
- [E] `assert true: "Hallo Welt";`
- [F] `assert false "Hallo Welt";`

### Aufgabe 17

```
interface If1{void methode1();}
interface If2{void methode2();}
class A{
    void a(){
    }
class B    {
  methode1()   {}
  methode2()   {}
}
```

Positionieren Sie unten aufgelistete Elemente so, dass der Code kompiliert.

### Aufgabe 18

```
import java.io.*;
public class ExceptionMatching{
    public static void main(String... args){
        int i=10;
```

```
int j=0;
int k=0;
try{
    try {
        k=i/j;
    } catch (ArithmeticException e){
        System.out.print("Hallo ");
    }
    File inputDatei = new File("input.txt");
    try {
        FileReader frInputDatei = new FileReader(inputDatei);
    } catch (IOException e){
        System.out.print("du ");
    }
    int[] l=new int[3];
    int m;
    try {
        m=l[3];
    }catch (IndexOutOfBoundsException e){
        System.out.print("wunder ");
    }
    finally {
        System.out.print("schoene ");
    }
    throw new Exception();
} catch (Exception e){
    System.out.print("Welt ");
}
}
```

Was ist der Output des obigen Codes (sofern *input.txt* nicht existiert)? Wählen Sie eine Antwort.

- [A] Kompilierfehler
- [B] Hallo du schoene
- [C] Hallo du schoene Welt
- [D] Hallo du wunder schoene Welt
- [E] Runtime-Exception durch nicht abgefangene Ausnahme

## Aufgabe 19

```
public class Loop{
    public static void main(String... args){
        mein_label:
        for(int j=0;j<10;j++){
            if (j%2==0)
                continue mein_label;
            System.out.print("Hallo ");
        }
    }
}
```

Wie viele Male wird »Hallo« ausgegeben? Wählen Sie eine Antwort.

- [A] 1
- [B] 2
- [C] 3
- [D] 4
- [E] 5

## Aufgabe 20

```
import java.io.*;
class A{
    public void test(String s) throws IOException{}
}
public class OverrideTest extends A {
    //Hier einfügen
}
```

Welcher der unten aufgelisteten Methoden eingefügt bei //Hier einfügen verursacht einen Kompilierfehler? Wählen Sie eine Antwort.

- [A] `public void test(String s) throws IOException{}`
- [B] `public void test(String s) {}`
- [C] `public void test(String s) throws Exception {}`
- [D] `public void test(String s) throws EOFException{}`
- [E] `public void test(String s) throws RuntimeException{}`

**Aufgabe 21**

Gegeben ist folgender Code

```
public class ParseIntValueOfFehler{
    public static void main(String[] args){
        Integer i=Integer.parseInt("10"); // #1
        int k=Integer.parseInt("10"); // #2
        Integer l=Integer.valueOf("10"); // #3
        int m=Integer.valueOf("10"); // #4
        Long n=Long.parseLong("10"); // #5
        long o=Long.parseLong("10"); // #6
        Integer p=10; // #7
        Integer q=p.intValue(); // #8
        Number x=0; // #9
        float y=10.0f; // #10
        x=y; // #11
    }
}
```

Wählen Sie zwei korrekte Antworten.

- [A] Das Programm kompiliert nicht wegen Zeile #1.
- [B] Das Programm kompiliert.
- [C] Würde das Programm mit Java 1.4 kompiliert, ergäben sich Kompilierfehler.
- [D] Das Programm kompiliert nicht wegen Zeile #5.
- [E] Das Programm kompiliert nicht wegen Zeile #8.

**Aufgabe 22**

```
import [ ] .*;
import [ ] [ ] ;
public class B_Reader{
    public static void main(String[] args){
        String zeile=null;
        try{
            [ ] br=new [ ] (new [ ] ([ ] ));
            while ([ ]){
                out.println(zeile);
            }
        }
    }
}
```

```
catch(Exception e){}
}
}
```

Ergänzen Sie per Drag & Drop die Lücken.

BufferedReader  
BuffererWriter  
FileReader  
"input.txt"  
zeile=br.readLine()!=null  
zeile!=null  
java.lang.System.\*  
static  
java.io

### Aufgabe 23

Welche der folgenden Zeilen würden in Java 1.4 einen Kompilierfehler hervorrufen? Wählen Sie drei korrekte Antworten.

- [A] Integer i=Integer.parseInt("10");
- [B] int k=Integer.parseInt("10");
- [C] Integer l=Integer.valueOf("10");
- [D] int m=Integer.valueOf("10");
- [E] Long n=Long.parseLong("10");
- [F] long o=Long.parseLong("10");
- [F] Integer i=10; int m=i.intValue();

### Aufgabe 24

```
import _____ ;
public class B_Writer{
public static void main(String[] args){
String zeile="Hallo Welt";
Console c=System.console();
try{
_____ pw=new _____ (new _____ (new _____ ("_____ ")));

pw. _____ ;
pw. _____ ;
}
}
```

```
catch( ) {c.printf("Fehler beim Schreiben");}
}
```

Füllen Sie die Lücken mit einer Auswahl folgender Elemente:

- java.io.\*
- java.lang.System.\*
- "output.txt"
- br.readLine()
- BufferedReader
- FileReader
- PrintWriter
- BufferedWriter
- FileWriter
- printf(zeile)
- close()
- IOException e

### Aufgabe 25

```
import java.util.*;
public class FormatterTest{
    public static void main(String[] args){
        Formatter f=new Formatter();
        Calendar cal=Calendar.getInstance();
        f.format("%td %tB %tY %tH %tM %tS",
                cal,cal,cal,cal,cal,cal);
        System.out.println(f);
    }
}
```

Welcher der aufgelisteten Outputs erscheint, falls %t der Formatierer für die Zeit ist? Es ist der 9. Februar 2007, 15:34:48. Wählen Sie eine Antwort.

- [A] 15 27 30
- [B] 09 Februar 2007 15 34 48
- [C] Februar 09 2007 15 34 48
- [D] 15 34 48 Februar 09 2007
- [E] Laufzeitfehler

## Aufgabe 26

```
import java.util.*;
public class ZahlenFormatierung{
public static void main(String... args){
    double i=new Double( )+new Double( );
    String s=String.( ) ( );
    System.out.println(s);
    Locale lde=new Locale( );
    String strDE=String.( ) ( );
    System.out.println(strDE);
}
}
```

100000.00

lde,"Deutsches-Format: %,.2f",i

50000.00

format

"de","DE"

Ich,"CH-Format: %,.2f",i Schieben Sie die Elemente per Drag & Drop in die Leerstellen (Slots) im obigen Listing, sodass die Ausgabe wie folgt dargestellt wird:

```
CH-Format: 150'000.00
Deutsches Format: 150.000,00
```

## Aufgabe 27

```
import ;
public class Serialisieren implements {
private int x=10;
private int y=20;
public static void main(String... args){
    try{
        ser=new ();
        oos=new (new
            ("s.ser"));
        oos. ;
        oos. ();
        ois=new (new ("s.ser"));
        Serialisieren ser_back=( )ois. ;
        ois. ();
        System.out.println(ser_back.x);
        System.out.println(ser_back.y);
    }
}
```

```

    }
    catch (Exception e){e.printStackTrace();}
    }
}

```

- readObject()
- ObjectInputStream
- close
- java.io.\*
- transient
- volatile
- FileInputStream
- FileOutputStream
- Serializable
- ObjectOutputStream
- Serialisieren

Schieben Sie die obigen Elemente per *Drag & Drop* in die Leerstellen (Slots). Das Programm sollte kompilieren und laufen. Die Variable x soll *nicht* zwischengespeichert werden.

### Aufgabe 28

Schieben Sie die Elemente an die korrekte Stelle, so dass folgender Output resultiert: Mit diesem Buch ist die SCJP-Prüfung sehr einfach!

```

public class StringBuilderTest {
    public static void main(String... args){
        StringBuilder s=new StringBuilder("Die SCJP-Prüfung ist sehr schwierig!");
        //Erwartet: Mit diesem Buch ist die SCJP-Prüfung sehr einfach!

        _____ .
        _____ .
        _____ .
        _____ .
        _____ .
        System.out.println(s);
    }
}

```

- s.insert(41," einfach");
- s.delete(37,41);
- s.insert(0,"Mit diesem Buch ist d");
- s.delete(0,1);
- s.delete(41,51);

## Aufgabe 29

```
01 import java.util.*;
02 public class Tokenizing{
03     public static void main(String... args){
04         String s= "Sun Certified Java  Programmer";
05         String[] tokens=s.split("\\d");
06         Arrays.sort(tokens);
07         Arrays.reverse(tokens);
08         for (String s1:tokens)//#3
09             System.out.print(s1+" ");
10     }
11}
```

Was gibt obiges Programm aus? Wählen Sie zwei Antworten.

- [A] Kompilierfehler (Arrays.reverse() existiert nicht)
- [B] Sun Certified Java Programmer
- [C] Certified Java Programmer Sun
- [D] Sun Programmer Java Certified
- [E] Der String wird nicht zerteilt, das Zeichen »\\d« ist falsch.

## Aufgabe 30

```
import java.util.Scanner;
import java.io.*;
public class ScannerTest{
    public static void main(String[] args){
        boolean b=false;
        String str=null;
        int i=0;
        try{
            Scanner s=new Scanner(□ □ □ □ □ □ □ □);
            //test.txt enthält Hallo true false 1 2 3 tschuess
            while(s.□□□□□){
                if(s.□□□□□)
                    i=s.□□□□□;
                else if(s.□□□□□□□□)
                    b=s.□□□□□□;
                str=s.next();
            }
            System.out.print(i+" "+b+" "+str);
            //gibt »3 true tschuess« aus
        }
```

```

    }
    catch(Exception e){}
  }
}

```

Positionieren Sie folgende Elemente im obigen Code, sodass dieser kompiliert und läuft. Die Datei text.txt enthält »Hallo true false 1 2 3 tschuess« .

nextInt()
hasNext()
new
)
FileReader
(
hasNextBoolean()
"test.txt"
hasNextInt()
nextBoolean()

### Aufgabe 31

```

import java.io.PrintWriter;
public class PrintWriterTest{
public static void main(String... args){
  try{
    //PrintWriter pw=new PrintWriter(new BufferedWriter(new FileWriter(new
    //File("test.dat"))));
    PrintWriter pw=new PrintWriter(new BufferedWriter(new File-
    Writer("test.dat")));
    pw.printf("Hallo Welt");
    pw.close();

    //BufferedReader br=new BufferedReader(new FileReader(new
    File("test.dat"));
    BufferedReader br=new BufferedReader(new FileReader("test.dat"));
    String meinString="";
    while((meinString=br.readLine())!=null)
      System.out.println(meinString);

  }catch (IOException e){
    e.printStackTrace();
  }
}
}

```

Welche Ausgabe erzeugt obiger Code? Wählen Sie zwei Antworten.

- [A] Keine Ausgabe
- [B] hallo Welt
- [C] Kompilierfehler
- [D] Laufzeitfehler
- [E] Die mit Doppelslash auskommentierten Zeilen sind gleichwertig zu der jeweils untenstehenden Zeile.

### Aufgabe 32

Betrachten Sie folgenden Code:

```
1 public class DoppelStart extends Thread implements Runnable{
2     public void run(){
3         System.out.println("ich funktioniere");
4     }
5     public static void main(String [] args){
6         Thread t =new Thread(new DoppelStart());
7         t.start();
8         t.start();
9     }
10}
```

Was trifft zu? Wählen Sie eine Antwort.

- [A] Das Programm funktioniert und startet einen neuen Thread.
- [B] Das Programm meldet einen Kompilierfehler in Zeile 6.
- [C] Das Programm verursacht einen Laufzeitfehler in Zeile 6.
- [D] Das Programm startet und erzeugt zwei neue Threads.
- [E] Das Programm erzeugt einen Laufzeitfehler in Zeile 8.

### Aufgabe 33

Gegeben ist folgender Code:

```
public class ThreadVariationen // #1 Hier einfügen
public void run(){
    System.out.println("Thread");
}
public static void main(String [] args){
    // #2 hier einfügen
```

```
t.start();
}
}
```

Welche Kombinationen ergeben bei #1 und #2 eingefügt einen korrekt funktionierenden Thread mit der Ausgabe »Thread« ? Wählen Sie drei korrekte Lösungen.

A

```
extends Thread implements Runnable{
Thread t =new Thread(new ThreadVariationen());
```

B

```
implements Runnable{
Thread t =new Thread(new ThreadVariationen());
```

C

```
extends Thread{
Thread t =new Thread();
```

D

```
extends Thread{
Thread t =new ThreadVariationen();
```

E

```
extends Thread{
Thread t =new Runnable();
```

F

```
implements Thread{
Thread t =new ThreadVariationen();
```

### Aufgabe 34

```
public class SynchroTest implements Runnable{
private int zaehler=0;
public void zaehlMethode(){
synchronized (this){
for(int i=0;i<10;i++) {
System.out.println(Thread.currentThread().getName()+" "+zaehler++);
try {Thread.sleep(1000);} catch (InterruptedException e){};
```

```
    }  
  }  
}  
public void run(){  
    zaehlMethode();  
}  
public static void main(String... args){  
    Thread t = new Thread(new SynchroTest());  
    t.setName("ping");  
    t.start();  
    Thread t1 = new Thread(new SynchroTest());  
    t1.setName("pong");  
    t1.start();  
}  
}  
Output  
ping 0  
pong 0  
ping 1  
pong 1  
...  
ping 9  
pong 9
```

Welche Aussagen zum obigen Code treffen zu? Wählen Sie vier Antworten.

- A Der Code innerhalb der Methode »zaehlMethode()« ist synchronisiert, es kann immer nur ein einziger Thread auf diesen Code zugreifen.
- B Anstatt des Blocks *synchronized (this) {}* hätte man genauso gut *public synchronized void zaehlMethode()* schreiben können.
- C Die Reihenfolge des Outputs (Abwechslung zwischen ping und pong) ist durch die Synchronisierung gegeben.
- D Ließe man den *synchronized*-Block weg, wäre die Ausgabe zehn Mal ping und zehn Mal pong.
- E Die Reihenfolge wird in diesem Beispiel von der Synchronisierung nicht beeinflusst. Vielmehr durch das *sleep(1000)*, das den Thread in einen Schlafzustand versetzt und dem anderen Thread Gelegenheit gibt zu laufen.
- F Ließe man *sleep()* weg, ist die Abwechslung zwischen den Threads unregelmäßig, da der Thread, der aktuell am Laufen ist, einem anderen Thread keine Zeit zum Laufen gibt.

**Aufgabe 35**

Ordnen Sie per Drag & Drop folgende Methoden den Klassen *java.lang.Object*, *java.lang.Thread* und dem Interface *Runnable* zu.

- 
- 
- 
- 
- 
- 
- 
- 

Java.lang.Object

Java.lang.Thread

Interface Runnable

**Aufgabe 36**

```

class A implements Runnable{
    public void run(){
        for (int i=1;i<5;i++)
            System.out.println(Thread.currentThread().getName()+" "+ i);
    }
}
class B extends Thread{
    public B(String s){
        super(s);
    }
    public void run(){
        for (int i=1;i<5;i++)
            System.out.println(Thread.currentThread().getName()+" "+ i);
    }
}
class D extends Thread{
    public void run() {
        int i=0;
        for (i=0;i<10;i++)
            System.out.println("Zaehler: "+i);
    }
}
class E implements Runnable{

```

```
public void run(){
    for (int i=1;i<10;i++)
        System.out.print("E "+i);
    }
}
public class StartMethoden{
public static void main(String ... args){
    // #1
    Thread a =new Thread(new A(), "A");
    a.start();
    // #2
    Thread b=new B("B");
    b.start();
    // #3
    Thread c = new Thread() { //Objekt-Körper!
    public void run() {
        int i=0;
        for (i=0;i<10;i++)
            System.out.println("Zaehler: "+i);
        }
    };
    c.start();
    // #4
    new Thread(new D()).start();
    // #5
    new Thread(new E()).start();
    // #6
    Runnable f=new Runnable();
    f.run();
    }
}
```

Im obigen Programm werden Threads auf verschiedene Arten gestartet. Wählen Sie eine Antwort.

- [A] Alle Startarten von #1 bis #6 sind korrekt.
- [B] #1 ist falsch.
- [C] #2 ist falsch.
- [D] #3 ist falsch.
- [E] #4 ist falsch.
- [F] #5 ist falsch.
- [G] #6 ist falsch.

## Aufgabe 37

```
class Stack{
    int stand=0;
} //Ende Klasse Stack
class Push extends Thread{
    Stack stack;
    public Push(Stack stack){
        this.stack=stack;
    }
    public void run(){
        while(true){ //Endlosschleife
            int s;
            synchronized (stack){
                s=stack.stand; //Bestand auslesen
                if (s>9){
                    try {
                        System.out.println("Kueche wartet...");
                        stack.wait(); //Warten, bis Produzent ein notify() sendet
                    } catch (InterruptedException e){}
                }else {
                    s++; // Produktions-Code
                    stack.stand=s; //Produktion einlagern
                    System.out.println("Teller eingefügt: "+s);
                    stack.notify(); //Dem anderen Thread melden, dass neue Teller vorhanden
                }
                try {
                    Thread.sleep(100); //nach der Produktion schlafen...
                }catch (InterruptedException e) {}
            } //Ende Synchronized
        } //Ende while(true)
    } //Ende Run
} //Ende Klasse Push

class Pop extends Thread{
    Stack stack;
    public Pop(Stack stack){
        this.stack=stack;
    }
    public void run(){
        while(true){
            synchronized(stack){
                int s;
                s=stack.stand;
                if (s<=1){
```

```
try {
    System.out.println("Kellner wartet...");
    stack.wait(); //Warten, bis Produzent ein notify() sendet
} catch (InterruptedException e){}
}else {
    s--;
    System.out.println("Teller entfernt "+s);
    stack.stand=s;
    stack.notify();
} //Ende if s<1
try {
    Thread.sleep(3000);
} catch (InterruptedException e){}
} //Ende Synchronized
} //Ende while(true)
} //Ende Run
} //Ende Klasse Pop
public class StackTest{
    public static void main(String... args){
        Stack s=new Stack();
        Push push=new Push(s);
        Push push1=new Push(s);
        Pop pop=new Pop(s);
        push.start();
        push1.start();
        pop.start();
    } //Ende Main
} //Ende StackTest
```

Welche Aussagen zum obigen Code treffen zu? Wählen Sie vier Antworten.

- [A] Push ist das gemeinsame Objekt, der Zugriff erfolgt über eine Objektreferenz.
- [B] Stack ist das gemeinsame Objekt, der Zugriff erfolgt über eine Objektreferenz.
- [C] Der Aufruf der *notify()*-Methode auf einem gemeinsamen Objekt weckt einen einzigen wartenden Thread.
- [D] Der Aufruf der *wait()*-Methode auf einem gemeinsamen Objekt bewirkt, dass der aktuelle Thread seinen Lock aufgibt.
- [E] Die Produktion ist viel rascher als der Konsum, weil zwei Push-Threads vorhanden sind. Deshalb ist der Stack bald gefüllt (10 Teller).
- [F] Die Produktion ist langsamer als der Konsum, deshalb schwankt der Tellerstand zwischen einem und zwei Stück.

**Aufgabe 38**

```
public class JoinTest extends Thread{
    public JoinTest(String name){
        super(name);
    }
    public void run(){
        for (int i=0;i<5;i++){
            try {
                Thread.sleep(1000);
            }catch (InterruptedException e){ }
            System.out.print(this.getName());
        }
    }

    public static void main(String... args){
        JoinTest t1=new JoinTest("T1 ");
        JoinTest t2=new JoinTest("T2 ");
        t1.start();
        try{
            t1.join();
        }catch (InterruptedException e){}
        t2.start();
    }
}
```

Welche Aussagen treffen zu? Wählen Sie zwei Antworten.

- [A] Der Output lautet: T1 T2 T1 T2 T1 T2 T1 T2 T1 T2
- [B] Der Output lautet: T1 T1 T1 T1 T1 T2 T2 T2 T2 T2
- [C] Die Output-Reihenfolge kann nicht vorhergesagt werden, er hängt von der JVM ab
- [D] Die `join()`-Methode bewirkt, dass der aufrufende Thread (hier der *main*-Thread) in den Status `blocked-for-join-completion` verfällt, also blockiert. Der *main*-Thread ist solange blockiert, bis `t1` tot ist, erst dann wird `t2` vom *main*-Thread gestartet.

**Aufgabe 39**

```
public class YieldTest extends Thread{
    public YieldTest(String name){
        super(name);
    }
}
```

```
public void run(){
    for (int i=0;i<5;i++){
        if (i>2) Thread.yield();
        /*****
        try {
            Thread.sleep(1000);
        }catch (InterruptedException e){ }
        *****/
        System.out.print(this.getName());
    }
}
public static void main(String... args){
    YieldTest t1=new YieldTest("T1 ");
    YieldTest t2=new YieldTest("T2 ");
    t1.start();
    t2.start();
}
}
```

Wie lautet der Output des obigen Codes? Wählen Sie drei.

- [A] Der Output lautet: T1 T2 T1 T2 T1 T2 T1 T2 T1 T2
- [B] Der Output lautet: T1 T1 T1 T1 T1 T2 T2 T2 T2 T2
- [C] Der Output lautet: T1 T1 T1 T2 T2 T2 T1 T2 T1 T2, die Reihenfolge der letzten Elemente ist zufällig.
- [D] Die Output-Reihenfolge kann nicht vorhergesagt werden, er hängt von der JVM ab.
- [E] Würde man die *sleep()*-Methode im Kommentarblock aktivieren, lautete die Ausgabe: T1 T2 T1 T2 T1 T2 T1 T2 T1 T2

#### Aufgabe 40

```
01 public void ersteMethode(){
02     synchronized(erstesObjekt){
03         synchronized(zweitesObjekt){
04             tuWas();
05         }
06     }
07}
08 public void zweiteMethode(){
09     synchronized(zweitesObjekt){
10         synchronized(erstesObjekt){
11             tuWasAnderes();
```

```
12    }  
13    }  
14}
```

Was ist mit obigem Code los? Wählen Sie zwei korrekte Antworten.

- [A] Normaler Code, kompiliert und läuft.
- [B] Verschachtelte Synchronisationsblöcke in umgekehrter Reihenfolge können zu einer Blockierung führen.
- [C] Verschachtelte Synchronisationsblöcke führen zu einem Kompilierfehler.
- [D] Die Output-Reihenfolge kann nicht vorhergesagt werden, sie hängt von der JVM ab.
- [E] Durch Änderung der Reihenfolge der Synchronisation können Blockierungen verhindert werden.

#### Aufgabe 41

Welches sind die Eigenschaften der Notify-Methode? Wählen Sie 3 Antworten.

- [A] Sie ist eine Instanzmethode der Klasse Object.
- [B] Sie ist eine statische Methode der Klasse Object.
- [C] Der aktuelle Thread benötigt einen Lock auf das gemeinsame Objekt, auf das die Notify-Methode ausgeführt wird, sonst resultiert eine IllegalMonitorStateException.
- [D] Die Notify-Methode kann nur in einem synchronisierten Block bzw. in einer synchronisierten Methode aufgerufen werden.
- [E] Falls mehrere Threads warten, kann bestimmt werden, welcher an die Reihe kommt.

#### Aufgabe 42

Folgender Code ist gegeben

```
class A{  
    Double test(){  
        return 5.0;  
    }  
}  
class B extends A{  
    int test(){  
        return 5;  
    }  
}
```

```
class C extends B{
    String test(){
        return ("Hallo");
    }
}
public class TestOverload{
public static void main(String...args){
    A a=new A();
    System.out.print(a.test()+" ");
    B b =new B();
    System.out.print(b.test()+" ");
    C c =new C();
    System.out.print(c.test()+" ");
    A d=new C();
    System.out.print(d.test()+" ");
    A e =new B();
    System.out.print(e.test()+" ");
}
}
```

Was wird ausgegeben? Wählen Sie eine Antwort.

- [A] 5.0 5 Hallo 5.0 5.0
- [B] 5.0 5 Hallo 5.0 5
- [C] Kompilierfehler
- [D] 5.0 5 Hallo 5 5
- [E] 5.0 5 Hallo 5 5

### Aufgabe 43

```
interface Iface{public void machWas();}
class A implements Iface{public void machWas(){
    System.out.print("A");
}
}
class B extends A{
    public void machWas(){
        System.out.print("B");
    }
    public void machWasAnderes(){
        System.out.print("Anderes");
    }
}
```

```
class C extends A{
    public void machWas(){
        System.out.print("C");
    }
}
public class SimpleCast{
    public static void main(String... args){
        A a0=new A();
        a0.machWas();
        A a1=(A) new B(); // #1
        a1.machWas();
        if (a1 instanceof B){
            B b1=(B) a1; // #2
            b1.machWasAnderes();
        }
        A a2=(A) new C();
        a2.machWas();
    }
}
```

Was trifft für den obigen Code zu? Wählen Sie 3 richtige Antworten.

- [A] Die Ausgabe ist ABanderesC.
- [B] Die Ausgabe ist AAAnderesC.
- [C] Bei #1 findet ein Upcasting statt.
- [D] Bei #1 findet ein Downcasting statt.
- [E] Bei #2 findet ein Upcasting statt.
- [F] Bei #2 findet ein Downcasting statt.

#### Aufgabe 44

```
public class AutoB{
    public static void main(String [] args){
        drucke(3.0);
        drucke(3);
        drucke(3.0f);
    }
    static void drucke(Float n){
        System.out.print("Float ");
    }
    static void drucke(Integer n){
        System.out.print("Integer ");
    }
}
```

```
}  
static void drucke(Number n){  
    System.out.print("Number ");  
}  
}
```

Welche zwei Aussagen treffen zu?

- [A] Die Ausgabe ist »Number Integer Float«.
- [B] Die Ausgabe ist »Integer Float«.
- [C] Das Programm kompiliert und läuft.
- [D] Das Programm gibt Kompilierfehler aus, *drucke(double)* wird nicht gefunden.
- [E] Die Ausgabe ist »Float, Integer, Number«.

#### Aufgabe 45

```
class Tier{}
```

```
class Hund{}
```

```
class Herrchen{}
```

Welche Relationen könnten für diese Klassen bestehen? Wählen Sie vier.

- [A] Ein Hund ist ein Herrchen.
- [B] Ein Hund hat ein Herrchen.
- [C] Ein Hund ist ein Tier.
- [D] Ein Tier ist ein Hund.
- [E] Ein Herrchen hat einen Hund.
- [F] Eine Hat-ein-Beziehung nennt man auch eine Komposition.

#### Aufgabe 46

```
#1  
class Bankkonto{  
    public druckeFeriengutschein(){}  
}  
#2  
class Bankkonto{  
    public einzahlen(){}  
}
```

```
public auszahlen(){}  
}  
#3  
class Bankkonto{  
Kunde kunde;  
printKunde();  
}
```

Ordnen Sie die untenstehenden Begriffe den Codes zu.

- |                   |
|-------------------|
| Enge Koppelung    |
| Lockere Koppelung |
| Hohe Kohäsion     |
| Schwache Kohäsion |

### Aufgabe 47

```
class SuperKlasse{  
class SubKlasse extends SuperKlasse{  
private SubKlasse(){  
System.out.println("im Konstruktor der Subklasse");  
}  
//public static void main(String [] args){  
//SubKlasse s=new SubKlasse();  
//}  
}  
public class TestConstr{  
public static void main(String [] args){  
SubKlasse s=new SubKlasse();  
}  
}
```

Was trifft für den obigen Code zu? Wählen Sie drei Antworten.

- [A] Der Code kompiliert einwandfrei.
- [B] Der Code kompiliert nicht.
- [C] Falls die Methode *main()* in die SubKlasse verlegt wird, kompiliert der Code einwandfrei.
- [D] Private Konstruktoren sind verboten.
- [E] Private Konstruktoren sind nur erlaubt, falls sich die *main()*Methode in der gleichen Klasse wie der private Konstruktor befindet.

### Aufgabe 48

```
class A{
    void methode(String s){}
}

abstract class B extends A{
    //Hier einfügen
}
```

Welcher der folgenden Code-Stücke kann unabhängig bei //Hier einfügen eingesetzt werden und ermöglicht ein einwandfreies Kompilieren? Wählen Sie drei Antworten.

- [A] abstract void methode(String s);
- [B] protected void methode(String s) throws Error{}
- [C] void methode(String s) {}
- [D] int methode(String s) {}
- [E] private void methode(String s) {}
- [F] abstract void methode(String s){}

### Aufgabe 49

Nehmen Sie an, Hund erweiteren Tier. Welche der folgenden Zuweisungen kompilieren korrekt? Wählen Sie drei richtige Antworten.

```
Tier t=new Tier();
Hund h=new Hund();
```

- [A] t=h
- [B] h=t
- [C] h=(Hund)t
- [D] t=(Tier)h
- [E] h=(Tier)t

### Aufgabe 50

```
class A{
    void methode(String s){}
}
```

```
abstract class B extends A{
//Hier einfügen
}
```

Welcher der folgenden Code-Stücke kann unabhängig bei //Hier einfügen eingesetzt werden und verursacht einen Kompilierfehler? Wählen Sie zwei Lösungen.

- [A] void methode(String s) {}
- [B] protected void methode(int i) throws Error{}
- [C] int methode(String s);
- [D] int methode(int i) {}
- [E] private void methode(int k) {}
- [F] abstract void methode(double d){}

### Aufgabe 51

```
class A{
    static void methode(){}
}
class B extends A{
    void methode() {//#1
        System.out.println("es funktioniert");}
}
public class ueberschreibenStatisch{
    public static void main(String... args){
        B b=new B();
        b.methode(); //#2
    }
}
```

Wählen Sie zwei korrekte Aussagen:

- [A] Der Code kompiliert korrekt und gibt »es funktioniert« aus.
- [B] Es resultiert eine Fehlermeldung:

```
methode() in B cannot override methode() in A; overridden method is static.
```

- [C] Es resultiert eine Fehlermeldung:

```
duplicate methods: methode() in B cannot be doubled.
```

- [D] Der Code funktioniert, falls man bei #1 `void methode(int i)` einsetzen würde und den Aufruf in #2 in `b.methode(1)` ändern würde.

**Aufgabe 52**

Methoden	Collection-Klasse
Viele Zugriffe über einen Index, keine Thread-Sicherheit	
Speicherung ohne Schlüssel, keine Duplikate, sehr schneller Zugriff	
Assoziative Speicherung mit Schlüssel, nicht thread-sicher, schneller Zugriff	
Viele Zugriffe über einen Index, mit Thread-Sicherheit	
Häufiges Einfügen/Löschen am Anfang der Liste, wenige Zugriffe über Index	
Assoziative Speicherung mit Schlüssel-Wert-Paar, thread-sicher	
Geordnete Liste mit assoziativem Schlüssel-Wert-Paar	
Warteschlange	

Ordnen Sie den Anforderungen per Drag & Drop Collection-Klassen zu.

- PriorityQueue
- HashTable
- LinkedList
- Vector
- HashMap
- HashSet
- ArrayList
- TreeMap

**Aufgabe 53**

Welche Eigenschaften treffen für *hashCode()* zu? Wählen Sie drei Möglichkeiten.

- [A] Falls *hashCode()* in einem Objekt, das in ein *HashSet* gespeichert werden sollte, nicht überschrieben wurde, werden Objekte mit gleichem Inhalt separat gespeichert.
- [B] *hashCode()* ist in *Object* nicht implementiert.
- [C] Die Methode *hashCode()*, falls sie nicht überschrieben wurde, liefert für Objekte gleichen Inhalts eine unterschiedliche *int*-Zahl.
- [D] In der Klasse *String* wird *hashCode()* so überschrieben, dass bei gleichem Inhalt ein identischer *int*-Wert zurückgegeben wird.
- [E] Bei *StringBuilder* und *StringBuffer* wurde *hashCode()* genau wie in der Klasse *String* überschrieben.

### Aufgabe 54

```
import java.util.*;
public class HilfsklasseArrays{
    public static void main(String... args){
        Integer a[]={5,7,3,77,9,8,1,3};
        .sort();
         <Integer> a1=  (a);
        int index=  .  (,);
        System.out.print(a1.  (index));
    }
}
```

Positionieren Sie per *Drag & Drop* folgende Elemente im obigen Listing, sodass 77 ausgegeben wird.

- 
- 
- 
- 
- 
- 
- 
- 
- 
- 

### Aufgabe 55

Weisen Sie folgenden Anwendungen die passende Collection per *Drag & Drop* zu.

Telefonbuch .

Eine sortierte Liste von Nachnamen .

Eine unsortierte Liste mit Lottozahlen .

Eine nach Einfügedatum sortierte Liste von Notizen .

- 
- 
- 
- 

### Aufgabe 56

Welche Aussagen treffen zu? Wählen Sie drei.

- [A] Comparable implementiert man in einer eigenen Klasse, wo *Comparator()* zum Einsatz kommt.
- [B] Das Interface *java.lang.Comparable* definiert als einzige Methode *int compareTo(T o)* und sorgt für eine Basissortierung (natürliche Reihenfolge).
- [C] Comparator implementiert man in einer eigenen Klasse, wo *CompareTo()* zum Einsatz kommt.
- [D] Wird ein Element bei einer binären Suche nicht gefunden, wird der Einfügeindex als positive ganze Zahl zurückgegeben.
- [E] Das Interface *java.util.Comparator* definiert die Methode *int compare(T o1, T o2)* und sorgt für eine Zusatzsortierung.

### Aufgabe 57

```
import java.util.*;
public class ArrayMax{
    public static void main(String[] args){
        int[] testarray=new int[]
            {10,19,33,1,3,5,7,8,99,22,55,77};
        Arrays. [ ];
        int endindex= [ ];
        System.out.println( [ ] [ ]);
    }
}
```

Es soll das Maximum des Arrays `testarray` gefunden werden. Positionieren Sie die Elemente per Drag & Drop so, dass 99 ausgegeben wird.

- 
- 
- 
- 
- 
- 
- 

### Aufgabe 58

```
class Generics<T>{
    T i;
    Generics(T i){
        this.i=i;
    }
}
```

```
T getI(){
    return i;
}
}
public class myGenerics{
    public static void main(String... args){
        Generics<Integer> gi=new Generics<Integer>(15);
        int i=gi.i;
        System.out.print(i);
    }
}
```

Gegeben ist obiges Programm. Was trifft zu? Wählen Sie eine Antwort.

- [A] Programm weist Kompilierfehler auf.
- [B] Programm kompiliert und läuft, gibt 15 aus.
- [C] Programm kompiliert und läuft, gibt 1 aus.
- [D] Programm kompiliert und wirft eine Ausnahme.

### Aufgabe 59

```
import java.util.*;
public class Refactoring{
    public static void main(String [] args){
        ArrayList al=new ArrayList();
        al.add("erstens");
        al.add("zweitens");
        al.add("drittens");
        String s=al.get(0);
        System.out.println(s);
    }
}
```

Gegeben ist obiges Programm. Was trifft zu? Wählen Sie eine Antwort.

- [A] Das Programm kompiliert einwandfrei, verursacht aber einen Laufzeitfehler.
- [B] Das Programm kompiliert und läuft einwandfrei.
- [C] Das Programm kompiliert wegen einer Typ-Inkompatibilität nicht.
- [D] Das Programm kompiliert mit einer Warnung, läuft aber.
- [E] Das Programm kompiliert nicht, weil mit `get()` auf das Element 0 zugegriffen wird, das nicht existiert.

### Aufgabe 60

Welche kompilieren? Wählen Sie zwei.

- [A] `LinkedList<Double>d = new LinkedList<Integer>();`
- [B] `List<Number> n=new LinkedList<Number>();`
- [C] `List<Number> n=new <Number>LinkedList();`
- [D] `List<Number> n=new LinkedList();`
- [E] `List<Number> n=new LinkedList<Integer>();`
- [F] `List<int> I = new LinkedList<int>();`

### Aufgabe 61

```
import java.util.*;
public class MapTest {
    public static void main(String args[]) {
        Map<[ ], [ ]> map = new HashMap<[ ], [ ]>();
        map.put(1, "Hallo");
        map.put(2, "du");
        map.put(3, "schoene");
        map.put(4, "Welt");
        for ([ ] < [ ], [ ]> eintrag : map.entrySet()){
            System.out.println("Map Eintrag: " + eintrag);
        }
    }
}
```

Positionieren Sie per *Drag & Drop* die richtigen Elemente in die Lücken des obigen Programms, sodass dieses funktioniert und Folgendes ausgibt:

```
Map Eintrag: 1=hallo
Map Eintrag: 2=du
Map Eintrag: 3=schoene
Map Eintrag: 4=Welt
```

### Aufgabe 62

```
import java.util.*;
public class HashSetGenerics{
```

```
public static void main(String[] args){
    String[] a=new String[]{"Hans", "Hans", "Fritz", "Kar1", "Kar1"};
    HashSet<String>hotelGast=new HashSet<String>();//#1 geändert
    for (int i=0;i<a.length;i++){
        if (!hotelGast.add(a[i]))
            System.out.print("Doppel ");
    }
    Iterator<String>it =hotelGast.iterator();//#2 geändert
    while (it.hasNext()) {
        String element = it.next();//#3 Auslesen direkt als String
        System.out.print(element + " ");
    }
}
```

Wie lautet die Ausgabe des obigen Programms in garantierter Reihenfolge?

- [A] Doppel Doppel Kar1 Fritz Hans
- [B] Kar1 Fritz Hans Doppel Doppel
- [C] Doppel Doppel Hans Fritz Kar1
- [D] Doppel Doppel Fritz Kar1 Hans
- [E] Hans Fritz Kar1
- [F] Keines der obigen

### Aufgabe 63

Welche drei Feststellungen zur Garbage-Collection treffen zu?

- [A] Sobald eine Referenz eines Objekts = o ist, wird es augenblicklich entsorgt.
- [B] Sobald alle Referenzen auf ein Objekt null sind , ist das Objekt reif für den Garbage Collector.
- [C] Die *finalize()*-Methode kann nicht überschrieben werden.
- [D] Die Implementation der Garbage-Collection ist von der JVM abhängig.
- [E] Die *finalize()*-Methode wird pro Objekt nur einmal aufgerufen.

### Aufgabe 64

Folgende jar-Datei ist gegeben, sie befindet sich im Ordner `c:\javaprogs\com`.

```
META-INF/
META-INF/MANIFEST.MF
projekt1/
```

```
projekt1/classes/  
projekt1/classes/library/  
projekt1/classes/library/Lib.class  
projekt1/classes/Aufruf.class
```

Der Klassenpfad lautet

```
c:\javaprogs\com\
```

Wie lauten die Package- und Import-Statements der Dateien Aufruf.class und Lib.class?

- [A] Package-Statement Aufruf.class: keines,  
Lib.class: library
- [B] Package-Statement Aufruf.class: classes  
Lib.class classes.library
- [C] Package-Statement Aufruf.class: projekt1.classes  
Lib.class: projekt1.classes.library
- [D] Import-Statement in Aufruf.class: import library.lib.class;
- [E] Import-Statement in Aufruf.class: import classes.library.lib.class;

### Aufgabe 65

Welche der folgenden Operatoren werten immer alle Operanden aus?

- [A] &&
- [B] &
- [C] ||
- [D] |
- [E] ?:

### Aufgabe 66

```
public class PrintTest{  
    public static void main(String... args){  
        double d=47114711.47611;  
        Locale lde=new Locale("de","DE");  
        System.out.printf(lde,"%,.2f",d);  
    }  
}
```

Wie lautet der Output dieses Programms? Wählen Sie eine Antwort.

- [A] 47,114,711.480
- [B] 47 114 711.47
- [C] 47.114.711,48
- [D] 47114711.48
- [E] 47114711.47

**Aufgabe 67**

```
01 public class Logo{
02     public static void main(String... args){
03         boolean b=false;
04         if (b=true)
05             System.out.print("Hallo ");
06         else
07             System.out.println("Tschuess ");
08         if (false ^ true)
09             System.out.print("Welt ");
10         else
11             System.out.print("Mond ");
12     }
13 }
```

Wie lautet das Resultat?

- [A] Hallo Welt
- [B] Welt Tschuess
- [C] Tschuess Mond
- [D] Laufzeitfehler
- [E] Kompilierfehler

**Aufgabe 68**

```
class Kommandozeile{
    public static void main(String... args){
        for (Object o:args)
            System.out.print(o);
        String[] s=new String[5];
        s=args;
        for (String o:args)
            System.out.print(o);
    }
}
```

Was wird ausgegeben, falls das Programm mit `java Kommandozeile 1 2` gestartet wird? Wählen Sie eine Antwort.

- [A] 1122
- [B] 1212
- [C] Kompilierfehler wegen Typen-Inkompatibilität
- [D] Laufzeitfehler

### Aufgabe 69

Welche der folgenden Ausdrücke kompilieren? Wählen Sie zwei Antworten.

```
int i=1; String s="test";
```

- [A] `s+=i;`
- [B] `i+=s;`
- [C] `s=s+i;`
- [D] `i=i+s;`
- [E] `if(i==s);`

### Aufgabe 70

```
//Datei Vater.class
package u;
public class Vater{
    protected int vaterVariable=10;
    public void vaterMethode(){
        System.out.println("vaterMethode()");
    }
}
//Datei Kind.class
01 import u.*;
02 public class Kind extends Vater{
03     public static void main(String... args){
04         Vater v = new Vater();
05         Kind k = new Kind();
06         v.vaterMethode();
07         System.out.println(v.vaterVariable);
08     }
09 }
```

Was passiert mit obigem Code? Wählen Sie eine Antwort.

- [A] Kompilierfehler bei Zeile 06
- [B] Kompilierfehler bei Zeile 07
- [C] Kompiliert einwandfrei und läuft.
- [D] Laufzeitfehler

### Aufgabe 71

```
class Mue11{
    Mue11 m;
}
Mue11 m0=new Mue11();
Mue11 m1=new Mue11();
Mue11 m2=new Mue11();
m0.m=m1;
m1.m=m2;
m2.m=m0;
m0=new Mue11Halde1();
m0=m1=m2;
}
```

Wieviele Objekte sind vor und nach dem Aufruf von `m0=m1=m2` referenziert?  
Wählen Sie eine Lösung.

- [A] 2, 4
- [B] 3, 4
- [C] 4, 3
- [D] 3, 2
- [E] 2, 3

### Aufgabe 72

```
public class Salat{
    static boolean hello(){
        System.out.print("Hello ");
        return true;
    }
    public static void main(String... args){
        boolean a=false;
        boolean b=true;
        int i=0;
```

```
if (a || b){
    System.out.print("Hallo ");
}
if (a && hello()){
    System.out.print("du ");
}
if (a & hello()){
    System.out.print("Welt ");
}
}
}
```

Welches ist die Ausgabe des obigen Programms? Wählen Sie eine Antwort.

- [A] Hallo Hello
- [B] Hallo Hello du Welt
- [C] Hallo Hello du
- [D] Hallo Hello Hallo Hello
- [E] Programm weist Kompilierfehler auf

## 10.2 Lösungen zu Simulationstest 1

Die Aufgaben und Lösungen sind nach Prüfungsziel gegliedert. Hinter der Aufgabenstellung, in Klammern, steht jeweils das Prüfungsziel und das Thema.

### 10.2.1 Prüfungsziel 1

#### Aufgabe 1 (1-1, Enum)

Gegeben ist folgender Code:

```
enum Wochentage{Mon,Tue,Wed,Thu,Fri,Sat,Sun}
public class Woche{
    enum Wochentage{MO,DI,MI,DO,FR,SA,SO}
    public static void main(String[] args){
        enum Wochentage{MO,DI,MI,DO,FR,SA,SO}
        System.out.println(Wochentage.MO);
    }
}
```

- [A] Die Ausgabe ist MO.
- [B] Das Programm kompiliert und läuft, gibt aber nichts aus.

- [C] Das Programm kompiliert, aber es ereignet sich ein Laufzeitfehler.
- [D] Das Programm erzeugt einen Kompilierfehler.

Wählen Sie eine Antwort.

### Lösung

D ist korrekt. Enums dürfen nicht *lokal* in einer Methode deklariert werden. Wird die Zeile unterhalb `public static void main(String[] args)` auskommentiert, funktioniert das Programm einwandfrei. *Bemerkung: Falls auf `Wochentage.Mon` zugegriffen werden will, also auf das äussere enum-Konstrukt, ereignet sich ein Kompilierfehler, da die äussere Deklaration von »Wochentage« durch die Deklaration in der Klasse überdeckt wird. Falls »enum Wochentage« innerhalb der Klasse auskommentiert wird, funktioniert der Zugriff.*

### Aufgabe 2 (1-3, Gültige Variablen)

Welche Variablennamen sind gültig? Wählen Sie drei richtige Antworten.

- [A] \$\_hallo
- [B] ämter3
- [C] 3er
- [D] \_test
- [E] #wert
- [F] Wert-0

### Lösung

A, B und D sind korrekt. \$ und \_ und a-z und A-Z sowie Umlaute sind erlaubt. Zahlen dürfen nicht am Anfang einer Variablen vorkommen. C ist falsch, weil eine Zahl an erster Stelle nicht erlaubt ist. E ist falsch, weil # ein Sonderzeichen ist, und F ist falsch, weil »-« ebenfalls ein Sonderzeichen ist.

### Aufgabe 3 (1-2, Schnittstellen)

Gegeben sind zwei Interfaces:

```
interface Fooable{void fooMethod();}
```

```
interface Barable{void barMethod();}
```

Welche Codes kompilieren? (3 korrekte Antworten)

❑ [A]

```
class Foo implements Fooable{public void fooMethod(){}  
class Bar extends Foo implements Barable{public void barMethod(){}  
class Baz extends Bar implements Fooable {}
```

❑ [B]

```
class Foo {public void fooMethod(){}  
class Bar extends Foo implements Barable {public void barMethod(){}  
class Baz extends Bar implements Fooable {}
```

❑ [C]

```
class Foo {public void fooMethod(){}  
class Bar extends Foo implements Barable {public void barMethod();}  
class Baz extends Bar implements Fooable {public void barMethod(){}}
```

❑ [D]

```
class Foo {public void fooMethod(){}  
class Bar extends Foo implements Barable, Fooable {}  
class Baz extends Bar implements Fooable {}
```

❑ [E]

```
class Foo {public void fooMethod(){}  
class Bar implements Barable {void barMethod(){};}  
class Baz implements Fooable {}
```

❑ [F]

```
class Foo {public void fooMethod(){}  
abstract class Bar extends Foo implements Barable {public abstract  
void barMethod();}  
class Baz extends Bar implements Fooable {public void barMethod(){}}
```

### Lösung

A und B und F sind korrekt. A: *fooMethod()* wird in Foo implementiert. B: Da die Klasse Foo nichts implementiert, müsste die Methode *fooMethod()* nicht unbedingt implementiert werden. Es kommt zu keinem Konflikt zwischen der aus Klasse Foo ererbten Methode *fooMethod()* und deren Deklaration im Interface *Fooble*.

F: In der Klasse Bar fehlt die Implementierung *barMethod()*. Weil aber Bar und die Methode *barMethod()* *abstract* deklariert sind, kann *barMethod()* eine Vererbungshierarchie tiefer in *Baz* implementiert werden. Für eine Implementierung reicht die Angabe des Methodenkörpers »{}«.

C ist falsch, weil in der Klasse Bar die Implementierung von `barMethod()` fehlt (es fehlt der Methodenkörper).

D ist falsch, weil in Bar die `public void barMethod(){};` nicht implementiert wird.

E ist falsch, `void barMethod(){};` müsste `public` sein, weil eine Methode in einem Interface implizit `public` ist. So muss auch die Implementierung `public` sein.

#### Aufgabe 4 (1-4, Aufruf von statischen Methoden)

```
class Vater{static void sagWas(){
    System.out.print("Hallo ");}
    void sagWasAnderes(){System.out.print("Hi ");}
}
public class Kind extends Vater{
    static void sagWas(){
        System.out.print("Tschuess ");
    }
    void sagWasAnderes(){
        super.sagWasAnderes();
        System.out.print("Bye ");
    }
    public static void main(String... args){
        Kind k=new Kind();
        k.sagWas();
        k.sagWasAnderes();
        Kind.sagWas();
        Vater.sagWas();
    }
}
```

Was gibt der obige Code aus? Wählen Sie eine Antwort.

- [A] Kompiliert nicht, Fehlermeldung: `non-static variable super cannot be referenced from a static context`
- [B] Tschuess Hi Bye Tschuess Hallo
- [C] Tschuess Bye Tschuess Hallo
- [D] Kompiliert nicht, `Kind.sagWas()` und `Vater.sagWas()` sind illegale Aufrufe.
- [E] Es ereignet sich ein Laufzeitfehler.

#### Lösung

B ist korrekt. »Hi« wird wegen des Aufrufs `super.sagWasAnderes()`, also wegen des Aufrufs der überschriebenen Methode in der Elternklasse ausgegeben.

A ist falsch, die Fehlermeldung käme, falls `super.sagWasAnderes()`; in `main()` stünde.

C ist falsch, weil mit `super.sagWasAnderes()` die Vater-Methode aufgerufen wird, die »Hi« ausgibt.

D ist falsch, `Kind.sagWas()` und `Eltern.sagWas()` sind legale Aufrufe von statischen Methoden.

E ist falsch, der Code kompiliert korrekt.

### Aufgabe 5 (1-3 Anonymer Array, Varargs)

```
public class ArrayTest{
    public void s(int... i) {//#1
        int m=0;
        for (int j:i)
            if (j>m) m=j;
        System.out.print(m);
    }
    public static void main(String... args){
        ArrayTest at=new ArrayTest();
        at.s(new int[] {10,60,3,77,88,65,44} ); //#2
    }
}
```

Was ist das Resultat des obigen Codes? Wählen Sie eine Antwort.

- [A] Kompiliert, läuft, gibt 10 aus.
- [B] Kompilierfehler, Übergabe der Werte funktioniert nicht (1).
- [C] Kompilierfehler, `new int[] . . .` in #2 verursacht Fehler.
- [D] Laufzeitfehler `ArrayIndexOutOfBoundsException`
- [E] Kompiliert, läuft, gibt 88 aus.

### Lösung

E ist korrekt, es handelt sich um eine Maximum-Berechnung.

A wäre korrekt, wenn es sich um eine Minimum-Berechnung handelte.

B, einem `VarArg` darf ein Array übergeben werden.

C, es handelt sich in #2 um ein anonymes Array, das korrekt deklariert ist.

D, eine `IndexOutOfBoundsException` kann sich in einer erweiterten `for`-Schleife nicht ereignen, da nicht über die Arraygrenzen hinaus iteriert wird

**Aufgabe 6 (1-6 Innere Klasse)**

```

public class [ ] {
    public void message(){
        System.out.print("von aussen ");
    }
    public class [ ] {
        public void message(){
            System.out.print("von innen ");
        }
    }
    public static void main(String ... args){
        [ ] a=new [ ];a.message();
        [ ].[ ] i=a.new [ ]; i.message();
    }
}

```

Setzen Sie folgende Elemente per Drag & Drop in die leeren Felder, sodass von aussen von innen ausgegeben wird.

- 
- 
- 
- 

**Lösung**

```

public class Aeussere{
    public void message(){
        System.out.print("von aussen ");
    }
    public class Innere{
        public void message(){
            System.out.print("von innen ");
        }
    }
    public static void main(String ... args){
        Aeussere a=new Aeussere();a.message();
        Aeussere.Innere i=a.new Innere(); i.message();
    }
}

```

**Aufgabe 7 (1-1, innere anonyme Klasse)**

```

new Thread(new Runnable() {
    public void run() {

```

```
try {  
    while (true) {  
        sleep(1000); System.out.print(".");  
    }  
}  
catch(InterruptedException ex) {}  
}  
}).start();
```

Welches Konstrukt liegt vor? Wählen Sie eine Antwort.

- [A] innere anonyme Klasse
- [B] innere lokale Klasse
- [C] innere Memberklasse
- [D] verschachtelte statische Klasse
- [E] verschachteltes Interface

### Lösung

A ist korrekt, es handelt sich um eine innere anonyme Klasse. Der gesamte Ausdruck wird mit einem Semikolon abgeschlossen, weil es sich um einen Ausdruck handelt. Alle anderen Antworten sind aufgrund der obigen Erklärung falsch.

### Aufgabe 8 (1-1, Statischer Import)

```
//Hier einfügen  
class Berechnung{  
    public static void main(String... args){  
        double alpha=30;  
        double rad=alpha*2*PI/360;  
        System.out.print(sin(rad));  
    }  
}
```

Welche Ausdrücke müssen Sie bei //Hier einfügen einsetzen, damit das Programm funktioniert. Wählen Sie zwei Antworten.

- [A]

```
import java.lang.Math.*;
```

- [B]

```
import static java.lang.Math.*;
```

[C]

```
import static java.lang.Math;
```

[D]

```
import static java.lang.Math.PI;
import static java.lang.Math.sin;
```

[E]

```
import static Math.*;
```

### Lösung

B und D sind korrekt. Bei einem statischen Import können entweder alle Member oder bestimmte Member importiert werden.

A ist falsch, weil im Programm nur *PI* und *sin()* anstatt *Math.PI* und *Math.sin()* verwendet werden, was bedeutet, dass ein statischer Import vorliegen muss.

C ist falsch, weil man statisch keine Packages importieren kann.

E ist falsch, weil *java.lang* benötigt wird, um die Math-Library zu laden.

### Aufgabe 9 (1-3, mehrdimensionale Arrays)

Welche Array-Deklarationen kompilieren ohne Fehlermeldung? Wählen Sie zwei Lösungen.

- [A] `int [][]a=new int [][];`
- [B] `int [][]a=new int [][5];`
- [C] `int [][]a=new int [10][];`
- [D] `int a[10]=new int [10][10];`
- [E] `int []a[]=new int [10][10];`

### Lösung

C und E sind korrekt. Bei der Deklaration darf bei C die zweite eckige Klammer auf der rechten Seite leer sein. Die Position der eckigen Klammer kann vor oder nach dem Bezeichner liegen.

Alle anderen sind aus folgenden Gründen falsch: A: Rechts des Gleichheitszeichens muss mindestens eine Dimension vorhanden sein. B: Bei einem mehrdimensionalen Array muss die Erste rechteckige Klammer eine Dimension enthalten. D: Eine Zahl innerhalb einer eckigen Klammer darf nie links vom Gleichheitszeichen vorkommen, daher ist D falsch. Die Dimensionen müssen links und rechts vom Gleichheitszeichen gleich sein, daher ist D nochmals falsch.

**Aufgabe 10 (1-5, Überschreiben, override)**

```

class Vater{};
class Sohn extends Vater{}
class A{
Vater meineMethode(Vater v){return v;}
}
public class B extends A{
//Hier einfügen
}

```

Welche Code-Stücke sind legal? Fügen Sie diese bei //Hier einfügen ein. Wählen Sie drei Antworten.

- [A] Sohn meineMethode(Sohn s){return s;};
- [B] private Vater meineMethode(Vater v){return v;};
- [C] Vater meineMethode(Vater V){return V;};
- [D] public Vater meineMethode(Vater V){return V;};
- [E] void meineMethode(Vater V){};

**Lösung**

A und C und D sind korrekt. Bei A wäre ein ist ein kovarianter Return-Typ vorhanden, A ist aber ein Overload, da in der Parameter-Klammer ein anderer Typ vorhanden ist. C ist ein normaler Override und D ebenfalls.

B ist falsch, weil die Zugriffsprivilegien eingeschränkt werden. Die überschriebene Methode muss den gleichen oder einen breiteren Zugriffsmodifizierer aufweisen. E ist falsch, weil der Rückgabetyt nicht mehr gleich ist wie in der überladenen Klasse, aber auch nicht kovariant ist.

**10.2.2 Prüfungsziel 2****Aufgabe 11 (2-2, 2D Array, erweiterte for-Schleife)**

```

public class ArrayDrucken{
    public static void main(String... args){
        String[][] s={{"1", "2", "3", "4"}, {"A", "B", "C"}};
        for( [ ] s1 [ ] [ ] )
            for( [ ] s2 [ ] [ ] )
                System.out.print( [ ] );
    }
}

```

String[]  
 String  
 :  
 s  
 s1  
 s2

Positionieren Sie per Drag & Drop die Elemente so, dass 1234ABC ausgegeben wird.

### Lösung

Ein zweidimensionales Array ist ein Array, das weitere Arrays enthält. Deshalb muss im ersten Schritt der String[] aufgelöst werden, dann jedes einzelne Array. s1 und s2 sind Hilfsvariablen.

```
public class ArrayDrucken{
    public static void main(String... args){
        String[][]s={{"1","2","3","4"},{"A","B","C"}};
        for(String[] s1: s)
            for(String s2: s1)
                System.out.print(s2);
    }
}
Output
1234ABC
```

### Aufgabe 12 (2-1, Case mit Integer Selector)

Gegeben ist folgender Code:

```
public class BreakFalle{
    public static void main(String[] args){
        Integer i=5;
        switch (i){
            case 2: System.out.print("2 ");
            default: System.out.print("Default ");
            case 3: System.out.print("3 ");
            break;
            case 4: System.out.print("4 ");
        }
    }
}
```

Was resultiert? Wählen Sie eine Antwort.

- [A] Default
- [B] 3
- [C] Kompilierfehler
- [D] 2
- [E] Default 3
- [F] Laufzeitfehler

### Lösung

E ist richtig. Der Selector Integer ist für die Versionen vor 1.5 falsch und führte zu einem Kompilierfehler. Dank Autoboxing wird ein Integer automatisch in einen int gewandelt und ist deshalb als Case-Selector zulässig.

### Aufgabe 13 (2-5 Exception werfen außerhalb von try und catch)

Gegeben ist folgender Code:

```
public class TryTest{
    public static void main(String... args) throws ArithmeticException {
        throw new IndexOutOfBoundsException(); // #1
        try{
            throw new ArithmeticException();
        }
        catch (ArithmeticException e){ // #2
            System.out.println("ArithmeticException gefangen");
        }
        catch (IndexOutOfBoundsException e){ // #3
            System.out.println("IndexOutOfBoundsException gefangen");
        }
    }
}
```

Welche Aussagen treffen zu? Wählen Sie zwei Antworten.

- [A] Der Code kompiliert und läuft.
- [B] Der Code kompiliert und läuft, falls die mit #1 markierte Zeile gelöscht wird.
- [C] Die geworfene IndexOutOfBoundsException wird in #3 gefangen.
- [D] Der Code gibt bei der Kompilierung die Fehlermeldung »unreachable statement« (unerreichbares Statement) aus.
- [E] Der Code verursacht einen Laufzeitfehler.

### Lösung

B und D sind korrekt. B: Falls die Zeile #1 gelöscht wird, kompiliert der Code. D korrekt, weil durch das Werfen einer Exception außerhalb eines *try-catch*-Blocks die nachfolgenden Statements nicht mehr erreichbar werden (wie wenn an dieser Stelle ein `return` stehen würde), daher die Fehlermeldung »unreachable Statement«.

A und E treffen wegen obigen Ausführungen nicht zu. Zu C kommt es nicht, weil der Code nicht kompiliert.

### Aufgabe 14 (2-4 Exceptions und Vererbung)

```
import java.io.*;
class A{
    void m1() throws ArithmeticException {
        throw new ArithmeticException();
    }
    void m2() throws IOException {
        throw new IOException();
    }
}
class B extends A{

    //Hier einfügen
}
public class Erbstest{
    public static void main(String... args) throws Exception {
        A a=new A();a.m1();a.m2();
        B b=new B();b.m1();b.m2();
    }
}
```

Welcher Codeblock, falls bei //Hier einfügen eingesetzt, erzeugt einen Kompilierfehler? Wählen Sie eine Antwort.

- [A]

```
void m1(){ }
void m2() throws EOFException{
    throw new EOFException();
}
```

- [B]

```
void m1(){ }
void m2() throws Exception{
```

```
throw new Exception();  
}
```

[C]

```
void m1() {}  
void m2() {}
```

[D]

```
void m1() {}  
void m2() throws ArithmeticException {  
    throw new ArithmeticException();  
}
```

[E]

```
void m1() throws ArithmeticException {  
    throw new ArithmeticException();  
}  
void m2() throws ArithmeticException {  
    throw new ArithmeticException();  
}
```

### Lösung

Es gilt: In einer überschriebenen Methode dürfen *keine* neuen oder breitere gecheckte Ausnahmen geworfen werden.

B kompiliert nicht: Erbtest.java:15: m2() in B cannot override m2() in A; overridden method does not throw java.lang.Exception. Exception ist eine breitere gecheckte Exception als EOFException.

A kompiliert, eine EOFException in der Kindklasse ist eine weniger breite Exception als die IOException in der Elternklasse.

C kompiliert, es ist erlaubt, keine Exception in der überschreibenden Methode aufzurufen.

D kompiliert, es ist erlaubt, eine ungecheckte Exception in der überschreibenden Methode aufzurufen.

E kompiliert, eine ungecheckte Ausnahme in m1() kann beliebig überschrieben werden.

### Aufgabe 15 (2-3 Assertions)

Welche der folgenden Blöcke stellen einen angepassten Gebrauch von Assertions dar? Wählen Sie zwei Antworten.

[A]

```
public String test(int i){  
    assert(value>0 && value <10):"Wert muss zwischen 1 und 9 liegen";}
```

 [B]

```
private String test(int i){  
    assert(value>0 && value <10): "Wert muss zwischen 1 und 9 liegen";}
```

 [C]

```
public static void main (String ... args) {  
    assert(args[0]>0 && value <10):"Wert muss zwischen 1 und 9 liegen";}
```

 [D]

```
switch(x) {default:assert false; }
```

### Lösung

B und D sind eine korrekte Verwendung von Assertions.

Assertions sollten nicht Werte überprüfen, die von außen ins Programm kommen, weil diese Werte einen beliebigen Wertebereich haben können. Bei privaten Methoden weiß der Programmierer, welche Bandbreite der Werte normalerweise zu erwarten ist. Assertions werden auch bei Switch-Statements eingesetzt, und zwar dort, wo nie ein Default-Wert zu erwarten ist.

A ist falsch, weil man mit einer Assertion nicht von außen eingespeiste Daten überprüfen soll.

C ist ebenfalls falsch, weil die Form der Kommandozeilenparameter nicht beeinflusst werden kann (an der SCJP-Prüfung kommen dennoch Assertions in Verbindung mit Kommandozeilenparameter vor).

Bemerkung: Assertions sind ein Entwicklertool, sie müssen beim Start mit java mit der Compiler-Option `-ea` oder `-enableassertions` explizit aktiviert werden.

### Aufgabe 16 (2-5 Assertions gültig/ungültig)

Welche der folgenden Ausdrücke kompilieren korrekt *und* erzeugen eine Ausgabe? Wählen Sie zwei korrekte Lösungen.

 [A] `assert false : null;` [B] `assert true : return;` [C] `assert 10<1: Hallo;` [D] `assert false: 8 + 9;`

- [E] `assert true: "Hallo Welt";`
- [F] `assert false "Hallo Welt";`

**Lösung**

A und D sind korrekt.

Prinzip: Es wird nur etwas ausgegeben, falls der boolesche Ausdruck des Assert-Statements *false* ist, also fallen alle Assertions mit *true* weg.

A ist korrekt. Der Code kompiliert, es wird der String »null« ausgegeben (nicht das Literal *null*). Die auszugebende Nachricht muss ein Ausdruck sein, der einen Wert hat. Null als auch eine Zahl ist ein Wert, Zeichenketten müssen von Anführungszeichen umgeben sein.

D ist korrekt, es wird »17« ausgegeben.

B kompiliert nicht, das Schlüsselwort *return* darf nicht in einem Assert-Statement vorkommen.

C kompiliert nicht, weil die Hochkommata bei Hallo fehlen.

E wäre korrekt, aber die Assertion wird nicht ausgelöst, weil die Bedingung *true* ist.

F kompiliert nicht, weil der Doppelpunkt im Assert-Statement fehlt.

**Aufgabe 17 (2-6 Programmstruktur mit Exception)**

```

interface If1{void methode1();}
interface If2{void methode2();}
class A{
  void a(){ }
}
class B    {
  methode1()   {}
  methode2()   {}
}
  
```

Positionieren Sie unten aufgelistete Elemente so, dass der Code kompiliert.

- 
- 
- 
- 
- 
- 
-

`public``RuntimeException``protected`

### Lösung

»protected« und »implements if1« sind überflüssig. In einem Interface definierte Methoden sind implizit *public*, deshalb ist der *public*-Modifizierer vor den Methoden obligatorisch.

```
interface If1{void methode1();}
interface If2{void methode2();}
class A{
    void a(){
    }
}
class B extends A implements If1, If2 {
    public void methode1() throws RuntimeException{}
    public void methode2() throws RuntimeException{}
}
```

### Aufgabe 18 (2-6 Exception Matching)

```
import java.io.*;
public class ExceptionMatching{
    public static void main(String... args){
        int i=10;
        int j=0;
        int k=0;
        try{
            try {
                k=i/j;
            } catch (ArithmeticException e){
                System.out.print("Hallo ");
            }
            File inputDatei = new File("input.txt");
            try {
                FileReader frInputDatei = new FileReader(inputDatei);
            } catch (IOException e){
                System.out.print("du ");
            }
            int[] l=new int[3];
            int m;
            try {
                m=l[3];
            }catch (IndexOutOfBoundsException e){
```

```
        System.out.print("wunder ");
    }
    finally {
        System.out.print("schoene ");
    }
    throw new Exception();
} catch (Exception e){
    System.out.print("Welt ");
}
}
}
```

Was ist der Output des obigen Codes (sofern *input.txt* nicht existiert)? Wählen Sie eine Antwort.

- [A] Kompilierfehler
- [B] Hallo du schoene
- [C] Hallo du schoene Welt
- [D] Hallo du wunder schoene Welt
- [E] Runtime-Exception durch nicht abgefangene Ausnahme

### Lösung

D ist korrekt. Alle Ausnahmen werden abgefangen. Deshalb sind alle restlichen Antworten falsch.

### Aufgabe 19 (2-2 Continue)

```
public class Loop{
    public static void main(String... args){
        mein_label:
        for(int j=0;j<10;j++){
            if (j%2==0)
                continue mein_label;
            System.out.print("Hallo ");
        }
    }
}
```

Wie viele Male wird »Hallo« ausgegeben? Wählen Sie eine Antwort.

- [A] 1
- [B] 2

- [C] 3
- [D] 4
- [E] 5

**Lösung**

E ist korrekt: Jedes zweite Mal, bei geradem j, wird nichts ausgegeben, sondern mit der nächsten Iteration begonnen. *Es wird mit der Iteration beim aktuellen Zählerstand weitergemacht. Zu beachten ist auch, das o%2 ebenfalls 0 ergibt.*

**Aufgabe 20 (2-3 Überschreiben von Methoden, die eine checked Exception werfen)**

```
import java.io.*;
class A{
    public void test(String s) throws IOException{}
}
public class OverrideTest extends A {
    //Hier einfügen
}
```

Welcher der unten aufgelisteten Methoden eingefügt bei //Hier einfügen verursacht einen Kompilierfehler? Wählen Sie eine Antwort.

- [A] public void test(String s) throws IOException{}
- [B] public void test(String s) {}
- [C] public void test(String s) throws Exception {}
- [D] public void test(String s) throws EOFException{}
- [E] public void test(String s) throws RuntimeException{}

**Lösung**

C ist korrekt. Eine Methode, die eine geprüfte Ausnahme (*checked Exception*) wirft, darf nicht mit einer breiteren geprüften Ausnahme (*checked Exception*) überschrieben werden.

Alle anderen Überlagerungen sind zugelassen. EOFException bei D ist eine engere geprüfte Ausnahme und deshalb korrekt.

**10.2.3 Prüfungsziel 3****Aufgabe 21 (3-1, parseX(), valueOf() )**

Gegeben ist folgender Code

```

public class ParseIntValueOfFehler{
  public static void main(String[] args){
    Integer i=Integer.parseInt("10"); // #1
    int k=Integer.parseInt("10"); // #2
    Integer l=Integer.valueOf("10"); // #3
    int m=Integer.valueOf("10"); // #4
    Long n=Long.parseLong("10"); // #5
    Long o=Long.parseLong("10"); // #6
    Integer p=10; // #7
    Integer q=p.intValue(); // #8
    Number x=0; // #9
    float y=10.0f; // #10
    x=y; // #11
  }
}

```

Wählen Sie zwei korrekte Antworten.

- [A] Das Programm kompiliert nicht wegen Zeile #1.
- [B] Das Programm kompiliert.
- [C] Würde das Programm mit Java 1.4 kompiliert, ergäben sich Kompilierfehler.
- [D] Das Programm kompiliert nicht wegen Zeile #5.
- [E] Das Programm kompiliert nicht wegen Zeile #8.

**Lösung**

B und C sind korrekt. Sämtliche Zuweisungen, vor allem diejenigen von *parseX()* an einen Hüllklassentyp sind in Java 5 durch das Auto-Boxing möglich. In Java 1.4 hätte ein Resultat von *parseX()* nicht einer Hüllklassenvariablen zugewiesen werden können, da Auto-Boxing noch nicht existierte.

Alle anderen Antworten sind aus den obigen Erklärungen falsch.

**Aufgabe 22 (3-2, Struktur Datei IO)**

```

import [ ] . * ;
import [ ] [ ] ;
public class B_Reader{
  public static void main(String[] args){
    String zeile=null;
    try{
      [ ] br=new [ ] (new [ ] ( [ ] ));

```

```

while (() {
    out.println(zeile);
}
}
catch(Exception e){}
}
}

```

Ergänzen Sie per Drag & Drop die Lücken.










### Lösung

```

import java.io.*;
import static java.lang.System.*;
public class B_Reader{
public static void main(String[] args){
String zeile=null;
try{
    BufferedReader br=new BufferedReader(new FileReader("input.txt"));
    while ((zeile=br.readLine()) !=null){
        out.println(zeile);
    }
}
catch(Exception e){}
}
}

```

### Aufgabe 23 (3-1, parseX(), valueOf(), intValue())

Welche der folgenden Zeilen würden in Java 1.4 einen Kompilierfehler hervorrufen? Wählen Sie drei korrekte Antworten.

- [A] Integer i=Integer.parseInt("10");
- [B] int k=Integer.parseInt("10");

- [C] Integer l=Integer.valueOf("10");
- [D] int m=Integer.valueOf("10");
- [E] Long n=Long.parseLong("10");
- [F] long o=Long.parseLong("10");
- [F] Integer i=10; int m=i.intValue();

**Lösung**

A, D, E würden einen Kompilierfehler hervorrufen.

*valueOf()* gibt einen Hüllklassentyp zurück, *parseX()* und *intValue()* einen primitiven Typ. Mit dem Auto-Boxing kann ab Java 5 nun das Resultat von *valueOf()* an einen primitiven Typ und *parseX()* (X steht für die Primitivtypen) an einen Hüllklassentyp zugewiesen werden. In Java 1.4 scheitern alle Zuweisungen, wo *valueOf()* an einen Primitiven zugewiesen wird und wo *parseX()* oder *intValue()* an einen Hüllklassentyp zugewiesen wird (Eselsleiter: Die Umwandlungsmethoden, die einen Typ im Namen führen, geben einen Primitivwert zurück).

**Aufgabe 24 (3-2, Struktur PrintWriter)**

```
import  ;
public class B_Writer{
public static void main(String[] args){
String zeile="Hallo Welt";
Console c=System.console();
try{
 pw=new  (new  (new  ("  ")));

pw.  ;
pw.  ;
}
catch() {c.printf("Fehler beim Schreiben");}
}
}
```

Füllen Sie die Lücken mit einer Auswahl folgender Elemente:

- 
- 
- 
- 
- 
-

```

PrintWriter
BufferedWriter
FileWriter
printf(zeile)
close()
IOException e

```

*Lösung*

```

import java.io.*;
public class B_Writer{
public static void main(String[] args){
String zeile="Hallo Welt";
Console c=System.console();
try{
PrintWriter pw=new PrintWriter(new BufferedWriter(new File-
Writer("output.txt")));

pw.printf(zeile);
pw.close();
}
catch(IOException e){c.printf("Fehler beim Schreiben");}
}
}

```

**Aufgabe 25 (3-4, Datumsformatierung mit Locale)**

```

import java.util.*;
public class FormatterTest{
public static void main(String[] args){
Formatter f=new Formatter();
Calendar cal=Calendar.getInstance();
f.format("%td %tB %tY %tH %tM %tS",
cal,cal,cal,cal,cal,cal);
System.out.println(f);
}
}

```

Welcher der aufgelisteten Outputs erscheint, falls %t der Formattierer für die Zeit ist? Es ist der 9. Februar 2007, 15:34:48. Wählen Sie eine Antwort.

- [A] 15 27 30
- [B] 09 Februar 2007 15 34 48
- [C] Februar 09 2007 15 34 48

- [D] 15 34 48 Februar 09 2007
- [E] Laufzeitfehler

### Lösung

B ist korrekt. Mit %td wird der Tag, mit %tB der Monat, voll ausgeschrieben, mit %tY das Jahr vierstellig, und mit %tH, %tM und mit %tS die Stunde, Minute und Sekunde ausgegeben.

### Aufgabe 26 (3-4, Zahlenformatierung mit Locale)

```
import java.util.*;
public class ZahlenFormatierung{
public static void main(String... args){
    double i=new Double( )+new Double( );
    String s=String.( ) ( );
    System.out.println(s);
    Locale lde=new Locale( );
    String strDE=String.( ) ( );
    System.out.println(strDE);
}
}
```

100000.00

lde,"Deutsches-Format: %,.2f",i

50000.00

format

"de","DE"

Ich,"CH-Format: %,.2f",i Schieben Sie die Elemente per Drag & Drop in die Leerstellen (Slots) im obigen Listing, sodass die Ausgabe wie folgt dargestellt wird:

```
CH-Format: 150'000.00
Deutsches Format: 150.000,00
```

### Lösung

```
import java.util.*;
public class ZahlenFormatierung{
public static void main(String... args){
    double i=new Double(100000.00)+new Double(50000.00);
    Locale lch=new Locale("de","CH");
    String s=String.format(lch,"CH-Format: %,.2f",i);
    System.out.println(s);
}
```

```

Locale lde=new Locale("de","DE");
String strDE=String.format(lde,"Deutsches-Format: %,.2f",i);
System.out.println(strDE);
}
}

```

**Aufgabe 27 (3-3, Struktur der Serialisierung/Deserialisierung)**

```

import [ ] ;
public class Serialisieren implements [ ] {
private [ ] int x=10;
private int y=20;
public static void main(String... args){
try{
[ ] ser=new [ ] ();
[ ] oos=new [ ] (new
[ ] ("s.ser"));
oos. [ ] ;
oos. [ ] ();
[ ] ois=new [ ] (new [ ] ("s.ser"));
Serialisieren ser_back=( [ ] )ois. [ ] ;
ois. [ ] ();
System.out.println(ser_back.x);
System.out.println(ser_back.y);
}
catch (Exception e){e.printStackTrace();}
}
}

```

- [ ] readObject()
- [ ] ObjectInputStream
- [ ] close
- [ ] java.io.\*
- [ ] transient
- [ ] volatile
- [ ] FileInputStream
- [ ] FileOutputStream
- [ ] Serializable
- [ ] ObjectOutputStream
- [ ] Serialisieren

Schieben Sie die obigen Elemente per *Drag & Drop* in die Leerstellen (Slots). Das Programm sollte kompilieren und laufen. Die Variable x soll *nicht* zwischengespeichert werden.

## Lösung

```
import java.io.*;
public class Serialisieren implements Serializable{
private transient int x=10;
private int y=20;
    public static void main(String... args){
        try{
            Serialisieren ser=new Serialisieren();
            ObjectOutputStream oos=new ObjectOutputStream(new
                FileOutputStream("s.ser"));

            oos.writeObject(ser);
            oos.close();
            ObjectInputStream ois=new ObjectInputStream(new FileInputStream("s.ser"));
            Serialisieren ser_back=(Serialisieren)ois.readObject();
            ois.close();
            System.out.println(ser_back.x);
            System.out.println(ser_back.y);
        }
        catch (Exception e){e.printStackTrace();}
    }
}
```

In diesem Beispiel wird das Objekt selbst serialisiert, daher muss es zuerst instanziiert werden. Für eine Variable, die nicht gespeichert werden soll, muss transient und nicht volatile als Modifizierer gewählt werden.

### Aufgabe 28 (3-1, String, String Buffer, StringBuilder, Einfügen)

Schieben Sie die Elemente an die korrekte Stelle, so dass folgender Output resultiert: Mit diesem Buch ist die SCJP-Prüfung sehr einfach!

```
public class StringBuilderTest {
    public static void main(String... args){
        StringBuilder s=new StringBuilder("Die SCJP-Prüfung ist sehr schwierig!");
        //Erwartet: Mit diesem Buch ist die SCJP-Prüfung sehr einfach!

        .
        .
        .
        .
        .
        System.out.println(s);
    }
}
```

```
s.insert(41," einfach");  
s.delete(37,41);  
s.insert(0,"Mit diesem Buch ist d");  
s.delete(0,1);  
s.delete(41,51);
```

### Lösung

```
public class StringBuilderTest {  
    public static void main(String... args){  
        StringBuilder s=new StringBuilder("Die SCJP-Prüfung ist sehr schwierig!");  
        //Mit diesem Buch ist die SCJP-Prüfung sehr einfach!  
        s.delete(0,1);  
        s.insert(0,"Mit diesem Buch ist d");  
        s.delete(37,41);  
        s.delete(41,51);  
        s.insert(41," einfach");  
        System.out.println(s);  
    }  
}
```

### Aufgabe 29 (3-5, Stringzerteilung und reguläre Ausdrücke)

```
01 import java.util.*;  
02 public class Tokenizing{  
03     public static void main(String... args){  
04         String s= "Sun Certified Java Programmer";  
05         String[] tokens=s.split("\\d");  
06         Arrays.sort(tokens);  
07         Arrays.reverse(tokens);  
08         for (String s1:tokens)//#3  
09             System.out.print(s1+" ");  
10     }  
11}
```

Was gibt obiges Programm aus? Wählen Sie zwei Antworten.

- [A] Kompilierfehler (Arrays.reverse() existiert nicht)
- [B] Sun Certified Java Programmer
- [C] Certified Java Programmer Sun
- [D] Sun Programmer Java Certified
- [E] Der String wird nicht zerteilt, das Zeichen »\\d« ist falsch.

### Lösung

A und E sind korrekt, `Arrays.reverse()` gibt es nicht. Der String wird nicht zerteilt, weil `\\d`, eine Zahl, als Trenner gewählt wurde. Korrekt wäre `\\s` für einen White-space. `»|«` wird benötigt, weil `»\«` ein Sonderzeichen in String-Literalen ist.

Entfernt man die Zeile 07, würde die Ausgabe `»Sun Certified Java Programmert«` lauten, d.h. der String würde nicht zerteilt. Ersetzt man das `\\d` durch `\\s`, erfolgt erst die erwartete Ausgabe von `»Certified Java Programmert Sun«`, also das lexikographisch sortierte Array.

Die Klasse `Collections` besitzt eine Methode `reverseOrder()`, die die Sortierung einer Collection umkehrt, für Arrays fehlt eine entsprechende Methode. Um einen Array absteigend zu sortieren muss ein `Comparator` implementiert werden.

Siehe <http://java.sun.com/javase/6/docs/api/java/util/Arrays.html>.

```
import java.util.*;
public class Tokenizing{
public static void main(String... args){
    String s= "Sun Certified Java Programmert";
    String[] tokens=s.split("\\s"); //\\d steht für Zahl, \\s für Leerzeichen!
    Arrays.sort(tokens);
    //Arrays.reverse(tokens);//Arrays.reverse() gibt es nicht
        for (String s1:tokens)
            System.out.print(s1+" ");
    }
}
```

Output

Certified Java Programmert Sun

### Aufgabe 30 (3-5, Scanner)

```
import java.util.Scanner;
import java.io.*;
public class ScannerTest{
public static void main(String[] args){
    boolean b=false;
    String str=null;
    int i=0;
    try{
        Scanner s=new Scanner(□ □ □ □);
        //test.txt enthält Hallo true false 1 2 3 tschuess
        while(s.□){
            if(s.□)
                i=s.□;
        }
    }
}
```

```

        else if(s. )
            b=s. ;
            str=s.next();
        }
        System.out.print(i+" "+b+" "+str);
        //gibt »3 true tschuess« aus
    }
    catch(Exception e){}
}
}

```

Positionieren Sie folgende Elemente im obigen Code, sodass dieser kompiliert und läuft. Die Datei text.txt enthält »Hallo true false 1 2 3 tschuess« .

nextInt()
hasNext()
new
)
FileReader
(
hasNextBoolean()
"test.txt"
hasNextInt()
nextBoolean()

### Lösung

```

import java.util.Scanner;
import java.io.*;
public class ScannerTest{
    public static void main(String[] args){
        boolean b=false;
        String str=null;
        int i=0;
        try{
            Scanner s=new Scanner(new FileReader ( "test.txt" ));
            //test.txt enthält Hallo true false 1 2 3 tschuess
            while(s.hasNext()){
                if(s.hasNextInt())
                    i=s.nextInt();
                else if(s.hasNextBoolean())
                    b=s.nextBoolean();
                str=s.next();
            }
        }
    }
}

```

```
    }  
    System.out.print(i+" "+b+" "+str);  
    //gibt "3 true tschuess" aus  
    }  
    catch(Exception e){}  
    }  
}
```

### Aufgabe 31 (3-2, PrintWriter)

```
import java.io.PrintWriter;  
public class PrintWriterTest{  
    public static void main(String... args){  
        try{  
            //PrintWriter pw=new PrintWriter(new BufferedWriter(new FileWriter(new  
            //File("test.dat"))));  
            PrintWriter pw=new PrintWriter(new BufferedWriter(new File-  
            Writer("test.dat")));  
            pw.printf("Hallo Welt");  
            pw.close();  
  
            //BufferedReader br=new BufferedReader(new FileReader(new  
            File("test.dat")));  
            BufferedReader br=new BufferedReader(new FileReader("test.dat"));  
            String meinString="";  
            while((meinString=br.readLine())!=null)  
                System.out.println(meinString);  
  
        }catch (IOException e){  
            e.printStackTrace();  
        }  
    }  
}
```

Welche Ausgabe erzeugt obiger Code? Wählen Sie zwei Antworten.

- [A] Keine Ausgabe
- [B] hallo Welt
- [C] Kompilierfehler
- [D] Laufzeitfehler
- [E] Die mit Doppelslash auskommentierten Zeilen sind gleichwertig zu der jeweils untenstehenden Zeile.

### Lösung

B und E sind korrekt. B ist korrekt, weil der Code kompiliert und läuft. E ist korrekt, weil `FileReader` einen Konstruktor mit einem Filenamen als Parameter besitzt.

## 10.2.4 Prüfungsziel 4

### Aufgabe 32 (4-1, Doppelstart eines Threads)

Betrachten Sie folgenden Code:

```
1 public class DoppelStart extends Thread implements Runnable{
2     public void run(){
3         System.out.println("ich funktioniere");
4     }
5     public static void main(String [] args){
6         Thread t =new Thread(new DoppelStart());
7         t.start();
8         t.start();
9     }
10}
```

Was trifft zu? Wählen Sie eine Antwort.

- [A] Das Programm funktioniert und startet einen neuen Thread.
- [B] Das Programm meldet einen Kompilierfehler in Zeile 6.
- [C] Das Programm verursacht einen Laufzeitfehler in Zeile 6.
- [D] Das Programm startet und erzeugt zwei neue Threads.
- [E] Das Programm erzeugt einen Laufzeitfehler in Zeile 8.

### Lösung

E ist richtig. Ein Doppelstart führt zu einem Laufzeitfehler. Zeile 6 ist korrekt, und das »`extends Thread`« ist überflüssig, führt aber zu keinem Fehler. Wird der zweite Ausdruck `t.start()` auskommentiert, funktioniert obiges Listing perfekt.

Fehlermeldung:

```
Exception in thread "main" java.lang.IllegalThreadStateException
ich funktioniere
    at java.lang.Thread.start(Thread.java:571)
    at DoppelStart.main(DoppelStart.java:9)
```

**Aufgabe 33 (4-1, Threads erzeugen)**

Gegeben ist folgender Code:

```
public class ThreadVariationen // #1 Hier einfügen
public void run(){
    System.out.println("Thread");
}
public static void main(String [] args){
    // #2 hier einfügen
    t.start();
}
}
```

Welche Kombinationen ergeben bei #1 und #2 eingefügt einen korrekt funktionierenden Thread mit der Ausgabe »Thread« ? Wählen Sie drei korrekte Lösungen.

A

```
extends Thread implements Runnable{
    Thread t =new Thread(new ThreadVariationen());
```

B

```
implements Runnable{
    Thread t =new Thread(new ThreadVariationen());
```

C

```
extends Thread{
    Thread t =new Thread();
```

D

```
extends Thread{
    Thread t =new ThreadVariationen();
```

E

```
extends Thread{
    Thread t =new Runnable();
```

F

```
implements Thread{
    Thread t =new ThreadVariationen();
```

### Lösung

A ist korrekt, *extends* ist zwar überflüssig, stört aber nicht.

B ist korrekt und entspricht der Konstruktion mit der Implementation von *Runnable*.

D ist korrekt und entspricht der Deklaration über die Erweiterung der Klasse *Thread*.

C kompiliert zwar korrekt, gibt aber keine Ausgabe, weil nur die Klasse *Thread* instanziiert wird.

E ist falsch, ein Interface (*Runnable*) besitzt keinen Konstruktor und kann nicht instanziiert werden.

F ist falsch, weil *Thread* eine Klasse und kein Interface ist und daher nicht implementiert werden kann.

### Aufgabe 34 (4-4, Synchronisation)

```
public class SynchroTest implements Runnable{
    private int zaehler=0;
    public void zaehlMethod(){
        synchronized (this){
            for(int i=0;i<10;i++) {
                System.out.println(Thread.currentThread().getName()+" "+zaehler++);
                try {Thread.sleep(1000);} catch (InterruptedException e){};
            }
        }
    }
    public void run(){
        zaehlMethod();
    }
    public static void main(String... args){
        Thread t = new Thread(new SynchroTest());
        t.setName("ping");
        t.start();
        Thread t1 = new Thread(new SynchroTest());
        t1.setName("pong");
        t1.start();
    }
}
```

Output

```
ping 0
pong 0
ping 1
pong 1
```

```
...  
ping 9  
pong 9
```

Welche Aussagen zum obigen Code treffen zu? Wählen Sie vier Antworten.

- A Der Code innerhalb der Methode »zaehlMethode()« ist synchronisiert, es kann immer nur ein einziger Thread auf diesen Code zugreifen.
- B Anstatt des Blocks *synchronized (this) {}* hätte man genauso gut *public synchronized void zaehlMethode()* schreiben können.
- C Die Reihenfolge des Outputs (Abwechslung zwischen ping und pong) ist durch die Synchronisierung gegeben.
- D Ließe man den *synchronized*-Block weg, wäre die Ausgabe zehn Mal ping und zehn Mal pong.
- E Die Reihenfolge wird in diesem Beispiel von der Synchronisierung nicht beeinflusst. Vielmehr durch das *sleep(1000)*, das den Thread in einen Schlafzustand versetzt und dem anderen Thread Gelegenheit gibt zu laufen.
- F Ließe man *sleep()* weg, ist die Abwechslung zwischen den Threads unregelmäßig, da der Thread, der aktuell am Laufen ist, einem anderen Thread keine Zeit zum Laufen gibt.

### Lösung

A, B, E und F sind korrekt.

C und D sind falsch, auch wenn der Code synchronisiert ist, heißt es nicht, dass die Reihenfolge der Ausführung damit festgelegt ist. Die Synchronisierung verhindert nur den gleichzeitigen Zugriff von mehreren Threads auf eine Ressource.

### Aufgabe 35 (4-2, Thread Methoden, Zugehörigkeit)

Ordnen Sie per Drag & Drop folgende Methoden den Klassen *java.lang.Object*, *java.lang.Thread* und dem Interface *Runnable* zu.

wait()
sleep()
join()
notify()
notifyAll()
start()
run()
yield()

Java.lang.Object	Java.lang.Thread	Interface Runnable
------------------	------------------	--------------------

*Lösung*

Java.lang.Object	Java.lang.Thread	Interface Runnable
notify()	start()	run()
notifyAll()	sleep()	
wait()	yield()	
	join()	

**Aufgabe 36 (4-1, Start eines Threads)**

```

class A implements Runnable{
    public void run(){
        for (int i=1;i<5;i++)
            System.out.println(Thread.currentThread().getName()+" "+ i);
    }
}
class B extends Thread{
    public B(String s){
        super(s);
    }
    public void run(){
        for (int i=1;i<5;i++)
            System.out.println(Thread.currentThread().getName()+" "+ i);
    }
}
class D extends Thread{
    public void run() {
        int i=0;
        for (i=0;i<10;i++)
            System.out.println("Zaehler: "+i);
    }
}
class E implements Runnable{
    public void run(){
        for (int i=1;i<10;i++)
            System.out.print("E "+i);
    }
}

```

```
public class StartMethoden{
public static void main(String ... args){
    //#1
    Thread a =new Thread(new A(), "A");
    a.start();
    //#2
    Thread b=new B("B");
    b.start();
    //#3
    Thread c = new Thread() {//Objekt-Körper!
    public void run() {
    int i=0;
        for (i=0;i<10;i++)
            System.out.println("Zaehler: "+i);
        }
    };
    c.start();
    //#4
    new Thread(new D()).start();
    //#5
    new Thread(new E()).start();
    //#6
    Runnable f=new Runnable();
    f.run();
    }
}
```

Im obigen Programm werden Threads auf verschiedene Arten gestartet. Wählen Sie eine Antwort.

- [A] Alle Startarten von #1 bis #6 sind korrekt.
- [B] #1 ist falsch.
- [C] #2 ist falsch.
- [D] #3 ist falsch.
- [E] #4 ist falsch.
- [F] #5 ist falsch.
- [G] #6 ist falsch.

### Lösung

G ist korrekt, nur #6, `Runnable f=new Runnable()`, ist falsch. Das Interface `Runnable` hat keinen Konstruktor. Bei #6 darf die `run()`-Methode zwar explizit auf-

gerufen werden, f. `run()`, `run()` läuft dann aber wie eine gewöhnliche Methode, und kein neuer Thread wird gestartet.

### Aufgabe 37 (4-4, Wait und Notify)

```

class Stack{
    int stand=0;
} //Ende Klasse Stack
class Push extends Thread{
    Stack stack;
    public Push(Stack stack){
        this.stack=stack;
    }
    public void run(){
        while(true){ //Endlosschleife
            int s;
            synchronized (stack){
                s=stack.stand; //Bestand auslesen
                if (s>9){
                    try {
                        System.out.println("Kueche wartet...");
                        stack.wait(); //Warten, bis Produzent ein notify() sendet
                    } catch (InterruptedException e){}
                }else {
                    s++; // Produktions-Code
                    stack.stand=s; //Produktion einlagern
                    System.out.println("Teller eingefügt: "+s);
                    stack.notify(); //Dem anderen Thread melden, dass neue Teller vorhanden
                }
                try {
                    Thread.sleep(100); //nach der Produktion schlafen...
                }catch (InterruptedException e) {}
            } //Ende Synchronized
        } //Ende while(true)
    } //Ende Run
} //Ende Klasse Push

class Pop extends Thread{
    Stack stack;
    public Pop(Stack stack){
        this.stack=stack;
    }
    public void run(){
        while(true){

```

```
synchronized(stack){
    int s;
    s=stack.stand;
    if (s<=1){
        try {
            System.out.println("Kellner wartet...");
            stack.wait(); //Warten, bis Produzent ein notify() sendet
        } catch (InterruptedException e){}
    }else {
        s--;
        System.out.println("Teller entfernt "+s);
        stack.stand=s;
        stack.notify();
    }//Ende if s<1
    try {
        Thread.sleep(3000);
    }catch (InterruptedException e){}
} //Ende Synchronized
} //Ende while(true)
} //Ende Run
} //Ende Klasse Pop
public class StackTest{
    public static void main(String... args){
        Stack s=new Stack();
        Push push=new Push(s);
        Push push1=new Push(s);
        Pop pop=new Pop(s);
        push.start();
        push1.start();
        pop.start();
    } //Ende Main
} //Ende StackTest
```

Welche Aussagen zum obigen Code treffen zu? Wählen Sie vier Antworten.

- [A] Push ist das gemeinsame Objekt, der Zugriff erfolgt über eine Objektreferenz.
- [B] Stack ist das gemeinsame Objekt, der Zugriff erfolgt über eine Objektreferenz.
- [C] Der Aufruf der *notify()*-Methode auf einem gemeinsamen Objekt weckt einen einzigen wartenden Thread.
- [D] Der Aufruf der *wait()*-Methode auf einem gemeinsamen Objekt bewirkt, dass der aktuelle Thread seinen Lock aufgibt.

- [E] Die Produktion ist viel rascher als der Konsum, weil zwei Push-Threads vorhanden sind. Deshalb ist der Stack bald gefüllt (10 Teller).
- [F] Die Produktion ist langsamer als der Konsum, deshalb schwankt der Tellerstand zwischen einem und zwei Stück.

**Lösung**

B, C, D und E sind korrekt.

A ist falsch, weil das gemeinsame Objekt der Stack ist.

F ist falsch, weil die Produktion bzw. das Füllen des Stacks viel rascher ist (weniger Verzögerung, zwei Threads push und push1, die produzieren).

**Aufgabe 38 (4-1, Methode join())**

```
public class JoinTest extends Thread{
    public JoinTest(String name){
        super(name);
    }
    public void run(){
        for (int i=0;i<5;i++){
            try {
                Thread.sleep(1000);
            }catch (InterruptedException e){ }
            System.out.print(this.getName());
        }
    }

    public static void main(String... args){
        JoinTest t1=new JoinTest("T1 ");
        JoinTest t2=new JoinTest("T2 ");
        t1.start();
        try{
            t1.join();
        }catch (InterruptedException e){}
        t2.start();
    }
}
```

Welche Aussagen treffen zu? Wählen Sie zwei Antworten.

- [A] Der Output lautet: T1 T2 T1 T2 T1 T2 T1 T2 T1 T2
- [B] Der Output lautet: T1 T1 T1 T1 T1 T2 T2 T2 T2 T2

- [C] Die Output-Reihenfolge kann nicht vorhergesagt werden, er hängt von der JVM ab
- [D] Die `join()`-Methode bewirkt, dass der aufrufende Thread (hier der `main`-Thread) in den Status `blocked-for-join-completion` verfällt, also blockiert. Der `main`-Thread ist solange blockiert, bis `t1` tot ist, erst dann wird `t2` vom `main`-Thread gestartet.

### Lösung

B und D sind korrekt. Durch die `join()`-Methode ist die Reihenfolge der Threads gegeben. Bemerkung: Im obigen Programm laufen drei Threads, der `main`-Thread und die Threads `t1` und `t2`. Mit `t1.join()` führt der `main`-Thread die `join()`-Methode auf den Thread `t1` aus und verfällt selbst in den Status `blocked-for-join-completion`.

### Aufgabe 39 (4-2, Methode `yield()`)

```
public class YieldTest extends Thread{
    public YieldTest(String name){
        super(name);
    }
    public void run(){
        for (int i=0;i<5;i++){
            if (i>2) Thread.yield();
            /*****
            try {
                Thread.sleep(1000);
            }catch (InterruptedException e){ }
            *****/
            System.out.print(this.getName());
        }
    }
    public static void main(String... args){
        YieldTest t1=new YieldTest("T1 ");
        YieldTest t2=new YieldTest("T2 ");
        t1.start();
        t2.start();
    }
}
```

Wie lautet der Output des obigen Codes? Wählen Sie drei.

- [A] Der Output lautet: T1 T2 T1 T2 T1 T2 T1 T2 T1 T2
- [B] Der Output lautet: T1 T1 T1 T1 T1 T2 T2 T2 T2 T2

- [C] Der Output lautet: T<sub>1</sub> T<sub>1</sub> T<sub>1</sub> T<sub>2</sub> T<sub>2</sub> T<sub>2</sub> T<sub>1</sub> T<sub>2</sub> T<sub>1</sub> T<sub>2</sub>, die Reihenfolge der letzten Elemente ist zufällig.
- [D] Die Output-Reihenfolge kann nicht vorhergesagt werden, er hängt von der JVM ab.
- [E] Würde man die *sleep()*-Methode im Kommentarblock aktivieren, lautete die Ausgabe: T<sub>1</sub> T<sub>2</sub> T<sub>1</sub> T<sub>2</sub> T<sub>1</sub> T<sub>2</sub> T<sub>1</sub> T<sub>2</sub> T<sub>1</sub> T<sub>2</sub>

### Lösung

C und E sind korrekt. Sobald  $i > 2$  ist, überlässt der aktuelle Thread die Ausführung einem anderen Thread.

E ist korrekt, weil der zweite Thread immer Laufzeit erhält, falls eine Verzögerung durch *sleep()* vorliegt. *yield()* ist in diesem Falle wirkungslos.

### Aufgabe 40 (4-3, Deadlock)

```

01 public void ersteMethode() {
02     synchronized(erstesObjekt) {
03         synchronized(zweitesObjekt) {
04             tuWas();
05         }
06     }
07 }
08 public void zweiteMethode() {
09     synchronized(zweitesObjekt) {
10         synchronized(erstesObjekt) {
11             tuWasAnderes();
12         }
13     }
14 }

```

Was ist mit obigem Code los? Wählen Sie zwei korrekte Antworten.

- [A] Normaler Code, kompiliert und läuft.
- [B] Verschachtelte Synchronisationsblöcke in umgekehrter Reihenfolge können zu einer Blockierung führen.
- [C] Verschachtelte Synchronisationsblöcke führen zu einem Kompilierfehler.
- [D] Die Output-Reihenfolge kann nicht vorhergesagt werden, sie hängt von der JVM ab.
- [E] Durch Änderung der Reihenfolge der Synchronisation können Blockierungen verhindert werden.

### Lösung

B und E sind richtig.

Szenario: Ein erster Thread führt die *ersteMethode()* aus. Er hat den Lock auf das ersteObjekt bekommen und will jetzt den Lock auf das »zweiteObjekt«. Zufällig führt ein zweiter Thread gleichzeitig die »zweitenMethode« aus. Er hat das zweite Objekt schon gelockt. Der erste Thread wartet auf das zweite Objekt. Der zweite Thread will nun das erste Objekt locken, dieses wird aber schon vom ersten Thread in Beschlag genommen. Es resultiert ein Deadlock.

Falls die Reihenfolge der Synchronisationsblöcke in beiden Methoden gleich ist und nicht wechselseitig verschieden, ist das Problem gelöst, weil dann zuerst das erste Objekt gelockt wird und damit implizit auch das zweite.

### Aufgabe 41 (4-4, Eigenschaften der Notify-Methode)

Welches sind die Eigenschaften der Notify-Methode? Wählen Sie 3 Antworten.

- [A] Sie ist eine Instanzmethode der Klasse Object.
- [B] Sie ist eine statische Methode der Klasse Object.
- [C] Der aktuelle Thread benötigt einen Lock auf das gemeinsame Objekt, auf das die Notify-Methode ausgeführt wird, sonst resultiert eine IllegalMonitorStateException.
- [D] Die Notify-Methode kann nur in einem synchronisierten Block bzw. in einer synchronisierten Methode aufgerufen werden.
- [E] Falls mehrere Threads warten, kann bestimmt werden, welcher an die Reihe kommt.

### Lösung

A, C und D sind korrekt.

B ist falsch, weil A richtig ist. E ist falsch, weil nicht bestimmt werden kann, welcher Thread den Lock bekommen soll.

Bemerkung: Die *notify()*-, *notifyAll()*- und *wait()*-Methoden werden immer auf den gemeinsamen Objekten ausgeführt.

## 10.2.5 Prüfungsziel 5

### Aufgabe 42 (5-4, Korrektes Überschreiben)

Folgender Code ist gegeben

```
class A{  
    Double test(){
```

```
        return 5.0;
    }
}
class B extends A{
    int test(){
        return 5;
    }
}
class C extends B{
    String test(){
        return ("Hallo");
    }
}
public class TestOverload{
public static void main(String...args){
    A a=new A();
    System.out.print(a.test()+" ");
    B b =new B();
    System.out.print(b.test()+" ");
    C c =new C();
    System.out.print(c.test()+" ");
    A d=new C();
    System.out.print(d.test()+" ");
    A e =new B();
    System.out.print(e.test()+" ");
}
}
```

Was wird ausgegeben? Wählen Sie eine Antwort.

- [A] 5.0 5 Hallo 5.0 5.0
- [B] 5.0 5 Hallo 5.0 5
- [C] Kompilierfehler
- [D] 5.0 5 Hallo 5 5
- [E] 5.0 5 Hallo 5 5

### Lösung

C ist richtig, ein Override mit verschiedenen Return-Typen führt zu einem Kompilierfehler.

A wäre richtig, falls die Methoden als Overload programmiert worden wären, also mit verschiedenen Parameterlisten wie im Listing unten.

```
class A{
    Double test(Double x){
        return 5.0;
    }
}
class B extends A{
    int test(int i){
        return 5;
    }
}

class C extends B{
    String test(String s){
        return ("Hallo");
    }
}

public class TestOverload{
public static void main(String...args){
    A a=new A();
    System.out.print(a.test(5.0)+" ");
    B b =new B();
    System.out.print(b.test(5)+" ");
    C c =new C();
    System.out.print(c.test("test"+" ");
    A d=new C();
    System.out.print(d.test(5.0)+" ");
    A e =new B();
    System.out.print(e.test(5.0)+" ");
}
}
```

### Aufgabe 43 (5-2, Up- und Downcasting)

```
interface Iface{public void machWas();}
class A implements Iface{public void machWas(){
    System.out.print("A");
}
}
class B extends A{
    public void machWas(){
        System.out.print("B");
    }
    public void machWasAnderes(){
        System.out.print("Anderes");
    }
}
```

```
    }  
  }  
  class C extends A{  
    public void machWas(){  
      System.out.print("C");  
    }  
  }  
  public class SimpleCast{  
    public static void main(String... args){  
      A a0=new A();  
      a0.machWas();  
      A a1=(A) new B(); // #1  
      a1.machWas();  
      if (a1 instanceof B){  
        B b1=(B) a1; // #2  
        b1.machWasAnderes();  
      }  
      A a2=(A) new C();  
      a2.machWas();  
    }  
  }  
}
```

Was trifft für den obigen Code zu? Wählen Sie 3 richtige Antworten.

- [A] Die Ausgabe ist ABAnderesC.
- [B] Die Ausgabe ist AAAAnderesC.
- [C] Bei #1 findet ein Upcasting statt.
- [D] Bei #1 findet ein Downcasting statt.
- [E] Bei #2 findet ein Upcasting statt.
- [F] Bei #2 findet ein Downcasting statt.

### Lösung

A, C und F sind richtig. Die Ausgabe ist ABAnderesC, weil die Polymorphie greift. Bei #1 findet ein Upcasting statt, einem allgemeineren Typ wird ein spezifischer Typ zugewiesen, die Angabe von (A) ist überflüssig. Bei #2 findet ein Downcasting statt, weil einer spezifischen Referenz eine allgemeine Referenz zugewiesen wird.

Der instanceof Operator wertet zu true aus, falls das Objekt, d.h. die rechte Seite der Instantiierungs-Gleichung, vom Typ B ist.

**Aufgabe 44 (5-4, Overloading und Auto-Boxing)**

```
public class AutoB{
    public static void main(String [] args){
        drucke(3.0);
        drucke(3);
        drucke(3.0f);
    }
    static void drucke(Float n){
        System.out.print("Float ");
    }
    static void drucke(Integer n){
        System.out.print("Integer ");
    }
    static void drucke(Number n){
        System.out.print("Number ");
    }
}
```

Welche zwei Aussagen treffen zu?

- [A] Die Ausgabe ist »Number Integer Float«.
- [B] Die Ausgabe ist »Integer Float«.
- [C] Das Programm kompiliert und läuft.
- [D] Das Programm gibt Kompilierfehler aus, *drucke(double)* wird nicht gefunden.
- [E] Die Ausgabe ist »Float, Integer, Number«.

**Lösung**

A und C sind richtig. Das Auto-Boxing von *double* nach *Number* funktioniert. Wäre keine Methode *drucke(Number n)* vorhanden, wäre Antwort D richtig, weil die Methode *drucke(double)* nicht gefunden würde. B und E sind aufgrund der obigen Ausführungen falsch.

**Aufgabe 45 (5-5, ist-ein, hat-ein)**

```
class Tier{}
```

```
class Hund{}
```

```
class Herrchen{}
```

Welche Relationen könnten für diese Klassen bestehen? Wählen Sie vier.

- [A] Ein Hund ist ein Herrchen.
- [B] Ein Hund hat ein Herrchen.
- [C] Ein Hund ist ein Tier.
- [D] Ein Tier ist ein Hund.
- [E] Ein Herrchen hat einen Hund.
- [F] Eine Hat-ein-Beziehung nennt man auch eine Komposition.

### Lösung

B, C, E und F sind richtig.

A ist falsch, weil ein Hund kein Herrchen ist.

D ist falsch, weil nicht jedes Tier ein Hund ist.

### Aufgabe 46 (5-1, Koppelung und Kohäsion)

```
#1
class Bankkonto{
public druckeFeriengutschein(){}
}
#2
class Bankkonto{
public einzahlen(){}
public auszahlen(){}
}
#3
class Bankkonto{
Kunde kunde;
printKunde();
}
```

Ordnen Sie die untenstehenden Begriffe den Codes zu.

- Enge Koppelung
- Lockere Koppelung
- Hohe Kohäsion
- Schwache Kohäsion

## Lösung

Enge Koppelung #3

Lockere Koppelung

Hohe Kohäsion #2

Schwache Kohäsion #1

Bemerkung:

Koppelung bedeutet den Zusammenhang zwischen Objekten. Ist ein Objekt stark von einem anderen abhängig, ist die Koppelung eng. Enge Koppelung ist unerwünscht. Bei #3 ist der Kunde eng an das Bankkonto gekoppelt.

Kohäsion bedeutet die Konzentration auf ein Thema. In #2 gehören *einzahlen()* und *auszahlen()* thematisch zum Bankkonto. Eine schwache Kohäsion findet sich bei #1, weil *druckeFeriengutschein()* nicht viel mit einem Bankkonto zu tun hat. Schwache Kohäsion ist unerwünscht.

## Aufgabe 47 (5-2, privater Konstruktor)

```
class SuperKlasse{}
class SubKlasse extends SuperKlasse{
    private SubKlasse(){
        System.out.println("im Konstruktor der SubKlasse");
    }
    //public static void main(String [] args){
    //SubKlasse s=new SubKlasse();
    //}
}
public class TestConstr{
    public static void main(String [] args){
        SubKlasse s=new SubKlasse();
    }
}
```

Was trifft für den obigen Code zu? Wählen Sie drei Antworten.

- [A] Der Code kompiliert einwandfrei.
- [B] Der Code kompiliert nicht.
- [C] Falls die Methode *main()* in die SubKlasse verlegt wird, kompiliert der Code einwandfrei.
- [D] Private Konstruktoren sind verboten.
- [E] Private Konstruktoren sind nur erlaubt, falls sich die *main()* Methode in der gleichen Klasse wie der private Konstruktor befindet.

### Lösung

B, C und E sind korrekt. A ist falsch, weil eine Fehlermeldung erscheint »ungültige Methode, fehlender Return-Typ«. D ist falsch, weil private Konstruktoren nicht generell verboten sind und funktionieren, falls die `main()`-Methode in der gleichen Klasse ist.

### Aufgabe 48 (5-2, Overriding)

```
class A{
    void methode(String s){}
}

abstract class B extends A{
    //Hier einfügen
}
```

Welcher der folgenden Code-Stücke kann unabhängig bei //Hier einfügen eingesetzt werden und ermöglicht ein einwandfreies Kompilieren? Wählen Sie drei Antworten.

- [A] `abstract void methode(String s);`
- [B] `protected void methode(String s) throws Error{}`
- [C] `void methode(String s) {}`
- [D] `int methode(String s) {}`
- [E] `private void methode(String s) {}`
- [F] `abstract void methode(String s){}`

### Lösung

A ist korrekt, weil eine abstrakte Methode implizit `public` ist und damit der Zugriffsmodifizierer der überschreibenden Methode auf jeden Fall weniger restriktiv ist. Korrekt ist auch, dass die Methode keine Implementation hat. Die Geschweiften Klammern fehlen korrekterweise. Eine abstrakte Klasse darf Methoden von einer konkreten Klasse überschreiben. Auch darf eine abstrakte Klasse konkrete Methoden besitzen.

B ist korrekt, weil `protected` ein weniger restriktiver Zugriffsmodifizierer ist als der `package`-Modifizierer (`default` oder `package` = kein Modifizierer). `Error` ist eine ungecheckte Ausnahme, weshalb das Überschreiben erlaubt ist.

C ist korrekt, weil der Zugriffsmodifizierer der überschreibenden Methode der gleiche wie der der überschriebenen ist, nämlich `default` bzw. `package`.

D ist falsch, weil der Rückgabotyp *int* nicht kompatibel zum Rückgabotyp (*void*) der überschriebenen Methode ist.

E ist falsch, weil die überschreibende Methode einen restriktiveren Modifizierer besitzt (Fehlermeldung: `cannot override methode(java.lang.String) in A; attempting to assign weaker access privileges; was package`).

F ist falsch, weil eine abstrakte Methode keinen Körper (Body) haben kann.

```
class A{
    void methode(String s){}
}
abstract class B extends A{
//Hier einfügen
abstract void methode(String s); //A ok
//protected void methode(String s) throws Error{} //B ok
//void methode(String s) {} //C ok

//int methode(String s) {} //D falscher Return-Typ
//private void methode(String s) {}//E restriktiverer Zugriffsmodifizierer
//abstract void methode(String s){} //F {} nicht erlaubt
}
```

#### Aufgabe 49 (5-2, Objektreferenz-Casting)

Nehmen Sie an, Hund erweitere Tier. Welche der folgenden Zuweisungen kompilieren korrekt? Wählen Sie drei richtige Antworten.

```
Tier t=new Tier();
Hund h=new Hund();
```

- [A] t=h
- [B] h=t
- [C] h=(Hund)t
- [D] t=(Tier)h
- [E] h=(Tier)t

#### Lösung

A ist korrekt, Zuweisung von etwas Speziellerem an etwas Generelleres funktioniert ohne expliziten Cast.

C ist korrekt, ein expliziter Cast ist notwendig, falls etwas Spezielleres etwas Generellerem zugewiesen wird.

D ist korrekt, kompiliert, der Cast ist allerdings überflüssig.

B ist falsch, ein Cast wäre nötig.

E ist falsch, innerhalb des Cast-Operators (der runden Klammern) müsste *Hund* stehen.

### Aufgabe 50 (5-5, Overloading)

```
class A{
    void methode(String s){}
}
abstract class B extends A{
    //Hier einfügen
}
```

Welcher der folgenden Code-Stücke kann unabhängig bei //Hier einfügen eingesetzt werden und verursacht einen Kompilierfehler? Wählen Sie zwei Lösungen.

- [A] void methode(String s) {}
- [B] protected void methode(int i) throws Error{}
- [C] int methode(String s);
- [D] int methode(int i) {}
- [E] private void methode(int k) {}
- [F] abstract void methode(double d){}

### Lösung

C und F verursachen einen Kompilierfehler. C ist ein falscher Override mit anderem Return-Typ, und F ist eine abstrakte Methode mit einer nicht zulässigen Implementation (Methodenkörper).

Prinzip: Beim Überladen kann nichts falsch gemacht werden. Falls ein Fehler auftritt, ist es meist ein nicht korrektes Überschreiben oder ein sonstiger Fehler.

```
class A{
    void methode(String s){}
}
abstract class B extends A{
    //Hier einfügen
    void methode(String s) {} //ok
    //protected void methode(int i) throws Error{} //ok
    //int methode(String s); //incompatible return type
    //int methode(int i) {return i;}; //Ok
```

```
//private void methode(int k) {} //ok
//abstract void methode(double d){} //abstract methods cannot have a body
}
```

### Aufgabe 51 (5-2, Überdecken statischer Methoden)

```
class A{
    static void methode(){}
}
class B extends A{
    void methode() {//#1
        System.out.println("es funktioniert");}
}
public class UeberschreibenStatisch{
    public static void main(String... args){
        B b=new B();
        b.methode(); //#2
    }
}
```

Wählen Sie zwei korrekte Aussagen:

- [A] Der Code kompiliert korrekt und gibt »es funktioniert« aus.
- [B] Es resultiert eine Fehlermeldung:

```
methode() in B cannot override methode() in A; overridden method is static.
```

- [C] Es resultiert eine Fehlermeldung:

```
duplicate methods: methode() in B cannot be doubled.
```

- [D] Der Code funktioniert, falls man bei #1 *void methode(int i)* einsetzen würde und den Aufruf in #2 in *b.methode(1)* ändern würde.

### Lösung

B und D sind richtig.

A ist falsch, weil das Programm nicht kompiliert. C ist falsch, weil eine solche Fehlermeldung nicht existiert.

Bei D macht man aus einem Override einen Overload, der funktioniert.

```
class A{
    static void methode(){}
}
```

```

class B extends A{
    void methode() { //#1
        //void methode(int i) {
            System.out.println("es funktioniert");
        }
    }
}
public class ueberschreibenStatisch{
    public static void main(String... args){
        B b=new B();
        b.methode();    //#2
        //b.methode(1);
    }
}
    
```

### 10.2.6 Prüfungsziel 6

#### Aufgabe 52 (6-1, Collection-Methoden)

Methoden	Collection-Klasse
Viele Zugriffe über einen Index, keine Thread-Sicherheit	
Speicherung ohne Schlüssel, keine Duplikate, sehr schneller Zugriff	
Assoziative Speicherung mit Schlüssel, nicht thread-sicher, schneller Zugriff	
Viele Zugriffe über einen Index, mit Thread-Sicherheit	
Häufiges Einfügen/Löschen am Anfang der Liste, wenige Zugriffe über Index	
Assoziative Speicherung mit Schlüssel-Wert-Paar, thread-sicher	
Geordnete Liste mit assoziativem Schlüssel-Wert-Paar	
Warteschlange	

Ordnen Sie den Anforderungen per *Drag & Drop* Collection-Klassen zu.

- PriorityQueue
- HashTable
- LinkedList
- Vector
- HashMap
- HashSet
- ArrayList
- TreeMap

## Lösung

Methoden	Collection-Klasse
Viele Zugriffe über einen Index, keine Thread-Sicherheit	ArrayList
Speicherung ohne Schlüssel, keine Duplikate, sehr schneller Zugriff	HashSet
Assoziative Speicherung mit Schlüssel, nicht thread-sicher, schneller Zugriff	HashMap
Viele Zugriffe über einen Index, mit Thread-Sicherheit	Vector
Häufiges Einfügen/Löschen am Anfang der Liste, wenige Zugriffe über Index	LinkedList
Assoziative Speicherung mit Schlüssel-Wert-Paar, thread-sicher	HashTable
Geordnete Liste mit assoziativem Schlüssel-Wert-Paar	TreeMap
Warteschlange	PriorityQueue

### Aufgabe 53 (6-2, hashCode())

Welche Eigenschaften treffen für *hashCode()* zu? Wählen Sie drei Möglichkeiten.

- [A] Falls *hashCode()* in einem Objekt, das in ein HashSet gespeichert werden sollte, nicht überschrieben wurde, werden Objekte mit gleichem Inhalt separat gespeichert.
- [B] *hashCode()* ist in *Object* nicht implementiert.
- [C] Die Methode *hashCode()*, falls sie nicht überschrieben wurde, liefert für Objekte gleichen Inhalts eine unterschiedliche *int*-Zahl.
- [D] In der Klasse *String* wird *hashCode()* so überschrieben, dass bei gleichem Inhalt ein identischer *int*-Wert zurückgegeben wird.
- [E] Bei *StringBuilder* und *StringBuffer* wurde *hashCode()* genau wie in der Klasse *String* überschrieben.

## Lösung

A, C und D sind korrekt. A und B sagen das Gleiche aus. D ist korrekt, da für gleichen Inhalt der gleiche *HashCode* zurückgegeben wird.

B ist falsch, weil *hashCode()* in *Object* implementiert ist, E ist falsch, weil weder *StringBuilder* noch *StringBuffer* *hashCode()* überschreiben.

### Aufgabe 54 (6-5, Arrays.sort, binarySearch())

```
import java.util.*;
public class HilfsklasseArrays{
```

```

public static void main(String... args){
    Integer a[]={5,7,3,77,9,8,1,3};
    [ ].sort([ ]);
    [ ]<Integer> a1= [ ](a);
    int index= [ ]. [ ]([ ], [ ]);
    System.out.print(a1. [ ](index));
}
}

```

Positionieren Sie per *Drag & Drop* folgende Elemente im obigen Listing, sodass 77 ausgegeben wird.

Arrays
List
get
binarySearch
Arrays.asList
ArrayList
Collections
a1
77
a

### Lösung

```
ArrayList
```

ist überflüssig.

```

import java.util.*;
public class HilfsklasseArrays{
    public static void main(String... args){
        Integer a[]={5,7,3,77,9,8,1,3};
        Arrays.sort(a);
        List<Integer> a1=Arrays.asList(a);
        int index=Collections.binarySearch(a1,77);
        System.out.print(a1.get(index));
    }
}

```

Programm mit Zwischenausgabe

```

import java.util.*;
public class HilfsklasseArrays{
    public static void main(String... args){

```

```

Integer a[]={5,7,3,77,9,8,1,3};
Arrays.sort(a);
List<Integer> al=Arrays.asList(a); //nicht ArrayList
//for (Integer i:a)
//System.out.print(i+" ");
//System.out.println();
int index=Collections.binarySearch(al,77);
System.out.print(al.get(index));
}
}

```

### Aufgabe 55 (6-1, Auswahl einer passenden Collection)

Weisen Sie folgenden Anwendungen die passende Collection per *Drag & Drop* zu.

Telefonbuch .

Eine sortierte Liste von Nachnamen .

Eine unsortierte Liste mit Lottozahlen .

Eine nach Einfügedatum sortierte Liste von Notizen .

- 
- 
- 
- 

### Lösung

Telefonbuch

Eine sortierte Liste von Nachnamen

Eine unsortierte Liste mit Lottozahlen

Eine nach Einfügedatum sortierte Liste von Notizen

### Aufgabe 56 (6-5, Comparable und Comparator)

Welche Aussagen treffen zu? Wählen Sie drei.

- [A] Comparable implementiert man in einer eigenen Klasse, wo *Comparator()* zum Einsatz kommt.
- [B] Das Interface *java.lang.Comparable* definiert als einzige Methode *int compareTo(T o)* und sorgt für eine Basissortierung (natürliche Reihenfolge).
- [C] Comparator implementiert man in einer eigenen Klasse, wo *CompareTo()* zum Einsatz kommt.

- [D] Wird ein Element bei einer binären Suche nicht gefunden, wird der Einfügeindex als positive ganze Zahl zurückgegeben.
- [E] Das Interface `java.util.Comparator` definiert die Methode `int compare(T o1, T o2)` und sorgt für eine Zusatzsortierung.

**Lösung**

B, C und E sind korrekt.

A ist falsch: Comparator, die Zusatzsortierung, implementiert man mit einem eigenen Objekt, nicht Comparable, die Basissortierung.

D ist falsch: Die Einfügestelle wird durch `-Einfüangepunkt-1` dargestellt.

**Aufgabe 57 (6-6, größer-Wert in einem Array finden)**

```
import java.util.*;
public class ArrayMax{
    public static void main(String[] args){
        int[] testarray=new int[]
            {10,19,33,1,3,5,7,8,99,22,55,77};
        Arrays. ;
        int endIndex=;
        System.out.println( []);
    }
}
```

Es soll das Maximum des Arrays `testarray` gefunden werden. Positionieren Sie die Elemente per Drag & Drop so, dass 99 ausgegeben wird.

- 
- 
- 
- 
- 
- 
- 

**Lösung**

`testarray.length`, `testarray.length()-1` und `1` sind falsch bzw. überflüssig. Die Sortierung erfolgt von 1 bis 99, also muss 99 an der Position `n-1` liegen.

```
import java.util.*;
public class ArrayMax{
    public static void main(String[] args){
```

```
int[] testarray=new int[]
    {10,19,33,1,3,5,7,8,99,22,55,77};
Arrays.sort(testarray);
int endindex=testarray.length-1;
System.out.println(testarray[endindex]);
}
}
```

### Aufgabe 58 (6-3, Generics)

```
class Generics<T>{
    T i;
    Generics(T i){
        this.i=i;
    }
    T getI(){
        return i;
    }
}
public class myGenerics{
    public static void main(String... args){
        Generics<Integer> gi=new Generics<Integer>(15);
        int i=gi.i;
        System.out.print(i);
    }
}
```

Gegeben ist obiges Programm. Was trifft zu? Wählen Sie eine Antwort.

- [A] Programm weist Kompilierfehler auf.
- [B] Programm kompiliert und läuft, gibt 15 aus.
- [C] Programm kompiliert und läuft, gibt 1 aus.
- [D] Programm kompiliert und wirft eine Ausnahme.

### Lösung

B ist korrekt. Das Programm kompiliert und läuft korrekt. Der Typparameter gibt den Typ `<Integer>` bei der Instanziierung ins Programm hinunter. Bei der Zeile `int i=gi.i;` greift das Auto-Boxing, d.h. die Zuweisung eines Hüllklassentyps an den korrespondierenden Primitivtyp.

**Aufgabe 59 (6-3, Refactoring)**

```
import java.util.*;
public class Refactoring{
public static void main(String [] args){
    ArrayList al=new ArrayList();
    al.add("erstens");
    al.add("zweitens");
    al.add("drittens");
    String s=al.get(0);
    System.out.println(s);
}
}
```

Gegeben ist obiges Programm. Was trifft zu? Wählen Sie eine Antwort.

- [A] Das Programm kompiliert einwandfrei, verursacht aber einen Laufzeitfehler.
- [B] Das Programm kompiliert und läuft einwandfrei.
- [C] Das Programm kompiliert wegen einer Typ-Inkompatibilität nicht.
- [D] Das Programm kompiliert mit einer Warnung, läuft aber.
- [E] Das Programm kompiliert nicht, weil mit `get()` auf das Element 0 zugegriffen wird, das nicht existiert.

**Lösung**

C ist richtig. Das Refactoring wurde nicht vollständig durchgeführt. Beim Auslesen wird ein Wert des Typs Object dem Typ String zugewiesen, was zur Typ-Inkompatibilität führt, falls ein Cast fehlt.

Die korrekte Lösung ist, den Typparametern `<String>` bei der Instanziierung der `ArrayList` zu verwenden oder den Typ `Object` in einen `String` zu casten.

```
import java.util.*;
public class Refactoring{
public static void main(String [] args){
    ArrayList<String> al=new ArrayList<String>();
    al.add("erstens");
    al.add("zweitens");
    al.add("drittens");
    String s=al.get(0);
    System.out.println(s);
}
}
```

### Aufgabe 60 (6-4, Welche Definitionen kompilieren?)

Welche kompilieren? Wählen Sie zwei.

- [A] `LinkedList<Double>d = new LinkedList<Integer>();`
- [B] `List<Number> n=new LinkedList<Number>();`
- [C] `List<Number> n=new <Number>LinkedList();`
- [D] `List<Number> n=new LinkedList();`
- [E] `List<Number> n=new LinkedList<Integer>();`
- [F] `List<int> I = new LinkedList<int>();`

#### Lösung

B, D sind korrekt.

B ist korrekt, weil List die Elternklasse von LinkedList ist.

D kompiliert mit einer Warnmeldung, aber es läuft. Man nennt Typen ohne Typparameter »raw types«.

A ist falsch, weil die Typparameter nicht auf beiden Seiten die gleichen sind.

C ist falsch, weil die Deklaration eine falsche Syntax hat (<Number> am falschen Ort).

E ist falsch, weil die Typparameter auf beiden Seiten nicht die gleichen sind.

F ist falsch, weil primitive Datentypen in Typparametern verboten sind.

### Aufgabe 61 (6-1, eine Map nutzen)

```
import java.util.*;
public class MapTest {
    public static void main(String args[]) {
        Map<[ ], [ ]> map = new HashMap<[ ], [ ]>();
        map.put(1, "Hallo");
        map.put(2, "du");
        map.put(3, "schoene");
        map.put(4, "Welt");
        for ([ ]<[ ], [ ]> eintrag : map.entrySet()){
            System.out.println("Map Eintrag: " + eintrag);
        }
    }
}
```

Positionieren Sie per *Drag & Drop* die richtigen Elemente in die Lücken des obigen Programms, sodass dieses funktioniert und Folgendes ausgibt:

```
Map Eintrag: 1=hallo
Map Eintrag: 2=du
Map Eintrag: 3=schoene
Map Eintrag: 4=Welt
```

### Lösung

```
import java.util.*;
public class MapTest {
    public static void main(String args[]) {
        Map<Integer, String> map = new HashMap<Integer, String>();
        map.put(1, "Hallo");
        map.put(2, "du");
        map.put(3, "Schoene");
        map.put(4, "Welt");
        for (Map.Entry<Integer, String> eintrag : map.entrySet()){
            System.out.println("Map Eintrag: " + eintrag);
        }
    }
}
```

### Aufgabe 62 (6-2, ein Set nutzen)

```
import java.util.*;
public class HashSetGenerics{
    public static void main(String[] args){
        String[] a=new String[]{"Hans", "Hans", "Fritz", "Karl", "Karl"};
        HashSet<String>hotelGast=new HashSet<String>();//#1 geändert
        for (int i=0;i<a.length;i++){
            if (!hotelGast.add(a[i]))
                System.out.print("Doppel ");
        }
        Iterator<String>it =hotelGast.iterator();//#2 geändert
        while (it.hasNext()) {
            String element = it.next();//#3 Auslesen direkt als String
            System.out.print(element + " ");
        }
    }
}
```

Wie lautet die Ausgabe des obigen Programms in garantierter Reihenfolge?

- [A] Doppel Doppel Karl Fritz Hans
- [B] Karl Fritz Hans Doppel Doppel
- [C] Doppel Doppel Hans Fritz Karl
- [D] Doppel Doppel Fritz Karl Hans
- [E] Hans Fritz Karl
- [F] Keines der obigen

### Lösung

F ist korrekt. Die genaue Reihenfolge beim Iterieren durch ein HashSet nicht garantieren.

## 10.2.7 Prüfungsziel 7

### Aufgabe 63 (7-4, Garbage-Collection)

Welche drei Feststellungen zur Garbage-Collection treffen zu?

- [A] Sobald eine Referenz eines Objekts = 0 ist, wird es augenblicklich entsorgt.
- [B] Sobald alle Referenzen auf ein Objekt null sind, ist das Objekt reif für den Garbage Collector.
- [C] Die *finalize()*-Methode kann nicht überschrieben werden.
- [D] Die Implementation der Garbage-Collection ist von der JVM abhängig.
- [E] Die *finalize()*-Methode wird pro Objekt nur einmal aufgerufen.

### Lösung

B, D und E sind korrekt.

A ist falsch, weil eine Referenz nicht = 0 ist, sondern = null. Die Entsorgung findet nicht zwingendermaßen sofort statt.

C ist falsch, die *finalize()*-Methode kann überschrieben werden.

### Aufgabe 64 (7-5, jar-Import)

Folgende jar-Datei ist gegeben, sie befindet sich im Ordner `c:\javaprogs\com`.

```
META-INF/  
META-INF/MANIFEST.MF  
projekt1/  
projekt1/classes/
```

```
projekt1/classes/library/  
projekt1/classes/library/Lib.class  
projekt1/classes/Aufruf.class
```

Der Klassenpfad lautet

```
c:\javaprogs\com\
```

Wie lauten die Package- und Import-Statements der Dateien Aufruf.class und Lib.class?

- [A] Package-Statement Aufruf.class: keines, Lib.class: library
- [B] Package-Statement Aufruf.class: classes  
Lib.class classes.library
- [C] Package-Statement Aufruf.class: projekt1.classes  
Lib.class: projekt1.classes.library
- [D] Import-Statement in Aufruf.class: import library.lib.class;
- [E] Import-Statement in Aufruf.class: import classes.library.lib.class;

### Lösung

C ist korrekt. Lib.class liegt in:

```
c:\javaprogs\com\projekt1\classes\library\Lib.class
```

Aufruf.class liegt in:

```
c:\javaprogs\com\projekt1\classes\Aufruf.class
```

wobei der fett gedruckte Teilpfad der Klassenpfad ist. Klassenpfad plus Package-Statement oder Klassenpfad und Import-Statement und müssen sich zum absoluten Pfad zusammensetzen. Ein jar-File benötigt noch einen eigenen Klassenpfad inklusive des Dateinamens .jar.

### Aufgabe 65 (7-6, Operanden)

Welche der folgenden Operatoren werten immer alle Operanden aus?

- [A] &&
- [B] &
- [C] ||
- [D] |
- [E] ?:

### Lösung

B und D sind richtig, alle anderen Operatoren sind »Kurzschlussoperatoren«.

### Aufgabe 66 (3-4, formatierte Zahlenausgabe)

```
public class PrintTest{
    public static void main(String... args){
        double d=47114711.47611;
        Locale lde=new Locale("de","DE");
        System.out.printf(lde,"%,.2f",d);
    }
}
```

Wie lautet der Output dieses Programms? Wählen Sie eine Antwort.

- [A] 47,114,711.480
- [B] 47 114 711.47
- [C] 47.114.711,48
- [D] 47114711.48
- [E] 47114711.47

### Lösung

C ist richtig, das .476 wird zu .48 aufgerundet. Das Tausender-Trennzeichen ist ein Punkt. Zur Erinnerung: Jeder Format-String beginnt mit dem Prozentzeichen. Das Tausender-Trennzeichen wird mit deutscher Formatierung mit einem Punkt angegeben.

Würde man die Locale-Instanz nicht der Methode printf() übergeben, würde die Zahl in der Formatierung der Default-Locale ausgegeben werden. Die Methode format() hat übrigens die gleiche Wirkung wie printf().

### Aufgabe 67 (7-6, Logische Operatoren)

```
01 public class Logo{
02     public static void main(String... args){
03         boolean b=false;
04         if (b=true)
05             System.out.print("Hallo ");
06         else
07             System.out.println("Tschuess ");
08         if (false ^ true)
09             System.out.print("Welt ");
```

```
10 else
11     System.out.print("Mond ");
12 }
13 }
```

Wie lautet der Output?

- [A] Hallo Welt
- [B] Welt Tschuess
- [C] Tschuess Mond
- [D] Laufzeitfehler
- [E] Kompilierfehler

### Lösung

A ist richtig.

In Zeile 4 wird der Variable `b` `true` zugewiesen, damit ist die Bedingung `true`. `^` ist der XOR-Operator. Er wertet sich zu `true` aus, falls beide Bedingungen unterschiedlich, also `true` und `false`, sind.

### Aufgabe 68 (7-2, Kommandozeilenparameter)

```
class Kommandozeile{
    public static void main(String... args){
        for (Object o:args)
            System.out.print(o);
        String[] s=new String[5];
        s=args;
        for (String o:args)
            System.out.print(o);
        }
    }
```

Was wird ausgegeben, falls das Programm mit `java Kommandozeile 1 2` gestartet wird?

- [A] 1122
- [B] 1212
- [C] Kompilierfehler wegen Typen-Inkompatibilität
- [D] Laufzeitfehler

### Lösung

B ist korrekt. Es wird 1212 ausgegeben. Es ereignet sich keine Typ-Inkompatibilität, weil String an ein Object zugewiesen werden darf und bei der Array-Zuweisung ein eindimensionales Array an ein eindimensionales Array zugewiesen wird.

### Aufgabe 69 (7-6, String Concatenation Operator)

Welche der folgenden Ausdrücke kompilieren? Wählen Sie zwei Antworten.

```
int i=1; String s="test";
```

- [A] `s+=i;`
- [B] `i+=s;`
- [C] `s=s+i;`
- [D] `i=i+s;`
- [E] `if(i==s);`

### Lösung

A und C sind richtig.

Im Falle eines Strings ist das `»+«` der String-Verkettungsoperator (*String Concatenation Operator*). Damit er ins Spiel kommt, muss der erste Ausdruck ein String sein.

Vorsicht: In der Methode `System.out.println()` findet immer eine String-Konkatenierung statt, falls eines der Elemente ein String ist. Es kommt nicht auf die Position des Strings an.

```
class Ausdruecke{
    public static void main(String... args){
        String s="test";
        int i=1;
        s+=i;
        System.out.println(s);
        //i+=s; //incompatible types, found int, required String
        s=s+i;
        System.out.println(s);
        //i=i+s; //incompatible types, found int, required String
        //if(i==s);
    }
}
```

**Aufgabe 70 (7-1, Package- und Importanweisungen)**

```
//Datei Vater.class
package u;
public class Vater{
    protected int vaterVariable=10;
    public void vaterMethode(){
        System.out.println("vaterMethode()");
    }
}
//Datei Kind.class
01 import u.*;
02 public class Kind extends Vater{
03     public static void main(String... args){
04         Vater v = new Vater();
05         Kind k = new Kind();
06         v.vaterMethode();
07         System.out.println(v.vaterVariable);
08     }
09 }
```

Was passiert mit obigem Code? Wählen Sie eine Antwort.

- [A] Kompilierfehler bei Zeile 06
- [B] Kompilierfehler bei Zeile 07
- [C] Kompiliert einwandfrei und läuft.
- [D] Laufzeitfehler

**Lösung**

B ist korrekt. Aus der Aufgabenstellung folgt, dass Vater.class in einem Unterordner u liegen muss, also z.B. `c:\javaprogs\u\Vater.class` und `c:\javaprogs\Kind.class`. Sind die Dateien so angeordnet, wird Vater.class gefunden und korrekt importiert.

Das Problem liegt beim Zugriff auf die Variable `vaterVariable`. Sie hat den Modifizierer `protected` und ist so nur über den Vererbungsweg sichtbar. Würde man anstatt `v.vaterVariable` `k.vaterVariable` setzen, würde das Programm funktionieren.

Also: Für eine Unterklasse außerhalb des Packages kann auf ein `protected` Member nur über den Vererbungsweg zugegriffen werden. Der Zugriff über die Referenzvariable der Elternklasse funktioniert nicht.

**Aufgabe 71 (7-4, java, Garbage-Collection)**

```
class Mue11{
    Mue11 m;
}

Mue11 m0=new Mue11();
Mue11 m1=new Mue11();
Mue11 m2=new Mue11();
m0.m=m1;
m1.m=m2;
m2.m=m0;
m0=new Mue11Halde1();
m0=m1=m2;
}
}
```

Wieviele Objekte sind vor und nach dem Aufruf von `m0=m1=m2` referenziert?  
Wählen Sie eine Lösung.

- [A] 2, 4
- [B] 3, 4
- [C] 4, 3
- [D] 3, 2
- [E] 2, 3

**Lösung**

C ist korrekt. Es handelt sich um eine sogenannte Referenz-Insel, so lange eine der Referenzen `m0`, `m1` oder `m2` noch original sind, sind die drei untereinander verknüpften Objekte referenziert. Durch die Zuweisung von `m0=m1=m2` wird nur die Referenz auf die Insel zerstört. Danach ist nur noch das am Schluss erzeugte Objekt referenziert.

```
class Mue11{
    Mue11 m;
}

Mue11 m0=new Mue11 ();
Mue11 m1=new Mue11 ();
Mue11 m2=new Mue11 (); //3 Objekte Obrejt-m0, Objekt-m1, Obejekt-m2
m0.m=m1; //Verknüpfung der Objekte untereinander
m1.m=m2;
m2.m=m0;
m0=new Mue11 (); //zusätzliches Objekt Objekt-m0-neu, 4 Objekte referenziert
```

```
m0=m1=m2; //alle Referenzen zeigen auf Objekt-m0-neu, 1 reif für GC,
//3 durch die Referenzinsel noch referenziert
```

Mithilfe einer Klassenvariable *zaehler* lassen sich die Instanzen durch den Konstruktor und die Methode *finalize()* zählen. Allerdings wirkt *System.gc()* nur als Empfehlung an den Garbage Collector, es besteht keine Garantie, dass der Garbage Collector sofort läuft. Um die Verhältnisse nachzuprogrammieren musste *m0=m1=m2* durch *m1=null*; und *m2=null* ersetzt werden, da sonst *System.gc()* nicht ablief. Zudem musste eine Verzögerung eingebaut werden, da sonst die Ausgabe zu früh ausgegeben wurde.

```
class Mue11{
    static int zaehler=0;
    Mue11 m; //Membervariable vom Typ des eigenen Objekts
    public Mue11(){
        zaehler++;
        System.out.println("Objekt erstellt: "+zaehler);
    }
    public void finalize(){
        zaehler--;
        System.out.println("Objekt zerstört: "+zaehler);
    }
} //Ende Klasse Mue11
public class Mue11Halde{
    public static void main(String[] args){
        Mue11 m0=new Mue11();
        Mue11 m1=new Mue11();
        Mue11 m2=new Mue11();
        //Referenzinsel
        m0.m=m1;
        m1.m=m2;
        m2.m=m0;
        m0=new Mue11(); //4 Objekte referenziert
        m0=m1=m2; //Referenzinsel ist immer noch referenziert
        m1=null;//müssen für GC explizit gnullt werden
        m2=null;
        System.out.println("Vor GC: "+Mue11.zaehler);
        System.gc();//1 Objekt wird zerstört
        //Verzögerung 1 Sekunde
        try{Thread.sleep(1000);}catch(InterruptedException e){}
        System.out.println("Nach GC: "+Mue11.zaehler);
        m0=null;
        System.gc();//Referenzinsel wird zerstört
        //Verzögerung 1 Sekunde
        try{Thread.sleep(1000);}catch(InterruptedException e){}
        System.out.println("nach m0=null: "+Mue11.zaehler);
```

```
}  
}  
Output:  
Objekt erstellt: 1  
Objekt erstellt: 2  
Objekt erstellt: 3  
Objekt erstellt: 4  
Vor GC: 4  
Objekt zerstört: 3  
Nach GC: 3  
Objekt zerstört: 2  
Objekt zerstört: 1  
Objekt zerstört: 0  
nach m0=null: 0
```

### Aufgabe 72 (7-3, Operatoren)

```
public class Salat{  
    static boolean hello(){  
        System.out.print("Hello ");  
        return true;  
    }  
    public static void main(String... args){  
        boolean a=false;  
        boolean b=true;  
        int i=0;  
        if (a || b){  
            System.out.print("Hallo ");  
        }  
        if (a && hello()){  
            System.out.print("du ");  
        }  
        if (a & hello()){  
            System.out.print("Welt ");  
        }  
    }  
}
```

Welches ist die Ausgabe des obigen Programms?

- [A] Hallo Hello
- [B] Hallo Hello du Welt
- [C] Hallo Hello du
- [D] Hallo Hello Hallo Hello
- [E] Programm weist Kompilierfehler auf

### Lösung

A ist korrekt. Der Operator `||` wird *true*, falls eine der beiden Bedingungen *true* ist. Der Operator `&&` führt die zweite Bedingung nicht mehr aus, falls die erste *false* ist, der Operator `&` hingegen schon (`&&` und `||` sind sogenannte Kurzschlussoperatoren).

## 10.3 Aufgabenstellungen zu Simulationstest 2

In der folgenden Simulation ist die Fragestellung als auch der Code in Englisch gehalten. Nur die Fragestellung hat eine deutsche Übersetzung. Bei der realen deutschen Prüfung ist ebenfalls nur die Fragestellung ins Deutsche übersetzt.

### Aufgabe 1

Given the following source:

```
package aufgabe1;

public class House {
    public class Room{void lamp(){System.out.println("lamp");}
        public class Furniture extends Room{void chair(){System.out.println("chair");}};
    };

    public static void main(String[] args) {
        House h=new House();
        House.Room r=h.new Room();
        House.Room.Furniture f=r.new Furniture();
        h.chair();
        r.chair();
        r.lamp();
        f.lamp();
        f.chair();
    }
}
```

Which invocations of methods cause the program to not compile? Choose one answer.

(Welche Methodenaufrufe bewirken, dass das Programm nicht kompiliert werden kann? Wählen Sie eine Antwort)

- [A] *h.chair()* alone
- [B] *r.chair()* alone

- [C] *only r.lamp()*
- [D] *r.lamp() and f.lamp()*
- [E] *r.lamp() and f.lamp() and f.chair()*
- [F] *h.chair() and r.chair()*

## Aufgabe 2

Given a library class and eight different test classes:

(Gegeben sind eine Bibliotheksklasse und acht Testklassen.)

Which one cannot be compiled? Choose one answer.

(Welche kann nicht kompiliert werden?)

```
package ch.geo.scjp;
public class MyLib{
    public static int i=1;
    public int j=2;
}
And
a)
class Test{
public static void main(String[] args) {
    System.out.println(ch.geo.scjp.MyLib.i);
}
}
b)
import ch.geo.scjp.*;
class Test{
public static void main(String[] args) {
    System.out.println(MyLib.i);
}
}
c)
import ch.geo.scjp.MyLib;
class Test{
public static void main(String[] args) {
    System.out.println(MyLib.i);
}
}
d)
import static ch.geo.scjp.MyLib.*;
class Test{
```

```
public static void main(String[] args) {
    System.out.println(i);
}
}
e)
import ch.geo.scjp.MyLib;
class Test{
public static void main(String[] args) {
    MyLib ml=new MyLib();
    System.out.println(ml.i);
}
}
f)
import ch.geo.scjp.MyLib.*;
class Test{
public static void main(String[] args) {
    MyLib ml=new MyLib();
    System.out.println(ml.j);
}
}
g)
class Test{
public static void main(String[] args) {
    ch.geo.scjp.MyLib ml=new ch.geo.scjp.MyLib();
    System.out.println(ml.j);
}
}
h)
import ch.geo.scjp.MyLib;
class Test{
public static void main(String[] args) {
    MyLib ml=new MyLib();
    System.out.println(ml.j);
}
}
}
```

- [A] a
- [B] b
- [C] c
- [D] d
- [E] e

- [F] f
- [G] g
- [H] h

### Aufgabe 3

Given following code:

```
1 int[] a={0,1,2,3,4};
2 Object o=a;
3 int[] myArray = (int[]) o;
4 for(int i:myArray)
5 System.out.print(i);
```

What is the result? Choose one answer.

(Was ist das Resultat? Wählen Sie eine Antwort.)

- [A] 01234
- [B] ClassCastException at line 3
- [C] RuntimeException due to an error at line 2
- [D] Does not compile.

### Aufgabe 4

Given:

```
public class EnumTest {
enum Auto{ //Deklaration innerhalb der aufrufenden Klasse
Ford(20000),Opel(22000),VW(25000), Mercedes(30000); //#1
private int price;
Auto (int price){
    this.price=price;
};
int getPrice(){
    return price;
}
}; //#2

public static void main(String[] args){
    Auto a;
    System.out.println(Auto.Mercedes.getPrice());
    System.out.println(Auto.Ford.getPrice()); //#3
}
}
```

What happens with this code? Choose one.

(Was passiert mit diesem Code? Wählen Sie eine Antwort.)

- [A] Does not compile, reason: »;« at #1 should be replaced by a »;,«.
- [B] Does not compile, reason: »;« at #1 should be deleted.
- [C] Compiles and runs, Output: 30000 20000
- [D] Runtime error
- [E] Runtime error at #3, getPrice() is only defined for Mercedes.

### Aufgabe 5

```
public class VarArgTest{

    public void method(int[] i) {
        for (int j:i)
            System.out.print(j+" ");
        System.out.println("Array int");
    }

    public void method(int... i) {
        for (int j:i)
            System.out.print(j+" ");
        System.out.println("Vararg int");
    }

    public void method(Integer... i){
        for (Integer j:i)
            System.out.print(j+" ");
        System.out.println("Vararg Integer");
    }

    public static void main(String... args){
        VarArgTest vat=new VarArgTest();
        vat.method(new int[] {1,2,3} );

        Integer[] j={5,6,7};
        vat.method(j);
    }
}
```

What is the result of the above code. Choose one.

(Was ist das Resultat des obigen codes, wählen Sie eine Antwort.)

- [A] Compilation error
- [B] Runtime error
- [C] Outputs  
1 2 3 Vararg int  
5 6 7 Vararg Integer
- [D] Outputs  
1 2 3 Array int  
5 6 7 Vararg Integer

### Aufgabe 6

Given:

```
1 interface Eatable {boolean eat(String thing);}
2 abstract class Apple implements Eatable {public boolean eat(String thing);}
3 abstract class Pear implements Eatable {boolean eat(){return true;}}
4 public class GoldenDelicious extends Apple implements Eatable{boolean
eat(){return true;}}
```

Which one statement is true?

(Welche Aussage trifft zu?)

- [A] Compilation succeeds
- [B] Compilation fails due to only an error in line 1.
- [C] Compilation fails due to only an error in line 2.
- [D] Compilation fails due to only an error in line 3.
- [E] Compilation fails due to only an error in line 4.

### Aufgabe 7

Given:

```
interface Iface{ String doit();}
class A implements Iface{public String doit(){return "Hello";}}
class B extends A {void doit(String s){System.out.println("Bye");}}
public class C extends B{public String doit(){System.out.println("Hey");return
"";}
public static void main(String [] args){
A a=new C();
```

```
a.doit();
}
}
```

What happens?

(Was passiert?)

- [A] Code compiles and runs, Output »Hey«
- [B] Code compiles and runs, Output »Hello«
- [C] Compiler error in Class A
- [D] Compiler error in Class B
- [E] Compiler error in Class C
- [F] Runtime error

### Aufgabe 8

Given:

```
class Foo{public static Foo createFoo(){
    return new Foo();
};
    public void hello(){System.out.print("Hello ");
}
}
public class Bar extends Foo{
    public static Bar createFoo(){return new Bar();};
    public void hello(){System.out.print("Hy "); }
    public static void main(String[] args){
        Foo f=createFoo();
        Bar b=createFoo();
        f.hello();
        b.hello();
    }
}
```

Which statement is true? Select two.

(Welche Aussagen treffen zu? Wählen Sie zwei.)

- [A] Code does not compile.
- [B] Code causes runtime error.
- [C] Code compiles and runs, outputs »Hy Hy«

- [D] CreateFoo in Bar has a covariant return when overriding.
- [E] createFoo() is not overrrden in this example.
- [F] Code compiles and runs, outputs »Hello Hy«

### Aufgabe 9

Given:

```
class Parent{
    private int age=0;
    public Parent(int age){//#1
        this.age=age;
    }
    public int getAge(){
        return this.age;
    }
}
public class Child extends Parent{
    public Child(){//#2
    }
    public static void main(String[] args){
        Child c=new Child();
        Parent p=new Parent();
        System.out.println(p.getAge());
    }
}
```

Which statements are true? Choose two answers:

(Welche Aussagen treffen zu? Wählen Sie zwei Antworten:)

- [A] Code compiles and runs.
- [B] There is a lacking default constructor in class Parent
- [C] There is a lacking default constructor in class Child
- [D] There is a runtime error.

### Aufgabe 10

Given:

```
class Temperature{
    private int t=2;
    {t+=t;}
    public Temperature(int t){
        this.t=t;
    }
}
```

```
}  
//insert here  
public static void main(String[] args){  
    Temperature myT0=new Temperature();  
    System.out.println(myT0.t);  
    Temperature myT1=new Temperature(20);  
    System.out.println(myT1.t);  
}  
}
```

What has to be inserted at `//insert here` so that the code compiles and runs and outputs 4 and 20? Choose one answer.

(Was muss bei `//insert here` eingegeben werden, dass der Code kompiliert und 4 und 20 ausgibt? Wählen Sie eine einzige Antwort.)

- [A] nothing
- [B] `t=t+2;`
- [C] `public Temperature(){}`
- [D] `public Temperature(){super(5)}`

### Aufgabe 11

```
public class IfChaos{  
    public static void main(String [] args){  
        int a=1,b=2;  
        if (a>b)  
            System.out.println("first");  
        else if (a+2>b)  
            System.out.println("second");  
        else if(true)  
            System.out.println("third");  
        else  
            System.out.println("fourth");  
    }  
}
```

What is true about the code? Choose one answer.

(Was trifft für den Code zu? Wählen Sie eine Antwort.)

- [A] Outputs *first*
- [B] Outputs *second*
- [C] Outputs *third*

- [D] Outputs *fourth*
- [E] Compilation fails

### Aufgabe 12

Given:

```
import java.util.*;
public class IteratorTest{
    public static void main(String [] args){
        int myArray[] ={0,1,2,3,4,5,6,7,8,9};
        List ar=Arrays.asList(myArray);
        Collections.reverse(ar);
        Iterator it=ar.iterator();
        while (it.hasNext()){
            System.out.print(it.next());
        }
    }
}
```

What ist true about the above program? Choose one.

(Was trifft für obiges Programm zu? Wählen Sie eine Antwort)

- [A] Some output like [I@42e816
- [B] 3
- [C] 0123456789
- [D] Compilation error
- [E] Runtime error
- [F] 9876543210

### Aufgabe 13

Given:

```
public class ExceptionTest{
    public static void main(String [] args){
        Exception e1=new Exception();
        ArithmeticException ae=new ArithmeticException();
        try{
            throw e1;
        }
        catch (Exception ex){
```

```
System.out.println("Exception caught");
try {
    throw ae;
}
catch (ArithmeticException aex){
    System.out.println("ArithmeticException caught");
}
}
}
```

Which statements are true? Choose two answers.

(Welche Aussagen treffen zu? Wählen Sie zwei Antworten.)

- [A] Code compiles and runs.
- [B] Compilation error
- [C] RuntimeException
- [D] Code generates the following compiler error message »ArithmeticException has already been caught«
- [E] Runs and writes »Exception caught« and »ArithmeticException caught«

#### Aufgabe 14

Given:

```
class ShortCut{
    public static void main(String... args) {
        byte x=2,y=4;
        if ((x++==2) && (y-- ==4)) x++;
        if ((x--==1) || (y++ ==3)) x++;
        System.out.println(x+" "+y);
    }
}
```

What is the output (select one)?:

(Wie lautet der Output (wählen Sie eine Antwort)?)

- [A] 4 3
- [B] 4 4
- [C] 3 4
- [D] 2 4
- [E] 2 3

### Aufgabe 15

Given:

```
class BreakTest {
    public static void main(String[] args){
        int [] myArray={1,2,3,4,5,6,7,8,9};
        for (int i=0;i<3;i++){
            myLabel:
            for(int j=0;j<myArray.length;j++){
                if (myArray[j] % 2==0) break myLabel;
                System.out.println(myArray[j]);
            }
        }
    }
}
```

What is the output (select one)?:

(Wie lautet der Output (wählen Sie eine Antwort)?)

- [A] I
- [B] II
- [C] III
- [D] no output
- [E] 1234

### Aufgabe 16

Given:

```
class AssertTest{
    static int value=10;

    public static void main(String [] args){
        AssertTest at=new AssertTest();
        at.myMethod(value);
    }
    private boolean myMethod(int x){
        assert(x<--value):"uups";
        return false;
    }
}
```

What is the output if the program is started by the command-line invocation `java -ea AssertTest`? Select one Answer.

(Wie lautet ist der Output, falls das Programm über folgende Kommandozeile gestartet wird: `java -ea AssertTest` ? Wählen Sie eine Antwort)

- [A] no output, no error
- [B] compilation error
- [C] an assertion error is thrown
- [D] assertions are not activated, no output

**Aufgabe 17**

Given:

Exception	Category runtime/ error/ checked	Triggered by	
		JVM	Program
ArrayIndexOutOfBoundsException			
ClassCastException			
IllegalArgumentException			
IllegalStateException			
NullPointerException			
NumberFormatException			
AssertionError			
ExceptionInInitializerError			
StackOverflowError			
NoClassDefFoundError			
IOException			
EOFException			
FileNotFoundException			

Use drag & drop to fill in the table. The meaning of the category is:  
runtime = unchecked runtime exceptions  
checked = checked exceptions  
error = errors  
x = triggered by JVM or Program

(Verwenden Sie *Drag & Drop*, um die Tabelle zu ergänzen. Die Bedeutung der Kategorie ist:

runtime = ungecheckte Laufzeitfehler  
checked = gecheckte Fehler  
error = Fehler  
x= ausgelöst durch JVM oder Programm)

### Aufgabe 18

Given:

```
class MyError extends Error{
    double pressure;
    MyError(double p){
        pressure=p;
    }
    public String toString(){
        return "MyError: Pressure is "+pressure+", caution";
    }
}
public class OwnExceptionTest {
    static void treshold(double p) throws MyError{
        if (p>20) throw new MyError(p);
        System.out.println("Normal state, pressure is below 20 bars");
    }
    public static void main(String[] args) {
        try{
            treshold(10);
            treshold(30);
        }
        catch (MyError e){
            System.out.println("Error "+e);
        }
    }
}
```

Which statements are true (choose two)?

(Welche Aussagen treffen zu? Wählen Sie zwei.)

- [A] The program does not compile because of the use of Error instead of Exception.
- [B] The program compiles and throws one error.
- [C] The program compiles and throws two exceptions.
- [D] The output is

```
Normal state, pressure is below 20 bars
Error MyError: Pressure is 30.0, caution
```

- [E] The output is

```
Normal state, pressure is below 20 bars
Normal state, pressure is below 30 bars
```

### Aufgabe 19

Given:

```
import java.io.Console;
public class TryCatchTest{
public static void divisionDurchNull() throws ArithmeticException{
    int i=10;
    int k=0;
    i=i/k;
}
public static void main(String[] args) {
    Console c=System.console();
    try{
        c.printf("vor Exception\n");
        divisionDurchNull();
        c.printf("nach Exception\n");
    }
    catch (ArithmeticException ae){
        c.printf(ae.getMessage()+"\n");
    }
    finally {c.printf("finally\n");
    }
    c.printf("Ende\n");
}
}
```

What is the result? Choose one.

(Welches ist das Resultat? Wählen Sie eine Antwort.)

[A]

```
vor Exception
nach Exception
/ by zero
finally
Ende
```

[B]

```
vor Exception
/ by zero
finally
Ende
```

[C]

```
vor Exception
/ by zero
nach Exception
finally
Ende
```

[D]

```
vor Exception
/ by zero
finally
```

## Aufgabe 20

Given:

```
import java.io.Console;
public class TryCatchTest{
public static void divisionDurchNull() throws ArithmeticException{
int i=10;
int k=0;
i=i/k;
}

public static void main(String[] args) {
Console c=System.console();
try{
c.printf("1 ");
```

```

        divisionDurchNull();
        c.printf("2 ");
    }
    catch (ArithmeticException ae){
        c.printf("3 ");
        try{
            throw new Exception();
            //c.printf("4 ");//#1
        }catch (Exception e){
            c.printf("5 ");
        }finally{
            c.printf("6 ");
        }
    }
    finally {c.printf("7 ");
    }
    c.printf("8 ");
}
}

```

Which statements are true (choose two)?

(Welche Aussagen treffen zu? Wählen Sie zwei.)

- [A] Output is 1 3 5 6 7 8
- [B] Output is 1 2 3 5 6 7 8
- [C] Output is 1 2 3 4 5 6 7
- [D] if `c.printf("4 ");` at `#1` would be uncommented, the compilation error »unreachable statement« occurs.
- [E] if `c.printf("4 ");` at `#1` would be uncommented there would be no influence on the program.

### Aufgabe 21

Given:

```

import java.io.Console;
public class HoneyTrap{
    public static void main(String[] args){
        Console c=System.console();
        short x=1;
        byte y=2;
        byte sum=0;
        do
            ++x;

```

```
while(false);
sum=x--+y;
c.printf("%7d",sum);
}
}
```

What is the result? Choose one Answer.

(Wie lautet das Resultat? Wählen Sie eine Antwort)

- [A] Output is 3
- [B] Output is 4
- [C] Compilation error
- [D] Runtime error

### Aufgabe 22

Given:

```
import java.io.*;
public class ParseIntValueOf{
    public static void main(String[] args){
        Boolean b=true;
        Console c=System.console();
        if(Boolean.valueOf("True"))
            c.printf("only work ");
        if(b.booleanValue())
            c.printf("and no play ");
        if(b.parseBoolean("true"));
            c.printf("makes Jack ");
        if (Boolean.valueOf(b.toString()))
            c.printf("a dull boy");
        }
    }
}
```

Which statements are true? Choose four correct answers.

(Welche Aussagen treffen zu? Wählen Sie vier Antworten, die korrekt sind.)

- [A] Result: only work and no play makes Jack a dull boy
- [B] Result: and no play makes Jack a dull boy
- [C] valueOf() converts String to boolean
- [D] valueOf() converts String to Boolean

- [E] booleanValue() converts Boolean to Boolean
- [F] booleanValue() converts Boolean to boolean
- [G] parseBoolean() converts String to Boolean
- [H] parseBoolean() converts String to boolean

### Aufgabe 22a

```
import [ ] ;
public class Navigation{
    public static void main(String[] args){
        File directory= [ ] ;
        [ ] filter=new [ ] (){
            public boolean [ ] {
                return name.startsWith("N");
            }
        };
        [ ] array= directory.[ ] ;
        for(File f:array)
            if (f. [ ] )
                System.out.println(f.getName());
    }
}
```

Drag & drop the elements into the slots.

(Ziehen Sie die Elemente in die Lücken.)

- 
- 
- 
- 
- 
- 
- 

### Aufgabe 23

What are the characteristics of the class java.io.File? Choose four correct answers.

Assume f is a valid reference to a File object.

(Was sind die Eigenschaften der Klasse java.io.File? Wählen Sie vier korrekte Antworten. Nehmen Sie an, f sei eine gültige Referenz auf ein File-Objekt.)

- [A] f.mkdir() creates a directory with the name given in f
- [B] f.mkdir() returns false if the directory already exists

- [C] `f.mkdir("\\java")` creates the folder `\java`
- [D] `f.mkdir("\\\\java")` creates the folder `\java`
- [E] `f.delete()` deletes a directory only if it is empty
- [F] `f.createNewFile("datei.txt");` is correct
- [G] The slashes in directories in Windows notation have to be escaped, i.e. `>>\\<<`, Unix slashes `>/<` may be used in Windows, too.

### Aufgabe 24

Given:

```
1 public class StringBufferTest{
2 public static void main(String[] args){
3 StringBuffer sb=new StringBuffer("123456789");
4 sb.insert(5,"abcdefghik");
5 sb.delete(7,10);
6 String sb1=sb.substring(3,9);
7 System.out.println(sb1.indexOf("f"));
8 }
9}
```

What is the output?

(Wie lautet der Output?)

- [A] 3
- [B] 4
- [C] 5
- [D] Compiler error in line 6

### Aufgabe 25

Given:

```
import java.io.Console;
public class Passwort{
    public static void main(String[] args) {
        Console c=System.console();
        char[] password=c.readPassword("%s","Password: ");
        for (char ch:password)
            c.printf("%c",ch);
    }
}
```

What is prompted on the screen while typing in the password? Assume the password to enter is »test«. Choose one.

(Was wird am Bildschirm ausgegeben, während man das Passwort eingibt? Nehmen Sie an, das Passwort hieße »test«. Wählen Sie eine richtige Antwort.)

- [A] Password: \*\*\*\*
- [B] Password:
- [C] Password: test
- [D] Runtime error

### Aufgabe 26

Given:

```
import  ;
public class MinimalRegex{
    public static void main(String[] args){
        Pattern p = Pattern. ;
        Matcher m = p. ;
        System.out.println("The Pattern: "+ m. );
        while (m.find()){
            System.out.println("Start: "+  +" found: "+  );
        }
    }
}
```

Position the elements so that the program compiles and runs.

(Positionieren Sie die Elemente so, dass das Programm kompiliert und läuft.)

m.start()  
 compile("-")  
 java.util.regex.\*  
 java.util.\*  
 pattern()  
 matcher("Hello-nice-world")  
 m.group()

### Aufgabe 27

Given:

```
import java.util.*;
public class FormatterTest{
```

```
public static void main(String[] args){
    Formatter fo=new Formatter(Locale.US);
    Double x=new Double(123.456789);
    fo.format("%.2f",x);
    double y=Float.parseFloat(fo.toString());
    if (y > 123.45)
        System.out.println(fo);
    else
        System.out.println("smaller than 123.45");
    StringBuffer sb=new StringBuffer();
    Formatter fo1=new Formatter(sb,Locale.GERMANY);
    fo1.format("%.3f",124.5678910);
    System.out.println(fo1);
    StringBuffer sb1=(StringBuffer)fo1.out();
    System.out.println(sb1);
}
}
```

What is the output? Choose one:

(Wie lautet der Output? Wählen Sie eine Antwort.)

[A]

```
123.44
Smaller than 123.45
124,568
```

[B]

```
123.46
124,568
124,568
```

[C]

```
123.46
124.568
124.568
```

[D] Laufzeitfehler

### Aufgabe 28

Which statements are true?

(Welche Aussagen treffen zu?)

- [A] The constructors of the serialized objects do not run when they are deserialized. But the parent constructor does, in case the parent object was not serialized.
- [B] A class to be serialized must implement Serializable. If the parent class implements Serializable, the child class does not need to implement Serializable to be serialized.
- [C] If you mark a class's instance variable transient, it will persist serialization.
- [D] If you mark a class's instance variable volatile, it will be lost during serialization.
- [E] If you mark a class's instance variable static, it will persist serialization.
- [F] If you serialize an object which contains another object as member, the member object has to implement Serializable, too.

### Aufgabe 29

Given:

```
import java.util.*;
import java.text.*;
public class FormatterTest{
    public static void main(String[] args){
        Date d=new Date();
        Calendar cal=Calendar.getInstance();
        cal.setTime(d);

        Formatter fo=new Formatter(Locale.GERMANY);
        fo.format("%td. %tB %tY",cal,cal,cal);
        System.out.println(fo);

        DateFormat df = DateFormat.getDateInstance(DateFormat.LONG, Locale.GERMANY);
        System.out.println(df.format(d));
    }
}
```

Which statements are correct if today is September 23, 2008?

(Welche Aussagen sind korrekt falls das heutige Datum der 23. September 2008 ist?)

- [A] Formatter is an older class than DateFormat.
- [B] The output is:

```
23. September 2008
23. September 2008
```

- [C] The output is:

```
23. 9.2008
23. 9.2008
```

- [D] Formatter is more universal than DateFormat, as it can format currencies too.

### Aufgabe 30

```
import java.util.*;
import java.text.*;
public class FormatterTest{
    public static void main(String[] args){
        double d=1000.50;
        Formatter fo=new Formatter(Locale.US);
        fo.format("$%, .2f",d);
        System.out.println(fo);
        NumberFormat nf = NumberFormat.getCurrencyInstance(Locale.US);
        System.out.println(nf.format(d));
    }
}
```

Which statements are correct?

(Welche Aussagen sind korrekt?)

- [A] Formatter existed before NumberFormat.
- [B] The output is:

```
$1000.50
$1000.50
```

- [C] The output is:

```
$1,000.50
$1,000.50
```

- [D] Formatter is more universal than NumberFormat, as it can format dates, too.

### Aufgabe 31 (3-5, Reguläre Ausdrücke)

Which pattern finds -xy@ and -xyzi@ und -xyzi@ in a String? Chosse one correct answers.

(Welches Muster findet `-xy@` und `-xyz@` und `-xyzi@` in einem String? Wählen Sie eine korrekte Antwort)

- [A] `-[xyzi]+ @`
- [B] `-xy[zi]+@`
- [C] `-(xyzi)+@`
- [D] `-xy(zi)+@`
- [E] `-xyz@`

### Aufgabe 32

Given:

```
public class ThreadStart implements Runnable{
    public void run(){
        System.out.println("I work");
    }
    public static void main(String [] args){
        //insert here
        t.start();
    }
}
```

and the following code snippets:

(und die folgenden Code-Schnipsel:)

- [A] `Thread t =new Thread(new ThreadStart());`
- [B] `Thread t=new Thread();`
- [C] `ThreadStart=new ThreadStart();`
- [D] `Thread t=new ThreadStart();`

Which ones, inserted independently at `//insert here` will compile, run and output »I work«? Choose one answer.

(Welche bei `//insert here` kompilieren, laufen und geben »I work« aus? Wählen Sie eine Antwort.)

### Aufgabe 33

If you create a class `TestThread` that instantiates a thread in the main method, which statements are true, if you want to run a thread which generates an output? Choose two.

(Falls Sie eine Klasse `TestThread` erzeugen, die einen Thread in der `main()`-Methode erzeugt, welche Aussagen treffen dann zu, falls Sie einen Thread mit einem Output erzeugen wollen? Wählen Sie zwei Antworten)

- A `TestThread` must override `public void run()`.
- B `TestThread` must implement `Thread`.
- C `TestThread` must inherit from `Thread`.
- D `TestThread` must inherit from `Runnable`.
- E None of the above is necessary.

### Aufgabe 34

Given:

```
1 public class JoinTest1 implements Runnable{
2     private static String s="";
3     public void run(){
4         //try{Thread.sleep(1000);} catch (InterruptedException e){}
5         for (int i=0;i<50;i++)
6             s+="r";
7     }
8     public static void main(String... args){
9         Thread t = new Thread(new JoinTest1());
10        t.start();
11        for (int i=0;i<50;i++){
12            s+="m";
13            try{t.join();} catch (InterruptedException e){}
14        }
15        System.out.println(s);
16    }
17}
```

What is the difference, if you run this example without and with a call to `join()`?  
What will happen, if you uncomment `sleep()` in line 4? Which two are correct?

(Was ist der Unterschied, falls Sie dieses Beispiel einmal mit und einmal ohne Aufruf von `join()` laufen lassen? Was passiert, falls Sie `sleep()` in Zeile 4 auskommentieren? Welche beiden Aussagen sind korrekt?)

- A With `join()` the `main()`-thread runs more.
- B With `join()` the `main()`-thread runs less.
- C `join()` has no influence on the output.

- D `join()` sets the calling thread (here main) in a blocked-for-join state.
- E If you activate `sleep(1000)` and deactivate line 13, there would be more outputs of r, as the run thread gets more runtime.

### Aufgabe 35

Given:

```
public class YieldTest implements Runnable{
    private static long max=1000;
    public void run(){
        for (long i=0;i<max;i++){
            if (i%(max/10)==0){
                System.out.print(i+"r ");
                //Thread.yield();
            }
        }
    }
    public static void main(String... args){
        Thread t = new Thread(new YieldTest());
        t.start();
        for (long i=0;i<max;i++){
            if (i%(max/10)==0)
                System.out.print(i+"m ");
            //Thread.yield();
        }
    }
}
```

Which statements are true? Choose three.

(Welche Aussagen treffen zu? Wählen Sie drei.)

- A With `Thread.yield()` commented out the Output is evenly distributed.
- B The thread, in which `Thread.yield()` is activated, has less opportunities to run.
- C The thread, in which `Thread.yield()` is activated, has more opportunities to run.
- D The output consists of 10 numbers (0, 100, ...).

### Aufgabe 36

Given a thread with a synchronized `run()` method:

What is the output? Choose one.

```
public class SyncRun implements Runnable {
    public synchronized void run() {
        long t=Thread.currentThread().getId();
        for (int i=0;i<5;i++){
            if(i==2) try{wait();}catch(InterruptedException e){}
            System.out.println(t+" "+i);
        }
        public static void main(String[] args) {
            new Thread(new SyncRun()).start();
            new Thread(new SyncRun()).start();
        }
    }
}
```

(Wie lautet der Output? Wählen Sie eine Antwort.)

[A]

```
8 0
8 1
8 2
9 0
9 1
9 2
```

[B]

```
8 0
9 0
8 1
9 1
```

[C]

```
8 0
8 1
9 0
9 1
```

[D]

```
8 0
9 0
```

[E] Runtime error

[F] Compile error

**Aufgabe 37**

Given:

```
class MyThread extends Thread{
    int sum=0;
    public void run(){
        synchronized(this){
            for (int i=0;i<1000;i++){ //producing action
                sum+=i;
            }
            notify();
        }
    }
}

public class MyThreadTest{
    public static void main(String[] args){
        MyThread t1=new MyThread();
        t1.start();
        synchronized(t1){
            try{
                t1.wait(); //waiting for producer
            } catch (InterruptedException e){}
            System.out.println(t1.sum);
        }
    }
}
```

Which statements are true for the above code? Choose three.

(Welche Aussagen treffen zu? Wählen Sie drei Antworten.)

- [A] The program outputs the sum of 499500. If the `t1.wait()` in the `main()` method would lack the output could be 0.
- [B] `try{Thread.sleep(2000);}catch (InterruptedException e){}` could be used in the `main()` method instead the `synchronized` block and the result would be more or less the same.
- [C] `notify()` or `notifyAll()` is usually placed in front of a code which wants to consume.
- [D] `wait()` is usually placed in front of producing code.
- [E] The key of the `synchronized` block has to be the same on which the `wait()` method is called.

## Aufgabe 38

```
public class SomeThreads {
    private int x=0;
    public void increment() {
        x++;//#1
    }
    public void tenThreads() {
        for(int i = 0; i < 10; i++) {
            new Thread() {
                public void run() {
                    increment();
                    System.out.print(x + ", ");
                }
            }.start();
        }
    }
    public static void main(String[] args){
        new SomeThreads().tenThreads();
    }
}
//Output: 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
```

What two changes are necessary to change the program so that the output is

1, 2, 3, 4, 5, 6, 7, 8, 9, 10,

with a granted sequence? Choose two.

(Welche zwei Änderungen sind im obigen Programm notwendig, das der Output garantiert 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, lautet? Wählen Sie zwei Antworten)

- [A] Change public void run() to public synchronized run();
- [B] Move the System.out.print(x+",") to the method increment, below x++ (below #1)
- [C] Move the System.out.print(x+",") to the method increment, below x++ (below #1). Change public void increment() to public synchronized void increment()
- [D] Change public void tenThreads() to public void synchronized tenThreads();
- [E] .Wrap the for loop in public void tenThreads in a synchronized(this){} block.

### Aufgabe 39

Given:

```
public class PC extends Thread{
private int s=1;
public void run(){
for (int i=0;i<10;i++){
s++;
try{Thread.sleep(100);} catch (InterruptedException e){}
notify();
System.out.print(s+" ");
if (s<1){
try{wait();}
catch (InterruptedException e){}
}
else{s--;System.out.print(s+" ");
}
}
}
public static void main(String[] args){
new PC().start();
}
}
```

Which statements are true? Choose two.

(Welche Aussagen treffen zu?)

- [A] The program compiles.
- [B] The program does not compile.
- [C] The output is 2 1 2 1 ...
- [D] The output is 1 2 1 2 ...
- [E] An `IllegalMonitorStateException` is thrown.

### Aufgabe 40

Given:

```
1 class ThreadInheritance extends Thread{
2 public void myStart(){
3 super.start();
4 }
5 public void run(){
```

```
6  int i=0;
7  do {
8    System.out.print("Hello ");
9    i++;
10 }while(i<5);
11 }
12 public static void main(String [] args){
13   Thread t=new ThreadInheritance();
14   t.myStart();
15 }
16}
```

What is about the above code? Choose one correct answer.

(Was ist mit obigem Code? Wählen Sie eine korrekte Antwort.)

- [A] Compiles and runs
- [B] Runtime error
- [C] Compilation error because of line 3.
- [D] Compilation error because of line 13.
- [E] Outputs five times »Hello«
- [F] Outputs four times »Hello«

#### Aufgabe 41

Given:

```
public class MySequence implements Runnable {
    public synchronized void run(){
        for (int i=0;i<11;i++){
            try{Thread.sleep(1000);} catch (InterruptedException e){}
            System.out.print(i+" ");
        }
    }
    public static void main(String[] args){
        new Thread(new MySequence()).start();
        try{Thread.sleep(1000);} catch (InterruptedException e){}
        new Thread(new MySequence()).start();
    }
}
```

What is the sequence of the output? Choose two.

(Welches ist die Ausgabereihenfolge? Wählen Sie zwei Lösungen.)

- [A] Outputs two times 1 to 10 in a predefined sequence.
- [B] Outputs two times 1 to 10 in a random sequence.
- [C] The synchronized statement in the run-method guarantees an ordered sequence.
- [D] The synchronized statement in the run-method is not enough to guarantee any ordered sequence.
- [E] Runtime error
- [F] Compilation error

### Aufgabe 42

Given:

```
class Car{
    Wheel [] wheels=new Wheel[4];
    public Car(){
        for (int i=0;i<wheels.length;i++)
            wheels[i]=new Wheel();
        for(Wheel p:wheels)
            p.setPressure(2.5);
    }
}
class Wheel {
    double pressure=2.0;
    public void setPressure(double p){
        this.pressure=p;
    }
}
public class TestC{
    public static void main(String...args){
        Car car=new Car();
        System.out.println(car.wheels[0].pressure);
    }
}
```

What can be said about the above code? Choose three correct answers.

(Was kann über obigen Code ausgesagt werden? Wählen Sie drei korrekte Antworten.)

- [A] It implements tight encapsulation.
- [B] It implements loose encapsulation.

- [C] It implements high cohesion.
- [D] It implements low cohesion.
- [E] It implements tight coupling.
- [F] It implements loose coupling.

### Aufgabe 43

Given:

```
1 interface MyInterface{}
2 class A implements MyInterface{public void methodA(){}
3 }
4 public class TestFinal extends A{
5     public void methodB(){}
6     public static void main(String[] args){
7         MyInterface a=new A();
8         A b=new TestFinal();
9         final A c=new TestFinal();
10        c=b;
11        b.methodA();
12        b.bethodB();
12    }
13}
```

What is the result? Select two.

(Was resultiert? Wählen Sie zwei Antworten.)

- [A] Code does not compile because of line 10.
- [B] Code does not compile because of line 12.
- [C] Runtime Exception.
- [D] Compiles and runs.

### Aufgabe 44

```
class Alpha{
    private static final void print(Float n){
        System.out.print("Float ");
    }
}
public class Beta extends Alpha{
    private static final void print(Integer i){
        System.out.println("Integer ");
    }
}
```

```
}  
  
public static void main(String [] args){  
    print(3.0f); // #1  
    print(3); // #2  
    print(3.0); // #3  
}  
}
```

What is the result? Choose one.

(Was ist das Resultat? Wählen Sie eine Antwort.)

- [A] The output is Float, Integer, Float.
- [B] The code does not compile because of the call at #1.
- [C] The code does not compile because of the call at #2.
- [D] The code does not compile because of the call at #3.
- [E] The code does not compile because of the call at #1 and #3.

#### Aufgabe 45

Given:

```
class Animal{  
    public void communicate(){System.out.println("give sound ");}  
}  
public class Dog extends Animal{  
    void communicate(){System.out.println("bark");}  
    public static void main(String [] args){  
        Animal[] animals={new Animal(), new Dog()};  
        for(Animal a:animals)  
            a.communicate();  
    }  
}
```

What is the correct answer?

(Welches ist die korrekte Antwort?)

- [A] Output: give sound bark
- [B] Output: give sound give sound
- [C] Compilation fails
- [D] Runtime error

## Aufgabe 46

Given:

```
class Animal{
    String name;
    Animal(String name){this.name=name;}
}
public class Dog extends Animal{
    public Dog(String name){super(" Fluffi ");}
    public static void main(String [] args){
        Animal[] animals={new Animal(" Tobi "), new Dog(" Miro ")};
        for(Animal a:animals)
            System.out.print(a.name);
    }
}
```

What is the output?

(Welches ist der Output?)

- [A] Tobi Miro
- [B] Tobi Fluffi
- [C] Compilation fails
- [D] Runtime error

## Aufgabe 47

Given:

```
class Os {
    static String s="";
    Os loadOs(){
        s+="os loaded ";
        return this;
    }
}
public class Vista extends Os{
    Vista loadOs(){
        s+="Vista loaded";
        return this;
    }
    public static void main(String [] args){
        Os o=new Os();Vista v=new Vista();o.loadOs();v.loadOs();
    }
}
```

```

System.out.println(s);
    }
}

```

What is true for the above code? Choose two correct answers.

(Was trifft für den obigen Code zu? Wählen Sie zwei korrekte Antworten.)

- [A] The code compiles without problems.
- [B] The code does not compile, there is an illegal override.
- [C] Output: os loaded Vista loaded.
- [D] Output: Vista loaded os loaded.
- [E] Code ends in a runtime error.

### Aufgabe 48

```

class A{
    void myMethod(String s){}
class B extends A{
    //insert here
}

```

Which of the following code fragments can be inserted at `//insert here` so that the code compiles? Choose three.

(Welche der folgenden Code-Fragmente können bei `//insert here` eingesetzt werden, damit der Code kompiliert? Wählen Sie drei Antworten.)

- [A] **void** myMethod(String s);
- [B] **protected void** myMethod(String s) **throws** Error{}
- [C] **void** myMethod(String s) {}
- [D] **public int** myMethod(String s) {}
- [E] **public void** myMethod(**int** i) {}
- [F] **abstract void** myMethod(String s){}

### Aufgabe 49

Given:

```

class Animal{
    String name;
    public void communicate(){
        System.out.println("Animal sound");
    }
}

```

```
}  
}  
class Goose extends Animal{  
    public Goose(String name){this.name=name;}  
    public void communicate(){System.out.println("quack");}  
    public void fly(){System.out.println("I fly away");}  
}  
class Dog extends Animal{  
    public Dog(String name){this.name=name;}  
    public void communicate(){System.out.println("bark");}  
    public void clean(){System.out.println("cleaning fur");}  
}  
public class AnimalTest{  
    public static void main(String [] args){  
        Animal[] animals={new Dog("Miro"),new Goose("Otto")};  
        for(Animal a:animals){a.communicate();  
            if(a instanceof Dog){Dog b=(Dog)a; b.clean();}  
            if (a instanceof Goose){Goose c=(Goose)a; c.fly(); }  
        }  
    }  
}
```

What's about the above code? Choose two.

(Was ist mit dem obigen Code? Wählen Sie zwei Antworten.)

[A] Output:

```
bark  
cleaning fur  
quack  
I fly away
```

[B] ClassCastException is thrown at runtime

[C] Compilation error

[D] Output:

```
cleaning fur  
I fly away
```

[E] Compiles and runs

### Aufgabe 50

```
class Foo{private Fooable f;}  
interface Fooable{}  
class Bar extends Foo implements Fooable{
```

```
void method(String s){}
}
```

Which are the relationships? Choose four.

(Welches sind die Verwandtschaftsbeziehungen? Wählen Sie vier Antworten.)

- [A] class Bar is a Fooable
- [B] Foo has a Fooable
- [C] class Foo is a Bar
- [D] class Bar is a Foo
- [E] class Bar has a method()
- [F] class Bar has a Fooable

### Aufgabe 51

Given:

```
class A{
    static void method(){System.out.print("in A ");}
}
class B extends A{
    static void method(){System.out.print("in B ");}
}
public class ShadowTest{
    public static void main(String... args){
        A a=new A();
        B b=new B();
        A c=new B();
        a.method();
        b.method();
        c.method();
    }
}
```

What is the correct output of the above code? Choose one.

(Welcher ist der korrekte Output des obigen Codes? Wählen Sie eine Lösung.)

- [A] in B in B in A
- [B] in B in B in B
- [C] in B in A in A
- [D] in A in B in A
- [E] if »static« in both methods is omitted, the Output is: in A in B in B

**Aufgabe 52**

Given:

```
class Parent {
    int x=10;
    public void f() {System.out.println("parent dynamic ");}
    public static void fs(){System.out.println("parent static ");}
}
class Child extends Parent {
    int x=20;
    public void f(){System.out.println("child dynamic ");}
    public static void fs(){System.out.println("child static ");}
}
public class OverrideShadow {
    public static void main(String[] args){
        Parent c = new Child();
        System.out.println(c.x);
        c.f();
        c.fs();
    }
}
```

What is the output?

 [A]

```
20
child dynamic
child static
```

 [B]

```
10
parent dynamic
parent static
```

 [C]

```
10
child dynamic
parent static
```

 [D]

```
20
child dynamic
parent static
```

**Aufgabe 53**

What output is generated by the following code?

(Welchen Output erzeugt folgendes Programm?)

```
class Foo{
    private int x;
    public Foo(int x){
        this.x=x;
    }
    public int getX(){
        return x;
    }
}

public class Bar{
    static Foo fooBar(Foo foo){
        foo = new Foo(100);
        return foo;
    }

    public static void main(String[] args){
        Foo foo = new Foo(200);
        System.out.print(foo.getX()+ "-");

        Foo fooFoo = fooBar(foo);
        System.out.print(foo.getX()+"-");
        System.out.print(fooFoo.getX()+"-");

        foo = fooBar(fooFoo);
        System.out.print(foo.getX() + "-");
        System.out.print(fooFoo.getX());
    }
}
```

Which output is correct? Choose one.

(Welcher Output ist korrekt? Wählen Sie eine Antwort.)

- [A] 200-100-100-100-100
- [B] 200-200-100-100-100
- [C] 200-200-200-100-100
- [D] 200-200-200-200-100

## Aufgabe 54

Given:

```
import java.util.*;
import java.util.concurrent.*;
public class Account1{
    public static void main(String[] args){
        String[] nameArr={"Hans","Xafer","Baldwin","Aaron"};
        Double[] balanceArr={2300.00,250000.00,30000.00,10000.00};
        //insert here

        for (int i=0;i<nameArr.length;i++){acc.put(nameArr[i],balanceArr[i]);}
        for (Map.Entry<String, Double> e : acc.entrySet())
            System.out.println(e.getKey() + " : " + e.getValue());
    }
}
```

Which line inserted at `//insert here` will result in an alphabetically sorted output? Choose two.

(Welche Zeile bei `//insert here` eingesetzt resultiert in einem alphabetisch sortierten Output? Wählen Sie zwei Antworten.)

- [A] `Map<String,Double> acc=new Map<String,Double>();`
- [B] `Map<String,Double> acc=new HashMap<String,Double>();`
- [C] `Map<String,Double> acc=new LinkedHashMap<String,Double>();`
- [D] `Map<String,Double> acc=new TreeMap<String,Double>();`
- [E] `Map<String,Double> acc=new NavigableMap<String,Double>();`
- [F] `Map<String,Double> acc=new ConcurrentSkipListMap<String,Double>();`

## Aufgabe 55

Given:

```
import java.util.*;
class GpsRecord implements Comparable<GpsRecord>{
    public int compareTo(GpsRecord gpsr){
        if (this.z>gpsr.z) return -1;
        if (this.z==gpsr.z) return 0;
        if (this.z<gpsr.z) return 1;
        return 0;
    }
}
```

```
long gpsTime=new Date().getTime();//#1
double x;double y;double z;
public GpsRecord(double x,double y,double z){
    this.x=x;this.y=y;this.z=z;
}
}
public class GpsTracker{
    public static void main(String[] args){
        TreeSet<GpsRecord> ts = new TreeSet<GpsRecord>();
        ts.add(new GpsRecord(2.0,3.0,77.0));
        ts.add(new GpsRecord(3.0,4.0,10.0));
        ts.add(new GpsRecord(4.0,5.0,22.0));
        for (GpsRecord g:ts)
            System.out.printf("\n%.2f %.2f %.2f",g.x,g.y,g.z);
        }
    }
}
```

Select the one correct sort order:

(Wählen Sie einen die eine korrekte Sortierung:)

[A]

```
4.00 5.00 10.00
3.00 4.00 22.00
2.00 3.00 77.00
```

[B]

```
2.00 3.00 77.00
4.00 5.00 10.00
3.00 4.00 22.00
```

[C]

```
2.00 3.00 77.00
3.00 4.00 22.00
4.00 5.00 10.00
```

[D]

```
3.00 4.00 22.00
2.00 3.00 77.00
4.00 5.00 10.00
```

## Aufgabe 56

```
import java.util.*;
class GpsRecord implements Comparable{
    //insert here
    long gpsTime=new Date().getTime();
    double x;double y;double z;
    String comment;
    public GpsRecord(double x,double y,double z, String comment){
        this.x=x;this.y=y;this.z=z;this.comment=comment;
    }
}
```

Which implementation of `compareTo()` is correct, if you wish to sort for the member variable `comment`? Choose one.

(Welche Implementation von `compareTo()` ist korrekt, falls Sie die Membervariable `comment` sortieren wollen? Wählen Sie eine Antwort.)

[A]

```
public int compareTo(GpsRecord gpsr){
    if (this.z>gpsr.z) return -1;if (this.z==gpsr.z) return 0;if
    (this.z<gpsr.z) return 1;return 0;
}
```

[B]

```
public int compareTo(GpsRecord gpsr){
    return comment.compareTo(gspr.comment);
}
```

[C]

```
public int compareTo(Object gpsr){
    return comment.compareTo(gspr.comment);
}
```

[D]

```
public int compare(Object gpsr){
    return comment.compareTo(gspr.comment);
}
```

[E]

```
public int compareTo(Object o){
    GpsRecord gr=(GpsRecord)o;
    return comment.compareTo(gr.comment);
}
```

## Aufgabe 57

Given:

```
import java.io.*;
import java.util.*;
public class BinarySearchTest{
    public static void main(String[] args){
        Console c=System.console();
        List<String> t=new ArrayList<String>();
        t.add("neh"); t.add("snoc");t.add("snochie"); t.add("boochies");
        t.add("snigi");t.add("snoogans");
        Collections.sort(t);
        int position1=Collections.binarySearch(t, "snigi");
        int position2=Collections.binarySearch(t, "snoc");
        c.printf("%d %d\n",position1,position2);
    }
}
```

Which are the correct positions ? Choose one.

(Welches sind die korrekten Positionen? Wählen Sie eine Lösung.)

- [A] 2 6
- [B] 2 -5
- [C] 2 -6
- [D] 1 -6
- [E] 1 -5

## Aufgabe 58

Given a legacy implementation of a method:

(Gegeben ist eine veraltete Implementation einer Methode:)

```
public static double avg(List list) {//#1
    double sum = 0;
    for ( Iterator iter = list.iterator(); iter.hasNext(); ) { //#2
        Double d = ((Double)iter.next()).doubleValue(); //#3
        sum += d; //#4
    }
    return (double)sum/list.size(); //#5
}
```

```

public static double avg(List<Double> list) {
    double sum = 0;
    for ( Double d:list) {
        sum += d;
    }
    return sum/list.size();
}

```

Which three changes must be made to the method avg() to use generics? Choose three.

(Welche drei Änderungen sind notwendig, damit die Methode avg() Generics verwendet? Wählen Sie drei Antworten.)

- [A] remove line #2
- [B] replace line #2 with »for ( Double d:list) {«
- [C] replace line #2 with »for (Iterator iter : list) {«
- [D] remove line #3
- [E] replace line #1 with »public static double avg(List<Double> list) {«

### Aufgabe 59

```

import java.util.*;
import java.util.concurrent.*;
public class CountyMap1{
    public static void main(String[] args) {
        NavigableMap <String, String>navMap = new ConcurrentSkipListMap<String,
String>();
        navMap.put("zh", "Zurich");navMap.put("be", "Berne");
        navMap.put("ag", "Argovia");navMap.put("sh", "Shaffouse");
        navMap.put("tg", "Thurgovie");
        SortedMap<String,String> navSub=navMap.subMap("be","zh");
        navSub.put("vd", "Vaud");
        for (Map.Entry countyPairs:navSub.entrySet()){
            System.out.println(countyPairs.getKey()+" = "+countyPairs.getValue());
        }
    }
}

```

What is the output? Select one.

(Was wird ausgegeben? Wählen Sie eine einzige Antwort.)

- [A] Runtime error, because put on navSub is out of range.

- [B] Compilation error
- [C]

```
be = Berne
sh = Shaffouse
tg = Thurgovie
vd = Vaud
```

- [D]

```
be = Berne
sh = Shaffouse
tg = Thurgovie
vd = Vaud
zh= Zurich
```

### Aufgabe 60

Given:

```
//insert here
private T max=null;
private T min=null;
private T[] value;
public MyMax(T[] v){ //Konstruktor
    this.value=v;
}
public T max(){
    T v=value[0];
    for (int i=1;i<value.length;i++)
        if(value[i].compareTo(v)>0) v=value[i];
    return v;
}
}
```

Which declaration inserted at //insert here compiles and runs? Choose one answer.

(Welche Deklaration eingefügt bei //insert here kompiliert und läuft? Wählen Sie eine Antwort)

- [A] class MyMax <T implements Comparable<T>> {
- [B] class MyMax <T extends Comparable<T>> {
- [C] class MyMax <T <T>> {
- [D] class MyMax <? extends Comparable<T>> {

### Aufgabe 61

Given:

```
public class [ ] {
    private [ ] var;
    public Generic ([ ] var) {
        this.var=var;
    }
    public [ ] getVar(){
        return var;
    }

    public static void main(String[] args){
        Generic<String> str=new Generic<String>("hello");
        Generic<Integer>i=new Generic<Integer>(10);
        System.out.println(str.getVar()+" "+i.getVar());
    }
}
```

Place some of the following into the correct slots, so that the code compiles and runs and writes »hello 10«.

(Platzieren Sie einige der folgenden Kästchen in die korrekten Fächer, sodass der Code kompiliert, läuft und »hello 10« ausgibt.)

Generic<T>

T

Generic

?

### Aufgabe 62

Given the following legacy code:

(Gegeben ist folgender veralteter Code:)

```
import java.util.*;
public class Conversion {
    public static void main(String []args){
        List list = new LinkedList();//replace here #1
        list.add("one");
        list.add("two");
        Object x=list.get(0); //replace here #2
        System.out.println(x);
    }
}
```

What must be replaced at #1 and inserted at #2 so that the code will compile and run without warnings?

(Was muss bei #1 und bei #2 ersetzt werden, damit der Code kompiliert und ohne Warnung läuft?)

[A]

```
List <String> list = new LinkedList ();
String x=list.get(0);
```

[B]

```
List list = new LinkedList <String> ();
String x=list.get(0);
```

[C]

```
List <String> list = new LinkedList<String>();
String x=list.get(0);
```

[D]

```
List <String>list = new <String>LinkedList ();
Object o=list.get(0);
```

### Aufgabe 63

Given :

```
public static void main(String [] args){
    String[] sarr = new String[]{"a","b","c","d","e" };
    List<String> l=Arrays.asList(sarr);
    //l.add("f"); //1UnsupportedOperationException, hinzufügen nicht erlaubt
    l.set(0,"z"); //ändern erlaubt!!!
    for (String i:l)
        System.out.print(i+" ");
    System.out.println();
    for (String i:sarr)
        System.out.print(i+" ");
    }
}
```

Which statements are true. Coose two.

(Welche Aussagen treffen zu, wählen Sie zwei?)

[A] Output:

```
a b c d e
z b c d e
```

[B] Output:

```
z b c d e
z b c d e
```

[C] `l.add("f");` works fine

[D] `l.add("f");` causes an error

[E] `sarr` is independent from `l`

### Aufgabe 64

On a linux system, a Java program called `MyProg` can be started from the directory

```
/home/user1
```

using the command:

```
java -classpath /test:/home/user1/downloads/*.jar games.MyProg
```

The CLASSPATH is set to

```
/usr/lib:/home/user1/classes:/opt/java/lib:/opt/java/lib/*.jar
```

at system boot.

What is a possible location for the `MyProg.class` file? Choose one.

(Welches ist ein möglicher Standort der Datei `MyProg.class`? Genau eine Antwort auswählen.)

- [A] `/test/MyProg.class`
- [B] `/home/user1/MyProg.class`
- [C] `/usr/lib/games/MyProg.class`
- [D] `/home/user1/games/ MyProg.class`
- [E] `/test/games/MyProg.class`
- [F] inside jarfile `/opt/java/lib/MyProg.jar` (with a correct manifest)
- [G] inside jarfile `/home/user1/downloads/MyProg.jar` (with a correct manifest)

**Aufgabe 65**

Given:

```
public class Operands {
    public static void main(String [] args) {
        int x = 5;
        boolean b1 = true;
        boolean b2 = false;
        if ((x == 4) && !b2 )
            System.out.print("1 ");
        System.out.print("2 ");
        if ((b2 = true) && b1 )
            System.out.print("3 ");
    }
}
```

What is the result?

(Was ist das Ergebnis?)

- [A] 2
- [B] 3
- [C] 2 3
- [D] 1 3

**Aufgabe 66**

Given:

```
public class StringNumberOutput{
    public static void main(String[] args){
        int izahl=5;
        String szahl="7";
        double dzahl=8.85;
        System.out.println(izahl+szahl+dzahl);//#1
        System.out.printf("%f7.2",izahl+szahl+dzahl);//#2
    }
}
```

What is the output?

(Wie lautet der Output?)

- [A] Compiler error
- [B] Runtime error due to line #1

- [C] 578.85 and Runtime error due to line #2
- [D] 578.85 578.85
- [E] 20.85 20.85

### Aufgabe 67

Given:

```
01 public class Test{
02     public static void main(String... args){
03         ArrayList al=new ArrayList();
04         for (int i=0;i<5;i+1){
05             Double d=new Double(i);
06             al.add(d);
07         }
08         System.out.println(al);
09     }
10 }
```

Which line of code marks the first point that an object referenced by d becomes eligible for garbage collection?

(Welche Codezeile markiert den ersten Punkt, wo das durch d referenzierte Objekt bereit ist für den Garbage-Collector?)

- [A] 10
- [B] 8
- [C] 9
- [D] 7

### Aufgabe 68

```
class CommandLine{
    public static void main(String... args){
        for (int i=1;i<args.length;i++)
            System.out.print(args[i]+ " " );

    }
}
```

What is the output if the program is started with a) `java CommandLine` and b) with `java CommandLine 1 2 3 4`?

(Was wird ausgegeben, falls das Programm mit a) `java CommandLine` und b) mit `java CommandLine 1 2 3 4` gestartet wird?)

- [A] (no Output) 1 2 3
- [B] (no Output) 2 3
- [C] (no Output) 2 3 4
- [D] Runtime Error, 1 2 3
- [E] Runtime Error, 2 3 4

### Aufgabe 69

Which relationship represents the relationship »Man's best friend is a Horse«?

(Welche Relation wird durch die Aussage »Ein Pferd ist des Menschen bester Freund« ausgedrückt?)

- [A] `class Man extends Horse { }`
- [B] `class Man implements Horse{ }`
- [C] `class Man { private Horse bestFriend; }`
- [D] `class Man { private bestFriend Horse; }`
- [E] `class Man { private Horse<bestFriend>; }`
- [F] `class Man { private BestFriend<Horse>; }`

### Aufgabe 70

Given:

```
1 class CommandLine{
2 public static void main(String... args){
3   Integer i= args[0];
4   int j=4;
5   if (i==j)
6     System.out.println("i==j");
7   else
8     System.out.println("i!=j");
9   }
10 }
```

The program is started by issuing `java CommandLine 4`. What is the Result?

(Das Programm wird mit `java CommandLine 4` gestartet. Was ist das Resultat?)

- [A] Compiler error at line 3
- [B] `i==j`

- [C] i!=j
- [D] Runtime error

### Aufgabe 71

```
class A {
    B b; //#1
    public A() { b = new B(this); }
}
class B {
    A a; //#2
    public B(A s) { a = s; }
}
public class GCTest{
public static void main(String[] args) {
    A ref = new A(); //#3
    ref = null;
    // more code
}
}
```

Which statement is true about the objects referenced by a (#2), b (#1), and ref (#3) immediately after line `ref=null` executes?

(Welche Aussage über die Objekte, die durch a (#2), b (#1) und ref (#3) referenziert sind, trifft direkt nach der Stelle, wo `ref=null` ausgeführt wird, zu?)

- [A] None of these objects are eligible for garbage collection.
- [B] Only the object referenced by a is eligible for garbage collection.
- [C] Only the object referenced by b is eligible for garbage collection.
- [D] Only the object referenced by ref is eligible for garbage collection.
- [E] The objects referenced by a and b are eligible for garbage collection.

### Aufgabe 72

The class `MyClass.class` is deployed in a JAR file named `Lib.jar`. Which statements will allow to use the `MyClass` in a program named `Test`? Choose three.

(Die Klasse `MyClass.class` wurde in einer jar-Datei namens `Lib.jar` bereitgestellt. Welche Befehle erlauben die Verwendung von `MyClass` in einem Programm namens `Test`? Wählen Sie drei Antworten.)

- [A] The JAR file is located at `$JAVA_HOME/lib/ext/Lib.jar`.
- [B] The JAR file is located at `$JAVA_HOME/jre/classes/Lib.jar`.

- [C] The JAR file is located at /test/Lib.jar and a classpath environment variable is set so that it includes /test/myLib.jar/MyClass.class.
- [D] The JAR file is located at /test/Lib.jar and a classpath environment variable is set so that it includes /test/Lib.jar.
- [E] The JAR file is located at /test/Lib.jar and the Test class is compiled using `javac -cp/test/myLib.jar/MyClass Test.java`.
- [F] The JAR file is located at /test/Lib.jar and the Book class is compiled using `javac -d /test/Lib.jar Test.java`.
- [G] The JAR file is located at /test/Lib.jar and the Test class is compiled using `javac -classpath/test/Lib.jar Test.java`.

## 10.4 Lösungen zu Simulationstest 2

Die Aufgaben und Lösungen sind nach Prüfungsziel gegliedert. Hinter der Aufgabenstellung steht in Klammern jeweils das Prüfungsziel und das Thema.

### 10.4.1 Prüfungsziel 1

#### Aufgabe 1 (1-1, Verschachtelte Klassen)

Given the following source:

```
package aufgabe1;

public class House {
    public class Room{void lamp(){System.out.println("lamp");}
        public class Furniture extends Room{void chair(){System.out.println("chair");}};
    };

    public static void main(String[] args) {
        House h=new House();
        House.Room r=h.new Room();
        House.Room.Furniture f=r.new Furniture();
        h.chair();
        r.chair();
        r.lamp();
        f.lamp();
        f.chair();
    }
}
```

Which invocations of methods cause the program to not compile? Choose one answer.

(Welche Methodenaufrufe bewirken, dass das Programm nicht kompiliert werden kann? Wählen Sie eine Antwort)

- [A] *h.chair()* alone
- [B] *r.chair()* alone
- [C] *only r.lamp()*
- [D] *r.lamp()* and *f.lamp()*
- [E] *r.lamp()* and *f.lamp()* and *f.chair()*
- [F] *h.chair()* and *r.chair()*

### Lösung

F ist korrekt. Von der Vererbungshierarchie aus *r.lamp()* and *f.lamp()* and *f.chair()* funktionieren diese Varianten, *h.chair()* und *r.chair()* hingegen nicht, weil *chair()* eine Methode von Room ist und mit Elternreferenzen nicht angesprochen werden kann.

```
package aufgabe1;
public class House {
    public class Room{void lamp(){System.out.println("lamp");}
        public class Furniture extends Room{void chair(){System.out.println("chair");}
        };//Ende Furniture
    };//Ende Room

    public static void main(String[] args) {
        House h=new House();
        House.Room r=h.new Room();
        House.Room.Furniture f=r.new Furniture();
        //h.chair(); //funktioniert nicht
        //r.chair();//funktioniert nicht
        r.lamp();
        f.lamp();
        f.chair();
    }
}
```

**Aufgabe 2 (1-1, Statische und normale Importe)**

Given a library class and eight different test classes:

(Gegeben sind eine Bibliotheksklasse und acht Testklassen.)

Which one cannot be compiled? Choose one answer.

(Welche kann nicht kompiliert werden?)

```
package ch.geo.scjp;
public class MyLib{
    public static int i=1;
    public int j=2;
}
And
a)
class Test{
public static void main(String[] args) {
    System.out.println(ch.geo.scjp.MyLib.i);
}
}
b)
import ch.geo.scjp.*;
class Test{
public static void main(String[] args) {
    System.out.println(MyLib.i);
}
}
c)
import ch.geo.scjp.MyLib;
class Test{
public static void main(String[] args) {
    System.out.println(MyLib.i);
}
}
d)
import static ch.geo.scjp.MyLib.*;
class Test{
public static void main(String[] args) {
    System.out.println(i);
}
}
e)
```

```
import ch.geo.scjp.MyLib;
class Test{
public static void main(String[] args) {
MyLib m1=new MyLib();
    System.out.println(m1.i);
    }
}
f)
import ch.geo.scjp.MyLib.*;
class Test{
public static void main(String[] args) {
MyLib m1=new MyLib();
    System.out.println(m1.j);
    }
}

g)
class Test{
public static void main(String[] args) {
ch.geo.scjp.MyLib m1=new ch.geo.scjp.MyLib();
    System.out.println(m1.j);
    }
}
h)
import ch.geo.scjp.MyLib;
class Test{
public static void main(String[] args) {
MyLib m1=new MyLib();
    System.out.println(m1.j);
    }
}
```

- [A] a
- [B] b
- [C] c
- [D] d
- [E] e
- [F] f
- [G] g
- [H] h

### Lösung

F ist der einzige Import, der scheitert.

Erläuterung

- Test.class befindet sich immer im Ordner oberhalb ch/geo/scjp, da kein Package angegeben wurde.
- Vollqualifizierter Zugriff auf statisches Member i, ok.
- Dynamischer Import aller Klassen, statisches Member i, ok
- Dynamischer Import der Klasse MyLib, statisches Member i, ok
- Statischer Import aller Member der Klasse MyLib, ok
- Dynamischer Import der Klasse MyLib, statisches Member i, ok
- Dynamischer Import aller Member Klasse MyLib, falsch, müsste static sein
- Vollqualifizierter Zugriff auf dynamisches Member, ok.
- Dynamischer Import der Klasse MyLib, dynamisches Member j, ok
- Regeln des Imports: Statischer Import importiert immer ein oder mehrere Member, nicht-statischer Import eine oder mehrere Klassen. Nur statische Member können statisch importiert werden.

### Aufgabe 3 (1-3, Array)

Given following code:

```
1 int[] a={0,1,2,3,4};
2 Object o=a;
3 int[] myArray = (int[]) o;
4 for(int i:myArray)
5 System.out.print(i);
```

What is the result? Choose one answer.

(Was ist das Resultat? Wählen Sie eine Antwort.)

- [A] 01234
- [B] ClassCastException at line 3
- [C] RuntimeException due to an error at line 2
- [D] Does not compile.

### Lösung

A ist korrekt. Es handelt sich in Zeile 3 um ein korrektes Downcasting mit vorangegangenem Upcasting in Zeile 2.

### Aufgabe 4 (1-3, Enum)

Given:

```
public class EnumTest {
    enum Auto{ //Deklaration innerhalb der aufrufenden Klasse
        Ford(20000), Opel(22000), VW(25000), Mercedes(30000); // #1
    }
    private int price;
    Auto (int price){
        this.price=price;
    };
    int getPrice(){
        return price;
    }
} ; // #2

public static void main(String[] args){
    Auto a;
    System.out.println(Auto.Mercedes.getPrice());
    System.out.println(Auto.Ford.getPrice()); // #3
}
}
```

What happens with this code? Choose one.

(Was passiert mit diesem Code? Wählen Sie eine Antwort.)

- [A] Does not compile, reason: »;« at #1 should be replaced by a »;,«.
- [B] Does not compile, reason: »;« at #1 should be deleted.
- [C] Compiles and runs, Output: 30000 20000
- [D] Runtime error
- [E] Runtime error at #3, getPrice() is only defined for Mercedes.

### Lösung

C ist korrekt. Die Methode `getPrice()` gilt für alle Autos. Das enum-Konstrukt enthält vier Werte. Im letzten Wert sind der Konstruktoraufruf, der Konstruktor, die Variable `price` und die Methode `getPrice()` gespeichert. Die Methoden folgen bei #1 auf das letzte Element und sind korrekt mit einem Semikolon abgetrennt.

**Aufgabe 5 (1-4 Varargs)**

```
public class VarArgTest{

    public void method(int[] i) {
        for (int j:i)
            System.out.print(j+" ");
        System.out.println("Array int");
    }

    public void method(int... i) {
        for (int j:i)
            System.out.print(j+" ");
        System.out.println("Vararg int");
    }

    public void method(Integer... i){
        for (Integer j:i)
            System.out.print(j+" ");
        System.out.println("Vararg Integer");
    }

    public static void main(String... args){
        VarArgTest vat=new VarArgTest();
        vat.method(new int[] {1,2,3} );

        Integer[] j={5,6,7};
        vat.method(j);
    }
}
```

What is the result of the above code. Choose one.

(Was ist das Resultat des obigen codes, wählen Sie eine Antwort.)

- [A] Compilation error
- [B] Runtime error
- [C] Outputs

```
1 2 3 Vararg int
5 6 7 Vararg Integer
```

- [D] Outputs

```
1 2 3 Array int
5 6 7 Vararg Integer
```

### Lösung

A ist korrekt. Der Kompiler kann nicht zwischen `public void s(int[] i)` und `public void s(int ... i)` unterscheiden:

```
VarArgTest.java:9:15: cannot declare both method(int...) and method(int[]) in VarArgTest
```

### Aufgabe 6 (1-2 Interfaces, abstrakte Klassen)

Given:

```
1 interface Eatable {boolean eat(String thing);}
2 abstract class Apple implements Eatable {public boolean eat(String thing);}
3 abstract class Pear implements Eatable {boolean eat(){return true;}}
4 public class GoldenDelicious extends Apple implements Eatable{boolean eat(){return true;}}
```

Which one statement is true?

(Welche Aussage trifft zu?)

- [A] Compilation succeeds
- [B] Compilation fails due to only an error in line 1.
- [C] Compilation fails due to only an error in line 2.
- [D] Compilation fails due to only an error in line 3.
- [E] Compilation fails due to only an error in line 4.

### Lösung

C ist korrekt. In der Klasse Apple muss eine Implementation der Methode `eat()` vorhanden sein.

```
interface Eatable {boolean eat(String thing);}
abstract class Apple implements Eatable {public boolean eat(String thing){return true;}}
abstract class Pear implements Eatable {boolean eat(){return true;}}
public class GoldenDelicious extends Apple implements Eatable{boolean eat(){return true;}}
```

Es gäbe noch mehrere andere Varianten. Eine ist, in Klasse Apple die Methode `eat()` als *abstract* zu deklarieren. Dies hat aber zur Folge, dass die Klasse GoldenDelicious `eat()` implementieren muss. Der Modifizierer muss auf *public* gestellt werden, weil in einem Interface deklarierte Methoden implizit *public* sind, sonst klappt das Überschreiben nicht.

```

interface Eatable {boolean eat(String thing);}
abstract class Apple implements Eatable {public abstract boolean eat(String
thing);}
abstract class Pear implements Eatable {boolean eat(){return true;}}
public class GoldenDelicious extends Apple implements Eatable{public boolean
eat(String thing){return true;}}

```

### Aufgabe 7 (1-1, Überschreiben und Überlagern)

Given:

```

interface Iface{ String doit();}
class A implements Iface{public String doit(){return "Hello";}}
class B extends A {void doit(String s){System.out.println("Bye");}}
public class C extends B{public String doit(){System.out.println("Hey");return
"";}}
    public static void main(String [] args){
        A a=new C();
        a.doit();
    }
}

```

What happens?

(Was passiert?)

- [A] Code compiles and runs, Output »Hey«
- [B] Code compiles and runs, Output »Hello«
- [C] Compiler error in Class A
- [D] Compiler error in Class B
- [E] Compiler error in Class C
- [F] Runtime error

### Lösung

A ist korrekt. Es wird »Hey« ausgegeben, da das Objekt (rechte Seite des Gleichheitszeichens) vom Typ C ist. In Klasse A wird doit() korrekt überlagert und nicht überschrieben (andere Parameterliste). Erst in B, der Modifizierer muss public sein, wird doit() korrekt überschrieben.

### Aufgabe 8 (1-6, Shadowing)

Given:

```

class Foo{public static Foo createFoo(){
    return new Foo();
}

```

```
};  
public void hello(){System.out.print("Hello ");  
}  
}  
public class Bar extends Foo{  
public static Bar createFoo(){return new Bar();};  
public void hello(){System.out.print("Hy "); }  
public static void main(String[] args){  
    Foo f=createFoo();  
    Bar b=createFoo();  
    f.hello();  
    b.hello();  
}  
}
```

Which statement is true? Select two.

(Welche Aussagen treffen zu? Wählen Sie zwei.)

- [A] Code does not compile.
- [B] Code causes runtime error.
- [C] Code compiles and runs, outputs »Hy Hy«
- [D] CreateFoo in Bar has a covariant return when overriding.
- [E] CreateFoo() is not overridden in this example.
- [F] Code compiles and runs, outputs »Hello Hy«

### Lösung

C und E sind korrekt. Obwohl der Return-Typ von createFoo() in Bar kovariant zum Rückgabebetyp in Foo ist, handelt es sich nicht um ein Überschreiben sondern nur um ein Überdecken (Shadowing) von statischen Methoden. Es wird zwei Mal createFoo() in der Klasse Bar aufgerufen, da die Polymorphie mit statischen Methoden nicht spielt. Damit werden auch nur Referenzen auf die Klasse Bar zurückgegeben und nur hello() von Bar aufgerufen.

### Aufgabe 9 (1-6, default-Konstruktor)

Given:

```
class Parent{  
    private int age=0;  
    public Parent(int age){//#1  
        this.age=age;  
    }  
}
```

```

public int getAge(){
    return this.age;
}
}
public class Child extends Parent{
    public Child(){//#2
        public static void main(String[] args){
            Child c=new Child();
            Parent p=new Parent();
            System.out.println(p.getAge());
        }
    }
}

```

Which statements are true? Choose two answers:

(Welche Aussagen treffen zu? Wählen Sie zwei Antworten:)

- [A] Code compiles and runs.
- [B] There is a lacking default constructor in class Parent
- [C] There is a lacking default constructor in class Child
- [D] There is a runtime error.

### Lösung

B und D sind korrekt. In der Klasse Parent fehlt der Default-Konstruktor. In Zeile #2 muss der Kindkonstruktor deshalb explizit den Parameterkonstruktor der Elternklasse aufrufen oder es muss in der Elternklasse ein Konstruktor ohne Parameter bereitgestellt werden.

Falls Vererbung vorliegt, muss in der Elternklasse immer ein Default-Konstruktor vorhanden sein oder die Instantiierung der Elternklasse muss den Parameterkonstruktor explizit aufrufen..

```

class Parent{
    private int age=0;
    //public Parent(){this(50);} Nicht-Parameter Konstruktor, der

    public Parent(int age){
        this.age=age;
    }
}
public class Child extends Parent{
    public Child(){super(1);}
    public static void main(String[] args){
        Child c=new Child();
    }
}

```

```
Parent p=new Parent(2);  
}  
}
```

### Aufgabe 10 (1-4, Default Konstruktor)

Given:

```
class Temperature{  
    private int t=2;  
    {t+=t;}  
    public Temperature(int t){  
        this.t=t;  
    }  
    //insert here  
    public static void main(String[] args){  
        Temperature myT0=new Temperature();  
        System.out.println(myT0.t);  
        Temperature myT1=new Temperature(20);  
        System.out.println(myT1.t);  
    }  
}
```

What has to be inserted at `//insert here` so that the code compiles and runs and outputs 4 and 20? Choose one answer.

(Was muss bei `//insert here` eingegeben werden, dass der Code kompiliert und 4 und 20 ausgibt? Wählen Sie eine einzige Antwort.)

- [A] nothing
- [B] `t=t+2;`
- [C] `public Temperature() {}`
- [D] `public Temperature(){super(5)}`

### Lösung

C ist korrekt. Der Default-Konstruktor existiert nicht mehr, falls ein Parameterkonstruktor programmiert wird. Deshalb muss der Konstruktor ohne Parameter von Hand eingesetzt werden.

```
class Temperature{  
    private int t=2;  
    {t+=t;}  
    public Temperature(int t){  
        this.t=t;  
    }  
}
```

```
}  
    public Temperature(){}  
    public static void main(String[] args){  
        Temperature myT0=new Temperature();  
        System.out.println(myT0.t);  
        Temperature myT1=new Temperature(20);  
        System.out.println(myT1.t);  
    }  
}
```

## 10.4.2 Prüfungsziel 2

### Aufgabe 11 (2-1, if ohne Klammern)

```
public class IfChaos{  
    public static void main(String [] args){  
        int a=1,b=2;  
        if (a>b)  
            System.out.println("first");  
        else if (a+2>b)  
            System.out.println("second");  
        else if(true)  
            System.out.println("third");  
        else  
            System.out.println("fourth");  
    }  
}
```

What is true about the code? Choose one answer.

(Was trifft für den Code zu? Wählen Sie eine Antwort.)

- [A] Outputs *first*
- [B] Outputs *second*
- [C] Outputs *third*
- [D] Outputs *fourth*
- [E] Compilation fails

### Lösung

B ist korrekt, es wird nur »second« ausgegeben, da die *else*-Abzweigungen nicht durchlaufen werden. *ifs* können ohne geschweifte Klammern geschachtelt werden, falls nach einem ersten *if* direkt ein zweites *if* folgt.

**Aufgabe 12 (2-1, Iterator)**

Given:

```
import java.util.*;
public class IteratorTest{
    public static void main(String [] args){
        int myArray[] ={0,1,2,3,4,5,6,7,8,9};
        List ar=Arrays.asList(myArray);
        Collections.reverse(ar);
        Iterator it=ar.iterator();
        while (it.hasNext()){
            System.out.print(it.next());
        }
    }
}
```

What ist true about the above program? Choose one.

(Was trifft für obiges Programm zu? Wählen Sie eine Antwort)

- [A] Some output like [I@42e816
- [B] 3
- [C] 0123456789
- [D] Compilation error
- [E] Runtime error
- [F] 9876543210

**Lösung**

A ist richtig. Weil `int[]` keine Objekte, sondern nur primitive Variablen speichert, werden undefinierte Werte (von Speicherstellen) ausgegeben. Abhilfe schafft man, indem man für das Array `Integer` anstatt `int` verwendet.

```
import java.util.*;
public class IteratorTest{
    public static void main(String [] args){
        Integer myArray[] ={0,1,2,3,4,5,6,7,8,9};
        //int myArray[] ={0,1,2,3,4,5,6,7,8,9};
        List ar=Arrays.asList(myArray);
        Collections.reverse(ar);
        Iterator it=ar.iterator();
        while (it.hasNext()){
            System.out.print(it.next());
        }
    }
}
```

```
}  
}  
}  
Output  
9876543210
```

### Aufgabe 13 (2-5 Exceptions werfen)

Given:

```
public class ExceptionTest{  
    public static void main(String [] args){  
        Exception e1=new Exception();  
        ArithmeticException ae=new ArithmeticException();  
        try{  
            throw e1;  
        }  
        catch (Exception ex){  
            System.out.println("Exception caught");  
            try {  
                throw ae;  
            }  
            catch (ArithmeticException aex){  
                System.out.println("ArithmeticException caught");  
            }  
        }  
    }  
}
```

Which statements are true? Choose two answers.

(Welche Aussagen treffen zu? Wählen Sie zwei Antworten.)

- [A] Code compiles and runs.
- [B] Compilation error
- [C] RuntimeException
- [D] Code generates the following compiler error message »ArithmeticException has already been caught«
- [E] Runs and writes »Exception caught« and »ArithmeticException caught«

### Lösung

A und E sind korrekt. Es darf eine neue spezifischere Exception, hier eine ArithmeticException, nach einer allgemeinen Exception geworfen werden, sofern das

»Werfen« und »Fangen« ineinander abgeschlossen ist, also zwei vollständige try-catch Konstrukte vorliegen.

Zur Fehlermeldung »ArithmeticException has already been caught« kommt es, falls innerhalb eines einzigen try-catch Konstrukt, also einem try mit mehreren catches, die Reihenfolge des Abfangens von spezifisch zu allgemein nicht eingehalten wird, wie folgendes Programm zeigt:

```
public class ExceptionTest1{
    public static void main(String [] args){
        Exception e1=new Exception();
        ArithmeticException ae=new ArithmeticException();
        try{
            throw e1;
        }
        catch (Exception ex){
            System.out.println("Exception caught");
        }
        catch (ArithmeticException aex){
            System.out.println("ArithmeticException caught");
        }
    }
}
```

Output

```
ExceptionTest1.java:11:5: exception java.lang.ArithmeticException has already
been caught.
```

Die Instantiierung der Exception und ArithmeticException ausserhalb von try-catch ist legal. Das Werfen eine Exception ausserhalb eines try-catch Konstruktes führt zu einer Fehlermeldung »unreachable statement«. Dies bedeutet, dass der Code nach throw ei niemals ausgeführt wird. Die gleiche Wirkung hätte ein return an der selben Stelle.

```
public class ExceptionTest2{
    public static void main(String [] args){
        Exception e1=new Exception();
        throw e1;
        try{
            System.out.println("Error");
        }
        catch (Exception ex){
            System.out.println("Exception caught");
        }
    }
}
```

```

}
}
Output
ExceptionTest2.java:5:5: unreachable statement
    try{
      ^
ExceptionTest2.java:4:7: unreported exception java.lang.Exception; must be
caught or declared to be thrown
    throw e1;
      ^
2 errors

```

### Aufgabe 14 (2-4 Kurzschlussoperatoren)

Given:

```

class ShortCut{
    public static void main(String... args) {
        byte x=2,y=4;
        if ((x++==2) && (y-- ==4)) x++;
        if ((x--==1) || (y++ ==3)) x++;
        System.out.println(x+" "+y);
    }
}

```

What is the output (select one)?:

(Wie lautet der Output (wählen Sie eine Antwort)?)

- [A] 4 3
- [B] 4 4
- [C] 3 4
- [D] 2 4
- [E] 2 3

#### Lösung

B ist richtig.

Bei einem Postincrement, z.B. `y++`, wird der Wert erst beim nächsten Aufruf der Variable dekrementiert. Deshalb sind beide `if`-Bedingungen wahr.

Verfolgen wir den Werdegang des `x`:

```

x=2
x++ => x=3

```

```
x++ => x=4  
x-- => x=3  
x++=>x=4
```

Werdegang für y:

```
y=4  
y=3  
y=4
```

Die eigentliche Falle bei Kurzschluss-Operatoren, nämlich dass, falls der erste Term des Arguments false ergibt, der zweite Term nicht ausgeführt wird, kommt in dieser Frage nicht vor. Das folgende Beispiel zeigt den Sachverhalt:

```
class ShortCut{  
    public static void main(String... args) {  
        byte x=2,y=4;  
        if ((x++==3) && (y-- ==4)) x++; //x++ wird ausgeführt y-- nicht =>x=3, y=4  
        if ((x--==1) || (y++ ==3)) x++;//=> x=2, y=5  
        System.out.println(x+ " "+y);  
    }  
}  
Output  
2 5
```

### Aufgabe 15 (2-2 Labeled breaks)

Given:

```
class BreakTest {  
    public static void main(String[] args){  
        int [] myArray={1,2,3,4,5,6,7,8,9};  
        for (int i=0;i<3;i++){  
            myLabel:  
            for(int j=0;j<myArray.length;j++){  
                if (myArray[j] % 2==0) break myLabel;  
                System.out.println(myArray[j]);  
            }  
        }  
    }  
}
```

What is the output (select one)?:

(Wie lautet der Output (wählen Sie eine Antwort)?)

- [A] I
- [B] II
- [C] III
- [D] no output
- [E] 1234

### Lösung

C ist korrekt. Die innere Schleife wird nach dem Break endgültig verlassen. Die äußere startet die innere Schleife drei Mal.

Die Bedingung `myArray[j] % 2` ist nicht zu verwechseln mit `j % 2`. Da `myArray[0] = 1` ist, wird 1 ausgegeben. `j % 2` wäre zu Beginn 0, es gäbe keinen Output.

### Aufgabe 16 (2-3 Assertions)

Given:

```
class AssertTest{
    static int value=10;

    public static void main(String [] args){
        AssertTest at=new AssertTest();
        at.myMethod(value);
    }
    private boolean myMethod(int x){
        assert(x<--value):"uups";
        return false;
    }
}
```

What is the output if the program is started by the command-line invocation `java -ea AssertTest`? Select one Answer.

(Wie lautet ist der Output, falls das Programm über folgende Kommandozeile gestartet wird: `java -ea AssertTest` ? Wählen Sie eine Antwort)

- [A] no output, no error
- [B] compilation error
- [C] an assertion error is thrown
- [D] assertions are not activated, no output

**Lösung**

C ist korrekt. Eine Assertion wird ausgelöst.

Prinzip: Es wird nur etwas ausgegeben, falls der boolesche Ausdruck des Assert-Statements *false* ist und Assertions »enabled« (mit `-ea`) sind.

Der Output lautet:

```
Exception in thread "main" java.lang.AssertionError: uups
    at AssertTest.myMethod(AssertTest.java:9)
    at AssertTest.main(AssertTest.java:6)
```

**Aufgabe 17 (2-6 Fehlerkategorien)**

Given:

Exception	Category runtime/ error/ checked	Triggered by	
		JVM	Program
ArrayIndexOutOfBoundsException			
ClassCastException			
IllegalArgumentException Exception			
IllegalStateException			
NullPointerException			
NumberFormatException			
AssertionError			
ExceptionInInitializer Error			
StackOverflowError			
NoClassDefFoundError			
IOException			
EOFException			
FileNotFoundException			
Exception			

Use drag & drop to fill in the table. The meaning of the category is:  
 runtime = unchecked runtime exceptions  
 checked = checked exceptions  
 error = errors  
 x = triggered by JVM or Program

(Verwenden Sie *Drag & Drop*, um die Tabelle zu ergänzen. Die Bedeutung der Kategorie ist:

runtime = ungecheckte Laufzeitfehler  
 checked = gecheckte Fehler  
 error = Fehler  
 x= ausgelöst durch JVM oder Programm)

- 
- 
- 
- 

**Lösung**

Exception	Category runtime/ error/ checked	Trigger	
		JVM	Program
ArithmeticException	runtime	x	
ArrayIndexOutOfBoundsException*	runtime	x	
ClassCastException*	runtime	x	
IllegalArgumentException Exception*	runtime		x
IllegalStateException*	runtime		x
NullPointerException	runtime	x	
NumberFormatException Exception*	runtime		x
AssertionError*	error		x
ExceptionInInitializer Error*	error	x	
StackOverflowError*	error	x	

Exception	Category runtime/ error/ checked	Trigger	
		JVM	Program
NoClassDefFoundError*	error	x	
IOException	checked	x	
EOFException	checked	x	
FileNotFoundException	checked	x	

Die Exceptions gemäß der Auflistung des Prüfungsziels 2-6 sind mit einem \* markiert. Die ArithmeticException wurde deshalb in der Fragestellung weggelassen. Selbstverständlich gibt es noch mehr Exceptions, doch hier werden nur genau jene behandelt, die in den Prüfungszielen erwähnt werden.

Nicht ganz einfach ist es, eine Struktur in diese Begriffsvielfalt zu bringen:

Die meisten Exception werden durch die JVM ausgelöst, sind also zur Kompilierzeit nicht feststellbar. Es gibt vier wesentliche, durch das Programm ausgelöste Exceptions: IllegalArgumentException, IllegalStateException, NumberFormatException und AssertionError.

Die drei Checked-Exceptions sind alle mit Datei-IO verbunden.

Errors gibt es nur vier, sie haben alle »Error« im Namen.

Alle restlichen Exceptions sind ungecheckte Laufzeitfehler.

### Aufgabe 18 (2-5 Eigene Exception)

Given:

```
class MyError extends Error{
    double pressure;
    MyError(double p){
        pressure=p;
    }
    public String toString(){
        return "MyError: Pressure is "+pressure+", caution";
    }
}
public class OwnExceptionTest {
    static void treshold(double p) throws MyError{
        if (p>20) throw new MyError(p);
    }
}
```

```
System.out.println("Normal state, pressure is below 20 bars");
}
public static void main(String[] args) {
try{
    treshold(10);
    treshold(30);
}
catch (MyError e){
    System.out.println("Error "+e);
}
}
}
```

Which statements are true (choose two)?

(Welche Aussagen treffen zu? Wählen Sie zwei.)

- [A] The program does not compile because of the use of Error instead of Exception.
- [B] The program compiles and throws one error.
- [C] The program compiles and throws two exceptions.
- [D] The output is

```
Normal state, pressure is below 20 bars
Error MyError: Pressure is 30.0, caution
```

- [E] The output is

```
Normal state, pressure is below 20 bars
Normal state, pressure is below 30 bars
```

### Lösung

B und D sind richtig. Es handelt sich um einen selbst definierten Error. Das Programm reagiert ähnlich wie bei einem AssertionError.

Der Output lautet:

```
Normal state, pressure is below 20 bars
Error MyError: Pressure is 30.0, caution
```

Ob im obigen Beispiel Error oder Exceptions verwendet wird, spielt keine Rolle.

**Aufgabe 19 (2-5 try catch)**

Given:

```
import java.io.Console;
public class TryCatchTest{
public static void divisionDurchNull() throws ArithmeticException{
    int i=10;
    int k=0;
    i=i/k;
}
public static void main(String[] args) {
    Console c=System.console();
    try{
        c.printf("vor Exception\n");
        divisionDurchNull();
        c.printf("nach Exception\n");
    }
    catch (ArithmeticException ae){
        c.printf(ae.getMessage()+"\n");
    }
    finally {c.printf("finally\n");
    }
    c.printf("Ende\n");
}
}
```

What is the result? Choose one.

(Welches ist das Resultat? Wählen Sie eine Antwort.)

 [A]

```
vor Exception
nach Exception
/ by zero
finally
Ende
```

 [B]

```
vor Exception
/ by zero
finally
Ende
```

[C]

```
vor Exception
/ by zero
nach Exception
finally
Ende
```

[D]

```
vor Exception
/ by zero
finally
```

### Lösung

B ist korrekt.

Nur der Ausdruck nach dem Aufruf der Exception (»nach Exception«) wird nicht ausgeführt. »Ende« wird ausgegeben, weil die `ArithmeticException` im `catch`-Block behandelt wurde.

### Aufgabe 20 (2-5 Mehrfache Exceptions fangen)

Given:

```
import java.io.Console;
public class TryCatchTest{
    public static void divisionDurchNull() throws ArithmeticException{
        int i=10;
        int k=0;
        i=i/k;
    }
    public static void main(String[] args) {
        Console c=System.console();
        try{
            c.printf("1 ");
            divisionDurchNull();
            c.printf("2 ");
        }
        catch (ArithmeticException ae){
            c.printf("3 ");
            try{
                throw new Exception();
                //c.printf("4 ");//#1
            }catch (Exception e){
                c.printf("5 ");
            }
        }
    }
}
```

```

    }finally{
        c.printf("6 ");
    }
}
finally {c.printf("7 ");
}
c.printf("8 ");
}
}

```

Which statements are true (choose two)?

(Welche Aussagen treffen zu? Wählen Sie zwei.)

- [A] Output is 1 3 5 6 7 8
- [B] Output is 1 2 3 5 6 7 8
- [C] Output is 1 2 3 4 5 6 7
- [D] if c.printf("4 "); at #1 would be uncommented, the compilation error »unreachable statement« occurs.
- [E] if c.printf("4 "); at #1 would be uncommented there would be no influence on the program.

### Lösung

A und D sind korrekt. Der Output 1 3 5 6 7 8 erklärt sich wie folgt: Die Ausgabe 2 folgt direkt nach dem ein Fehler geworfen wurde. 4 ist auskommentiert.

Wird der Kommentar bei Zeile bei #1 entfernt, resultiert der Kompilierfehler »unreachable statement«. Der Ausdruck »throw new Exception()« hat die gleiche Wirkung wie ein return. Nachfolgende Ausdrücke werden nicht mehr ausgeführt, ausser diejenigen, die im nachfolgenden catch-Block stehen.

### Aufgabe 21 (2-1 Rechnen mit Short und Byte)

Given:

```

import java.io.Console;
public class HoneyTrap{
    public static void main(String[] args){
        Console c=System.console();
        short x=1;
        byte y=2;
        byte sum=0;
        do

```

```
++x;
while(false);
sum=x--+y;
c.printf("%7d",sum);
}
}
```

What is the result? Choose one Answer.

(Wie lautet das Resultat? Wählen Sie eine Antwort)

- [A] Output is 3
- [B] Output is 4
- [C] Compilation error
- [D] Runtime error

### Lösung

C ist richtig, Fehlermeldung »possible loss of precision«. Rechnet man mit short und byte, muss der resultierende Typ ein int sein. Falls man *byte sum* durch *int sum* ersetzt, resultiert 4.

## 10.4.3 Prüfungsziel 3

### Aufgabe 22 (3-1, parseX(), valueOf(), typValue(), toString() )

Given:

```
import java.io.*;
public class ParseIntValueOf{
    public static void main(String[] args){
        Boolean b=true;
        Console c=System.console();
        if(Boolean.valueOf("True"))
            c.printf("only work ");
        if(b.booleanValue())
            c.printf("and no play ");
        if(b.parseBoolean("true"));
        c.printf("makes Jack ");
        if (Boolean.valueOf(b.toString()))
            c.printf("a dull boy");
        }
    }
```

Which statements are true? Choose four correct answers.

(Welche Aussagen treffen zu? Wählen Sie vier Antworten, die korrekt sind.)

- [A] Result: only work and no play makes Jack a dull boy
- [B] Result: and no play makes Jack a dull boy
- [C] valueOf() converts String to boolean
- [D] valueOf() converts String to Boolean
- [E] booleanValue() converts Boolean to Boolean
- [F] booleanValue() converts Boolean to boolean
- [G] parseBoolean() converts String to Boolean
- [H] parseBoolean() converts String to boolean

### Lösung

A, D, F und H *sind* korrekt.

Der gesamte Satz wird ausgegeben.

Eselsleiter: Alle Umwandlungsmethoden, die einen Typnamen enthalten, wandeln in einen Primitivtyp um. Irrtümer bei der Zuweisung sind ab Java Version 1.5 nicht mehr zu erkennen, weil ein boolean in ein Boolean und umgekehrt per Autoboxing und Autounboxing gewandelt wird, so sind valueOf() und parseInt() ab Version 1.5 gleichwertig.

*valueOf()* konvertiert String in einen Boolean (kein Typ-Name im Namen).

*booleanValue()* konvertiert einen Boolean in einen boolean (Typ-Name im Namen).

*parseBoolean()* konvertiert einen String in einen boolean (Typ-Name im Namen).

*toString()* konvertiert einen Boolean in einen String.

### Aufgabe 22a (3-2, File)

```
import [ ];
public class Navigation{
    public static void main(String[] args){
        File directory= [ ];
        [ ] filter=new [ ](){
            public boolean [ ]{
                return name.startsWith("N");
            }
        };
        [ ] array= directory.[ ];
```

```

for(File f:array)
  if (f. )
    System.out.println(f.getName());
  }
}

```

Drag & drop the elements into the slots.

(Ziehen Sie die Elemente in die Lücken.)

*Lösung*

```

import java.io.*;
public class Navigation{
  public static void main(String[] args){
    File directory=new File(".");
    FilenameFilter filter=new FilenameFilter(){
      public boolean accept(File dir,String name){
        return name.startsWith("N");
      }
    };
    File [] array= directory.listFiles (filter);
    for(File f:array)
      if (f.isFile())
        System.out.println(f.getName());
  }
}

```

### Aufgabe 23 (3-2, File)

What are the characteristics of the class `java.io.File`? Choose four correct answers.

Assume `f` is a valid reference to a `File` object.

(Was sind die Eigenschaften der Klasse `java.io.File`? Wählen Sie vier korrekte Antworten. Nehmen Sie an, `f` sei eine gültige Referenz auf ein `File`-Objekt.)

- [A] `f.mkdir()` creates a directory with the name given in `f`
- [B] `f.mkdir()` returns `false` if the directory already exists
- [C] `f.mkdir("\java")` creates the folder `\java`
- [D] `f.mkdir("\\java")` creates the folder `\java`
- [E] `f.delete()` deletes a directory only if it is empty
- [F] `f.createNewFile("datei.txt");` is correct
- [G] The slashes in directories in Windows notation have to be escaped, i.e. `»\«`, Unix slashes `»/«` may be used in Windows, too.

### Lösung

A, B, E und G sind korrekt.

Außer `renameTo()`, `list()` und `listFiles()` haben die Methoden von `File` durchwegs keine Parameter, der Dateiname ist indirekt im Objekt `File` enthalten. `f.mkdir(parameter)` und `f.createNewFile(parameter)` sind deshalb falsch. Falls in Windows Backslashes verwendet werden, müssen diese escaped werden. Unix Slashes sind auch unter Windows erlaubt und müssen nicht escaped werden.

### Aufgabe 24 (3-1, StringBuilder)

Given:

```
1 public class StringBufferTest{
2     public static void main(String[] args){
3         StringBuffer sb=new StringBuffer("123456789");
4         sb.insert(5,"abcdefghik");
5         sb.delete(7,10);
6         String sb1=sb.substring(3,9);
7         System.out.println(sb1.indexOf("f"));
8     }
9 }
```

What is the output?

(Wie lautet der Output?)

- [A] 3
- [B] 4
- [C] 5
- [D] Compiler error in line 6

**Lösung**

B ist korrekt. In den Kommentaren des untenstehenden Listings sehen Sie, wie die Lösung zustande gekommen ist. Die Position der einzelnen Buchstaben wird immer von 0 aus gerechnet.

Wird ein Bereich *von bis* angegeben, ist der von-Wert immer enthalten, aber nicht der bis-Wert. `delete(7,10)` löscht Stelle 7 bis 9 und `substring(3,9)` kopiert den String von Stelle 3 bis 8.

```
public class StringBufferTest{
    public static void main(String[] args){
        StringBuffer sb=new StringBuffer("123456789");
        sb.insert(5,"abcdefghik");
        System.out.println(sb); //12345abcdefghik6789
        sb.delete(7,10);
        System.out.println(sb); //12345abfghik6789
        String sb1=sb.substring(3,9);
        System.out.println(sb1); //45abfg
        System.out.println(sb1.indexOf("f"));
    }
}
```

**Aufgabe 25 (3-2, Console)**

Given:

```
import java.io.Console;
public class Passwort{
    public static void main(String[] args) {
        Console c=System.console();
        char[] password=c.readPassword("%s", "Password: ");
        for (char ch:password)
            c.printf("%c", ch);
    }
}
```

What is prompted on the screen while typing in the password? Assume the password to enter is »test«. Choose one.

(Was wird am Bildschirm ausgegeben, während man das Passwort eingibt? Nehmen Sie an, das Passwort hieße »test«. Wählen Sie eine richtige Antwort.)

- [A] Password: \*\*\*\*
- [B] Password:
- [C] Password: test
- [D] Runtime error

### Lösung

B ist korrekt. Während des Eingebens des Passworts wird nichts ausgegeben.

Beachten Sie, dass an der Prüfung auch eine ganz normale Eingabe mit `Console.ReadLine()` geprüft werden kann. `ReadLine()` liest einen String ein und keinen char-Array.

### Aufgabe 26 (3-5, Reguläre Ausdrücke)

Given:

```
import [ ];
public class MinimalRegex{
    public static void main(String[] args){
        Pattern p = Pattern.[ ];
        Matcher m = p.[ ];
        System.out.println("The Pattern: "+ m.[ ] );
        while (m.find()){
            System.out.println("Start: "+ [ ] +" found: "+ [ ] );
        }
    }
}
```

Position the elements so that the program compiles and runs.

(Positionieren Sie die Elemente so, dass das Programm kompiliert und läuft.)

m.start()  
 compile("-")  
 java.util.regex.\*  
 java.util.\*  
 pattern()  
 matcher("Hello-nice-world")  
 m.group()

### Lösung

```
import java.util.regex.*;
public class MinimalRegex{
    public static void main(String[] args){
        Pattern p = Pattern.compile("-");
        Matcher m = p.matcher("Hello-nice-world");
        System.out.println("The Pattern: "+ m.pattern());//#5
        while (m.find()){
            System.out.println("Start: "+m.start()+" found: "+ m.group());
        }
    }
}
```

```
}  
}  
}  
Output:  
The Pattern: -  
Start: 5 found: -  
Start: 10 found: -
```

### Aufgabe 27 (3-3, Formatter)

Given:

```
import java.util.*;  
public class FormatterTest{  
    public static void main(String[] args){  
        Formatter fo=new Formatter(Locale.US);  
        Double x=new Double(123.456789);  
        fo.format("%.2F",x);  
        double y=Float.parseFloat(fo.toString());  
        if (y > 123.45)  
            System.out.println(fo);  
        else  
            System.out.println("smaller than 123.45");  
        StringBuffer sb=new StringBuffer();  
        Formatter fo1=new Formatter(sb,Locale.GERMANY);  
        fo1.format("%.3F",124.5678910);  
        System.out.println(fo1);  
        StringBuffer sb1=(StringBuffer)fo1.out();  
        System.out.println(sb1);  
    }  
}
```

What is the output? Choose one:

(Wie lautet der Output? Wählen Sie eine Antwort.)

[A]

```
123.44  
Smaller than 123.45  
124,568
```

[B]

```
123.46  
124,568  
124,568
```

[C]

```
123.46  
124.568  
124.568
```

[D] Laufzeitfehler

### Lösung

B ist korrekt. Die zweite und dritte Zahl sind im deutschen Format mit Komma formatiert.

```
123.46  
124,568  
124,568
```

### Aufgabe 28 (3-1, Serialisierung)

Which statements are true?

(Welche Aussagen treffen zu?)

- [A] The constructors of the serialized objects do not run when they are deserialized. But the parent constructor does, in case the parent object was not serialized.
- [B] A class to be serialized must implement Serializable. If the parent class implements Serializable, the child class does not need to implement Serializable to be serialized.
- [C] If you mark a class's instance variable transient, it will persist serialization.
- [D] If you mark a class's instance variable volatile, it will be lost during serialization.
- [E] If you mark a class's instance variable static, it will persist serialization.
- [F] If you serialize an object which contains another object as member, the member object has to implement Serializable, too.

### Lösung

A, B und F sind korrekt. A: Der Konstruktor des serialisierten Objekts läuft nicht, da sonst die deserialisierten Werte mit den Anfangswerten überschrieben würden. B: Es reicht aus, falls in der Vererbungshierarchie einmal Serializable implementiert wurde. F: Wird eine Komposition serialisiert, müssen die Komponenten auch Serializable implementieren.

Alle anderen sind falsch bzw. genau umgekehrt wäre die Antwort korrekt: C: Transiente Variablen werden nicht serialisiert. D: Volatile Variablen werden serialisiert. E: Statische Variablen werden nicht serialisiert, weil sie nicht zum Objekt sondern zur Klasse gehören.

### Aufgabe 29 (3-4, Formatter und DateFormat)

Given:

```
import java.util.*;
import java.text.*;
public class FormatterTest{
    public static void main(String[] args){
        Date d=new Date();
        Calendar cal=Calendar.getInstance();
        cal.setTime(d);

        Formatter fo=new Formatter(Locale.GERMANY);
        fo.format("%td. %tB %tY",cal,cal,cal);
        System.out.println(fo);

        DateFormat df = DateFormat.getDateInstance(DateFormat.LONG, Locale.GERMANY);
        System.out.println(df.format(d));
    }
}
```

Which statements are correct if today is September 23, 2008?

(Welche Aussagen sind korrekt falls das heutige Datum der 23. September 2008 ist?)

- [A] Formatter is an older class than DateFormat.
- [B] The output is:

```
23. September 2008
23. September 2008
```

- [C] The output is:

```
23. 9.2008
23. 9.2008
```

- [D] Formatter is more universal than DateFormat, as it can format currencies too.

### Lösung

B und D sind korrekt. DateFormat wird durch die universellere Formatter-Klasse abgelöst. Der Output beider Methoden ist 23. September 2008.

### Aufgabe 30 (3-5, Formatter und NumberFormat)

```
import java.util.*;
import java.text.*;
public class FormatterTest{
    public static void main(String[] args){
        double d=1000.50;
        Formatter fo=new Formatter(Locale.US);
        fo.format("%$, .2f",d);
        System.out.println(fo);
        NumberFormat nf = NumberFormat.getCurrencyInstance(Locale.US);
        System.out.println(nf.format(d));
    }
}
```

Which statements are correct?

(Welche Aussagen sind korrekt?)

- [A] Formatter existed before NumberFormat.
- [B] The output is:

```
$1000.50
$1000.50
```

- [C] The output is:

```
$1,000.50
$1,000.50
```

- [D] Formatter is more universal than NumberFormat, as it can format dates, too.

### Lösung

C und D sind korrekt. NumberFormat ist älter als Formatter.

### Aufgabe 31 (3-5, Reguläre Ausdrücke)

Which pattern finds -xy@ and -xyzi@ und -xyzi@ in a String? Chosse one correct answers.

(Welches Muster findet -xy@ und -xyz@ und -xyzi@ in einem String? Wählen Sie eine korrekte Antwort)

- [A] `-[xyzi]+ @`
- [B] `-xy[zi]+@`
- [C] `-(xyzi)+@`
- [D] `-xy(zi)+@`
- [E] `-xyz@`

**Lösung**

A ist korrekt. `[]` bedeutet ein einziges Zeichen, das x,y,z und i annehmen kann. Das `+` bedeutet einmaliges oder mehrmaliges Vorkommen.

B: Es werden `»-xyz@«` und `»-xyzi@«` gefunden. `»-xy@«` kann nicht gefunden werden, weil dazwischen noch ein `»z«` oder eine `»i«` stehen muss.

C: `(xyzi)+` bedeutet eine feste Abfolge, die einmal oder mehrmals gefunden wird (`+`). Daher wird nur `»xyzi@«` gefunden.

D: `(zi)+` ist ebenfalls eine feste Abfolge, es wird auch nur diese mit dem voran stehenden `»-xy«` und dem folgenden `»@«` gefunden.

E: findet nur `»-xyz@«`

```
import java.util.regex.*;
public class Regular1 {
    public static void main(String... args){
        String s="-xy@m und -xyz@ und -xyz1@ ";

        String muster="-[xyz1]+@";//a) findet -xy@ -xyz@ -xyz1@
        //String muster="-xy[zi]+@";//b) findet -xyz@ und -xyz1@
        //String muster="-(xyz1)+@";//c) findet -xyz1@
        //String muster="-xy(zi)+@";// d) findet -xyz1@
        //String muster="-xyz@";//e)findet -xyz@

        Pattern p = Pattern.compile(muster);
        Matcher m = p.matcher(s);
        boolean b = false;
        System.out.println("Das Muster: "+ m.pattern());
        while (b = m.find()){//find() ist true, falls ein Treffer vorhanden ist
            System.out.println("Start: "+m.start()+" gefunden: "+ m.group());
            //start() = Anfangsposition, group() = gefundener Teilstring
        }
    }
}
```

## 10.4.4 Prüfungsziel 4

### Aufgabe 32 (4-1, Thread erzeugen)

Given:

```
public class ThreadStart implements Runnable{
    public void run(){
        System.out.println("I work");
    }
    public static void main(String [] args){
        //insert here
        t.start();
    }
}
```

and the following code snippets:

(und die folgenden Code-Schnipsel:)

- [A] Thread t =new Thread(new ThreadStart());
- [B] Thread t=new Thread();
- [C] ThreadStart=new ThreadStart();
- [D] Thread t=new ThreadStart();

Which ones, inserted independently at `//insert here` will compile, run and output »I work«? Choose one answer.

(Welche bei `//insert here` kompilieren, laufen und geben »I work« aus? Wählen Sie eine Antwort.)

### Lösung

A ist richtig. B kompiliert zwar, gibt aber keinen Output. C und D kompilieren nicht: C nicht, weil die Methode `start()` nicht in `ThreadStart()` gefunden wird, und D nicht, weil die Typen `Thread` und `ThreadStart` keine Verwandtschaftsbeziehung aufweisen.

### Aufgabe 33 (4-1, Threads erzeugen)

If you create a class `TestThread` that instantiates a thread in the main method, which statements are true, if you want to run a thread which generates an output? Choose two.

(Falls Sie eine Klasse `TestThread` erzeugen, die einen Thread in der `main()`-Methode erzeugt, welche Aussagen treffen dann zu, falls Sie einen Thread mit einem Output erzeugen wollen? Wählen Sie zwei Antworten)

- A `TestThread` must override `public void run()`.
- B `TestThread` must implement `Thread`.
- C `TestThread` must inherit from `Thread`.
- D `TestThread` must inherit from `Runnable`.
- E None of the above is necessary.

### Lösung

A und C sind korrekt: A ist korrekt, sofern man einen Output erwartet. Über den Vererbungsweg erzeugt, braucht die Methode `run()` nicht unbedingt implementiert zu werden. Hingegen zwingt einen die Schnittstelle, falls der Thread über die Schnittstelle `Runnable` erzeugt wurde, die Methode `run()` zu implementieren.

### Aufgabe 34 (4-2, join)

Given:

```

1 public class JoinTest1 implements Runnable{
2     private static String s="";
3     public void run(){
4         //try{Thread.sleep(1000);} catch (InterruptedException e){}
5         for (int i=0;i<50;i++){
6             s+="r";
7         }
8     public static void main(String... args){
9         Thread t = new Thread(new JoinTest1());
10        t.start();
11        for (int i=0;i<50;i++){
12            s+="m";
13            try{t.join();} catch (InterruptedException e){}
14        }
15        System.out.println(s);
16    }
17}

```

What is the difference, if you run this example without and with a call to `join()`? What will happen, if you uncomment `sleep()` in line 4? Which two are correct?

(Was ist der Unterschied, falls Sie dieses Beispiel einmal mit und einmal ohne Aufruf von `join()` laufen lassen? Was passiert, falls Sie `sleep()` in Zeile 4 auskommentieren? Welche beiden Aussagen sind korrekt?)

- A With `join()` the `main()`-thread runs more.
- B With `join()` the `main()`-thread runs less.
- C `join()` has no influence on the output.
- D `join()` sets the calling thread (here `main`) in a blocked-for-join state.
- E If you activate `sleep(1000)` and deactivate line 13, there would be more outputs of `r`, as the `run` thread gets more runtime.

### Lösung

B und D sind korrekt. `join()` setzt den aufrufenden Thread in einen Blocked-for-join-Zustand. Deshalb hat der `run`-Thread mehr Laufzeit zur Verfügung.

Wird Zeile 13 (`join`) deaktiviert und Zeile 4 aktiviert (`sleep(1000)`), läuft nur noch der `Main`-Thread, und es werden nur `m` ausgegeben.

### Output

```
//ohne join mmmmmmmmmmmrrrrmr...
//mit join mrrrrrrrrrrrrrrrrrrr...
//ohne join, mit sleep(2000):mmmmmmmmmmmmmmmmmmmm...
```

### Aufgabe 35 (4-3, yield)

Given:

```
public class YieldTest implements Runnable{
private static long max=1000;
public void run(){
for (long i=0;i<max;i++){
if (i%(max/10)==0){
System.out.print(i+"r ");
//Thread.yield();
}
}
}
public static void main(String... args){
Thread t = new Thread(new YieldTest());
t.start();
for (long i=0;i<max;i++){
if (i%(max/10)==0)
System.out.print(i+"m ");
//Thread.yield();
}
}
}
```

Which statements are true? Choose three.

(Welche Aussagen treffen zu? Wählen Sie drei.)

- A With `Thread.yield()` commented out the Output is evenly distributed.
- B The thread, in which `Thread.yield()` is activated, has less opportunities to run.
- C The thread, in which `Thread.yield()` is activated, has more opportunities to run.
- D The output consists of 10 numbers (0, 100, ...).

### Lösung

A, B und D sind korrekt. A: Ohne `yield()` laufen Main- und Run-Thread etwa gleichmäßig. B: Der Thread, der `yield()` aufruft, lässt einem anderen Thread den Vortritt. D: Wegen der Modulo-Berechnung werden nur Zahlen in Hunderter-schritten ausgegeben.

C stimmt nicht, da der Thread, der `yield()` aufruft, weniger Laufzeit erhält.

### Aufgabe 36 (4-3, Synchronized)

Given a thread with a synchronized `run()` method:

```
public class SyncRun implements Runnable {
    public synchronized void run() {
        long t=Thread.currentThread().getId();
        for (int i=0;i<5;i++){
            if(i==2) try{wait();}catch(InterruptedException e){}
            System.out.println(t+" "+i);}
        }
    public static void main(String[] args) {
        new Thread(new SyncRun()).start();
        new Thread(new SyncRun()).start();
    }
}
```

What is the output? Choose one.

(Wie lautet der Output? Wählen Sie eine Antwort.)

- [A]

```
8 0
8 1
8 2
```

```
9 0
9 1
9 2
```

[B]

```
8 0
9 0
8 1
9 1
```

[C]

```
8 0
8 1
9 0
9 1
```

[D]

```
8 0
9 0
```

[E] Runtime error

[F] Compile error

### Lösung

C ist korrekt. `getId()` bezeichnet einen Thread eindeutig und beginnt mit einer Zahl von 6 bis 9. Hier ist es die Zahl 8.

Der erste Thread mit der Nummer 8 besitzt den Lock und arbeitet alles ab, bevor der zweite starten kann. Dadurch, dass `wait()` bei `i=2` aufgerufen wird, wartet ein gestarterter Thread nach zwei Ausgaben ewig.

### Aufgabe 37 (4-4, Wait und Notify)

Given:

```
class MyThread extends Thread{
    int sum=0;
    public void run(){
        synchronized(this){
            for (int i=0;i<1000;i++){ //producing action
                sum+=i;
            }
            notify();
        }
    }
}
```

```
    }  
    }  
}  
public class MyThreadTest{  
    public static void main(String[] args){  
        MyThread t1=new MyThread();  
        t1.start();  
        synchronized(t1){  
            try{  
                t1.wait(); //waiting for producer  
            } catch (InterruptedException e){}  
            System.out.println(t1.sum);  
        }  
    }  
}
```

Which statements are true for the above code? Choose three.

(Welche Aussagen treffen zu? Wählen Sie drei Antworten.)

- [A] The program outputs the sum of 499500. If the `t1.wait()` in the `main()` method would lack the output could be 0.
- [B] `try{Thread.sleep(2000);}catch (InterruptedException e){}` could be used in the `main()` method instead the `synchronized` block and the result would be more or less the same.
- [C] `notify()` or `notifyAll()` is usually placed in front of a code which wants to consume.
- [D] `wait()` is usually placed in front of producing code.
- [E] The key of the `synchronized` block has to be the same on which the `wait()` method is called.

### Lösung

A, B und E sind korrekt. Das `wait()` in der `main()`-Methode baut eine Verzögerung ein, in dem es wartet, bis die Summe berechnet worden ist.

A: Falls die Verzögerung fehlt, wird 0 ausgegeben, da noch nichts produziert wurde und nicht auf die Produktion gewartet wurde.

B: Die Verzögerung kann auch mit einem `Thread.sleep(2000)` erreicht werden, sofern die Produktion innerhalb 2 Sekunden vollständig ist. Die Methode mit `wait()` ist aber eleganter, weil genau solange gewartet wird, bis die Berechnung beendet wird.

E: Der Schlüssel des `synchronized`-Blocks muss die Objektreferenz sein, auf die die `wait()`-Methode aufgerufen wurde. Sonst ereignet sich eine `IllegalMonitorStateException`.

C und D sind falsch. `notify()` kommt nach einem produzierenden Code und `wait()` vor den konsumierenden. Die Struktur eines Producer-Consumer-Programms ist:

```
Produzierender Code (lange Schleife)
this.notify()
```

und

```
rt.wait()
konsumierender Code (Bildschirm-Ausgabe einer großen Zahl)
```

### Aufgabe 38 (4-4, synchronized)

```
public class SomeThreads {
    private int x=0;
    public void increment() {
        x++;//#1
    }
    public void tenThreads() {
        for(int i = 0; i < 10; i++) {
            new Thread() {
                public void run() {
                    increment();
                    System.out.print(x + ", ");
                }
            }.start();
        }
    }
    public static void main(String[] args){
        new SomeThreads().tenThreads();
    }
}
//Output: 10, 10, 10, 10, 10, 10, 10, 10, 10,
```

What two changes are necessary to change the program so that the output is

1, 2, 3, 4, 5, 6, 7, 8, 9, 10,

with a granted sequence? Choose two.

(Welche zwei Änderungen sind im obigen Programm notwendig, das der Output garantiert 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, lautet? Wählen Sie zwei Antworten)

- ❑ [A] Change public void run() to public synchronized run();
- ❑ [B] Move the System.out.print(x+",") to the method increment, below x++ (below #1)
- ❑ [C] Move the System.out.print(x+",") to the method increment, below x++ (below #1). Change public void increment() to public synchronized void increment()
- ❑ [D] Change public void tenThreads() to public void synchronized tenThreads();
- ❑ [E] .Wrap the for loop in public void tenThreads in a synchronized(this){} block.

### Lösung

A und C sind korrekt.

A: Die synchronisierte run() Methode lässt nur ein Thread pro Zeiteinheit zu.

C: Die elegantere Lösung ist, den synchronized Abschnitt möglichst klein zu halten in dem increment() synchronisiert wird und die Ausgabe in increment() verlegt wird (siehe Listing unten).

B: In dem die Ausgabe nach der Inkrementierung erfolgt, wird jeder Schritt ausgegeben. Output: 2, 2, 3, 4, 5, 6, 7, 8, 9, 10,. Aber ohne Synchronisierung von increment() stimmt die Reihenfolge nicht.

D: Die Synchronisation von thenThreads() bringt nichts, da ja die 10 Threads innerhalb dieser Methode gestartet werden.

E: Ein Synchronisationsblock um die for-Schleife hat die gleiche Wirkung wie die Lösung in D.

```
public class SomeThreads1 {
    private int x=0;
    public synchronized void increment() {
//public void increment() {
        x++;
        System.out.print(x + ", ");
    }
    public void tenThreads() {
        for(int i = 0; i < 10; i++) {
            new Thread() {
                public void run() {
                    increment();
                    //System.out.print(x + ", ");
                }
            }
        }
    }
}
```

```
    }.start();  
  }  
}  
public static void main(String[] args){  
  new SomeThreads1().tenThreads();  
}  
}  
//Output:1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
```

### Aufgabe 39 (4-4, IllegalMonitorStateException)

Given:

```
public class PC extends Thread{  
private int s=1;  
public void run(){  
  for (int i=0;i<10;i++){  
    s++;  
    try{Thread.sleep(100);} catch (InterruptedException e){}  
    notify();  
    System.out.print(s+ " ");  
    if (s<1){  
      try{wait();}  
      catch (InterruptedException e){}  
    }  
    else{s--;System.out.print(s+ " ");  
    }  
  }  
}  
public static void main(String[] args){  
  new PC().start();  
}  
}
```

Which statements are true? Choose two.

(Welche Aussagen treffen zu?)

- [A] The program compiles.
- [B] The program does not compile.
- [C] The output is 2 1 2 1 ...
- [D] The output is 1 2 1 2 ...
- [E] An IllegalMonitorStateException is thrown.

**Lösung**

A und E sind richtig. Die Methode `run()` müsste synchronisiert sein:

```
public synchronized void run(){
```

Ist dies der Fall, wird 2 1 2 1 (insgesamt 10 Zahlen) ausgegeben. Es handelt sich hier um ein einfaches Producer-Consumer-Problem, das eigentlich keine interthread Synchronisation benötigen würde, weil nur ein einziger Thread involviert ist.

Eine `IllegalMonitorStateException` wird geworfen, weil `wait()` und `notify()` in einem unsynchronisierten Codeabschnitt aufgerufen wird.

```
public class PC extends Thread{
private int s=1;
public synchronized void run(){
for (int i=0;i<10;i++){
s++;
try{Thread.sleep(100);} catch (InterruptedException e){}
notify();
System.out.print(s+" ");
if (s<1){
try{wait();}
catch (InterruptedException e){}
}
else{s--;System.out.print(s+" ");
}
}
}
public static void main(String[] args){
new PC().start();
}
}
//Output: 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1
```

**Aufgabe 40 (4-1, Thread-Erzeugung)**

Given:

```
1 class ThreadInheritance extends Thread{
2 public void myStart(){
3 super.start();
4 }
5 public void run(){
```

```
6  int i=0;
7  do {
8    System.out.print("Hello ");
9    i++;
10 }while(i<5);
11 }
12 public static void main(String [] args){
13   Thread t=new ThreadInheritance();
14   t.myStart();
15 }
16}
```

What is about the above code? Choose one correct answer.

(Was ist mit obigem Code? Wählen Sie eine korrekte Antwort.)

- [A] Compiles and runs
- [B] Runtime error
- [C] Compilation error because of line 3.
- [D] Compilation error because of line 13.
- [E] Outputs five times »Hello«
- [F] Outputs four times »Hello«

### Lösung

D ist korrekt. `Thread t=new ThreadInheritance();` ist falsch, weil die Referenz `t` die Methode `myStart()` nicht finden kann. Ändert man den Code in Zeile 13 in

```
ThreadInheritance t=new ThreadInheritance();
```

gibt das Programm fünf Mal »Hello« aus.

```
class ThreadInheritance extends Thread{
  public void myStart(){
    super.start();
  }
  public void run(){
    int i=0;
    do {
      System.out.print("Hello ");
      i++;
    }while(i<5);
  }
}
```

```

public static void main(String [] args){
    ThreadInheritance t=new ThreadInheritance();
    t.myStart();
}
}
//Output: Hello Hello Hello Hello Hello

```

#### Aufgabe 41 (4-4, Ausgabereihenfolge)

Given:

```

public class MySequence implements Runnable {
    public synchronized void run(){
        for (int i=0;i<11;i++){
            try{Thread.sleep(1000);} catch (InterruptedException e){}
            System.out.print(i+" ");
        }
    }
    public static void main(String[] args){
        new Thread(new MySequence()).start();
        try{Thread.sleep(1000);} catch (InterruptedException e){}
        new Thread(new MySequence()).start();
    }
}

```

What is the sequence of the output? Choose two.

(Welches ist die Ausgabereihenfolge? Wählen Sie zwei Lösungen.)

- [A] Outputs two times 1 to 10 in a predefined sequence.
- [B] Outputs two times 1 to 10 in a random sequence.
- [C] The synchronized statement in the run-method guarantees an ordered sequence.
- [D] The synchronized statement in the run-method is not enough to guarantee any ordered sequence.
- [E] Runtime error
- [F] Compilation error

#### Lösung

B und D sind korrekt.

Es handelt sich um zwei Threads, die in einem Abstand von einer Sekunde gestartet werden. Jeder Thread hat seine eigene run()-Methode. Deshalb ist der Output ungeordnet, z. B.:

```
0 1 0 1 2 2 3 4 3 4 5 5 6 7 6 7 8 9 8 10 9 10
```

Will man eine zweimalige Sequenz von 1 bis 10 garantieren, ist eine Synchronisation mit einer nur einmalig vorhandenen Referenzvariablen notwendig, die man mit `getClass()` oder mit `.class` erhalten kann:

```
public class MySequence1 implements Runnable {
    public void run(){
        synchronized(MySequence1.class){ //gleichwertig wie MySequence1.getClass()
            for (int i=0;i<11;i++){
                try{Thread.sleep(1000);} catch (InterruptedException e){}
                System.out.print(i+" ");
            }
        }
    }
    public static void main(String[] args){
        new Thread(new MySequence1()).start();
        try{Thread.sleep(1000);} catch (InterruptedException e){}
        new Thread(new MySequence1()).start();
    }
}
```

Output

```
0 1 2 3 4 5 6 7 8 9 10 0 1 2 3 4 5 6 7 8 9 10 0 1 2 3 4 5 6 7 8 9 10
```

## 10.4.5 Prüfungsziel 5

### Aufgabe 42 (5-1, Koppelung)

Given:

```
class Car{
    Wheel [] wheels=new Wheel[4];
    public Car(){
        for (int i=0;i<wheels.length;i++)
            wheels[i]=new Wheel();
        for(Wheel p:wheels)
            p.setPressure(2.5);
    }
}
class Wheel {
    double pressure=2.0;
    public void setPressure(double p){
        this.pressure=p;
    }
}
```

```

}
public class TestC{
public static void main(String...args){
    Car car=new Car();
    System.out.println(car.wheels[0].pressure);
}
}
}

```

What can be said about the above code? Choose three correct answers.

(Was kann über obigen Code ausgesagt werden? Wählen Sie drei korrekte Antworten.)

- [A] It implements tight encapsulation.
- [B] It implements loose encapsulation.
- [C] It implements high cohesion.
- [D] It implements low cohesion.
- [E] It implements tight coupling.
- [F] It implements loose coupling.

### Lösung

B, C und F sind korrekt.

Kapselung: Die Kapselung ist nicht eng, weil *pressure* nicht privat gesetzt ist und keine Getter- und Setter-Methoden vorhanden sind für die Variable *p*.

Kohäsion: C ist korrekt, weil sich jedes Objekt auf seine Aufgabe fokussiert.

Koppelung: F ist korrekt. Koppelung ist das Mass, wie viel von der Klasse A der Klasse B bekannt ist. Die Koppelung ist locker, sofern der Klasse B nur das Interface (die öffentlichen Methoden) bekannt sind. Klasse Car greift nur mit `public setPreassure()` auf Klasse Wheel zu, weshalb die Koppelung locker ist.

### Aufgabe 43 (5-2, final)

Given:

```

1 interface MyInterface{}
2 class A implements MyInterface{public void methodA(){}
3 }
4 public class TestFinal extends A{
5     public void methodB(){}
6     public static void main(String[] args){
7         MyInterface a=new A();

```

```
8   A b=new TestFinal();
9   final A c=new TestFinal();
10  c=b;
11  b.methodA();
12  b.bethodB();
12  }
13}
```

What is the result? Select two.

(Was resultiert? Wählen Sie zwei Antworten.)

- [A] Code does not compile because of line 10.
- [B] Code does not compile because of line 12.
- [C] Runtime Exception.
- [D] Compiles and runs.

### Lösung

A und B sind korrekt.

A: Variable c ist final, ihr kann in Zeile 10 kein neuer Wert zugewiesen werden.

B: *b.methodB()* ist falsch, weil die Variable b vom Typ A ist. Die Referenzvariable b ist vom Typ A und sieht *methodB()* nicht. Sie sieht nur die von A ererbten Methoden.

### Aufgabe 44 (5-3, Vererbung und Modifizierer)

```
class Alpha{
    private static final void print(Float n){
        System.out.print("Float ");
    }
}
public class Beta extends Alpha{
    private static final void print(Integer i){
        System.out.println("Integer ");
    }

    public static void main(String [] args){
        print(3.0f); // #1
        print(3); // #2
        print(3.0); // #3
    }
}
```

What is the result? Choose one.

(Was ist das Resultat? Wählen Sie eine Antwort.)

- [A] The output is Float, Integer, Float.
- [B] The code does not compile because of the call at #1.
- [C] The code does not compile because of the call at #2.
- [D] The code does not compile because of the call at #3.
- [E] The code does not compile because of the call at #1 and #3.

### Lösung

E ist korrekt. `print(3.0f)`: Die Methode wird nicht gefunden, da sie in der Elternklasse `private` deklariert ist (Fehlermeldung: `cannot be applied to float`), weil nur `print()` im aktuellen Objekt gefunden wird (`print()` wird nicht vererbt).

`print(3.0)` wird ebenfalls nicht erkannt (Fehlermeldung: `cannot find method print(double)`).

Der Aufruf bei #1 würde funktionieren, falls `print()` nicht `private` wäre. Der Aufruf bei #3 funktioniert nie, weil keine auf den Typ `double` oder `Double` passende Methode existiert.

### Aufgabe 45 (5-3, Vererbung und Modifizierer)

Given:

```
class Animal{
    public void communicate(){System.out.println("give sound ");}
}
public class Dog extends Animal{
    void communicate(){System.out.println("bark");}
    public static void main(String [] args){
        Animal[] animals={new Animal(), new Dog()};
        for(Animal a:animals)
            a.communicate();
    }
}
```

What is the correct answer?

(Welches ist die korrekte Antwort?)

- [A] Output: give sound bark
- [B] Output: give sound give sound

- [C] Compilation fails
- [D] Runtime error

### Lösung

C ist richtig. Der Zugriffsmodifizierer von `communicate()` in der Klasse `Dog` ist mit `default` eingengerter als der Modifizierer `public`.

```
class Animal{
    public void communicate(){System.out.println("give sound ");}
}
public class Dog extends Animal{
    public void communicate(){System.out.println("bark");}
    public static void main(String [] args){
        Animal[] animals={new Animal(), new Dog()};
        for(Animal a:animals)
            a.communicate();
    }
}
Output
give sound bark
```

### Aufgabe 46 (5-4, Konstruktorverkettung)

Given:

```
class Animal{
    String name;
    Animal(String name){this.name=name;}
}
public class Dog extends Animal{
    public Dog(String name){super(" Fluffi ");}
    public static void main(String [] args){
        Animal[] animals={new Animal(" Tobi "), new Dog(" Miro ")};
        for(Animal a:animals)
            System.out.print(a.name);
    }
}
```

What is the output?

(Welches ist der Output?)

- [A] Tobi Miro
- [B] Tobi Fluffi

- [C] Compilation fails
- [D] Runtime error

### Lösung

B ist korrekt. Im Dog-Konstruktor wird der Elternkonstruktor mit `super()` aufgerufen und der Name »Fluffi« übergeben. Der Name »Miro« wird nicht in die Membervariable `name` gespeichert.

### Aufgabe 47 (5-4, kovariantes Überschreiben)

Given:

```
class Os {
    static String s="";
    Os loadOs(){
        s+="os loaded ";
        return this;
    }
}
public class Vista extends Os{
    Vista loadOs(){
        s+="Vista loaded";
        return this;
    }
    public static void main(String [] args){
        Os o=new Os();Vista v=new Vista();o.loadOs();v.loadOs();
        System.out.println(s);
    }
}
```

What is true for the above code? Choose two correct answers.

(Was trifft für den obigen Code zu? Wählen Sie zwei korrekte Antworten.)

- [A] The code compiles without problems.
- [B] The code does not compile, there is an illegal override.
- [C] Output: os loaded Vista loaded.
- [D] Output: Vista loaded os loaded.
- [E] Code ends in a runtime error.

### Lösung

A und C sind korrekt. Es handelt sich hier um einen korrekten kovarianten Override, bei dem der Rückgabewert der Methode eine Kindklasse ist.

**Aufgabe 48 (5-4, Overriding)**

```
class A{
void myMethod(String s){}
class B extends A{
//insert here
}
```

Which of the following code fragments can be inserted at `//insert here` so that the code compiles? Choose three.

(Welche der folgenden Code-Fragmente können bei `//insert here` eingesetzt werden, damit der Code kompiliert? Wählen Sie drei Antworten.)

- [A] **void** myMethod(String s);
- [B] **protected void** myMethod(String s) **throws** Error{}
- [C] **void** myMethod(String s) {}
- [D] **public int** myMethod(String s) {}
- [E] **public void** myMethod(**int** i) {}
- [F] **abstract void** myMethod(String s){}

**Lösung**

B, C und E sind korrekt.

B: Protected ist weniger restriktiv als default und deshalb für das Überschreiben erlaubt.

C: Die Methode lautet genau gleich wie diejenige in der Elternklasse und damit ist ein Überschreiben erlaubt.

E: Es handelt sich hier um einen Overload, weil die Parameterliste einen int anstatt einen String besitzt.

A: Es fehlt der Methodenkörper.

D: Der Return-Typ falsch, er müsste vom gleichen Typ oder kovariant sein.

F: Es ist erlaubt, eine nicht abstrakte Methode mit einer abstrakten Methode zu überschreiben. Bei F ist aber der Methodenkörper vorhanden, obwohl er fehlen müsste, da eine abstrakte Methode keine Implementierung haben darf.

```
class A{
    void myMethod(String s){}
}
public class B extends A{
```

```
//hier einfügen
//void myMethod(String s); //a) Fehlender Methodenkörper
//protected void myMethod(String s) throws Error{} //b) ok
//void myMethod(String s) {} //c) ok
//public int myMethod(String s) {return 1;} //d) Inkompatibler Return-Typ
//public void myMethod(int i) {} //e) ok (ist Overload)
abstract void myMethod(String s){} // Abstrakte Methoden dürfen keinen Body
//haben
public static void main(String[] args){
}
```

### Aufgabe 49 (5-2, Up- und Downcasting)

Given:

```
class Animal{
    String name;
    public void communicate(){
        System.out.println("Animal sound");
    }
}
class Goose extends Animal{
    public Goose(String name){this.name=name;}
    public void communicate(){System.out.println("quack");}
    public void fly(){System.out.println("I fly away");}
}
class Dog extends Animal{
    public Dog(String name){this.name=name;}
    public void communicate(){System.out.println("bark");}
    public void clean(){System.out.println("cleaning fur");}
}
public class AnimalTest{
    public static void main(String [] args){
        Animal[] animals={new Dog("Miro"),new Goose("Otto")};
        for(Animal a:animals){a.communicate();
            if(a instanceof Dog){Dog b=(Dog)a; b.clean();}
            if (a instanceof Goose){Goose c=(Goose)a; c.fly();}
        }
    }
}
```

What's about the above code? Choose two.

(Was ist mit dem obigen Code? Wählen Sie zwei Antworten.)

- [A] Output:

```
bark
cleaning fur
quack
I fly away
```

- [B] ClassCastException is thrown at runtime
- [C] Compilation error
- [D] Output:

```
cleaning fur
I fly away
```

- [E] Compiles and runs

### *Lösung*

A und E sind korrekt.

In das Array vom Elterntyp Animal werden Kindobjekte gespeichert, was einem Upcast gleichkommt.

Der Downcast, d.h. die Zuweisung der Elternreferenz an eine Kindreferenz erfolgt nach der Kontrolle der Elternreferenz mit Hilfe des instanceof Operators: `if(a instanceof Dog)`. Erst jetzt darf der Downcast erfolgen: `Dog b=(Dog)a` und per Referenzvariable b auf die Methode `clean()` zugegriffen werden.

Ein Downcast benötigt immer einen vorgängigen Upcast.

### **Aufgabe 50 (5-5, Verwandtschaftsbeziehungen)**

```
class Foo{private Fooable f;}
interface Fooable{}
class Bar extends Foo implements Fooable{
    void method(String s){}
}
```

Which are the relationships? Choose four.

(Welches sind die Verwandtschaftsbeziehungen? Wählen Sie vier Antworten.)

- [A] class Bar is a Fooable
- [B] Foo has a Fooable
- [C] class Foo is a Bar

- [D] class Bar is a Foo
- [E] class Bar has a method()
- [F] class Bar has a Fooable

### Lösung

A, B, D und E sind korrekt.

A ist korrekt, weil ein Interface eine Ist-ein-Relation definiert.

B ist korrekt, weil Foo ein Member f von Fooable enthält.

D ist korrekt, weil Bar von Foo erbt.

E ist korrekt, weil Bar eine Methode namens method() besitzt.

C ist falsch, weil Foo nicht von Bar erbt, sondern umgekehrt.

F ist falsch, weil Bar die private Membervariable f nicht erbt.

### Aufgabe 51 (5-2, Shadowing)

Given:

```
class A{
    static void method(){System.out.print("in A ");}
}
class B extends A{
    static void method(){System.out.print("in B ");}
}
public class ShadowTest{
    public static void main(String... args){
        A a=new A();
        B b=new B();
        A c=new B();
        a.method();
        b.method();
        c.method();
    }
}
```

What is the correct output of the above code? Choose one.

(Welcher ist der korrekte Output des obigen Codes? Wählen Sie eine Lösung.)

- [A] in B in B in A
- [B] in B in B in B

- ❑ [C] in B in A in A
- ❑ [D] in A in B in A
- ❑ [E] if »static« in both methods is omitted, the Output is: in A in B in B

### Lösung

D und E sind korrekt. Mit statischen Methoden existiert keine Polymorphie.

Heikel ist die Deklaration `A c=new B();`: Eine statische Methode wird gemäß des Referenztyps (links des Gleichheitszeichens, also A) gewählt. Eine nichtstatische gemäß des Objekttyps (rechts des Gleichheitszeichens, also B).

Werden die beiden Methoden `method()` statisch deklariert, resultiert:

```
in A in B in A
```

Deklariert man beide nichtstatisch (mit Polymorphie), resultiert:

```
in A in B in B
```

### Aufgabe 52 (5-2, Overloading/Overriding/Shadowing)

Given:

```
class Parent {
    int x=10;
    public void f() {System.out.println("parent dynamic ");}
    public static void fs(){System.out.println("parent static ");}
}
class Child extends Parent {
    int x=20;
    public void f(){System.out.println("child dynamic ");}
    public static void fs(){System.out.println("child static ");}
}
public class OverrideShadow {
    public static void main(String[] args){
        Parent c = new Child();
        System.out.println(c.x);
        c.f();
        c.fs();
    }
}
```

What is the output?

[A]

```
20
child dynamic
child static
```

[B]

```
10
parent dynamic
parent static
```

[C]

```
10
child dynamic
parent static
```

[D]

```
20
child dynamic
parent static
```

### Lösung

C ist korrekt. Nur für dynamische Methoden gilt die Polymorphie bzw. das Late Binding. Beim Late Binding wird der Typ rechts des Gleichheitszeichens von *Parent* `c = new Child()`; also der Typ *Child* verwendet. Bei statischen Methoden und Variablen erfolgt die Bindung bei der Kompilierung. Dort ist der Typ links des Gleichheitszeichens maßgebend, also *Parent*. Siehe auch vorangehende Aufgabe.

### Aufgabe 53 (5-1 Reihenfolge es Aufrufs)

What output is generated by the following code?

(Welchen Output erzeugt folgendes Programm?)

```
class Foo{
    private int x;
    public Foo(int x){
        this.x=x;
    }
    public int getX(){
        return x;
    }
}
public class Bar{
```

```
static Foo fooBar(Foo foo){
    foo = new Foo(100);
    return foo;
}

public static void main(String[] args){
    Foo foo = new Foo(200);
    System.out.print(foo.getX()+"-");

    Foo fooFoo = fooBar(foo);
    System.out.print(foo.getX()+"-");
    System.out.print(fooFoo.getX()+"-");

    foo = fooBar(fooFoo);
    System.out.print(foo.getX() + "-");
    System.out.print(fooFoo.getX());
}
}
```

Which output is correct? Choose one.

(Welcher Output ist korrekt? Wählen Sie eine Antwort.)

- [A] 200-100-100-100-100
- [B] 200-200-100-100-100
- [C] 200-200-200-100-100
- [D] 200-200-200-200-100

### Lösung

B ist korrekt. Hier wird mit Referenzvariablen jongliert. Beachten Sie die Kommentare im folgenden Listing:

```
class Foo{
    private int x;
    public Foo(int x){//Konstruktor
        this.x=x;
    }
    public int getX(){
        return x;
    }
}

public class Bar{
    static Foo fooBar(Foo foo){//gibt eine Referenz des Typs Foo zurück
```

```

    foo = new Foo(100); //x wird auf 100 initialisiert
    return foo;
}

public static void main(String[] args){
    Foo foo = new Foo(200); //x des Elternobjekts wird auf 200 gestellt
    System.out.print(foo.getX() + "-"); //Output: 200

    Foo fooFoo = fooBar(foo); //fooBar stellt x auf 100
    System.out.print(foo.getX()+"-"); //x ist immer noch 200
    System.out.print(fooFoo.getX()+"-"); //x ist 100

    foo = fooBar(fooFoo); //die Referenz foo wird überschrieben, x ist 100
    System.out.print(foo.getX() + "-"); //x ist 100
    System.out.print(fooFoo.getX()); //x ist 100
}
}
//Output: 200-200-100-100-100

```

## 10.4.6 Prüfungsziel 6

### Aufgabe 54 (6-1, Map)

Given:

```

import java.util.*;
import java.util.concurrent.*;
public class Account1{
    public static void main(String[] args){
        String[] nameArr={"Hans","Xafer","Baldwin","Aaron"};
        Double[] balanceArr={2300.00,250000.00,30000.00,10000.00};
        //insert here

        for (int i=0;i<nameArr.length;i++){acc.put(nameArr[i],balanceArr[i]);}
        for (Map.Entry<String, Double> e : acc.entrySet())
            System.out.println(e.getKey() + ": " + e.getValue());
    }
}

```

Which line inserted at `//insert here` will result in an alphabetically sorted output? Choose two.

(Welche Zeile bei `//insert here` eingesetzt resultiert in einem alphabetisch sortierten Output? Wählen Sie zwei Antworten.)

- ❑ [A] `Map<String,Double> acc=new Map<String,Double>();`
- ❑ [B] `Map<String,Double> acc=new HashMap<String,Double>();`
- ❑ [C] `Map<String,Double> acc=new LinkedHashMap<String,Double>();`
- ❑ [D] `Map<String,Double> acc=new TreeMap<String,Double>();`
- ❑ [E] `Map<String,Double> acc=new NavigableMap<String,Double>();`
- ❑ [F] `Map<String,Double> acc=new ConcurrentSkipListMap<String,Double>();`

### Lösung

D und F sind korrekt. Die Inhalte dieser beiden Collections werden nach der natürlichen Sortierung geordnet, die im Falle eines Strings lexikographisch ist. Die Klasse `String` implementiert das `Comparable`-Interface schon von vornherein, weshalb die Sortierung automatisch ausgeführt wird.

Würden wir ein selbst definiertes Objekt in eine geordnete Collection speichern, müssten wir `Comparable` implementieren und die Methode `compareTo()` überschreiben. Ohne die Implementierung von `Comparable` würde eine `ClassCastException` zur Laufzeit resultieren.

A ist falsch, weil `Map` ein Interface ist und keinen Konstruktor hat.

B ist falsch, weil eine `HashMap` die Werte in zufälliger Reihenfolge ausgibt.

C ist falsch, weil eine `LinkedHashMap` nach der Einfügereihenfolge sortiert.

E ist falsch, weil `NavigableMap` wie `Map` ein Interface ist und nicht instantiiert werden kann.

```
import java.util.*;
import java.util.concurrent.*;
public class Account1{
    public static void main(String[] args){
        String[] nameArr={"Hans","Xafer","Baldwin","Aaron"};
        Double[] balanceArr={2300.00,250000.00,30000.00,10000.00};
        //insert here
        Map<String,Double> acc=new TreeMap<String,Double>();//=> d) nach ABC
        //Map<String,Double> acc=new ConcurrentSkipListMap<String,Double>();//=> f)
        //nach ABC
        //Map<String,Double> acc=new HashMap<String,Double>();//=> b) zufällig;
        //Xafer, Aaron, Baldwin, Hans
        //Map<String,Double> acc=new LinkedHashMap<String,Double>();//=> c)
        //Einfügereihenfolge
        for (int i=0;i<nameArr.length;i++){acc.put(nameArr[i],balanceArr[i]);}
        for (Map.Entry<String, Double> e : acc.entrySet())
```

```

        System.out.println(e.getKey() + ": " + e.getValue());
    }
}

```

### Aufgabe 55 (6-2, Comparable, compareTo())

Given:

```

import java.util.*;
class GpsRecord implements Comparable<GpsRecord>{
    public int compareTo(GpsRecord gpsr){
        if (this.z>gpsr.z) return -1;
        if (this.z==gpsr.z) return 0;
        if (this.z<gpsr.z) return 1;
        return 0;
    }
    long gpsTime=new Date().getTime();//#1
    double x;double y;double z;
    public GpsRecord(double x,double y,double z){
        this.x=x;this.y=y;this.z=z;
    }
}
public class GpsTracker{
    public static void main(String[] args){
        TreeSet<GpsRecord> ts = new TreeSet<GpsRecord>();
        ts.add(new GpsRecord(2.0,3.0,77.0));
        ts.add(new GpsRecord(3.0,4.0,10.0));
        ts.add(new GpsRecord(4.0,5.0,22.0));
        for (GpsRecord g:ts)
            System.out.printf("\n%5.2f %5.2f %5.2f",g.x,g.y,g.z);
    }
}

```

Select the one correct sort order:

(Wählen Sie einen die eine korrekte Sortierung:)

[A]

```

4.00  5.00 10.00
3.00  4.00 22.00
2.00  3.00 77.00

```

[B]

```

2.00  3.00 77.00
4.00  5.00 10.00
3.00  4.00 22.00

```

○ [C]

```
2.00 3.00 77.00
3.00 4.00 22.00
4.00 5.00 10.00
```

○ [D]

```
3.00 4.00 22.00
2.00 3.00 77.00
4.00 5.00 10.00
```

**Lösung**

C ist korrekt.

Es handelt sich hier um die Implementierung des Interfaces `Comparable` und dem Überschreiben der Methode `compareTo()`. Es wird nach der Variable `z` in umgekehrter Reihenfolge sortiert, weil `compareTo()` in dieser Weise überschrieben wurde. Bei selbstdefinierten Objekten, die man in eine geordnete Collection speichern will, muss `compareTo()` immer überschrieben werden, da sonst eine `ClassCastException` zur Laufzeit geworfen wird.

Die `getTime()`-Methode der Klasse `Date` bei #1 gibt einen `long`-Wert zurück, dessen Zahl die Anzahl der Millisekunden vor oder nach dem 1. Januar 1970 angibt. Diese wurde im Output der Übersichtlichkeit halber weggelassen.

**Aufgabe 56 (6-1, Comparable)**

```
import java.util.*;
class GpsRecord implements Comparable{
    //insert here
    long gpsTime=new Date().getTime();
    double x;double y;double z;
    String comment;
    public GpsRecord(double x,double y,double z, String comment){
        this.x=x;this.y=y;this.z=z;this.comment=comment;
    }
}
```

Which implementation of `compareTo()` is correct, if you wish to sort for the member variable `comment`? Choose one.

(Welche Implementation von `compareTo()` ist korrekt, falls Sie die Membervariable `comment` sortieren wollen? Wählen Sie eine Antwort.)

[A]

```
public int compareTo(GpsRecord gpsr){
    if (this.z>gpsr.z) return -1;if (this.z==gpsr.z) return 0;if
    (this.z<gpsr.z) return 1;return 0;
}
```

[B]

```
public int compareTo(GpsRecord gpsr){
    return comment.compareTo(gspr.comment);
}
```

[C]

```
public int compareTo(Object gpsr){
    return comment.compareTo(gspr.comment);
}
```

[D]

```
public int compare(Object gpsr){
    return comment.compareTo(gspr.comment);
}
```

[E]

```
public int compareTo(Object o){
    GpsRecord gr=(GpsRecord)o;
    return comment.compareTo(gr.comment);
}
```

### Lösung

E ist korrekt. Falls nach einem String sortiert wird, nutzt man die schon überschriebene Methode *compareTo()* der Klasse String. Da **implements Comparable** ohne Typparameter `<GpsRecord>` verwendet wurde, ist ein Cast innerhalb von *compareTo()* notwendig.

A ist falsch, weil diese Art der Implementation von *compareTo()* für Zahlenwerte gedacht ist.

B wäre richtig, falls dem Interface der Typparameter `<GpsRecord>` angehängt worden wäre.

C ist falsch, weil ein Object ohne Inhalt und ohne Casting übergeben wird.

D ist falsch, weil die falsche Funktion verwendet wird, *compare()* von *Comparator* anstatt *compareTo()* von *Comparable*.

Im folgenden Listing ist Lösung E implementiert:

```
import java.util.*;
class GpsRecord implements Comparable{
    public int compareTo(Object o){
        GpsRecord gr=(GpsRecord)o;
        return comment.compareTo(gr.comment);
    }
    long gpsTime=new Date().getTime();//Timestamp
    double x;double y;double z;
    String comment;

    public GpsRecord(double x,double y,double z,String comment){
        this.x=x;this.y=y;this.z=z;this.comment=comment;
    }
}
public class GpsTracker1{
    public static void main(String[] args){
        TreeSet<GpsRecord> ts = new TreeSet<GpsRecord>();
        ts.add(new GpsRecord(2.0,3.0,77.0,"Hallo"));
        ts.add(new GpsRecord(3.0,4.0,22.0,"Welt"));
        ts.add(new GpsRecord(4.0,5.0,10.0,"Allo"));
        for (GpsRecord g:ts)
            System.out.printf("\n%d %5.2f %5.2f %5.2f %s",g.gpsTime, g.x,g.y,g.z, g.com-
ment);
    }
}
Output
1223496420218 4.00 5.00 10.00 Allo
1223496420218 2.00 3.00 77.00 Hallo
1223496420218 3.00 4.00 22.00 Welt
```

Bemerkung; 1223496420218 ist der Zeitstempel

### Aufgabe 57 (6-1, binarySearch())

Given:

```
import java.io.*;
import java.util.*;
public class BinarySearchTest{
    public static void main(String[] args){
        Console c=System.console();
        List<String> t=new ArrayList<String>();
        t.add("neh"); t.add("snooc");t.add("snochie"); t.add("boochies");
```

```
t.add("snigi");t.add("snoogans");
Collections.sort(t);
int position1=Collections.binarySearch(t, "snigi");
int position2=Collections.binarySearch(t, "snooch");
c.printf("%d %d\n",position1,position2);
}
}
```

Which are the correct positions ? Choose one.

(Welches sind die korrekten Positionen? Wählen Sie eine Lösung.)

- [A] 2 6
- [B] 2 -5
- [C] 2 -6
- [D] 1 -6
- [E] 1 -5

### Lösung

C ist korrekt. Sortiert lautet die Ausgabe

```
boochies neh snigi snochie snooc snoogans
```

snigi ist an Position 2, snooch ist nicht vorhanden und käme an Position 5 zu stehen. Dieser Wert, da nicht vorhanden, wird negativ genommen, und davon wird noch eins subtrahiert, Resultat -6.

Die Begriffe snoochy und booch stammen übrigens aus der amerikanischen Komödie »Jay and Silent Bob«. Sehr viele Prüfungsfragen werden mit diesen mühsam zu sortierenden Ausdrücken erstellt. Falls sie nach diesen Ausdrücken googlen, finden Sie unter Umständen interessante Brain-Dumps von vergangenen SCJP-Prüfungen.

### Aufgabe 58 (6-5, Legacy Collections)

Given a legacy implementation of a method:

(Gegeben ist eine veraltete Implementation einer Methode:)

```
public static double avg(List list) {//#1
    double sum = 0;
    for ( Iterator iter = list.iterator(); iter.hasNext(); ) { //#2
        Double d = ((Double)iter.next()).doubleValue(); //#3
        sum += d; //#4
```

```
    }  
    return (double)sum/list.size();//#5  
}  
  
public static double avg(List<Double> list) {  
    double sum = 0;  
    for ( Double d:list) {  
        sum += d;  
    }  
    return sum/list.size();  
}
```

Which three changes must be made to the method avg() to use generics? Choose three.

(Welche drei Änderungen sind notwendig, damit die Methode avg() Generics verwendet? Wählen Sie drei Antworten.)

- [A] remove line #2
- [B] replace line #2 with »for ( Double d:list) {«
- [C] replace line #2 with »for (Iterator iter : list) {«
- [D] remove line #3
- [E] replace line #1 with »public static double avg(List<Double> list) {«

### Lösung

B, D, E sind korrekt, siehe folgender Code:

Legacy:

```
import java.util.*;  
public class AverageLegacy{  
    public static double avg(List list) {  
        double sum = 0;  
        for ( Iterator iter = list.iterator(); iter.hasNext(); ) {  
            Double d = ((Double)iter.next()).doubleValue();  
            sum += d;  
        }  
        return (double)sum/list.size();  
    }  
  
    public static void main(String[] args){  
        ArrayList a=new ArrayList();  
        a.add(new Double(1.0));  
    }  
}
```

```

a.add(new Double(1.5));
a.add(new Double(2.0));
Double av=avg(a);
System.out.println(av);
}
}

```

Generisch:

```

import java.util.*;
public class AverageGenerics{
    public static double avg(List<Double> list) {
        double sum = 0;
        for ( Double d:list) {
            sum += d;
        }
        return (double)sum/list.size();
    }

    public static void main(String[] args){
        ArrayList<Double> a=new ArrayList<Double>();
        a.add(new Double(1.0));
        a.add(new Double(1.5));
        a.add(new Double(2.0));
        Double av=avg(a);
        System.out.println(av);
    }
}

```

### Aufgabe 59 (6-6, Werte in einen Auszug aus einer Collection einfügen)

```

import java.util.*;
import java.util.concurrent.*;
public class CountyMap1{
    public static void main(String[] args) {
        NavigableMap <String, String>navMap = new ConcurrentSkipListMap<String,
String>();
        navMap.put("zh", "Zurich");navMap.put("be", "Berne");
        navMap.put("ag", "Argovia");navMap.put("sh", "Shaffouse");
        navMap.put("tg", "Thurgovie");
        SortedMap<String,String> navSub=navMap.subMap("be","zh");
        navSub.put("vd", "Vaud");
        for (Map.Entry countyPairs:navSub.entrySet()){
            System.out.println(countyPairs.getKey()+" = "+countyPairs.getValue());
        }
    }
}

```

```

    }
  }
}

```

What is the output? Select one.

(Was wird ausgegeben? Wählen Sie eine einzige Antwort.)

- [A] Runtime error, because put on navSub is out of range.
- [B] Compilation error
- [C]

```

be = Berne
sh = Shaffouse
tg = Thurgovie
vd = Vaud

```

- [D]

```

be = Berne
sh = Shaffouse
tg = Thurgovie
vd = Vaud
zh= Zurich

```

### Lösung

C ist korrekt. Eine `ConcurrentSkipListMap` sortiert wie `TreeMap` nach der natürlichen Reihenfolge, die bei einem `String` alphabetisch ist. Der Ausschnitt berücksichtigt den ersten angegebenen Schlüssel, der zweite Schlüssel ist nicht mehr in der Auswahl eingeschlossen.

`subMap()` liefert eine »gebackte« `Collection`, d.h. sie ist mit der originalen `Collection` verlinkt. An `subMap` angebrachte Änderungen finden sich in der originalen `navMap` wieder. Wird hingegen versucht, einen Eintrag ausserhalb des kopierten Bereichs hinzuzufügen, resultiert ein Laufzeitfehler (im folgenden Listing wurde die Auswahl auf »sh« beschränkt und versucht, »vd«, das ausserhalb des Bereichs »be« bis exklusive »sh« einzufügen).

A ist falsch: Es ist möglich, einem Ausschnitt weitere Daten hinzuzufügen, sofern das Element, das hinzugefügt wird, im Bereich des Ausschnitts liegt.

```

import java.util.*;
import java.util.concurrent.*;
public class CountyMap1{
    public static void main(String[] args) {

```

```

NavigableMap<String, String>navMap = new ConcurrentSkipListMap<String, String>();
navMap.put("zh", "Zurich");navMap.put("be", "Berne");
navMap.put("ag", "Argovia");navMap.put("sh", "Shaffouse");
navMap.put("tg", "Thurgovie");
SortedMap<String,String> navSub=navMap.subMap("be","sh"); //!!!!
navSub.put("vd", "Vaud"); //key out of range
for (Map.Entry countyPairs:navSub.entrySet()){
    System.out.println(countyPairs.getKey()+" = "+countyPairs.getValue());
}
}
}
/*
//Output:
//Exception in thread "main" java.lang.IllegalArgumentException: key out of
//range

```

### Aufgabe 6o (6-5, Generics, Comparable)

Given:

```

//insert here
private T max=null;
private T min=null;
private T[] value;
public MyMax(T[] v){ //Konstruktor
    this.value=v;
}
public T max(){
    T v=value[0];
    for (int i=1;i<value.length;i++)
        if(value[i].compareTo(v)>0) v=value[i];
    return v;
}
}

```

Which declaration inserted at //insert here compiles and runs? Choose one answer.

(Welche Deklaration eingefügt bei //insert here kompiliert und läuft? Wählen Sie eine Antwort)

- [A] class MyMax <T implements Comparable<T>> {
- [B] class MyMax <T extends Comparable<T>> {

- [C] class MyMax <T <T>> {
- [D] class MyMax <? extends Comparable<T>> {

### Lösung

B ist korrekt. In der generischen Klasse wird der Typparameter T verlangt. Ein eingeschränkter Typ-Parameter kennt nur das Schlüsselwort extends, implements wie in Antwort A existiert nicht. Comparable selbst besitzt einen Typparameter <T>.

A ist falsch, weil implements nicht in einem eingeschränkten Typparameter vorkommen kann.

C ist falsch, weil ein verschachtelter Typparameter keinen Sinn macht. Vor dem zweiten Typparameter fehlt ein Identifizierer.

D ist falsch, weil ein Typ T und keine Wildcard (>?<) erwartet wird.

```
class MyMax <T extends Comparable<T>> {
    private T max=null;
    private T min=null;
    private T[] value;
    public MyMax(T[] v){ //Konstruktor
        this.value=v;
    }
    public T max(){
        T v=value[0];
        for (int i=1;i<value.length;i++)
            if(value[i].compareTo(v)>0) v=value[i];
        return v;
    }
}
public class MaxTest{
    public static void main(String... args){
        Integer i[]={2,12,11,3,7,9};
        MyMax<Integer> m=new MyMax<Integer>(i);
        System.out.println(m.max());
        Double d[]={2.0,12.0,3.0,7.0,9.0};
        MyMax<Double> n=new MyMax<Double>(d);
        System.out.println(n.max());
    }
}
//Output:
//12
//12.0
```

**Aufgabe 61 (6-1, Generics drag & drop)**

Given:

```

public class  {
    private  var;
    public Generic ( var){
        this.var=var;
    }
    public  getVar(){
        return var;
    }

    public static void main(String[] args){
        Generic<String> str=new Generic<String>("hello");
        Generic<Integer>i=new Generic<Integer>(10);
        System.out.println(str.getVar()+" "+i.getVar());
    }
}

```

Place some of the following into the correct slots, so that the code compiles and runs and writes »hello 10«.

(Platzieren Sie einige der folgenden Kästchen in die korrekten Fächer, sodass der Code kompiliert, läuft und »hello 10« ausgibt.)

**Lösung**

? wird nicht verwendet.

```

public class Generic<T> {
    private T var;
    public Generic (T var){
        this.var=var;
    }
    public T getVar(){
        return var;
    }

    public static void main(String[] args){

```

```
Generic<String> str=new Generic<String>("hello");
Generic<Integer>i=new Generic<Integer>(10);
System.out.println(str.getVar()+" "+i.getVar());
}
}
```

### Aufgabe 62 (6-2, Legacy zu Generics)

Given the following legacy code:

(Gegeben ist folgender veralteter Code:)

```
import java.util.*;
public class Conversion {
    public static void main(String []args){
        List list = new LinkedList();//replace here #1
        list.add("one");
        list.add("two");
        Object x=list.get(0); //replace here #2
        System.out.println(x);
    }
}
```

What must be replaced at #1 and inserted at #2 so that the code will compile and run without warnings?

(Was muss bei #1 und bei #2 ersetzt werden, damit der Code kompiliert und ohne Warnung läuft?)

[A]

```
List <String> list = new LinkedList ();
String x=list.get(0);
```

[B]

```
List list = new LinkedList <String> ();
String x=list.get(0);
```

[C]

```
List <String> list = new LinkedList<String>();
String x=list.get(0);
```

[D]

```
List <String>list = new <String>LinkedList ();
Object o=list.get(0);
```

**Lösung**

C ist korrekt. Eine generisch deklarierte Collection gibt den per Typ-Parameter deklarierten Typ zurück und nicht mehr Object.

A und B kompilieren mit einer Warnung,. Die List ist immer noch im legacy Stil, falls auf einer Seite der Typ-Parameter fehlt. Auslesen muss man einen Wert mit dem Typ Object.

D ist syntaktisch verkehrt und provoziert einen Kompilierfehler.

```
import java.util.*;
public class Conversion {
    public static void main(String []args){
        //List list = new LinkedList();//replace here #1
        List <String>list=new LinkedList<String>();
        list.add("one");
        list.add("two");
        //Object x=list.get(0); //replace here #2
        String x=list.get(0);
        System.out.println(x);
    }
}
```

**Aufgabe 63 (6-4, asArray())**

Given :

```
public static void main(String [] args){
    String[] sarr = new String[]{"a","b","c","d","e" };
    List<String> l=Arrays.asList(sarr);
    //l.add("f"); //1UnsupportedOperationException, hinzufügen nicht erlaubt
    l.set(0,"z"); //ändern erlaubt!!!
    for (String i:l)
        System.out.print(i+" ");
    System.out.println();
    for (String i:sarr)
        System.out.print(i+" ");
    }
}
```

Which statements are true. Coose two.

(Welche Aussagen treffen zu, wählen Sie zwei?)

❑ [A] Output:

```
a b c d e  
z b c d e
```

❑ [B] Output:

```
z b c d e  
z b c d e
```

❑ [C] `l.add("f");` works fine

❑ [D] `l.add("f");` causes an error

❑ [E] `sarr` is independent from `l`

### *Lösung*

B und D sind korrekt. `sarr` und `l` hängen zusammen. Eine Änderung an `l` ist erlaubt aber wirkt sich auf `sarr` aus (`l` ist »backed« durch `sarr`). Daher ist der Output beider Konstrukte der gleiche.

Es ist nicht erlaubt, einer List, die per `asList()` erzeugt wurde, neue Elemente hinzuzufügen. Die fixe Länge des Arrays wird übernommen.

## 10.4.7 Prüfungsziel 7

### Aufgabe 64 (7-5, jar-Import)

On a linux system, a Java program called `MyProg` can be started from the directory

```
/home/user1
```

using the command:

```
java -classpath /test:/home/user1/downloads/*.jar games.MyProg
```

The CLASSPATH is set to

```
/usr/lib:/home/user1/classes:/opt/java/lib:/opt/java/lib/*.jar
```

at system boot.

What is a possible location for the `MyProg.class` file? Choose one.

(Welches ist ein möglicher Standort der Datei `MyProg.class`? Genau eine Antwort auswählen.)

- [A] /test/MyProg.class
- [B] /home/user1/MyProg.class
- [C] /usr/lib/games/MyProg.class
- [D] /home/user1/games/ MyProg.class
- [E] /test/games/MyProg.class
- [F] inside jarfile /opt/java/lib/MyProg.jar (with a correct manifest)
- [G] inside jarfile /home/user1/downloads/MyProg.jar (with a correct manifest)

### Lösung

E ist korrekt.

-classpath überschreibt die Systemeinstellungen. Es kommen daher nur noch die beiden Pfade /test und /home/user1/downloads/\*.jar in Frage. MyProg muss sich im Package games befinden (games.MyProg).

Der einzige passende Pfad ist daher /test/games/MyProg.class.

### Aufgabe 65 (7-6, Operanden)

Given:

```
public class Operands {
    public static void main(String [] args) {
        int x = 5;
        boolean b1 = true;
        boolean b2 = false;
        if ((x == 4) && !b2 )
            System.out.print("1 ");
        System.out.print("2 ");
        if ((b2 = true) && b1 )
            System.out.print("3 ");
    }
}
```

What is the result?

(Was ist das Ergebnis?)

- [A] 2
- [B] 3

- [C] 2 3
- [D] 1 3

### Lösung

C ist richtig. Das erste *if* ist *false*, die Ausgabe von »1« wird übersprungen. Die Bedingung (*b2 = true*) ist immer *true* (Zuweisung anstatt Vergleich, weshalb »3« ausgegeben wird).

### Aufgabe 66 (7-6, Ausgabe von String und Zahlen)

Given:

```
public class StringNumberOutput{
    public static void main(String[] args){
        int izahl=5;
        String szahl="7";
        double dzahl=8.85;
        System.out.println(izahl+szahl+dzahl);//#1
        System.out.printf("%f7.2", izahl+szahl+dzahl);//#2
    }
}
```

What is the output?

(Wie lautet der Output?)

- [A] Compiler error
- [B] Runtime error due to line #1
- [C] 578.85 and Runtime error due to line #2
- [D] 578.85 578.85
- [E] 20.85 20.85

### Lösung

C ist korrekt. Der erste Output gibt 578.85 aus, doch der zweite scheitert, weil mit einem String ein falscher Datentyp in der *printf()*-Methode vorhanden ist.

578.85 bei *println()* kommt durch das konkatenieren von Strings zu Stande. Es findet keine Summenbildung statt, das Resultat ist nie 20.85.

Eine Summation in einener *printf()* Methode ist nicht erlaubt.

**Aufgabe 67 (7-4, Garbage-Collection)**

Given:

```
01 public class Test{
02     public static void main(String... args){
03         ArrayList al=new ArrayList();
04         for (int i=0;i<5;i+1){
05             Double d=new Double(i);
06             al.add(d);
07         }
08         System.out.println(al);
09     }
10 }
```

Which line of code marks the first point that an object referenced by *d* becomes eligible for garbage collection?

(Welche Codezeile markiert den ersten Punkt, wo das durch *d* referenzierte Objekt bereit ist für den Garbage-Collector?)

- [A] 10
- [B] 8
- [C] 9
- [D] 7

**Lösung**

C ist richtig. Das Objekt ist erst außerhalb des Blocks nicht mehr referenziert, den die Methode *main()* markiert.

**Aufgabe 68 (7-2, Kommandozeilenparameter)**

```
class CommandLine{
    public static void main(String... args){
        for (int i=1;i<args.length;i++)
            System.out.print(args[i]+ " ");

    }
}
```

What is the output if the program is started with

- a) `java CommandLine` and
- b) with `java CommandLine 1 2 3 4`?

(Was wird ausgegeben, falls das Programm mit a) `java CommandLine` und b) mit `java CommandLine 1 2 3 4` gestartet wird?)

- [A] (no Output) 1 2 3
- [B] (no Output) 2 3
- [C] (no Output) 2 3 4
- [D] Runtime Error, 1 2 3
- [E] Runtime Error, 2 3 4

### Lösung

C ist korrekt: Da das `args`-Array `o`-basiert ist, aber mit dem Output erst ab eins begonnen wird, werden nur drei Zahlen ausgegeben.

Falls das Programm ohne Parameter gestartet wird ereignet sich kein Laufzeitfehler. Der Array `args` hat dann einfach die Länge `o`.

### Aufgabe 69 (7-6, Is-a/has-a Relation)

Which relationship represents the relationship »Man's best friend is a Horse«?

(Welche Relation wird durch die Aussage »Ein Pferd ist des Menschen bester Freund« ausgedrückt?)

- [A] `class Man extends Horse { }`
- [B] `class Man implements Horse{ }`
- [C] `class Man { private Horse bestFriend; }`
- [D] `class Man { private bestFriend Horse; }`
- [E] `class Man { private Horse<bestFriend>; }`
- [F] `class Man { private BestFriend<Horse>; }`

### Lösung

C ist korrekt. A und B bezeichnen eine Ist-ein-Beziehung, die nicht existiert. D, E und F sind syntaktisch falsch.

### Aufgabe 70 (7-1, Kommandozeilenparameter)

Given:

```
1 class CommandLine1{
2 public static void main(String... args){
3   Integer i= args[0];
4   int j=4;
```

```

5  if (i==j)
6    System.out.println("i==j");
7  else
8    System.out.println("i!=j");
9  }
10}

```

The program is started by issuing `java CommandLine1 4`. What is the Result?

(Das Programm wird mit `java CommandLine1 4` gestartet. Was ist das Resultat?)

- [A] Compiler error at line 3
- [B] `i==j`
- [C] `i!=j`
- [D] Runtime error

### Lösung

A ist korrekt. Zeile 3 würde eine Konversion von String in *int* bzw *Integer* benötigen.

```

int i=Integer.parseInt(args[0]);
Integer=Integer.valueOf(args[0]);

```

Gestartet mit `java CommandLine1 4` ist das Resultat `i==j`. Integer *i* wird beim Vergleich automatisch unboxed.

```

class CommandLine1{
public static void main(String... args){
    Integer i= Integer.valueOf(args[0]);
    int j=4;
    if (i==j)
        System.out.println("i==j");
    else
        System.out.println("i!=j");
    }
}
//Output: i==j

```

### Aufgabe 71 (7-4, java, Garbage-Collection)

```

class A {
    B b;//#1
    public A() { b = new B(this); }
}

```

```
}  
class B {  
    A a; // #2  
    public B(A s) { a = s; }  
}  
public class GCTest {  
    public static void main(String[] args) {  
        A ref = new A(); // #3  
        ref = null;  
        // more code  
    }  
}
```

Which statement is true about the objects referenced by a (#2), b (#1), and ref (#3) immediately after line `ref=null` executes?

(Welche Aussage über die Objekte, die durch a (#2), b (#1) und ref (#3) referenziert sind, trifft direkt nach der Stelle, wo `ref=null` ausgeführt wird, zu?)

- [A] None of these objects are eligible for garbage collection.
- [B] Only the object referenced by a is eligible for garbage collection.
- [C] Only the object referenced by b is eligible for garbage collection.
- [D] Only the object referenced by ref is eligible for garbage collection.
- [E] The objects referenced by a and b are eligible for garbage collection.

### Lösung

E ist richtig.

Klasse A besitzt eine Membervariable des Typs der Klasse B und umgekehrt. Im Konstruktor von A wird der Konstruktor von B aufgerufen. Die Referenzvariable a in B wird mit der Referenz von A initialisiert. Die jeweiligen Membervariablen zeigen somit auf die gegenüberliegenden Objekte. Mit `ref=null` zeigen die Referenzen von A und B immer noch aufeinander, aber der Link zur eigentlichen Welt ist abgebrochen. Man nennt ein solches Konstrukt eine »Referenzinsel«. Diese wird vom Garbage-Collector erst bereinigt, falls jede einzelne Referenz null ist.

### Aufgabe 72 (7-3, jar-Files)

The class `MyClass.class` is deployed in a JAR file named `Lib.jar`. Which statements will allow to use the `MyClass` in a program named `Test`? Choose three.

(Die Klasse `MyClass.class` wurde in einer jar-Datei namens `Lib.jar` bereitgestellt. Welche Befehle erlauben die Verwendung von `MyClass` in einem Programm namens `Test`? Wählen Sie drei Antworten.)

- [A] The JAR file is located at `$JAVA_HOME/lib/ext/Lib.jar`.
- [B] The JAR file is located at `$JAVA_HOME/jre/classes/Lib.jar`.
- [C] The JAR file is located at `/test/Lib.jar` and a classpath environment variable is set so that it includes `/test/myLib.jar/MyClass.class`.
- [D] The JAR file is located at `/test/Lib.jar` and a classpath environment variable is set so that it includes `/test/Lib.jar`.
- [E] The JAR file is located at `/test/Lib.jar` and the Test class is compiled using `javac -cp/test/myLib.jar/MyClass Test.java`.
- [F] The JAR file is located at `/test/Lib.jar` and the Book class is compiled using `javac -d /test/Lib.jar Test.java`.
- [G] The JAR file is located at `/test/Lib.jar` and the Test class is compiled using `javac -classpath/test/Lib.jar Test.java`.

### *Lösung*

A, D und G sind korrekt.

A: `$JAVA_HOME/lib/ext` ist das Verzeichnis, in das jar-Dateien abgelegt werden können, ohne dass ein Klassenpfad notwendig ist. `$JAVA_HOME` ist die absolute Verzeichnisangabe des jre-Verzeichnis (meist `c:\programme\java`). Die Syntax für Linux lautet `$JAVA_HOME` und `%JAVA_HOME%` für Windows.

D: Der Klassenpfad muss `/test/Lib.jar` enthalten.

G: Der Klassenpfad `/test/Lib.jar` kann durch die Kompileroption `-cp` angegeben werden.

B: Der Pfad ist falsch, er muss `$JAVA_HOME/lib/ext/` lauten.

C: Es fehlt der Klassenpfad auf die `.jar`-Datei.

E: `MyClass` darf in einer Klassenpfadangabe nicht enthalten sein.

F: Die Option `-d` ist hier falsch eingesetzt (Destination-Directory), sie müsste `-cp` (Classpath) lauten.