Lösungen

In diesem Dokument finden Sie die Lösungen zu allen Übungsaufgaben aus dem »Java Schnelleinstieg« sortiert nach Kapiteln.

Kapitel 2

Übung 1

```
class HalloWelt {
    public static void main(String[] args) {
        System.out.println("Hallo Nutzer!");
    }
}
```

Listing 1.1: Kapitel 2 Übung 1

Übung 2

```
class HalloWelt {
    public static void main(String[] args) {
        System.out.println("Hallo");
        System.out.println("Welt!");
    }
}
```

Listing 1.2: Kapitel 2 Übung 2

Übung 3

Das Programm erzeugt, wie das Programm in Listing 1.2, ebenfalls die Ausgabe »Hallo« in einer Zeile und »Welt!« in der nächsten. Der Zeichencode `\n` erzeugt einen Zeilenumbruch.

Kapitel 3

Übung 1

In der Mathematik gilt die Regel »Punkt vor Strich«, der Term `1 + 7 * 2` wird also, wenn er ohne Klammern geschrieben wird, mathematisch als `1 + (7 * 2)` interpretiert, das ergibt `1 + 14` also `15`. So auch in Java. Der Term `(1 + 7) * 2` dagegen ergibt `8 * 2` also `16`. Java verhält sich hier also, wie man es aus der Mathematik kennt.

Übung 2

Wie bei Übung 1 stellt sich auch hier die Frage nach der Reihenfolge, in der die Operatoren ausgeführt werden. Tatsächlich ist das aber im konkreten Fall egal, denn sowohl `(true && false) || true` wertet zu `true` aus, als auch `true && (false || true)`. Generell gilt in Java aber, dass der und-Operator vor dem oder-Operator ausgeführt wird. Der Ausdruck `true && false || true` wird von Java also als `(true && false) || true` interpretiert.

Übung 3

```
import java.util.Scanner;

public class Summe {
    public static void main(String[] args){
        Scanner scanner = new Scanner(System.in);

        System.out.println("Erster Summand:");
        double s1 = Double.valueOf(scanner.next());

        System.out.println("Zweite Summand:");
        double s2 = Double.valueOf(scanner.next());

        String summe = String.valueOf(s1 + s2);
        System.out.println("Die Summe ist " + summe);
```

```
        }
    }
```

Listing 1.3: Kapitel 3 Übung 3

Kapitel 4

Übung 1

Codeausschnitt B verwendet eine if-else-Verzweigung. Da die Bedingung für die if-Verzweigung erfüllt ist, wird der Code darin ausgeführt und »b ist wahr« ausgegeben. Der else-Block wird nicht ausgeführt, da die vorangehende Bedingung bereits erfüllt war.

In Codeausschnitt A befinden sich dagegen zwei unabhängige if-Blöcke. Auch hier ist die Bedingung des ersten if-Block erfüllt und der Text »b ist wahr« wird ausgegeben. Danach wird aber auch noch die zweite Bedingung geprüft. Diese ist nach Ausführung des ersten Blocks ebenfalls erfüllt und es wird daher auch noch »b ist falsch« ausgegeben.

Übung 2

Nach der Ausführung von Codeausschnitt C ist der Wert von i immer noch 0, nach der Ausführung von Codeausschnitt D ist der Wert dagegen 1. Der Grund hierfür ist, dass in Codeausschnitt D eine fußgesteuerte Schleife verwendet wird, deren Rumpf immer mindestens einmal ausgeführt wird, auch wenn die Bedingung niemals erfüllt ist. Codeausschnitt C dagegen verwendet eine kopfgesteuerte Schleife, deren Laufbedingung niemals erfüllt ist, somit bleibt der Wert von i unverändert auf 0.

Übung 3

Der Wert von x ist am Ende 10, da die äußere Schleife genau zehnmal ausgeführt wird. Die innere Schleife wird in jedem Durchlauf der äußeren Schleife zwanzigmal wiederholt, also 10 * 20, entsprechend ist der Wert von y am Ende 200.

Kapitel 5

Übung 1

Im Kopf der Funktion testFunktion ist festgelegt, dass die Funktion einen Wert vom Typ Integer zurückgibt. Bei den Werten, die zurückgegeben werden sollen, handelt es sich aber um Strings. Das führt zu einem Fehler beim Kompilieren.

Übung 2

```
public int potenz(int a, int b) {
    if (b == 0) {
        return 1;
    }
    else if (b == 1) {
        return a;
    }
    else {
        return a * potenz(a, b-1);
    }
}
```

Listing 1.4: Kapitel 5 Übung 2

Übung 3

```
public int fakultaet(int n) {
    long fak =1;

    for(int i = 1; i <= n; i++) {
            fak = fak * i;
    }
    return fak;
}
```

Listing 1.5: Kapitel 5 Übung 3

Kapitel 6

Übung 1

A ist eine Oberklasse von B, denn B erweitert (`extends`) A.

Übung 2

Das Attribut `y` der Klasse `A` ist privat, ein Zugriff von außerhalb über `a.y` ist daher nicht möglich.

Übung 3

```
class Viereck {
    int laenge;
    int breite;

    public Viereck(int laenge, int breite){
        this.laenge = laenge;
        this.breite = breite;
    }
}

class Quadrat extends Viereck {
    public Quadrat(int laenge){
        super(laenge, laenge);
    }
}
```

Listing 1.6: Kapitel 6 Übung 3

Kapitel 7

Übung 1

```
public String[] swap(String[] array, int i1, int i2){
    String temp = array[i1];
    array[i1] = array[i2];
```

```
    array[i2] = temp;
    return array;
}
```

Listing 1.7: Kapitel 7 Übung 1

Übung 2

Ja, auch Listen können, wie in Listing 1.8 gezeigt, verschachtelt werden. Die äußere Liste ist dabei eine Liste von Listen, die innere eine normale Liste des entsprechenden Typs.

```
List<List<String>> aussen = new ArrayList<List<String>>();
List<String> innen = new ArrayList<String>();

innen.add("Test");
aussen.add(innen);

System.out.println(aussen.get(0).get(0));
```

Listing 1.8: Kapitel 7 Übung 2

Übung 3

Im skizzierten Fall haben beide Arten von Listen Vor- und Nachteile. Für die LinkedList spricht, dass effizient neue Elemente hinzugefügt werden und alte gelöscht werden können, Änderungen sind selten notwendig.

Gegen die LinkedList spricht aber, dass pro Tag wahrscheinlich deutlich mehr Nachrichten gelesen als geschrieben werden. Das spricht für die ArrayList, in der Elemente besonders effizient gelesen werden können.

Kapitel 8

Übung 1

Der Wert des Parameters `nenner` kann 0 sein und somit eine Division mit dem Nenner 0 auftreten. Das führt zu einem Fehler vom Typ `ArithmeticException`.

Übung 2

Beim Kompilieren wird eine Fehlermeldung auf der Konsole ausgegeben. Der Grund hierfür ist, dass der Fehler `ArrayIndexOutOfBoundsException` doppelt abgefangen wird, das ist aber nicht zulässig.

Übung 3

```
try {
    Liste<Integer> liste = new ArrayList<>();
    int x = liste.get(0);
}
catch (IndexOutOfBoundsException e) {
    System.out.println("Ungültiger Listenindex.");
}
```

Listing 1.9: Kapitel 8 Übung 3

Kapitel 9

Übung 1

Listing 1.10 zeigt, wie eine CSV-Datei aussehen könnte, die sowohl Objekte vom Typ `Kunde` als auch Objekte vom Typ `Mitarbeiter` speichern kann.

```
typ,vorname,nachname,strasse,hausnummer,plz,ort,gehalt
Kunde,Max,Mustermann,Musterstrasse,7,01234,Musterstadt,null
```

```
Mitarbeiter,Mareike,Musterfrau,Musterstrasse,9,01234,Musterstadt,
2800
```

Listing 1.10: Kapitel 9 Übung 1

Übung 2

Um zu prüfen, ob ein Java-Objekt zu einer bestimmten Klasse gehört, kann der `instanceof`-Befehl verwendet werden. Zum Beispiel in der Form `x instanceof Mitarbeiter`.

Kapitel 10

Übung 1

```
import java.io.Serializable;
import org.json.JSONObject;

public class Person implements Serializable {
    String vorname;
    String nachname;
    Adresse adresse;

    public String getVorname() {
        return vorname;
    }

    public void setVorname(String vorname) {
        this.vorname = vorname;
    }

    public String getNachname() {
        return nachname;
    }

    public void setNachname(String nachname) {
        this.nachname = nachname;
```

```
    }

    public Adresse getAdresse() {
        return adresse;
    }

    public void setAdresse(Adresse adresse) {
        this.adresse = adresse;
    }

    public String ganzerName() {
        return vorname + " " + nachname;
    }

    public Person(String vorname, String nachname, String
strasse, int hausnummer, String plz, String ort) {
        this.vorname = vorname;
        this.nachname = nachname;
        this.adresse = new Adresse(strasse, hausnummer, plz,
ort);
    }

    public String toString() {
        return this.getVorname() + "," + this.getNachname() +
","
                + this.getAdresse().getStrasse() + "," +
this.getAdresse().getHausnummer() + ","
                + this.getAdresse().getPlz() + "," +
this.getAdresse().getOrt();
    }

    public JSONObject toJson(){
        JSONObject obj = new JSONObject();
        obj.put("vorname", this.getVorname());
        obj.put("nachname", this.getNachname());

        JSONObject adresse = new JSONObject();
```

```
        adresse.put("strasse",
this.getAdresse().getStrasse());
        adresse.put("hausnummer",
this.getAdresse().getHausnummer());
        adresse.put("plz", this.getAdresse().getPlz());
        adresse.put("ort", this.getAdresse().getOrt());


        obj.put("adresse", adresse);
        return obj;
    }
}
```

Listing 1.11: Kapitel 10 Übung 1

Übung 2

Es erscheint eine Fehlermeldung, da das Programm versucht, auf einen übergebenen Startparameter zuzugreifen. Da allerdings kein Parameter beim Starten übergeben wurde, enthält `args` keine Elemente und der versuchte Zugriff auf das erste Element führt zu einer Fehlermeldung.

Kapitel 11

Übung 1

```
JButton buttonNeu = new JButton("Neu");
buttonNeu.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
        personenListe.add(new Person("", "", "", -1, "", ""));
        ((AbstractTableModel)
dataModel).fireTableDataChanged();
    }
});
f.add(buttonNeu, BorderLayout.PAGE_START);
```

Listing 11.14: Kapitel 11 Übung 1

Übung 2

```
public void setValueAt(Object value, int row, int col) {
    Person p = personenListe.get(row);

    switch (col) {
        case 0:
            p.setVorname(String.valueOf(value));
            break;
        case 1:
            if(String.valueOf(value).length() < 1){
              personenListe.remove(row);
              this.fireTableDataChanged();
            }
            else {
              p.setNachname(String.valueOf(value));
            }
            break;
        case 2:
            p.getAdresse().setStrasse(String.valueOf(value));
            break;
        case 3:

p.getAdresse().setHausnummer(Integer.valueOf(String.valueOf(
value)));
            break;
        case 4:
            p.getAdresse().setPlz(String.valueOf(value));
            break;
        case 5:
            p.getAdresse().setOrt(String.valueOf(value));
            break;
    }
}
```

Listing 1.12: Kapitel 11 Übung 2

Kapitel 12

Übung 1

Da die `run`-Funktion direkt aufgerufen wird, findet keine parallele Ausführung statt, was aber vermutlich beabsichtigt war. Hierzu muss die `start`-Funktion aufgerufen werden.

Kapitel 13

Übung 1

```
public String toSQL(String tabelle){
    String sql = "INSERT INTO " + tabelle;
    sql += " (Vorname, Nachname, Strasse, Hausnummer, PLZ,
Ort)";
    sql += " VALUES ('" + this.getVorname() + "', " +
           "'" + this.getNachname() + "', " +
           "'" + this.getAdresse().getStrasse() + "', " +
           String.valueOf(this.getAdresse().getHausnummer()) +
", "
           "'" + this.getAdresse().getPlz()+ "', " +
           "'" + this.getAdresse().getOrt()+ "');";
    return sql;
}
```

Listing 1.13: Kapitel 13 Übung 1

Übung 2

Der Befehl würde zwei Einträge zurückgeben, nämlich die für Max Mustermann und Mareike Musterfrau, da in beiden Fällen das Alter der Personen kleiner 40 beziehungsweise größer 42 ist. Auf Mike Musterperson trifft das nicht zu, da das Alter zwar gleich, aber nicht größer als 42 ist.

Kapitel 14

Übung 1

```
private Response get (Request request, Response response) {
        String id =
request.getAttributes().get("id").toString();

        if(id == null){
            JSONObject array = new JSONArray();
            String sql = "SELECT * FROM Personen";

            Statement statement = null;
            try {
              statement =
datenbankVerbindung.createStatement();
              ResultSet res = statement.executeQuery(sql);

              while(res.next()) {
                    JSONObject person = new JSONObject();
                person.put("id", res.getInt("id"));
                person.put("vorname",
res.getString("vorname"));
                person.put("nachname",
res.getString("nachname"));

                JSONObject adresse = new JSONObject();
                adresse.put("strasse",
res.getString("strasse"));
                adresse.put("plz", res.getString("plz"));
                adresse.put("hausnummer",
res.getInt("hausnummer"));
                adresse.put("ort", res.getString("ort"));

                person.put("adresse", adresse);
                array.add(person);
              }
```

```
                response.setEntity(array.toString(4),
MediaType.APPLICATION_JSON);
        }

        else {
             JSONObject person = new JSONObject();

        String sql = "SELECT * FROM Personen WHERE ID = " +
id;
        Statement statement = null;
        try {
            statement = datenbankVerbindung.createStatement();
            ResultSet res = statement.executeQuery(sql);

            if(res.next()) {
                person.put("id", res.getInt("id"));
                person.put("vorname",
res.getString("vorname"));
                person.put("nachname",
res.getString("nachname"));

                JSONObject adresse = new JSONObject();
                adresse.put("strasse",
res.getString("strasse"));
                adresse.put("plz", res.getString("plz"));
                adresse.put("hausnummer",
res.getInt("hausnummer"));
                adresse.put("ort", res.getString("ort"));

                person.put("adresse", adresse);

                response.setEntity(person.toString(4),
MediaType.APPLICATION_JSON);
            }
            else {
                response.setStatus(new
Status(Status.CLIENT_ERROR_NOT_FOUND, "Not found."));
```

```
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        }

        return response;
}
```

Listing 14.8: Kapitel 14 Übung 1

Übung 2

```
private Response put (Request request, Response response) {
        aget("id").toString();
        Representation entity = request.getEntity();

        try {
            JSONObject person = new
JSONObject(entity.getText());
            JSONObject adresse =
person.getJSONObject("adresse");

            String sql = "UPDATE Personen SET ";
            if(!person.getString("vorname") == null)
                    sql += "vorname = '" +
person.getString("vorname") + "',";
            if(!person.getString("nachname ") == null)
                    sql += "nachname = '" +
person.getString("nachname") + "',";
            if(!adresse.getString("strasse") == null)
                    sql += "strasse = '" +
adresse.getString("strasse") + "',";
            if(!adresse.getInt("hausnummer") == null)
                    sql += "hausnummer = "
+String.valueOf(adresse.getInt("hausnummer")) + ",";
            if(!adresse.getString("plz") == null)
```

```
                    sql += "plz = '" +
adresse.getString("plz") + "',";
            if(!adresse.getString("ort") == null)
                    sql += "ort = '" +
adresse.getString("ort");
                    sql += "' WHERE id = " + id;

            Statement statement =
datenbankVerbindung.createStatement();
            statement.execute(sql);
        } catch (Exception e) {
            e.printStackTrace();
        }

        return response;
}
```

Listing 14.8: Kapitel 14