

HANS-GEORG SCHUMANN

ANDROID-APPS PROGRAMMIEREN

FÜR **KIDS**

2. AUFLAGE

SMARTPHONE-APPS ENTWICKELN
OHNE VORKENNTNISSE



INHALT

EINLEITUNG	9
-------------------------	---

WILLKOMMEN IN ANDROID STUDIO	17
Android Studio starten	18
Der SDK Manager	21
Ein neues Projekt erzeugen	25
Der AVD Manager	31
Die Emulation starten	36
Android Studio beenden	38
Zusammenfassung	39
Zwei Fragen	40
... aber noch keine Aufgaben	40

1

DAS ERSTE EIGENE PROJEKT	41
Design und Layout	42
Von Hello zu Hallo	47
Gut oder Schlecht	51
Antwort auf Antwort	60
Ein Projekt importieren	65
Zusammenfassung	67
Zwei Fragen	68
... und zwei Aufgaben	68

2

HALLO MIT KNOPFDRUCK	69
Was ist im Paket?	70
Variablen verknüpfen	73
onCreate und import	76
Den Buttons Leben einhauchen	79
Ressourcen-Strings benutzen	83

3

Der passende Titel	85
Ein Projekt kopieren	86
Zusammenfassung	91
Ein paar Fragen	92
... und ein paar Aufgaben	92

4

RECHNEN MIT DEM ZUFALL	93
Alles auf neu	93
Komponentenschwemme	99
Zufallszahlen und Zeichenketten	108
Jetzt wird gerechnet	112
Lokal-global, privat oder öffentlich?	114
Zusammenfassung	119
Ein paar Fragen	120
... und eine Aufgabe	120

5

BEDINGUNGEN	121
Von 1 bis 6	122
Wenn ... dann	127
Die if-Struktur	131
Von Fall zu Fall	135
Punkt für Punkt	137
Und und Oder – oder?	139
Zusammenfassung	141
Zwei Fragen	142
... und eine Aufgabe	142

6

GELD UND SPIELE	143
Auf dem Weg zum Millionär	143
while oder do-while?	146
Ich denke mir eine Zahl	149
Zu groß, zu klein	152
Feintuning	155
Schiebungen	158
Neues Spiel?	161
Zusammenfassung	163
Ein paar Fragen	164
... und ein paar Aufgaben	165

ES BEWEGT SICH WAS	167
Neue Komponenten	168
Show and Hide	175
Animation-XML	179
Film ab!	184
Gehen oder Drehen?	185
Zusammenfassung	192
Ein paar Fragen	194
... jedoch nur eine Aufgabe	194

7

ANIMATIONEN	195
Buttons mit Bild	196
Kommen und Gehen	199
Die Kugel rollt	202
Zufallsziele	204
Das Ziel selbst bestimmen	208
Grenzkontrollen	212
Eine eigene Methode	215
Zusammenfassung	217
Ein paar Fragen	218
... und ein paar Aufgaben	219

8

EIN KÄFER KRABBELT SICH FREI	221
Bug oder Käfer?	221
Der richtige Winkel	227
Noch mehr Methoden	232
Der »Runnable-Handler«	234
Eine eigene Klasse	237
Zusammenfassung	244
Ein paar Fragen	246
... doch nur eine Aufgabe	246

9

VOM KÄFER ZUR WANZE	247
Reparaturarbeiten	247
Das Spiel-Objekt einsetzen	250
Wanzenjagd	253
Feintuning	257
Eine Frage der Zeit?	259

10

Wiederbelebung	262
Treffer zählen	265
Zusammenfassung	267
Keine Frage	267
... und nur eine Aufgabe	267

11	SPRINGEN ODER DUCKEN?	269
	Hoch- oder Querformat	269
	Jump Et Duck	274
	Die Mitte finden	278
	Drücken oder loslassen	281
	Das richtige Layout	283
	Emulator-Wahl	288
	Zusammenfassung	292
	Ein paar Fragen	293
	... und eine Aufgabe	293

12	KONTAKTVERMEIDUNG	295
	Was für ein Ding?	295
	Setzen und zeigen	298
	Von rechts nach links	303
	Kontaktaufnahme	307
	»Dodges« zählen	312
	Game over?	315
	Zusammenfassung	319
	Eine Frage	320
	... und ein paar Aufgaben	320

13	ARRAYS	321
	Ein schönes Paar?	321
	Methoden mit Nummer	326
	Anpassen und einpassen	328
	Eigene View-Objekte erzeugen	331
	Layout-Tuning	336
	Wanzenschwemme	338
	Zusammenfassung	343
	Ein paar Fragen	344
	... und ein paar Aufgaben	344

DAS GROSSE KRABBELN	345
Jagd auf alle?	345
Im Wechselschritt	349
Punkte und Ende	351
Wiederbelebung	354
Käfer und Spinnen	357
Bis zum Ende	360
Zusammenfassung	363
Ein paar Fragen	364
... doch nur eine Aufgabe	364

14

BUNTES ALLERLEI	365
Treffer-Sound	366
... und Schluss-Sound	370
Vibrationen	373
Sensor-Management	376
Kugel-Roller	379
In Bewegung	381
Zusammenfassung (und Schluss)	384
Keine Fragen	386
... und nur eine Aufgabe	386

15

ANHANG	387
---------------------	-----

A

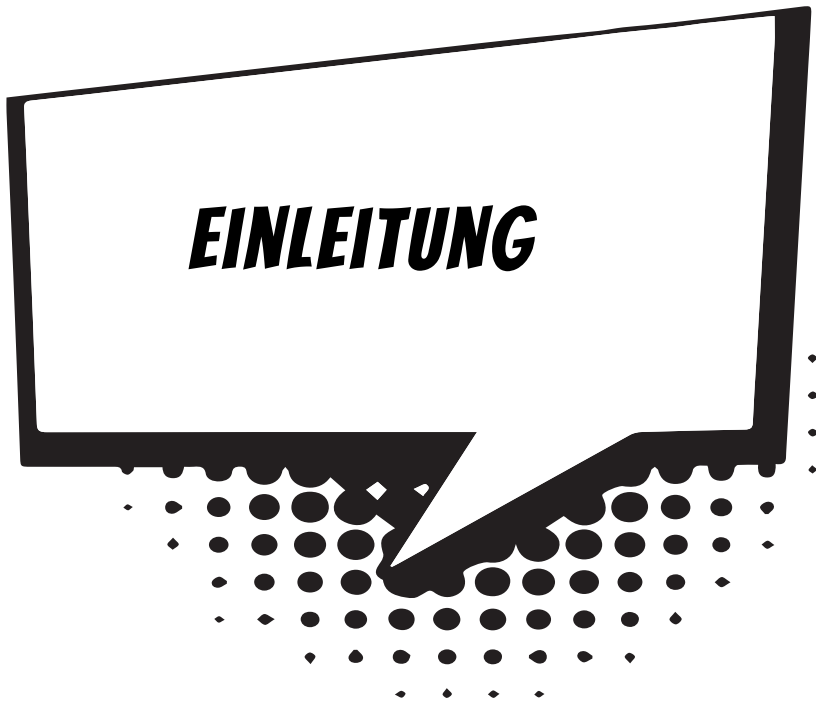
ANHANG	401
---------------------	-----

B

ANHANG	413
---------------------	-----

C

STICHWORTVERZEICHNIS	415
-----------------------------------	-----



Android? Ist das nicht so etwas wie eine Mensch-Maschine? Ein Roboter, der einem Menschen (täuschend) ähnlich sieht? Ja, und gleichzeitig der Name eines Betriebssystems, das **Google** entwickelt hat und kostenfrei zur Verfügung stellt.

Während man Windows vorwiegend auf »großen« PCs findet, ist Android auf Smartphones und Tablets am meisten verbreitet. Rund Dreiviertel aller Handys (wie man sie auch immer noch nennt) laufen mit dem Android-System.

Computer sind sie eigentlich alle: Geräte, die man auf oder unter den Tisch stellt (Desktop-PCs), Geräte, die man mitnehmen kann (Notebooks und Tablets), und dann noch eine Nummer kleiner: die Smartphones.

Vorwiegend um Android für Smartphones geht es hier. Die meisten Projekte funktionieren aber auch auf Tablets. Programme für dieses System werden meisten kurz **Apps** genannt (als Abkürzung für Applikationen).

WAS HEIßT EIGENTLICH PROGRAMMIEREN?

Wenn du aufschreibst, was ein Computer tun soll, nennt man das **Programmieren**. Das Tolle daran ist, dass du selbst bestimmen kannst, was getan werden soll. Lässt du dein Programm laufen, macht der Computer die Sachen, die du ausgeheckt hast. Natürlich wird er dann nicht dein Zimmer aufräumen und dir auch keine Tasse Kakao ans Bett bringen. Aber kannst du erst mal programmieren, kannst du den Computer sozusagen nach deiner Pfeife tanzen lassen.

Allerdings passiert es gerade beim Programmieren, dass der Computer nicht so will, wie du es gerne hättest. Meistens ist das ein Fehler im Programm. Das Problem kann aber auch irgendwo anders im Computer oder im Betriebssystem liegen. Das Dumme bei Fehlern ist, dass sie sich gern so gut verstecken, dass die Suche danach schon manchen Programmierer zur Verzweiflung gebracht hat.

Vielleicht hast du nun trotzdem Lust bekommen, das Programmieren zu erlernen. Und ausgerechnet noch für die »ganz Kleinen«. Dann brauchst du ja nur noch eine passende **Entwicklungsumgebung**, und schon kann's losgehen.

WAS IST EINE ENTWICKLUNGSUMGEBUNG?

Um ein Programm zu erstellen, musst du erst etwas eintippen. Das ist wie bei einem Brief oder einer Geschichte, die man schreibt. Das Textprogramm dafür kann sehr einfach sein, weil es ja nicht auf eine besondere Schrift oder Darstellung ankommt wie bei einem Brief oder einem Referat. So etwas wird **Editor** genannt.

Ist das Programm eingetippt, kann es der Computer nicht einfach lesen und ausführen. Jetzt muss es so übersetzt werden, dass der PC versteht, was du von ihm willst. Weil er aber eine ganz andere Sprache spricht als du, muss ein Dolmetscher her.

Du programmierst in einer Sprache, die du verstehst, und der Dolmetscher übersetzt es so, dass es dem Computer verständlich wird. So etwas heißt dann **Compiler**.

Im Prinzip kann man für Android mehrere Programmiersprachen benutzen, am meisten verbreitet ist Java. Hier gibt es als Dolmetscher die **Java Virtual Machine** (kurz JVM). Diese Art »Zwischencomputer« lässt sich kostenlos installieren.

Eigentlich wird ein Java-Programm also an die JVM weitergereicht, die es dann für den jeweiligen Computer passend zubereitet: Das kann ein PC z.B. mit Windows sein. Oder ein Smartphone z.B. mit Android. Ein und dasselbe Java-Programm kann so im Prinzip auf jedem beliebigen Gerät funktionieren, das über eine JVM verfügt.

Schließlich müssen Programme getestet, überarbeitet, verbessert, wieder getestet und weiterentwickelt werden. Dazu gibt es noch einige zusätzliche Hilfen. Daraus wird dann ein ganzes System, die Entwicklungsumgebung.

WARUM GERADE JAVA?

Leider kannst du nicht einfach programmieren, wie dir der Mund gewachsen ist. Eine **Programmiersprache** muss so aufgebaut sein, dass möglichst viele Menschen in möglichst vielen Ländern einheitlich damit umgehen können.

Weil in der ganzen Welt Leute zu finden sind, die wenigstens ein paar Brocken Englisch können, besteht auch fast jede Programmiersprache aus englischen Wörtern. Es gab auch immer mal Versuche, z.B. in Deutsch zu programmieren, aber meistens klingen die Wörter dort so künstlich, dass man lieber wieder aufs Englische zurückgreift.

Eigentlich wäre es egal, welche Programmiersprache du benutzt. Am besten eine, die möglichst leicht zu erlernen ist. Doch in der Android-Welt ist die Nummer 1 die Programmiersprache **Java**, mit der du es auch in diesem Buch zu tun hast. Diese gehört inzwischen zu den am meisten verbreiteten Sprachen im Computer-Bereich. Sie ist nicht einfach, aber auch für Anfänger geeignet, die mit Java ihre erste Programmiersprache lernen wollen. Und: Es ist eine Sprache, die Smartphones und Tablets mit Android gut verstehen.

In letzter Zeit hat sich mit **Kotlin** eine neue Sprache verbreitet, die man auch unter Android Studio benutzen kann. Wir bleiben hier jedoch bei Java.



Der Weg zum guten Programmierer kann ganz schön steinig sein. Nicht selten kommt es vor, dass man die Lust verliert, weil einfach gar nichts klappen will. Das Programm tut etwas ganz anderes, man kann den Fehler nicht finden und man fragt sich: Wozu soll ich eigentlich programmieren lernen, wo es doch schon genug Apps gibt? Und dann noch ausgerechnet für Android. Aber du verspürst da einen Reiz, eigene Apps zu schreiben? Es ist also nicht nur einen Versuch wert, es kann sich durchaus lohnen, das Programmieren zu erlernen.

ANDROID STUDIO, DIE ENTWICKLUNGSUMGEBUNG ZUM BUCH

Um den Kauf einer Entwicklungsumgebung für Java und für Android musst du dich nicht kümmern, beides bekommst du kostenlos aus dem Internet.



Google stellt dir **Android Studio** zur Verfügung, eine komfortable und leistungsstarke Entwicklungsumgebung, mit der du unter allen Versionen von Windows programmieren kannst. Android Studio macht es auch möglich, Smartphones und Tablets zu simulieren und damit alle Apps erst mal auf dem PC unter Windows zu testen.

Wie man Android Studio und Java aus dem Internet holt und installiert, erfährst du in **Anhang A**.

UND WAS BIETET DIESES BUCH?

Über eine ganze Reihe von Kapiteln verteilt lernst du

- ❖ die Grundlagen der Programmierung kennen
- ❖ mit Android Studio unter Windows umzugehen
- ❖ mit Komponenten zu arbeiten (das sind Bausteine, mit denen du dir viel Programmierarbeit sparen kannst)
- ❖ auch komplexere Programm-Elemente einzusetzen
- ❖ eine ganze Reihe von Spiel-Projekten zu erstellen
- ❖ wie man Apps für den Play Store vorbereitet

Im **Anhang** gibt es dann noch zusätzliche Informationen und Hilfen, u.a. über Installationen und den Umgang mit Fehlern.

WIE ARBEITE ICH MIT DIESEM BUCH?

Grundsätzlich besteht dieses Buch aus einer Menge Text mit vielen Abbildungen dazwischen. Natürlich habe ich mich bemüht, alles so zuzubereiten, dass daraus lauter gut verdauliche Happen werden. Damit das Ganze noch genießbarer wird, gibt es zusätzlich noch einige Symbole, die ich dir hier gern erklären möchte:

ARBEITSSCHRITTE

➤ Wenn du dieses Zeichen siehst, heißt das: Es gibt etwas zu tun. Damit kommen wir beim Programmieren Schritt für Schritt einem neuen Ziel immer näher.

Grundsätzlich lernt man besser, wenn man einen Programmtext selbst eintippt oder ändert. Aber nicht immer hat man große Lust dazu. Deshalb gibt es alle Projekte im Buch auch als Download:

<http://www.mitp.de/899>

Und hinter einem Programmierschritt findest du auch den jeweiligen Namen des Projekts oder einer Datei (z.B. → *GameView.java*). Wenn du also das Projekt nicht selbst erstellen willst, kannst du stattdessen den passenden Ordner laden (zu finden im Download-Paket).

AUFGABEN

Am Ende eines Kapitels findest du jeweils eine Reihe von Fragen und Aufgaben. Diese Übungen sind nicht immer ganz einfach, aber sie helfen dir, noch besser zu programmieren. Lösungen zu den Aufgaben findest du in verschiedenen Formaten ebenfalls im Verzeichnis *Projekte*. Du kannst sie dir alle im Editor von Windows oder auch in deinem Textverarbeitungsprogramm anschauen. Oder du lässt sie dir ausdrucken und hast sie dann schwarz auf weiß, um sie neben deinen Computer zu legen. (Auch die Programme zu den Aufgaben findest du im Download-Paket.)

NOTFÄLLE

Vielleicht hast du irgendetwas falsch gemacht oder etwas vergessen. Oder es wird gerade knifflig. Dann fragst du dich, was du nun tun sollst. Bei diesem Symbol findest du eine Lösungsmöglichkeit. Notfalls kannst du aber auch ganz hinten im **Anhang B** nachschauen, wo einige Hinweise zur Pannenhilfe aufgeführt sind.



WICHTIGE STELLEN IM BUCH



Hin und wieder findest du ein solch dickes Ausrufezeichen im Buch. Dann ist das eine Stelle, an der etwas besonders Wichtiges steht.

EXPERTENWISSEN



Wenn du ein solches »Wow« siehst, geht es um ausführlichere Informationen zu einem Thema.

WAS BRAUCHST DU FÜR DIESES BUCH?

Installiert wird Java in ein Verzeichnis deiner Wahl, z.B. `C:\Programme\Java`. Auch Android Studio muss installiert werden und auch hier kannst du das Verzeichnis selbst bestimmen, z.B. `C:\Programme\Android`. Zusätzlich empfiehlt es sich, auf einem anderen Laufwerk einen Arbeitsordner für die Projekte einzurichten, z.B. `D:\Projekte`.

Die Beispielprogramme in diesem Buch gibt es alle als Download von der Homepage des Verlages, falls du keine Lust zum Abtippen hast:

<http://www.mitp.de/899>

Und auch die Lösungen zu den Fragen und Aufgaben sind dort untergebracht.

BETRIEBSSYSTEM

Die meisten Computer arbeiten heute mit dem Betriebssystem Windows. Davon brauchst du eine Version zwischen 7 und 10. Außerdem wäre es gut, ein Smartphone oder Tablet mit Android zur Hand zu haben. Du kannst aber auch nur einen Emulator von Android Studio benutzen. Er reicht für die Mehrzahl der Projekte aus (es gibt aber einige wenige, die nur mit dem Smartphone oder Tablet funktionieren).

SPEICHERMEDIEN

Auf jeden Fall benötigst du etwas wie einen USB-Stick oder eine SD-Card, auch wenn du deine Programme auf die Festplatte speichern willst. Auf einem externen Speicher sind deine Arbeiten auf jeden Fall zusätzlich sicher aufgehoben.

Gegebenenfalls bitte deine Eltern oder Lehrer um Hilfe.

HINWEISE FÜR LEHRER

Dieses Buch lässt sich natürlich auch für den Informatik-Unterricht verwenden. Dort setzt natürlich jeder Lehrer seine eigenen Schwerpunkte. Benutzen Sie an Ihrer Schule bereits ein Werk aus einem Schulbuchverlag, so können Sie dieses Buch in Ergänzung zu dem vorhandenen Schulbuch einsetzen. Weil wir hier sozusagen »bei null« beginnen, ist auch ein direkter Einstieg in Java möglich – ohne irgendwelche anderen Programmierkenntnisse.

Der wesentliche Schwerpunkt in diesem Buch ist die Programmierung für das vor allem bei Smartphones weitverbreitete Betriebssystem Android. Dabei kommt auch die objektorientierte Programmierung nicht zu kurz.

In den Projekten werden die wesentlichen Elemente des Java-Wortschatzes wie auch die wichtigsten Komponenten für eine Android-Applikation (App) eingesetzt. In den Lösungen zu den Aufgaben finden Sie weitere Vorschläge zur Programmierung.

AUF DIE DATEIEN ZUM BUCH VERZICHTEN?

Vielleicht ist es Ihnen lieber, wenn Ihre Schüler die Projekte alle selbst erstellen. Dann lassen Sie die Download-Dateien einfach (erst einmal) weg.

ÜBUNGSMEDIEN

Für den Informatik-Unterricht sollte jeder Schüler ein anderes externes Speichermedium haben, um darauf seine Programmierversuche zu sichern. So wird verhindert, dass sich auf der Festplatte des Schulcomputers mit der Zeit allerlei »Datenmüll« ansammelt. Außerdem dient der eigene Datenträger dem Datenschutz: Nur der betreffende Schüler kann seine Daten manipulieren.

REGELMÄßIG SICHERN

Es kann nicht schaden, die Programmdateien, an denen gerade gearbeitet wird, etwa alle zehn Minuten zu speichern. Denn Computer pflegen gern gerade dann »abzustürzen«, wenn man seine Arbeit längere Zeit nicht gespeichert hat. In der Regel aber sorgt Android Studio dafür, dass bei Programmschluss alles gespeichert ist.



Du willst gleich loslegen? Dein Smartphone brauchst du dazu erst mal nicht, aber einen Windows-PC. Auf dem entwerfen wir dann gemeinsam dein erstes Programmprojekt. Wunder darfst du dabei nicht erwarten, es wird sicher nicht so bunt wie die fast schon unzählbaren Apps, die man aus dem Play Store herunterladen kann. Aber wir sind ja erst ganz am Anfang. Und für das allererste Mal gibt es viel zu tun.

In diesem Kapitel lernst du

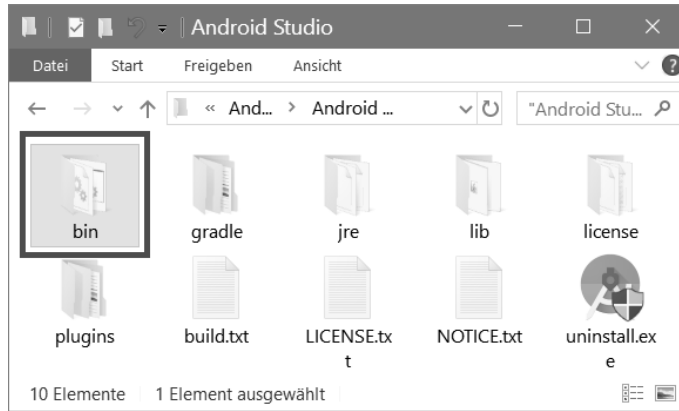
- ⊙ wie man Android Studio startet
- ⊙ etwas über den Einsatz des SDK Managers
- ⊙ wie man eine neue App erzeugt
- ⊙ etwas über den Einsatz des AVD Managers
- ⊙ wie man den Android-Emulator startet
- ⊙ wie man seine App ausführt
- ⊙ wie man Android Studio beendet

ANDROID STUDIO STARTEN

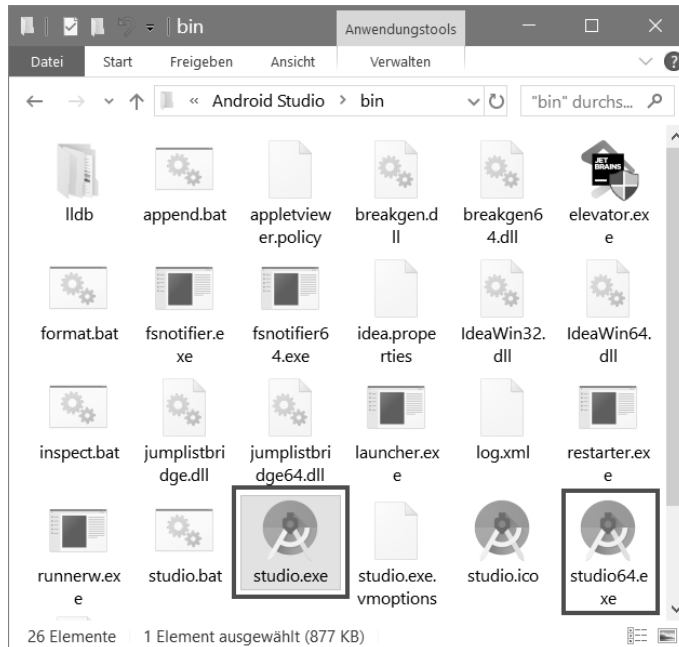
Bevor wir mit dem Programmieren anfangen können, muss das Entwicklungssystem Android Studio installiert werden. Genaues erfährst du im **Anhang A**. Hier solltest du dir von jemandem helfen lassen, wenn du dir das Einrichten nicht allein zutraust.

Eine Möglichkeit, Android Studio zu starten, ist diese:

- Öffne den Ordner, in den du Android Studio untergebracht hast (z.B. `C:\programme\Android`).



Dort musst du nun weiter in einen Unterordner mit dem Namen *BIN* wechseln:



Hier suchst du unter den vielen Symbolen eines der grünen heraus, und zwar das mit dem Namen *studio.exe*.

Falls es ein zweites mit dem Namen *studio64.exe* gibt, kannst du auch das ausprobieren. Doch das funktioniert nicht auf jedem PC, hier muss die sogenannte 64-Bit-Version von Windows installiert sein.

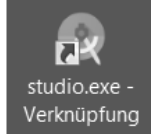
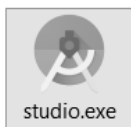


➤ Nun kannst du das Programm mit einem Doppelklick auf das Symbol starten.

Ich empfehle dir, eine **Verknüpfung** auf dem Desktop anzulegen:

- ❖ Dazu klickst du mit der rechten Maustaste auf das Symbol für Android Studio (STUDIO.EXE). Im Kontextmenü wählst du KOPIEREN.
- ❖ Dann klicke auf eine freie Stelle auf dem Desktop, ebenfalls mit der rechten Maustaste. Im Kontextmenü wählst du VERKNÜPFUNG EINFÜGEN.
- ❖ Es ist sinnvoll, für das neue Symbol auf dem Desktop den Text *studio.exe – Verknüpfung* durch *Android Studio* zu ersetzen.

Von nun an kannst du auf das neue Symbol doppelklicken und damit Android Studio starten.

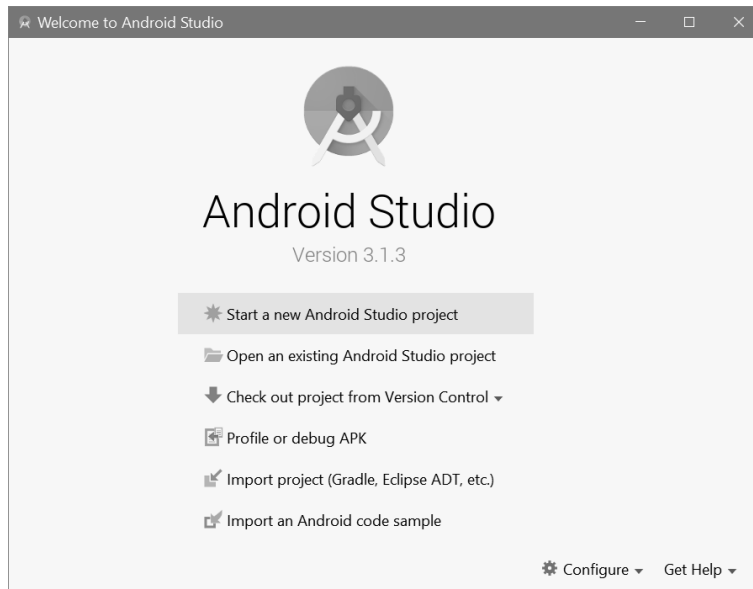
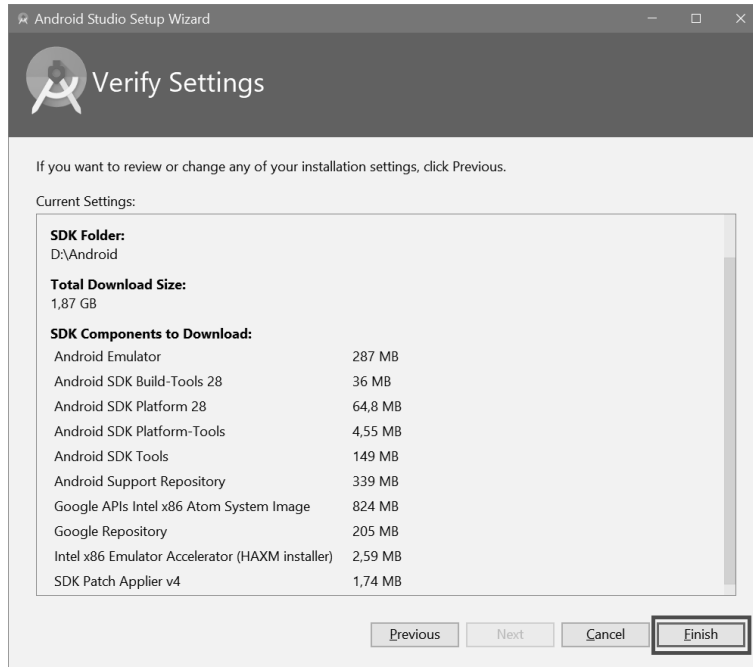


Beim ersten Start kann es sein, dass dich noch ein Überbleibsel von der Installation erwartet.

Ein **Setup Wizard** informiert dich über einige installierte Elemente (die du zum Teil später noch kennenlernen wirst).

➤ Schließe dieses Fenster mit einem Klick auf FINISH.

Je nach Computer kann es eine Weile dauern, bis Android Studio geladen ist. Einige Zeit später landest du in einem Willkommen-Fenster.



Du möchtest gern gleich mit einem neuen Projekt beginnen? Das könntest du mit einem Klick auf **START A NEW ANDROID STUDIO PROJECT** direkt in die Wege leiten.

Besser aber ist es, wir leisten noch ein bisschen Vorarbeit.

Um ein Android-Projekt zu erstellen und zum Laufen zu bringen, benötigen wir die Hilfe von zwei Managern:

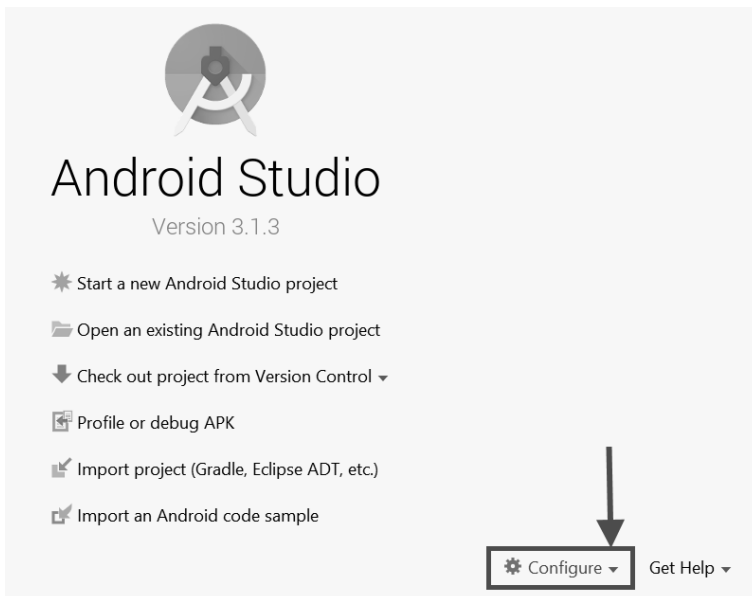
- ❖ Der **Software Development Kit (SDK) Manager** stellt die nötigen Elemente des Betriebssystems Android zur Verfügung. Davon gibt es ja inzwischen einige Versionen. Und wenn wir für möglichst viele Smartphones programmieren wollen, brauchen wir nicht nur die neueste, sondern auch einige der älteren Versionen.
- ❖ Der **Android Virtual Device (AVD) Manager** bietet die geeigneten Smartphone-Emulationen (das heißt, er sorgt dafür, dass die jeweiligen Smartphones unter Windows künstlich nachgeahmt werden). Natürlich gibt es auch Emulationen für Tablets und andere Geräte.



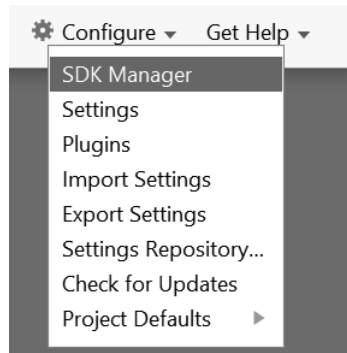
Android Studio hat bei der Installation schon dafür gesorgt, dass die wichtigsten Pakete mitinstalliert wurden. Aber das heißt nicht, dass ein Projekt sofort reibungslos läuft. Denn es gibt noch deutlich mehr Pakete im Angebot, und einige davon werden zusätzlich benötigt. Um selbst zu bestimmen, was letztendlich installiert ist, brauchen wir beide Manager. Dann können deine Apps störungsfrei laufen.

DER SDK MANAGER

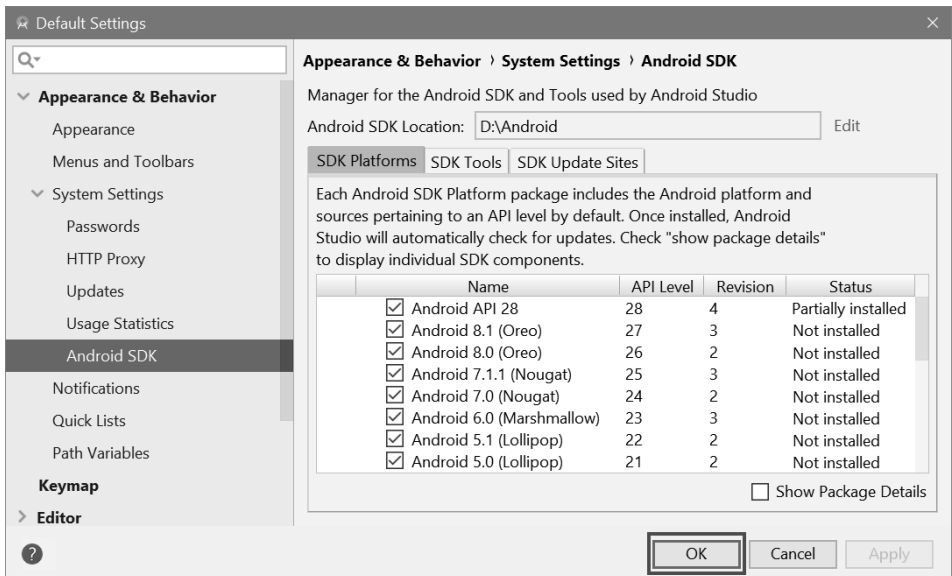
- Um das System auf den neuesten Stand zu bringen und das nötige »Drumherum« zu installieren, klicke unten rechts auf das kleine Dreieck hinter CONFIGURE.



➤ Es öffnet sich ein kleines Menü, wo du auf SDK MANAGER klickst.



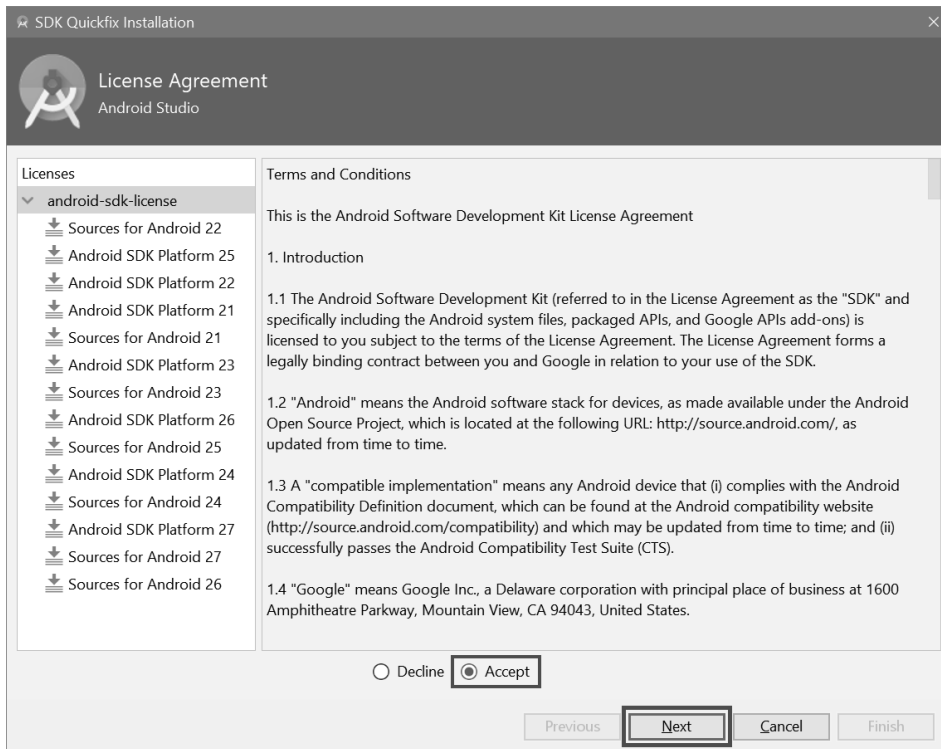
Es öffnet sich ein neues Zusatzfenster. (Möglicherweise wird noch einiges geladen, ehe es dir zur Verfügung steht.)



Unter der Option ANDROID SDK kannst du dir nun die Android-Versionen aussuchen, für die du deine Apps programmieren willst. Bedenke, dass es bis zu einer Stunde (und länger) dauern kann, bis alles heruntergeladen und installiert ist. Ich habe mich entschieden, alle älteren Versionen mindestens ab 5.0 (Level 21) zu bedienen, musste dafür jedoch teilweise sehr lange warten.

➤ Nach deiner Auswahl klickst du auf OK.

Nun musst du noch die Lizenz-Bestimmungen absegnen.



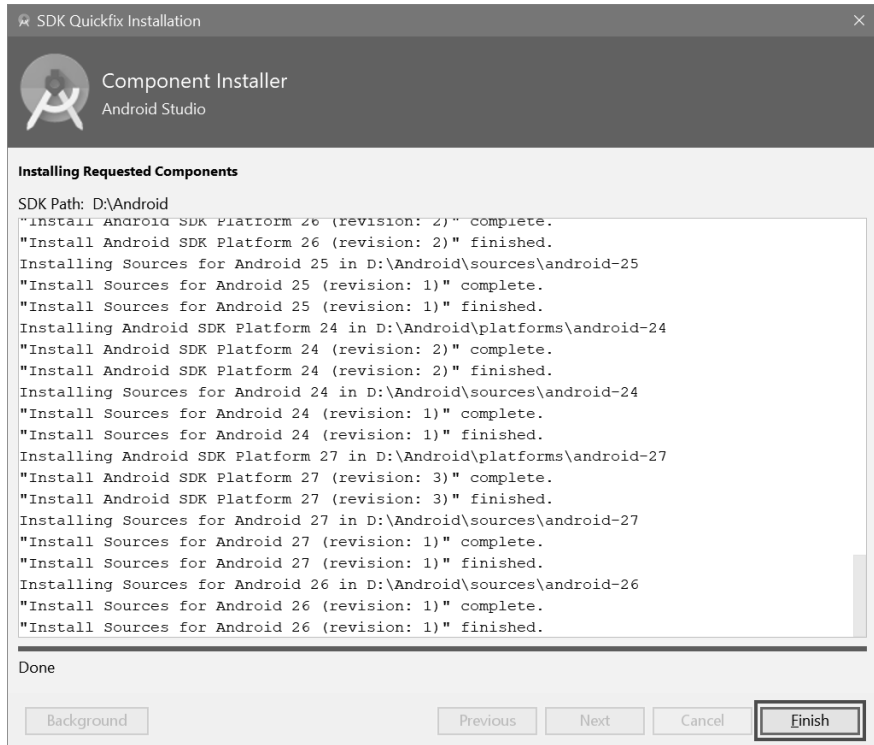
➤ Markiere unten den Eintrag ACCEPT und klicke dann auf NEXT. Damit startest du die Installationen.

Da die Pakete (Packages) für die SDK Tools aus dem Internet geladen werden, ist nun eine **Internet-Verbindung** nötig. Es wird immer das angezeigt, was gerade aktuell ist, deshalb kann das, was du siehst, von der Abbildung im Buch abweichen.

So wie Windows seine Versionsnummer hat, gibt es so etwas natürlich auch bei Android. Weil die meisten sich nicht jedes Jahr ein neues Smartphone oder Tablet kaufen, sind Versionen ab 5.0 noch sehr verbreitet. Du kannst natürlich auch alle Versionen auswählen, womit sich auch sehr viel ältere Smartphones bedienen lassen.



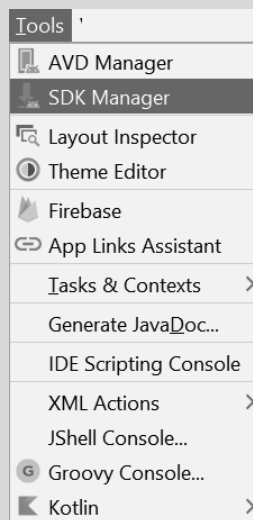
In einem neuen Fenster kannst du die langwierigen Downloads und Installationen mitverfolgen, wenn du willst.



➤ Am Schluss beendest du das Ganze mit einem Klick auf FINISH.

Und du landest wieder im Willkommen-Fenster von Android Studio.

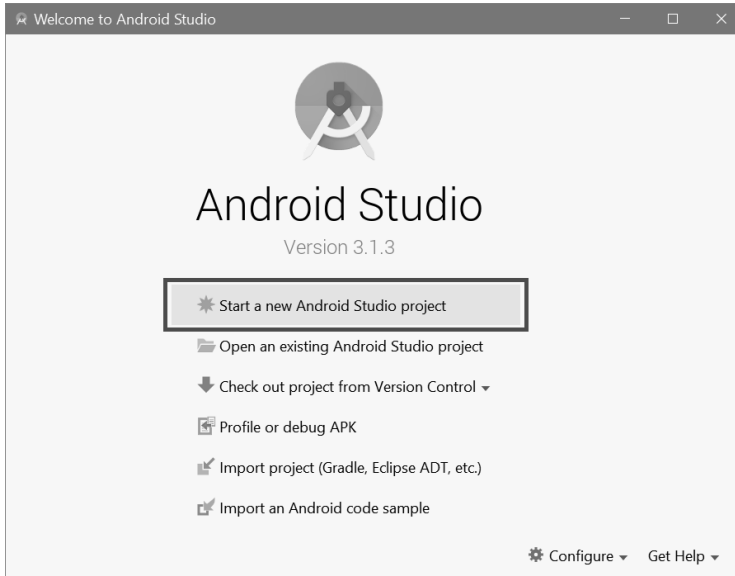
Der SDK Manager lässt sich auch über das TOOLS-Menü starten:



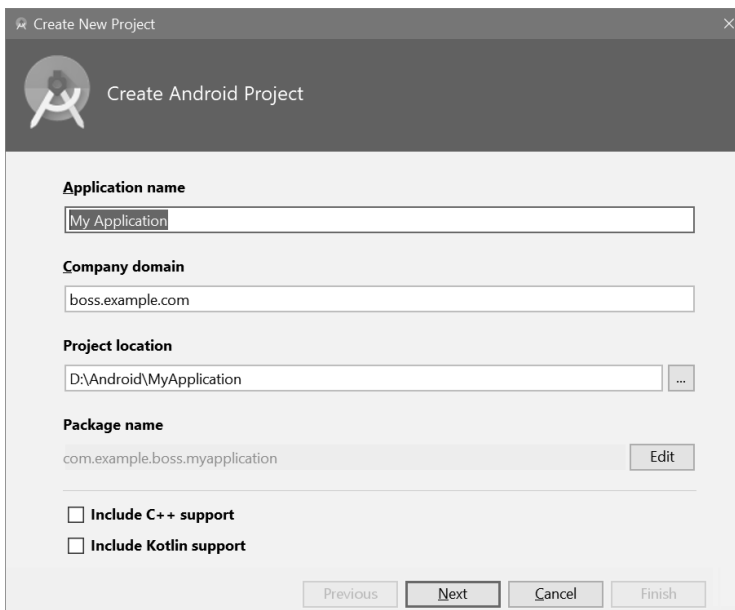
EIN NEUES PROJEKT ERZEUGEN

Und nun kannst du endlich in dein erstes Projekt einsteigen.

➤ Dazu klickst du auf die Schaltfläche **START A NEW ANDROID STUDIO PROJECT**.

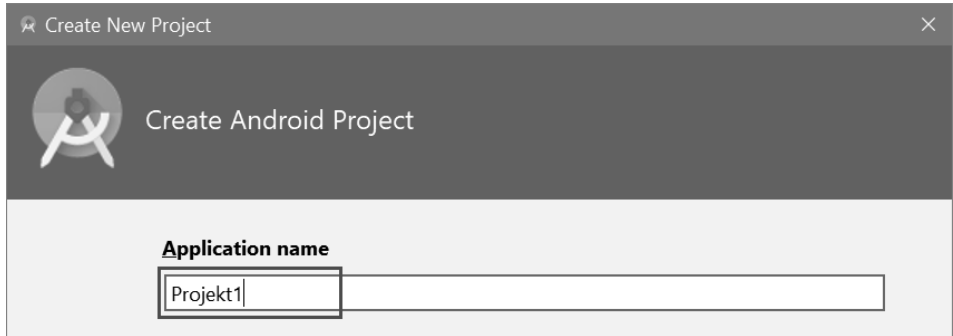


Ein neues Dialogfeld mit dem Titel **CREATE NEW PROJECT** erscheint und fordert dich heraus. Dann musst du etwas eingeben und gleich eine weitere Entscheidung treffen.



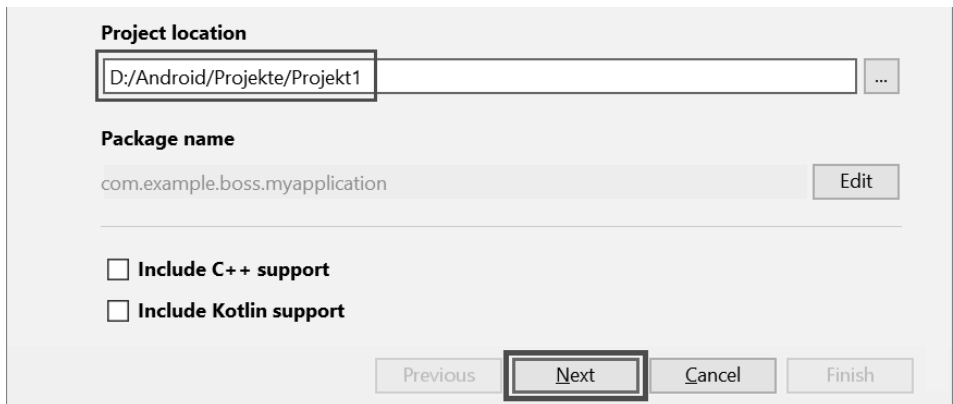
Zuerst braucht dein Projekt einen Namen. Du kannst natürlich *My Application* stehen lassen, doch ich empfehle dir, stets einen eigenen neuen Namen zu vergeben. Nennen wir das erste Projekt einfach *Projekt1*, auch weil wir noch nicht wissen, was genau die App, die nachher herauskommt, macht.

➤ Tippe hinter APPLICATION NAME einen neuen Namen ein.



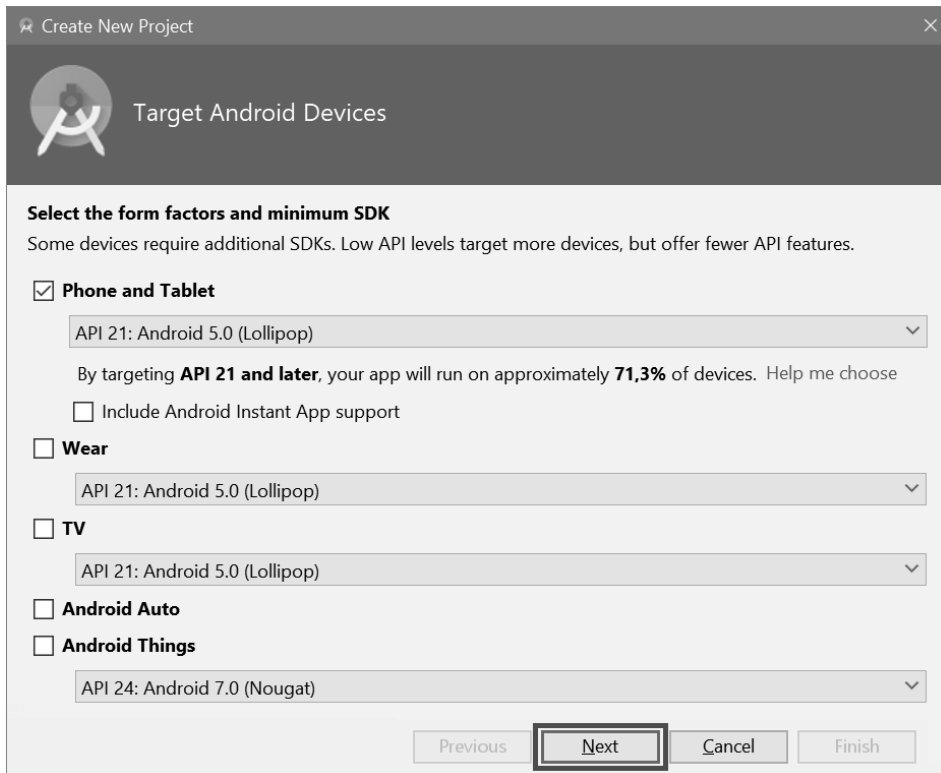
Weiter unten wird das Verzeichnis angezeigt, in dem dein Projekt untergebracht werden soll. Im Allgemeinen bietet Android Studio einen Platz im *Benutzer*-Ordner auf Laufwerk C: an. Ich schlage vor, die Projektdateien in einem anderen Laufwerk und Ordner unterzubringen.

Dazu könntest du einen weiteren Ordner in *D:\Android* mit dem Namen *Projekte* erstellen. Wenn dir das zu umständlich ist, lasse einfach den angebotenen Speicherort hinter PROJECT LOCATION stehen.



➤ Wenn Name und Speicherort feststehen, kannst du auf NEXT klicken.

Im folgenden Fenster genügt es, wenn ein Haken vor PHONE AND TABLET steht. Wir wollen eine App (= Applikation) für Smartphones entwickeln, aber wenn das Ding auch auf einem Tablet läuft, umso besser (an anderen Geräten sind wir derzeit nicht interessiert).



Create New Project

Target Android Devices

Select the form factors and minimum SDK

Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

☒ **Phone and Tablet**

API 21: Android 5.0 (Lollipop)

By targeting **API 21 and later**, your app will run on approximately **71,3%** of devices. Help me choose

☐ Include Android Instant App support

☐ **Wear**

API 21: Android 5.0 (Lollipop)

☐ **TV**

API 21: Android 5.0 (Lollipop)

☐ **Android Auto**

☐ **Android Things**

API 24: Android 7.0 (Nougat)

Previous **Next** Cancel Finish

Da nicht alle Smartphones mit den neuesten Versionen von Android versorgt werden, ist es sinnvoll, ein Projekt für eine (etwas) frühere Android-Version zu entwickeln. So läuft deine App auf möglichst vielen (auch älteren) Smartphones.

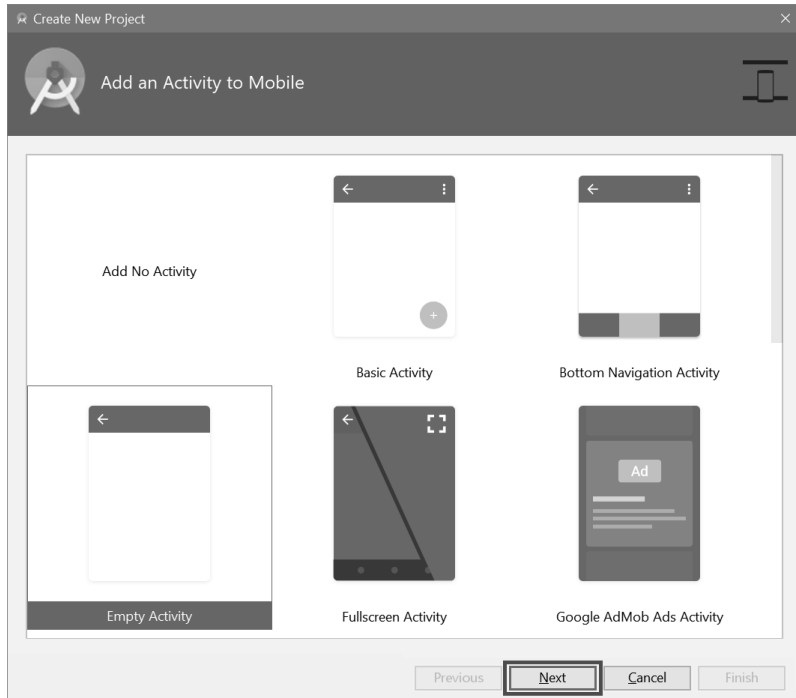
Auf jeden Fall laufen sämtliche Apps, die für ältere Android-Versionen entwickelt wurden, auch auf allen neuen Smartphones.

Wichtig ist: Für die eingestellte Version muss das **exakte** SDK installiert sein. Notfalls musst du also noch mal den SDK Manager bemühen, um Fehlendes nachträglich zu installieren.



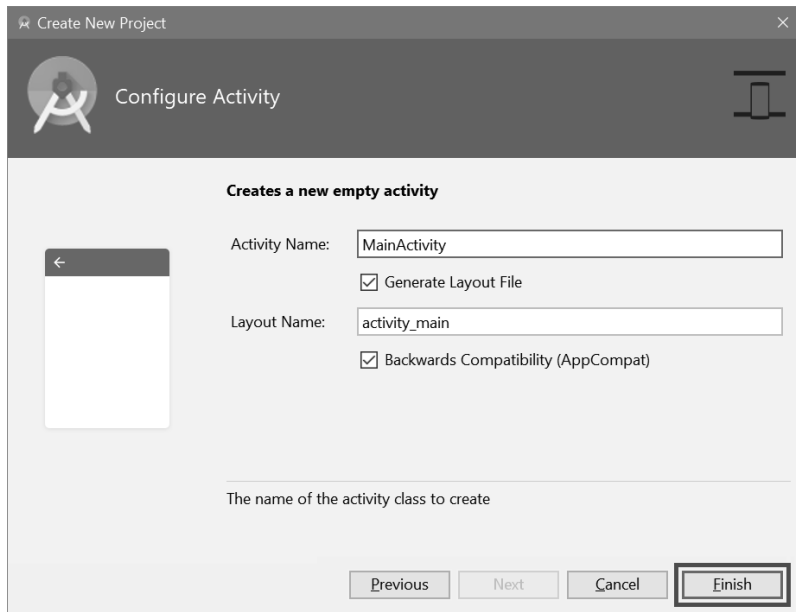
- Sorge dafür, dass die Android-Version eingestellt ist, mit der du arbeiten willst. Dann klicke auf **NEXT**.

Und nun geht es um die **Activity**. So nennt man ein Benutzerinterface. Das ist die Verbindungsstelle zwischen dem Benutzer (z.B. dir) und einem Smartphone. Vereinfacht kann man hier sagen: ein Fenster auf dem Display. Also quasi der Rahmen für die Aktivitäten einer App. Du solltest dich hier für eine »Empty Activity« entscheiden, also sozusagen ein »Fenster mit nix drin«.



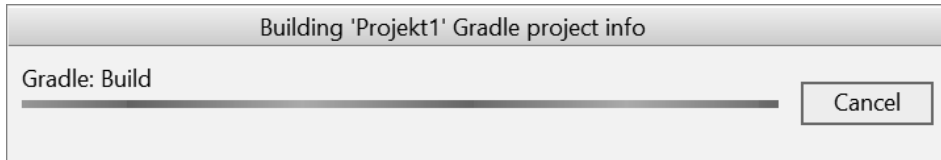
➤ Sorge dafür, dass EMPTY ACTIVITY markiert ist, und klicke dann auf NEXT.

Auch im folgenden Fenster kannst du alles für dein aktuelles Projekt so stehen lassen.

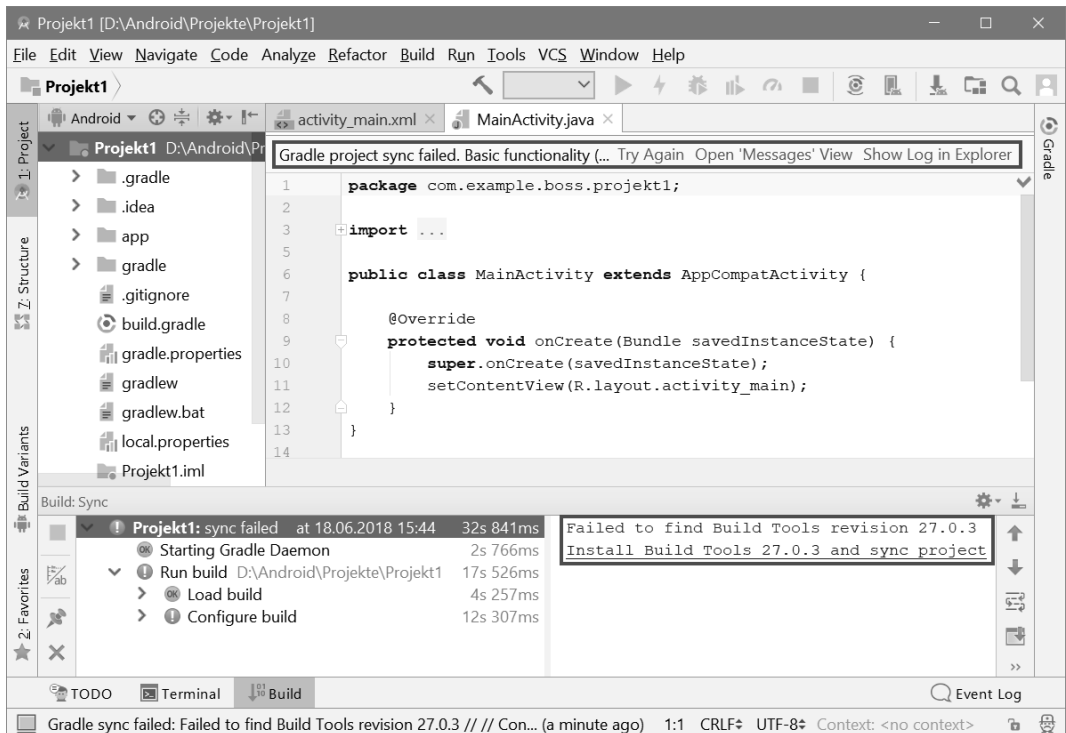


➤ Diesmal klickst du auf FINISH und schließt damit auch das Fenster.

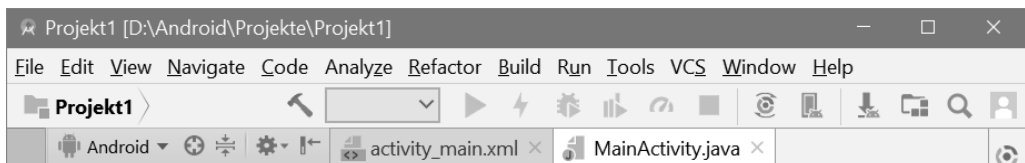
Nun beginnt Android Studio, deine Einstellungen zu verarbeiten. Das kann eine Weile dauern.



Was dich dann erwartet, könnte etwa so oder ähnlich aussehen:



Für den ersten Augenblick mag das ein bisschen sehr verwirrend sein. Doch fangen wir ganz oben an. Dort sitzt die Menüleiste. Darunter befinden sich mehrere Symbole, die man mit der Maus anklicken kann.



Von den vielen Menüs wirst du wahrscheinlich diese am meisten benutzen:

- ❖ Über das FILE-Menü kannst du Dateien speichern, laden (öffnen), ausdrucken, neu erstellen oder Android Studio beenden.
- ❖ Das Menü EDIT hilft dir bei der Bearbeitung deines Programmtextes, aber auch bei anderen Programmelementen. Außerdem kannst du dort bestimmte Arbeitsschritte rückgängig machen oder wiederherstellen.
- ❖ Über das RUN-Menü sorgst du dafür, dass dein Projekt ausgeführt wird.
- ❖ Und das HELP-Menü bietet dir vielfältige Hilfsinformationen (auf Englisch) an.

Unter dem Menübereich erwarten dich ein paar Fensterabschnitte:

- ❖ Links siehst du eine Spalte, die dem Explorer in einem Fenster unter Windows entspricht, hier werden die Bestandteile deines Projekts aufgelistet. (Möglicherweise sieht es bei dir anders aus, z.B. wenn oben nicht ANDROID eingestellt ist und die Listen eingeklappt sind.)
- ❖ Rechts steht der sogenannte Quelltext, mit dem du jetzt sicher noch nichts anfangen kannst.
- ❖ Ganz unten sind weitere Informationen. Dort werden unter anderem auch Fehler gemeldet.

Möglicherweise gibt es bei dir ebenso wie bei mir gleich zu Anfang Fehlermeldungen. Von GRADLE und BUILD TOOLS kann da die Rede sein.



```
Gradle project sync failed.

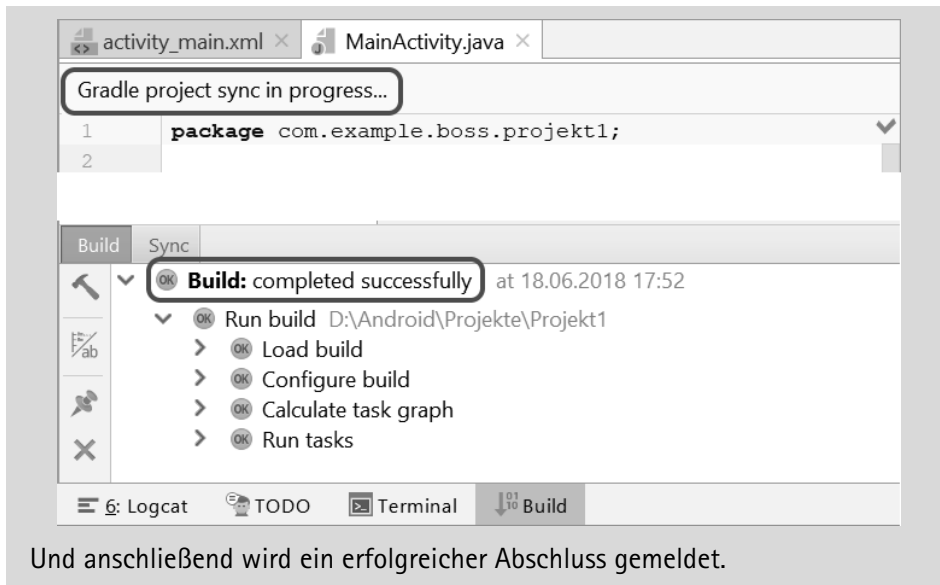
Failed to find Build Tools
Install Build Tools and sync project
```

```
▼ [i] Failed to sync Gradle project
  Error: failed to find target android : D:\Android\sdk
         Install missing platform(s) and sync project

▼ [i] Failed to sync Gradle project
  Error: failed to find Build Tools
         Install Build Tools and sync project
```

Im Falle einer solchen Meldung klickst du unten auf INSTALL BUILD TOOLS, danach werden einige Updates ausgeführt.

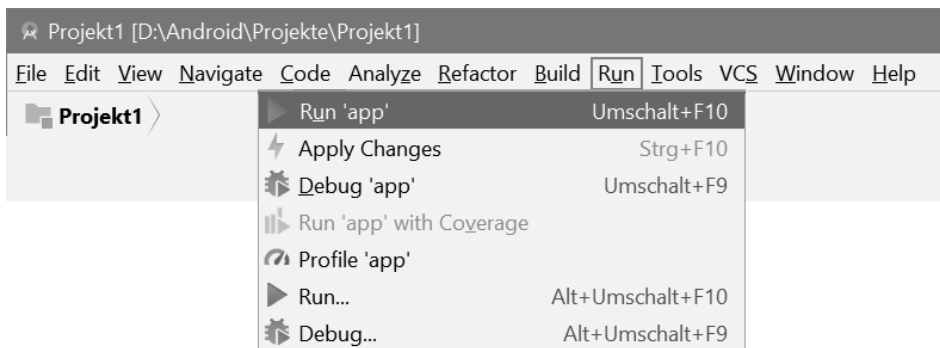
Anschließend kommt die Meldung, dass Android Studio erneut versucht, ein funktionierendes Programm zu erstellen.



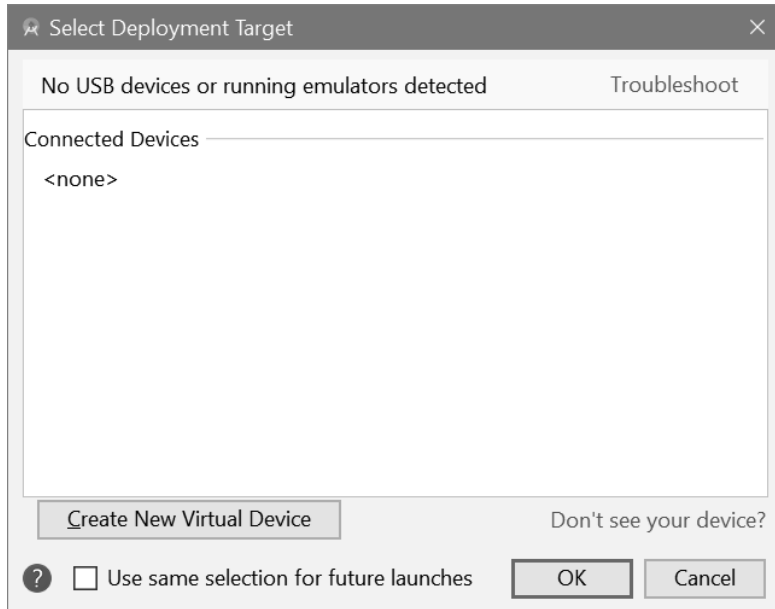
DER AVD MANAGER

Haben wir etwa schon ein Programm bzw. eine App?

- Probiere einfach aus, was passiert, wenn du im RUN-Menü auf den ersten Eintrag RUN 'APP' klickst (oder auf den grünen Pfeil in der Symbolleiste).



Nach einer Weile bekommst du dieses Dialogfeld zu sehen:



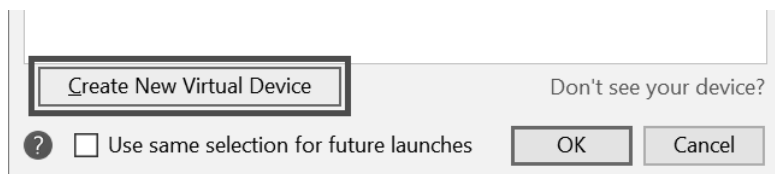
Mit »No USB devices« ist gemeint, dass offenbar kein Smartphone oder Tablet per USB am Computer angeschlossen ist. Und das »none« unter CONNECTED DEVICES weist darauf hin, dass auch kein Smartphone emuliert wird. Ansonsten ist darunter gähnende Leere.



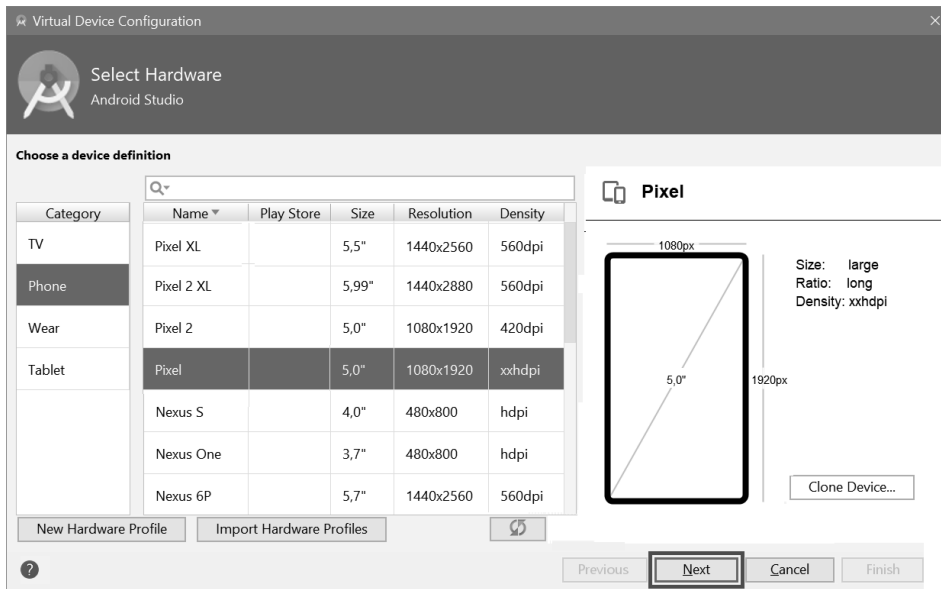
Später werden wir direkt ein Smartphone an den PC anschließen, um unsere Apps auch dort zu testen. Jetzt aber brauchen wir ein virtuelles (also un-echtes) Gerät, das so tut, als sei es ein Smartphone (genau das tut ein **Emulator**).

Wir müssen also nur ein solches Gerät, genannt **Android Virtual Device** (kurz: AVD) finden und das aktivieren.

➤ Klicke auf CREATE NEW VIRTUAL DEVICE.

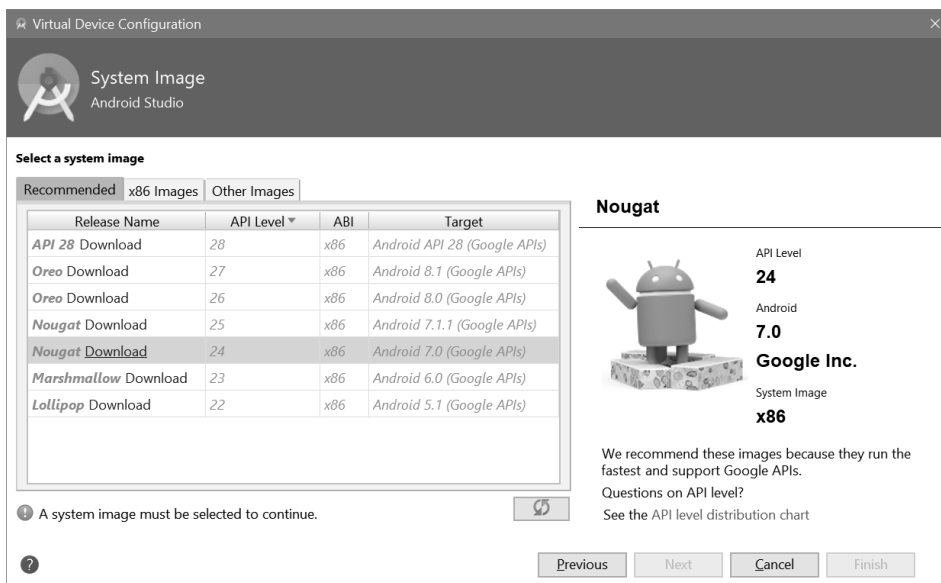


Und schon erscheint ein neues Dialogfeld mit dem Titel **SELECT HARDWARE**. Hier findest du unter **PHONE** eine ganze Reihe von Smartphones, die offenbar unter Android Studio emuliert werden können.



➤ Markiere das Gerät, das du haben willst, und klicke dann auf NEXT.

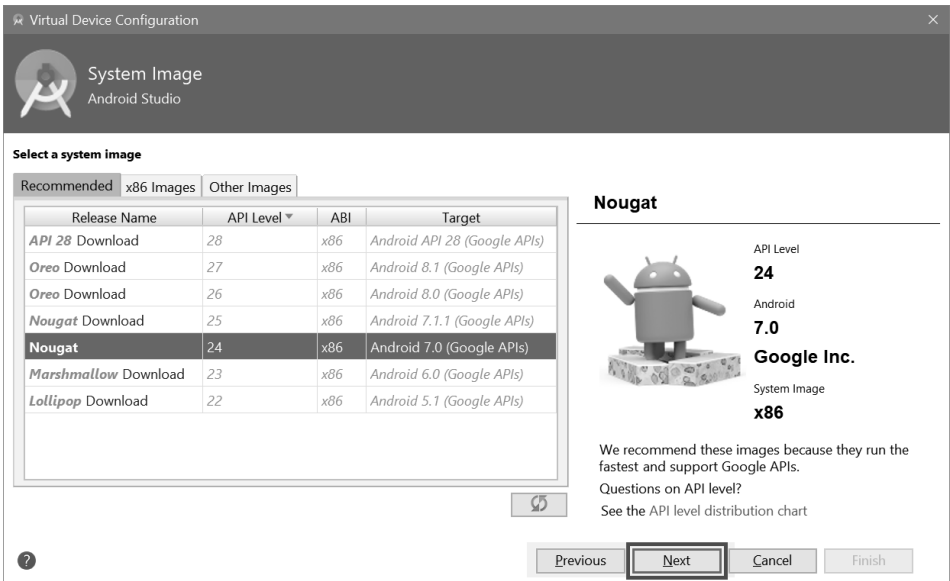
Und schon landest du im nächsten Dialogfeld mit dem Titel SYSTEM IMAGE.



Hier wählst du eines der Android-Systeme aus der Liste. Was darin steht, hängt auch davon ab, wie viel du mit dem SDK Manager installiert hast.

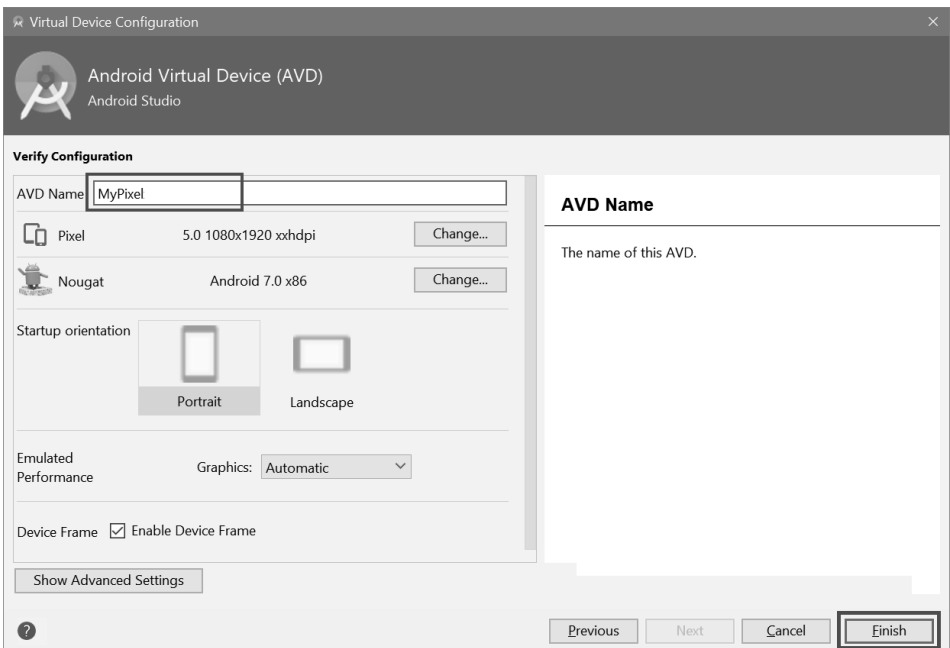
➤ Gegebenenfalls musst du noch Dateien für die betreffenden Versionen nachladen. Klicke also hinter dem Namen der Version auf DOWNLOAD.

Anschließend sollte das System deiner Wahl markiert sein.



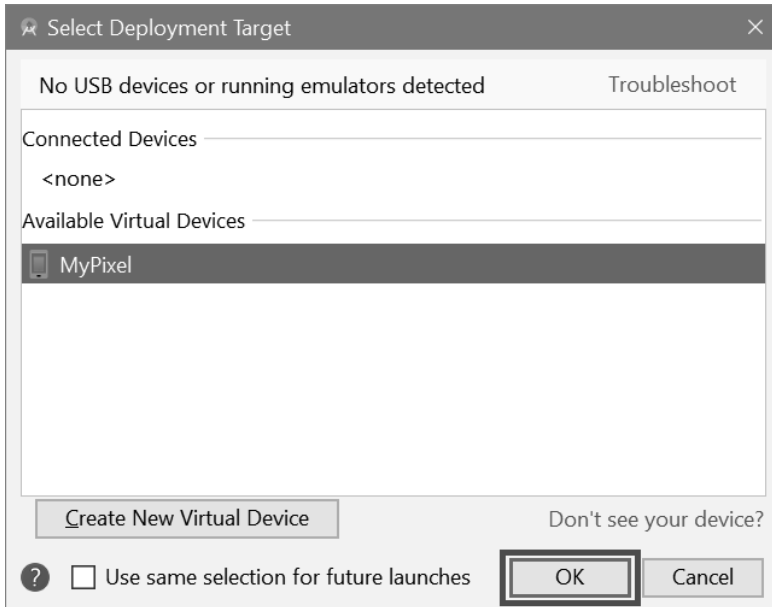
➤ Klicke nun auf NEXT.

Und damit bist du fast fertig.



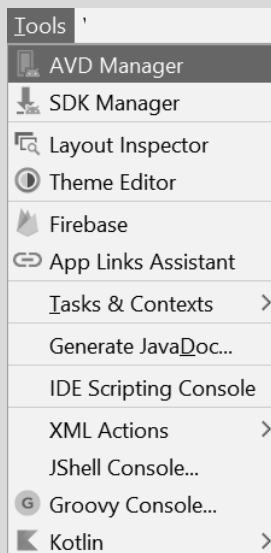
- Im letzten Dialogfeld kannst du für den Emulator (AVD) noch einen eigenen Namen eingeben, wenn du willst. Klicke abschließend auf FINISH.

Und damit landest du wieder am Anfang. Nun hast du eine Liste mit einem neuen Emulator. (Diese Liste kannst du natürlich später beliebig erweitern.)



- Um das Ganze abzuschließen, klicke auf OK.

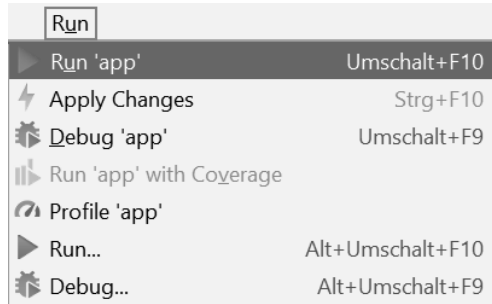
Der AVD Manager lässt sich auch über das TOOLS-Menü starten:



DIE EMULATION STARTEN

Und nun wollen wir schauen, ob wir unser erstes Programm zum Laufen kriegen.

- Dazu klickst du wieder im RUN-Menü auf RUN 'APP' (oder auf den grünen Pfeil in der Symbolleiste).



Das letzte Dialogfeld öffnet sich erneut.

- Klicke hier einfach (noch mal) auf OK. Und warte.

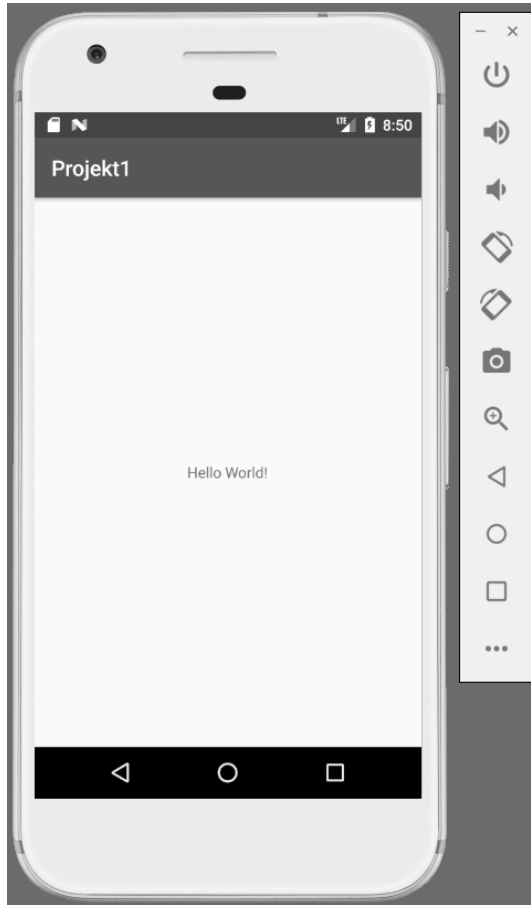
Kurz darauf tut sich ein weiteres, diesmal eindrucksvoll großes bzw. hohes Fenster auf, das wie ein Smartphone aussieht.



Und nun kann es viele Minuten dauern, bis das Android-System sich »aufgerappelt« hat. Das ist so ähnlich, als wenn du dein Smartphone komplett ausgeschaltet hast und neu startest. Also ist Geduld gefragt. (Aber die hast du ja bis jetzt ohnehin bewiesen.)



Warten wir jetzt erst einmal gelassen, bis das Emulations-Fenster so oder ähnlich aussieht:



Obwohl wir in Sachen Programmierung noch keinen Finger gerührt haben, erscheint hier schon ein Gruß an alle Welt (auf Englisch).

Früher oder später willst du natürlich deine Apps auf dem Smartphone testen. Wie das funktioniert, steht in **Anhang B**.

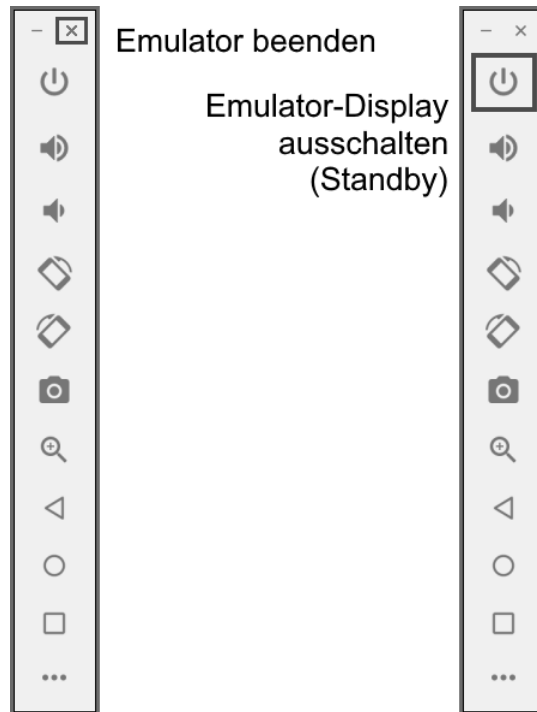


ANDROID STUDIO BEENDEN

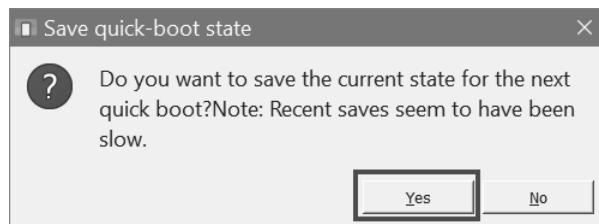
Das erste Projekt ist erstellt und gelaufen. Was will man bei diesem Aufwand mehr? Natürlich willst du mehr, doch jetzt solltest du für eine Pause erst mal Android Studio verlassen.

Das geht in zwei Schritten. Neben dem Smartphone siehst du eine Steuer-Leiste. Mit Klick auf das Symbol für POWER schaltest du das emulierte Smartphone aus.

➤ Klicke auf das kleine X ganz oben, um den Emulator zu schließen.



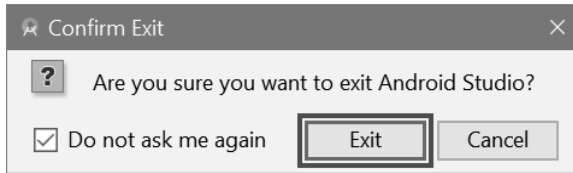
➤ Wenn dir dieses Meldedefenster begegnet, solltest du auf YES klicken.



Damit musst du beim nächsten Mal nicht mehr so lange warten, bis der Emulator gestartet ist.

- Zurück im Hauptbereich von Android Studio klickst du auf FILE und dann auf EXIT.
(Oder du klickst auch hier ganz oben rechts auf das kleine X.)

In einem weiteren Meldefenster wird noch einmal nachgefragt:



- Klicke auf EXIT. (Wenn du dieses Fenster nicht mehr sehen willst, musst du vorher DO NOT ASK ME AGAIN markieren.)

ZUSAMMENFASSUNG

Eine eigene App. Dieses Ziel haben wir hier nicht erreicht. Noch nicht. Aber du hast schon mal einen ersten Eindruck über die Arbeit mit Android Studio gewonnen. Aller Anfang ist schwer? Hier passt es, doch der erste Hürdenlauf ist geschafft.

Du weißt jetzt etwas über

- ❖ den SDK Manager, der für die Android-Entwicklungspakete zuständig ist.
- ❖ den AVD Manager, der sich um die Smartphone-Emulatoren kümmert.

Du weißt, dass du eine Activity, ein Benutzerinterface als Aktionsrahmen für dein Projekt brauchst. Und du kennst schon ein paar Operationen im Umgang mit Android Studio:

Android Studio starten	Doppelklicke auf das Symbol für Android Studio. Oder klicke auf START/AUSFÜHREN und tippe den kompletten Pfad für <i>STUDIO.exe</i> ein.
SDK Manager starten	Klicke auf TOOLS/SDK MANAGER.
AVD Manager starten	Klicke auf TOOLS/AVD MANAGER.
App-Projekt starten	Klicke auf RUN/RUN 'APP'.
Hilfesystem aufrufen	Klicke auf HELP.
Android Studio beenden	Klicke auf FILE/EXIT.

ZWEI FRAGEN ...

1. Was bedeuten SDK und AVD?
2. Was ist eine Activity?

... ABER NOCH KEINE AUFGABEN

STICHWORTVERZEICHNIS

@-Symbol 333
/* */ 234

A

Accelerometer 377
ActionBar 285
action_down 281
action_up 281
Activity 27, 76
addView 334
Alpha 199
and 138
Android 9
Android SDK 22
Android Studio 12
 beenden 38
 Hauptfenster 29
 installieren 391
 Menüs 30
 starten 18
Android Virtual Device 32
AndroidManifest 375
animate 200
Animation
 drawable 183
 duration 183
 item 182
 oneshot 183
 start 184
 stop 184
animation.xml 181
AnimationDrawable 180
animation-list 182
Ansichtsmodus 271
Anweisungsblock 131, 147
 Markierung 147
APK 407
App
 Play Store 406
 signieren 406
 Smartphone 401

app-release.apk 410
Apps 9
Argument 83
Arkustangens 228
Array 324
 Startwerte 358
atan2 227
Attributes 47
Ausrufezeichen 176, 191, 236
Ausweichen 276
 zählen 312
AVD
 Manager 21

B

Bedingung 131, 147
Benutzerinterface 27
Bibliothek 72
Bild-Button 197
Bildfeld 169
Bildformat 168
BMP 168
boolean 176
break 137, 298
Button 51
 beschriften 54
 einfügen 52, 99

C

case 135, 298
class 71
Compiler 10
Constraint 59, 337
ConstraintLayout 285, 333
Constraints 178
ConstraintSet 336
Context 241
controlLimits 381
controlPosition 215
create 366

D

Debug-Release 406
 default 137
 Dezimalzahl 198
 Dodger 271
 Doppelpunkt 136
 double 228
 do-while 148
 dp 58
 drawable 169

E

Editor 10
 EditText 127
 einfügen 125
 Eigene Klasse 237
 Eigene Komponente 331
 Eigene Methode 215
 Ein-Aus-Schalter 172
 Eingabefeld 124
 einfügen 125
 else 132
 Emulator 32
 Tablet 288
 Entwicklungsumgebung 10
 event.values 378
 extends 117, 240

F

Fading 196
 Fallunterscheidung 135
 false 176
 Fehler 30, 67, 104, 114, 118, 159, 204, 236,
 240, 248, 251, 332, 375
 Feld 324
 Feldelement 325
 final 118
 findViewById 75
 FIT_XY 276
 Float
 valueOf 148
 float 145
 for 340
 Funktion 212, 279

G

GameView 239, 274
 getAction 281
 getAltitude 299
 getBottom 308
 getContact 307, 310
 getContext 334
 getDefaultSensor 377
 getDirection 229, 326
 getDisplayMetrics 205
 getDistance 260, 327
 getDrawable 184
 getLayoutParams 287, 299
 getLeft 308
 getMedian 279
 getProgress 154
 getResources 205
 getRight 308
 getSystemService 374, 382
 getText 128
 getTop 308
 getX 211
 getY 211
 GIF 168
 Globale Variable 114
 GONE 177
 Google 9

H

Hallo-Projekt 42
 Handler 234, 305
 heightPixels 205
 Hintergrund
 Bildfeld 208
 Farbe 215
 Hintergrundmusik 373
 Hochformat 271

I

ID 64, 75
 Identifikations-Kennzeichen 64
 if-Struktur 128, 132
 ImageButton 197
 ImageView 169
 implements 378
 import 72

Index 325
Initialisierung 341
Installation 387, 391
Integer
 valueOf 129
INVISIBLE 177
isChecked 176, 185
isKilled 349
isPlaying 370

J

Java 11, 70
 installieren 387
JDK 387
JKS 409
JPG 168
JRE 387
JVM 10

K

Kapselung 249
killObject 255, 348
Klammer 138
 geschweifte {} 71
Klammern
 geschweifte {} 136
Klasse 71
 eigene 237
Kollision 307
Kommentar 72, 234, 327
Komponente 52
 einfügen 52, 102
 kopieren 101
 selbst erzeugen 331
 umbenennen 323
 Zentrieren 313
Komponete
 verschieben 52
Konstante 74, 177, 279, 296
Konstruktor 241, 250
Kontrollstruktur 132, 135, 147, 297, 340
Kotlin 11

L

Landscape 272
Laufzeitfehler 252, 375

Layout 105, 284
 Breite 150
 Container 332
 Höhe 150
 Rand 150
layout_height 64, 285
Layout-ID 333
LayoutParams 286, 296, 328
layout_width 64, 285
leftMargin 299
Lokale Variable 114

M

makeText 319
margin 65
marginBottom 307
marginEnd 307
marginStart 307
marginTop 307
match_constraint 150
match_parent 338
Math 227
Mathe-Projekt 93
MediaPlayer 366
Methode 71
 eigene 215
 mit Typ 212
 typlos 212
Mittellinie 279
MotionEvent 281
moveObject 233, 254, 304, 327
Multitask 350

N

new 79, 109, 235, 241, 325
nextInt 110
Nicht-Operator 176, 236

O

Objekt 52
 Array 324
 ausblenden 255
 bewegen 179, 202, 221, 253, 304, 327
 drehen 226
 einblenden 262
 erneuern 354

Größe 329
 killen' 255
 Kollision 307
 Maße 329
 Richtung 227
 steuern 208, 277
 Strecke 260
 zeigen 298
 Zufallsziel 204
 Objektfeld 324
 Oder-Operator 139
 onClick 79
 onClick-Array 346
 OnClickListener 79
 onCreate 76
 onDestroy 372
 onProgressChanged 158
 OnSeekBarChangeListener 158
 onSensorChanged 378
 onTouch 209
 onTouchListener 209
 Operator
 - 113
 ! 176, 191
 != 139
 . 113
 * 113
 / 113
 & 139
 + 113
 <> 139
 = 113
 == 139
 > 139
 || 139
 nicht 191
 oder 139
 Rechnen 113
 Referenz 64
 Umkehr 191
 und 139
 Verbindung 80
 Vergleichen 139
 Verknüpfen 139
 Zuweisung 110
 or 139
 Oracle 387

P

package 71
 padding 285
 Parameter 83
 Permission 375
 Pi 228
 Pixel 58
 Platzhalter 74
 Play Store 406
 playSound 367, 370
 PNG 168
 Portrait 272
 Position
 Pythagoras 259
 Start 226
 Ziel 203, 211, 226
 post 236
 postDelayed 235
 prepare 367
 private 117, 249
 Programmieren 10
 Programmiersprache 11
 Projekt
 Hallo 42
 importieren 65
 kopieren 86, 223, 322
 Mathe 93
 Zensur 122
 protected 117
 public 117, 249
 Pythagoras 259

Q

Quadratwurzel 260
 Quelltext 42
 Querformat 271

R

R.id 75
 R.string 83
 Random 109, 207, 232
 raw 367, 369
 Rechenoperator 113
 Referenz-Operator 64
 registerListener 377, 383
 release 367

- removeCallbacks 236, 361
- Ressource 64
 - Bild kopieren 179
 - ID finden 75
 - Sound 367
 - String benutzen 83
 - String einfügen 60
 - XML erstellen 180
 - XML holen 184
- restoreObject 263, 355
- return 212, 347
- rotate 226
- Rotation 227
- Rückgabe 279
- run 235
- Runnable 235, 305, 355
- Runtime Error 252, 375

S

- Scale 199
- ScaleType 276
- Schalter 172
- Schaltfläche 51
 - einfügen 52
- Schaltvariable 176, 236
- Schieberegler 151
- Schleife 147
- Schrägstriche 72
- SDK
 - Manager 21
 - Versionen 23
- SeekBar 150
- seekTo 370
- Sensormanager 376
- SENSOR_SERVICE 376
- setAlpha 199
- setDelay 258
- setDestination 232
- setDuration 200
- setImageResource 192
- setImageView 334
- setLayoutParams 287, 298
- setLooping 373
- setMargins 307
- setMax 153
- setObject 297
- setOnClickListener 79
- setOnSeekBarChangeListener 158

- setOnTouchListener 209
- setPosition 304
- setProgress 157
- setRotation 226
- setScaleType 276
- setScaleX 199
- setScaleY 199
- setSound 366
- setText 79
- Setup Wizard 19, 396
- setVisibility 177
- show 319
- showObject 278, 298, 336
- Sign-App 406
- Software Development Kit 23
- Sound
 - Ressource 367
- Source 180
- sp 59
- Speichern
 - alles 42
 - automatisch 42
- sqrt 260
- start 367
- static final 280
- stop 367
- stopSound 367
- String 49
 - Ressource 60, 98
 - valueOf 111
- string name 86, 128
- strings.xml 85, 104, 126, 144
- super 77, 241
- switch 135, 297
- switch-Struktur 135
- Syntax 134
- Syntaxfehler 134

T

- Tangens 227
- Task
 - Multi 350
- textAlignment 101
- textAppearance 49, 101
- Textfeld 52
 - beschriften 47
 - einfügen 105
- textSize 101

TextView 52
 einfügen 105
 this 235, 241
 Toast 318
 toDegrees 228
 ToggleButton 172
 topMargin 299
 toRadians 228
 toString 129
 Touch-Point 211, 278
 Treffer
 zählen 265
 true 176
 Typecasting 76
 Typisierung 76, 198, 206, 231
 Typumwandlung 76, 206

U

ueff 67
 Umbenennen 223, 411
 Und-Operator 139
 USB-Debugging 402

V

valueOf 111, 129, 148
 Variable
 globale 114
 lokale 114
 Variablenfeld 324
 Verbindungsoperator 80
 Vererbung 240
 Vergleich
 gleich 131
 ungleich 133
 Vergleichsoperator 131, 139
 Verknüpfungsoperator 139
 Verzweigung 133
 vibrate 374

Vibration 373
 Vibrator 374
 VIBRATOR_SERVICE 374
 View 80, 176
 animate 200
 ViewPropertyAnimatorObjekt 201
 Virtual Machine 10
 VISIBLE 177
 void 212

W

while 146
 while-Struktur 147
 Widget 52
 widthPixels 205
 Winkelfunktion 227
 wrap_content 64, 150, 352

X

XML 63

Z

Zahlenformat
 Of 198
 float 145, 198
 int 110
 Zählschleife 340
 Zensur-Projekt 122
 Zentrieren 313
 Zugriff
 private 117
 protected 117
 public 117
 Zugriffsmodifizierer 116
 Zuweisung 110
 Zuweisungsoperator 131
 Zweig 133