Kapitel 1
Einführung

Kotlin ist strukturiert, funktional, objektorientiert und leicht zu erlernen.

Kotlin ist eine kompakte und vielseitige Sprache, die auch für Einsteiger geeignet ist. Die erste stabile Version von Kotlin erschien im Februar 2016. Als junge Sprache bietet Kotlin den Vorteil, moderne Entwicklungen innerhalb der Programmiersprachen umsetzen zu können, ohne den Ballast einer Abwärtskompatibilität berücksichtigen zu müssen.

1.1 Kotlin und Android

Im Jahr 2017 erklärte Google Kotlin zu einer offiziellen Sprache für die Entwicklung von Apps für das Betriebssystem Android. Im Mai 2019 erfolgte der Aufstieg zur bevorzugten Sprache für diese Zwecke.

Laut Google verwendet im Mai 2019 bereits über die Hälfte der Android-Entwicklerinnen und -Entwickler Kotlin. Im Vergleich zur bisher bevorzugten Sprache Java wird weniger Code benötigt, um dieselbe Aufgabe zu lösen. Kotlin bietet zahlreiche Möglichkeiten, Android-Apps übersichtlicher und sicherer zu entwickeln.

Im Kotlin-Code kann direkt auf Java-Bibliotheken zugegriffen werden, was es u. a. ermöglicht, vorhandene Java-Projekte schrittweise und zügig auf Kotlin umzurüsten. Zudem wurden die Java-Bibliotheken für Kotlin um viele Funktionen erweitert.

Sie erlernen zunächst die *strukturierte* Programmierung in Kotlin mithilfe von kleinen, übersichtlichen Programmen. Die Prinzipien der *funktionalen* Programmierung verhelfen Ihnen zu kompaktem und dennoch leicht verständlichem Code. Es folgt der Übergang zur *objektorientierten* Programmierung, die sich für größere Projekte eignet. Gut gerüstet widmen wir uns dann der Erstellung von *Android-Apps* mit Kotlin.

1.2 Aufbau dieses Buchs

In diesem Buch wird mit zwei frei verfügbaren und weit verbreiteten Entwicklungsumgebungen gearbeitet: zunächst mit IntelliJ IDEA und anschließend mit dem darauf basierenden Android Studio.

1.2.1 Der Einstieg in Kotlin

Nach der Installation der Entwicklungsumgebung *IntelliJ IDEA Community Edition* von JetBrains wird mit den Grundlagen der strukturierten Programmierung in Kotlin begonnen, also Variablen, Datentypen, Operatoren und Kontrollstrukturen. Funktionale Anweisungen, die alternativ genutzt werden können, vermitteln bereits einen ersten Eindruck von der funktionalen Programmierung.

Darüber hinaus lernen Sie die besonderen Mittel von Kotlin zur Vermeidung von Programmabstürzen kennen. Außerdem setzen Sie Zufallsgeneratoren ein, die häufig in Spiel- und Trainingsprogrammen benötigt werden.

Anschließend geht es um die vielfältige Definition und Nutzung von Funktionen im Rahmen der Modularisierung Ihrer Programme. Mit den *anonymen Funktionen* gelingt der Einstieg in die funktionale Programmierung. Zudem wird die komfortable Fehlerfindung mithilfe des Debuggens erläutert.

Es folgt der Einsatz von vorgefertigten Klassen, Eigenschaften und Methoden zur Bearbeitung von mathematischen Berechnungen, Zeichenketten und Zeitangaben. Verschiedene Typen von Datenstrukturen dienen zur Verarbeitung von großen Datenmengen.

Die objektorientierte Programmierung und damit die Erstellung von eigenen Klassen, Objekten, Interfaces, abstrakten Klassen sowie die Prinzipien der Vererbung werden als Nächstes beschrieben. Damit vertieft sich auch das Verständnis für die vielen vorgefertigten Elemente, die Sie bereits einsetzen.

1.2.2 Apps mit dem Android Studio

Nach dem gründlichen Einstieg in Kotlin startet die Entwicklung von Android-Apps. Zunächst installieren und konfigurieren Sie die Entwicklungsumgebung *Android Studio* von Google. Anschließend lernen Sie die einzelnen Teile von Android-Apps und der zugehörigen Projekte kennen. Sie arbeiten mit realen und virtuellen Geräten, setzen verschiedene Ressourcen ein und gestalten Ihre Apps mit unterschiedlichen Layouts.

Sie lassen Ihre Apps auf Ereignisse und Aktionen der Benutzerinnen und Benutzer reagieren und setzen dabei verschiedene Formen von *Listenern* ein. Anschließend werden vielfältige Views beschrieben, die zur Bedienung Ihrer Benutzeroberflächen dienen.

Es folgen Elemente zur Automatisierung von Abläufen, wie sie in Spiel- und Trainingsprogrammen benötigt werden. Danach wird gezeigt, auf welche Weise die verschiedenen Teile einer komplexen App untereinander Daten austauschen. Sie lernen, wie die Benutzerinnen und Benutzer Ihre Apps mithilfe von Gesten, Dialogen und Menüs steuern. Die Benutzeroberflächen Ihrer Apps können sich während des Ablaufs mithilfe von Transformationen und Animationen verändern.

Sollen Ihre Apps Daten speichern, die beim nächsten Start wieder benötigt werden? Kotlin bietet dazu Datenbanken und andere Möglichkeiten. Viele Sensoren liefern Daten, die zur Steuerung Ihrer Apps eingesetzt werden können. Sie lernen diese und andere Systemzugriffe zusammen mit den dazu erforderlichen Berechtigungen kennen.

Anhand von größeren Beispielprojekten zeige ich, wie die verschiedenen Elemente, die Sie in diesem Buch kennengelernt haben, verknüpft werden können. Der Anhang bietet Informationen zur Erstellung unterschiedlicher virtueller Geräte zum Testen Ihrer Apps und beschreibt, wie Sie Ihre Apps veröffentlichen. Zudem werden die Installationen unter Ubuntu Linux und macOS beschrieben.

1.2.3 Optimierung des Codes

Ich lege Wert darauf, dass die Leserinnen und Leser ein Verständnis für die Zusammenhänge entwickeln. Daher benötigen meine Programme besonders zu Anfang des Buchs mehr Code als notwendig. Den bereits fortgeschrittenen Programmiererinnen und Programmierern unter Ihnen wird das auffallen. Außerdem macht sich das in den Code-Editoren der beiden Entwicklungsumgebungen bemerkbar. Daher werden an einigen Stellen Vorschläge zur Optimierung des Codes gemacht. Mit wachsenden Kenntnissen in der Sprache Kotlin wird der Code immer kompakter.

1.3 Installation von IntelliJ IDEA

Nachfolgend beschreibe ich die Installation der Entwicklungsumgebung IntelliJ IDEA von JetBrains. Es wird die frei verfügbare Community Edition in der im August 2021 aktuellen Version 2021.2 für Windows verwendet. Die Installationen unter Ubuntu Linux und macOS werden in Anhang C beziehungsweise Anhang D erläutert.

Über die Adresse *https://www.jetbrains.com/de-de/idea* kommen Sie zu der Seite, auf der Sie IntelliJ IDEA herunterladen können. Dort finden Sie im Bereich COMMUNITY den Button HERUNTERLADEN (siehe Abbildung 1.1). Wählen Sie die exe-Version der Installationsdatei.

Abbildung 1.1 IntelliJ IDEA herunterladen

Speichern Sie die Installationsdatei auf Ihrem PC (siehe Abbildung 1.2).

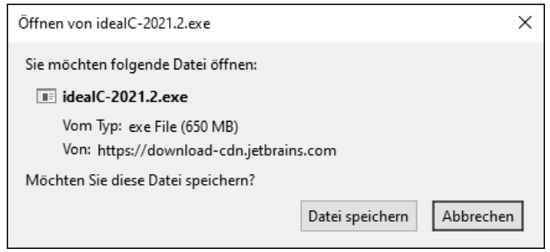

Abbildung 1.2 Installationsdatei speichern

Rufen Sie die Datei auf und übernehmen Sie den Standard-Installationspfad. Abhängig von Ihrem PC soll die 32-Bit-Version oder die 64-Bit-Version gestartet werden. Sowohl Java- als auch Kotlin-Dateien sollen anhand ihrer Dateiendungen erkannt werden (siehe Abbildung 1.3).

Starten Sie IntelliJ IDEA nach der Installation über das Icon auf dem Desktop. Es erscheint die Startseite von IntelliJ IDEA (siehe Abbildung 1.4). Über den Menüpunkt CUSTOMIZE gelangen Sie auf eine Seite, auf der Sie verschiedene Einstellungen vornehmen können. Ich habe in der Liste COLOR THEME den Eintrag INTELLIJ LIGHT für eine helle Darstellung gewählt.

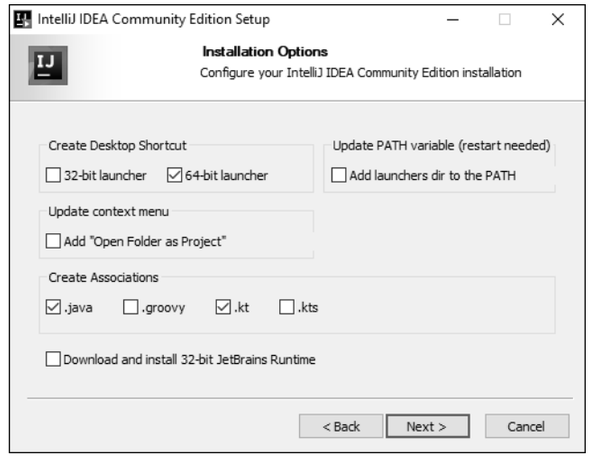

Abbildung 1.3 Installationsoptionen

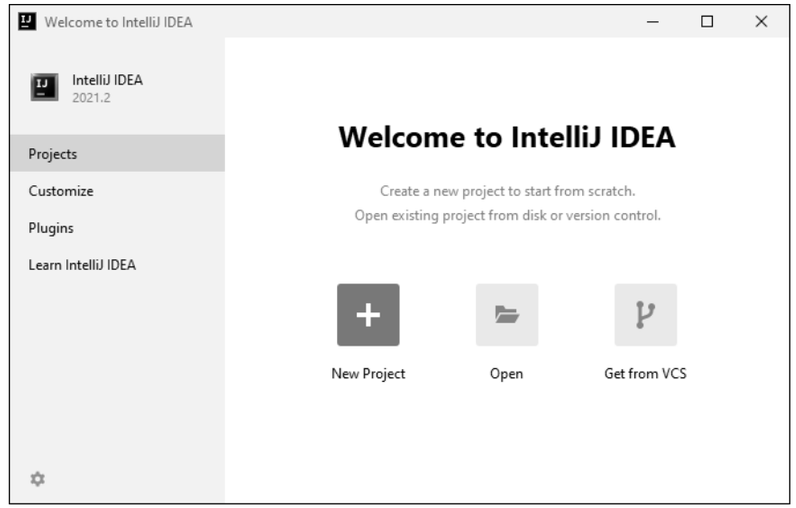

Abbildung 1.4 Startseite von IntelliJ IDEA

Links unten können Sie über das Menü EINSTELLUNGEN, erkennbar am Zahnradsymbol, den Menüpunkt CHECK FOR UPDATES aufrufen, um Ihr IntelliJ IDEA auf den neuesten Stand zu bringen. Das sollten Sie unmittelbar nach der Installation auf jeden Fall durchführen. Nach einem Update starten Sie IntelliJ IDEA neu.

1.4 Das erste Projekt

Zum Erzeugen des ersten Projekts betätigen Sie auf der Startseite von IntelliJ IDEA den Button NEW PROJECT. Im Dialogfeld NEW PROJECT (siehe Abbildung 1.5) wählen Sie auf der linken Seite den Eintrag KOTLIN aus. Als Name des Projekts habe ich *HalloWelt* gewählt. Der Name des Verzeichnisses wird entsprechend eingestellt.

Der Eintrag lautet bei mir *D:\IntelliJ_Idea_Projekte\HalloWelt*.

Wir starten mit Konsolen-Projekten, daher wird der Eintrag CONSOLE APPLICATION in der Liste PROJECT TEMPLATE gewählt. Bei dem *Build System* handelt es sich um das System zur automatisierten Erzeugung des fertigen Programms aus den einzelnen Bestandteilen. Hier genügt es, das eigene System INTELLIJ in der Liste BUILD SYSTEM auszuwählen.

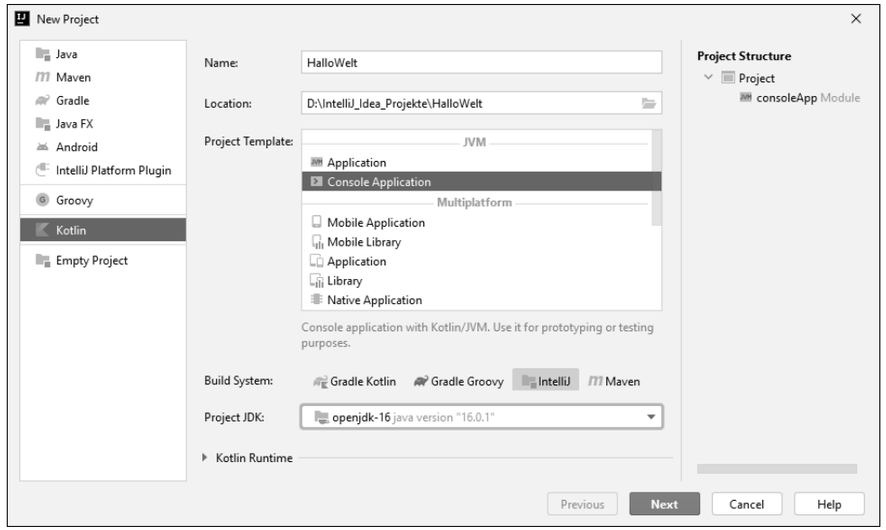

Abbildung 1.5 Projekt konfigurieren

In der Liste PROJECT JDK wird zunächst der Eintrag DOWNLOAD JDK gewählt. Im Dialogfeld DOWNLOAD JDK kann die vorgeschlagene Auswahl ORACLE OPEN JDK bestätigt werden (siehe Abbildung 1.6). Nach dem einmaligen Herunterladen und Installieren des JDK sieht es anschließend aus wie in Abbildung 1.5. In den späteren Projekten muss das *Oracle Open JDK* nur noch in der Liste PROJECT JDK ausgewählt werden.

JDK steht für *Java Development Kit*. Dabei handelt es sich um eine Sammlung von Werkzeugen und Bibliotheken zur Entwicklung von Software, die auf Java basiert und zurückgreift.

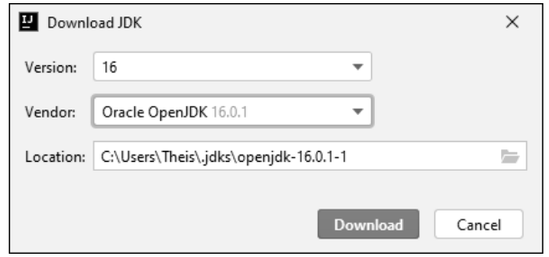

Abbildung 1.6 Herunterladen eines JDK

Nach dem Betätigen der Buttons NEXT und FINISH wird das Projekt erstellt (siehe Abbildung 1.7). Bei diesem ersten Projekt wird eine längere Phase der Konfiguration und Synchronisation durchlaufen. Das ist während dieser Zeit an der Anzeige rechts unten zu erkennen.

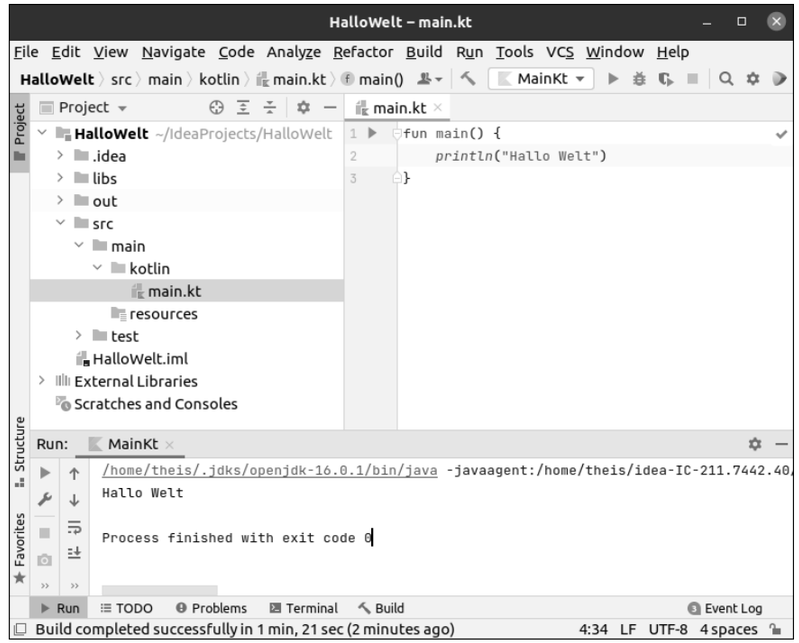

Abbildung 1.7 Erstes Projekt »HalloWelt«

Klappen Sie anschließend im Projektfenster auf der linken Seite den Verzeichnispfad bis zur Datei *src/main/kotlin/main.kt* auf. Nach einem Doppelklick auf diese Datei erscheint ihr Inhalt rechts im Codefenster, und zwar das Kotlin-Programm mit der Funktion `main()`.

Der Code der Funktion wird wie folgt geändert:

```
fun main() {
    println("Hallo Welt")
}
```

Listing 1.1 Projekt »HalloWelt«, Datei »main.kt«

Nach dem Betätigen des Menüpunkts Run • Run und des Eintrags MainKt wird das Programm übersetzt und gestartet. Das Ergebnis des Programms sehen Sie unten im Ausgabefenster in Abbildung 1.7. Mehr zum Kotlin-Code folgt in Abschnitt 2.1.

1.4.1 Die Arbeit mit Projekten

Während der Entwicklung können Sie das gesamte Projekt mithilfe des Menüpunkts File • Save All (Tastenkombination Strg + S, Mac: Cmd + S) speichern. Über den Menüpunkt File • Close Project schließen Sie das Projekt. Dabei werden alle geänderten Projektdateien automatisch gespeichert.

Ein bereits vorhandenes Projekt können Sie vom Startbildschirm von IntelliJ IDEA aus öffnen. Dazu wählen Sie den betreffenden Eintrag aus der Liste aus. Alternativ betätigen Sie den Button Open. Im darauf erscheinenden Dialogfeld Open File or Project wählen Sie das Projektverzeichnis aus dem Feld Location in Abbildung 1.5 aus.

Bei mir ist das *D:\IntelliJ_Idea_Projekte\HalloWelt*.

Die Auswahl wird vorgenommen, indem Sie die Verzeichnishierarchie Ebene für Ebene über den Pfeil neben dem jeweiligen Verzeichnisnamen aufklappen. Zum Öffnen des Projekts markieren Sie das betreffende Projektverzeichnis und betätigen den Button OK. Ein Doppelklick auf das Projektverzeichnis würde nur das Verzeichnis aufklappen, aber nicht das Projekt öffnen.

Alle Projekte in den ersten Kapiteln werden auf die in diesem Abschnitt beschriebene Weise erzeugt.

Kapitel 4
Daten strukturieren und speichern

Sie strukturieren größere Datenmengen in Ihren Programmen mithilfe verschiedener Datenstrukturen und legen sie dauerhaft in Dateien ab.

Zur Speicherung von größeren Datenmengen bietet Kotlin verschiedene Datenstrukturen mit unterschiedlichen Eigenschaften. Sie beinhalten meist Daten desselben Typs, also zum Beispiel nur `Int`-Werte, nur `Double`-Werte oder nur `String`-Werte. In diesem Kapitel werden einige Möglichkeiten der beiden Datenstrukturen *Array* und *ArrayList* vorgestellt. In Kapitel 7 werden die Kenntnisse zu den Datenstrukturen vertieft.

Es schließt sich die Speicherung von Daten in Textdateien an.

4.1 Array

Ein Array stellt eine einfache Datenstruktur dar. Kotlin bietet allgemeine Arrays, die für jeden Datentyp geeignet sind. Für bestimmte Datentypen stehen eigene Typen von Arrays mit zugeschnittenen Eigenschaften und Methoden zur Verfügung. Falls vorhanden, sollten diese bevorzugt werden.

Arrays bieten im Vergleich zu anderen Datenstrukturen eine schnelle Bearbeitung der Daten. Im Projekt *StrukturArray* folgt eine Einführung.

4.1.1 Allgemeine Arrays

Es folgt der erste Teil des Programms:

```
fun main() {
    val a1 = arrayOf(7, -15, 23, 8)
    println("Array<Int>: " + a1.joinToString())

    val a2 = arrayOf(4.7, -9.8, 13.2)
```

```
    println("Array<Double>: " + a2.joinToString() + "\n")
    ...
}
```

Listing 4.1 Projekt »StrukturArray«, Teil 1 von 4

Die Funktion `arrayOf()` liefert einen allgemeinen Array. Die einzelnen Werte innerhalb eines Arrays heißen Elemente. Die Parameter der Funktion werden zu den Elementen des Arrays. Haben sie alle denselben Typ, entsteht ein allgemeiner Array dieses Typs. Die Variable `a1` verweist auf einen allgemeinen Array aus `Int`-Werten. Dabei handelt es sich um ein Objekt der Klasse `Array<Int>`.

Die Methode `joinToString()` kann auf unterschiedliche Datenstrukturen angewandt werden. Sie verbindet die einzelnen Elemente mithilfe einer Trennzeichenfolge zu einer Zeichenkette, die für eine einfache Ausgabe geeignet ist. Die Trennzeichenfolge, auch Separator genannt, besteht standardmäßig aus einem Komma und einem Leerzeichen. Sie können der Methode `joinToString()` auch einen anderen Separator als Parameter übergeben.

Die Variable `a2` verweist auf einen allgemeinen Array aus `Double`-Werten. Dabei handelt es sich um ein Objekt der Klasse `Array<Double>`.

Die Ausgabe dieses Programmteils:

```
Array<Int>: 7, -15, 23, 8
Array<Double>: 4.7, -9.8, 13.2
```

Bei `Array<Int>` und `Array<Double>` handelt es sich um sogenannte generische Klassen. Sie bieten einheitliche Eigenschaften und Methoden für Variablen des angegebenen Datentyps.

4.1.2 Besondere Typen von Arrays

Nachfolgend sehen Sie den zweiten Teil des Programms:

```
fun main() {
    ...
    val b = intArrayOf(3, -28, 12, -4, 7)
    println("IntArray: " + b.joinToString())
    println("Anzahl der Elemente: " + b.size)
    b[4] = -11
```



```
    println("Einzelne Elemente: ${b[0]} ${b[4]}")
    print("for-in mit Index: ")
    for (x in b.indices)
        print("$x:${b[x]} ")
    println("\n")
    ...
}
```

Listing 4.2 Projekt »StrukturArray«, Teil 2 von 4

Die Funktion `intArrayOf()` liefert ein Objekt der Klasse `IntArray`. Ein solcher Array ist besonders für `Int`-Werte geeignet. Die Eigenschaft `size` beinhaltet die Größe eines Arrays, also die Anzahl seiner Elemente.

Die Elemente eines Arrays werden mithilfe des sogenannten Index nummeriert. Die Nummerierung beginnt bei 0 und ist fortlaufend. Der Index innerhalb von rechteckigen Klammern ermöglicht den Zugriff auf ein einzelnes Element. Die Variable `b` verweist auf einen Array mit fünf Elementen. Das erste Element wird mit `b[0]`, das letzte mit `b[4]` erreicht.

Hier wird ein komplexerer Ausdruck, zum Beispiel `b[0]`, mithilfe des Zeichens `$` in eine Zeichenkette eingebettet. Dabei muss der Ausdruck in geschweifte Klammern gesetzt werden.

Sie können einzelnen Elementen eines Arrays neue Werte zuweisen. Lassen Sie sich nicht davon irritieren, dass die Variable `b` unveränderlich ist. Das bedeutet nur, dass Sie dieser Variablen kein neues Array zuweisen können. Die einzelnen Elemente des Arrays können verändert werden.

Dies gilt allgemein für Variablen, die auf Objekte einer Klasse verweisen: Handelt es sich um eine unveränderliche Variable, können Sie dieser kein neues Objekt zuweisen. Sie können aber einzelne Eigenschaften des Objekts verändern.

Die `for-in`-Schleife eignet sich für den Zugriff auf alle Elemente eines Arrays. Die Eigenschaft `indices` liefert einen Zahlenbereich vom ersten bis zum letzten Index.

Es folgt die Ausgabe des zweiten Programmteils:

```
IntArray: 3, -28, 12, -4, 7
Anzahl der Elemente: 5
Einzelne Elemente: 3 7
for-in mit Index: 0:3 1:-28 2:12 3:-4 4:7
```

4.1.3 Arrays bestimmter Größe

Es folgt der dritte Teil des Programms:

```
fun main() {
    ...
    val c = IntArray(10)
    for (x in c.indices)
        c[x] = (1 .. 6).random()
    println("Mit Größe erzeugt: " + c.joinToString())
    ...
}
```

Listing 4.3 Projekt »StrukturArray«, Teil 3 von 4

Ein Objekt der Klasse `IntArray` kann mithilfe eines Konstruktors erstellt werden, dem die gewünschte Größe des Arrays übergeben wird. Die Variable `c` verweist auf ein Objekt der Klasse `IntArray` mit 10 Elementen. Alle Elemente besitzen zu Beginn den Wert 0. Mithilfe der Methode `random()` erhalten sie in der `for-in`-Schleife einen zufälligen Würfelwert.

Die Ausgabe des dritten Teils des Programms:

```
Mit Größe erzeugt: 6, 2, 2, 6, 3, 5, 6, 2, 6, 5
```

4.1.4 Arrays vergrößern

Es folgt der vierte und letzte Teil des Programms:

```
fun main() {
    ...
    var d = intArrayOf()
    for (x in 1 .. 10)
        d += (1 .. 6).random()
    println("Umspeicherung: ${d.joinToString()}")
}
```

Listing 4.4 Projekt »StrukturArray«, Teil 4 von 4

Bleiben die Klammern beim Aufruf der Funktion `intArrayOf()` leer, wird ein leeres Objekt der Klasse `IntArray` ohne Elemente erzeugt.

Wird auf einen Array mit einer veränderlichen Variablen verwiesen, können Sie ihn mithilfe des Operators `+=` um einzelne Elemente vergrößern. Hier wird das innerhalb einer

Schleife durchgeführt, die zehnmal durchlaufen wird. Anschließend besitzt das Array `d` zehn Elemente.

Sie können auch den Aufruf einer Methode, die einen Wert liefert, mithilfe des Zeichens `$` unmittelbar in eine Zeichenkette einbetten. Dabei müssen ebenfalls geschweifte Klammern verwendet werden.

Die Vorgehensweise in diesem Abschnitt hat einen großen Nachteil. Der Operator `+=` vermittelt den Eindruck, dass der Array veränderbar wäre. Das trifft nicht zu. Stattdessen wird viel Aufwand betrieben: Es wird ein neuer Array erzeugt und gespeichert, der aus den Inhalten des alten Arrays und einem weiteren Element besteht.

Benötigen Sie eine veränderbare Datenstruktur mit ähnlichen Eigenschaften wie ein Array, sollten Sie stattdessen eine ArrayList verwenden (siehe nachfolgender Abschnitt).

4.2 ArrayList

Eine ArrayList bietet ähnliche Zugriffsmöglichkeiten wie ein Array, ist aber flexibler. Sie ist dazu konzipiert, Elemente hinzufügen oder entfernen zu lassen. In diesem Abschnitt sehen Sie im Projekt *StrukturArrayList* einige Möglichkeiten von ArrayLists.

4.2.1 Elemente hinzufügen

Es folgt der Code des ersten Programmteils:

```
fun main() {
    val a = arrayListOf(5, -12)
    println("Original: $a")

    a += 8
    println("mit +=: $a")
    a.add(13)
    println("add() am Ende: $a")
    a.add(1, 26)
    println("add() in Liste: $a")
    val b = arrayListOf(8, 17)
    a.addAll(b)
    println("addAll(): $a")
    ...
}
```

Listing 4.5 Projekt »StrukturArrayList«, Teil 1 von 2

Mithilfe der Funktion `arrayListOf()` und ganzzahligen Parametern erstellen Sie ein Objekt der allgemeinen Klasse `ArrayList<Int>`. Der Operator `+=` dient dem Hinzufügen eines Elements am Ende der ArrayList.

Mithilfe der Methode `add()` können Sie ebenfalls Elemente hinzufügen. Sie ist etwas flexibler und kann auf zwei verschiedene Arten aufgerufen werden:

- Hat sie nur einen Parameter, handelt es sich dabei um den Wert, der am Ende der ArrayList angehängt wird.
- Hat sie zwei Parameter, wird mit dem ersten Parameter der Index desjenigen Elements angegeben, vor dem der zweite Parameter als neuer Wert in der ArrayList eingefügt wird.

Eine Funktion oder Methode, die auf mehrere Arten aufgerufen werden kann, wird *überladen* genannt (mehr dazu in Abschnitt 5.1.6). Mithilfe der Methode `addAll()` können Sie sämtliche Elemente einer anderen ArrayList am Ende der aktuellen ArrayList hinzufügen.

Die Ausgabe dieses Programmteils:

```
Original: [5, -12]
mit +=: [5, -12, 8]
add() am Ende: [5, -12, 8, 13]
add() in Liste: [5, 26, -12, 8, 13]
addAll(): [5, 26, -12, 8, 13, 8, 17]
```

Wie für Arrays gibt es die Methode `joinToString()` zur Verbindung der Elemente zu einer Zeichenkette, die ausgegeben wird. Allerdings kann eine ArrayList auch unmittelbar in einer lesbaren Form ausgegeben werden. Sie wird dabei in rechteckige Klammern eingebettet.

Im vorliegenden Programm verweist eine unveränderliche Variable auf die ArrayList. Das bedeutet, dass Sie dieser Variablen keine neue ArrayList zuweisen können. Allerdings können Sie die Elemente der ArrayList ändern. Ebenso können Sie Elemente hinzufügen oder entfernen.

4.2.2 Elemente entfernen

Es folgt der zweite Teil des Codes:

```
fun main() {
    ...
```



```
    a.remove(8)
    println("\nremove(): $a")
    a.removeAt(3)
    println("removeAt(): $a")
    a -= 26
    println("mit -=: $a")

    if(a.isNotEmpty())
        println("Nicht leer")
    a.clear()
    println("clear(): $a")
    if(a.isEmpty())
        println("Leer")
}
```

Listing 4.6 Projekt »StrukturArrayList«, Teil 2 von 2

Die Methode `remove()` entfernt das erste Element aus der ArrayList, das den angegebenen Wert besitzt. Mithilfe der Methode `removeAt()` entfernen Sie das Element mit dem angegebenen Index. Die Methode `clear()` entfernt alle Elemente der ArrayList. Die Methoden `isEmpty()` und `isNotEmpty()` prüfen, ob eine ArrayList leer beziehungsweise nicht leer ist.

Die Ausgabe des zweiten Programmteils:

```
remove(): [5, 26, -12, 13, 8, 17]
removeAt(): [5, 26, -12, 8, 17]
mit -=: [5, -12, 8, 17]
Nicht leer
clear(): []
Leer
```

4.3 Speichern und Lesen in Textdateien

Objekte der Klasse `File` ermöglichen das dauerhafte Speichern von Daten in Textdateien und das Lesen von Daten aus Textdateien. Im Projekt *KonsoleSpeichern* lernen Sie verschiedene Methoden zum Speichern und Lesen kennen.

4.3.1 Speichern in Textdateien

Es folgt Teil 1 des Programms:

```
import java.io.*
fun main() {
    val datei = File("datei.txt")
    datei.writeText("Das ist weg\n")
    datei.writeText("Erste Zeile\n")
    datei.appendText("abc 123 äöü ÄÖÜ ß\n")
    for(i in 1 .. 8)
        datei.appendText("$i ")
    datei.appendText("\n")
    for(i in 1 .. 8)
        datei.appendText("" + i/10.0 + " ")
    datei.appendText("\n")
    ...
}
```

Listing 4.7 Projekt »KonsoleSpeichern«, Teil 1 von 2

Objekte von Klassen werden standardmäßig mithilfe von besonderen Methoden erzeugt, den Konstruktor-Methoden (kurz: Konstruktoren). Viele Klassen bieten mehrere Konstruktoren an. Damit ist die Erstellung eines Objekts auf mehrere Arten möglich. Mehr zu Klassen und Objekten folgt in Kapitel 8.

Ein Objekt der Klasse `File` aus dem Paket `java.io` dient zum Schreiben von Daten in eine Datei und zum Lesen von Daten aus einer Datei. Die Datei wird standardmäßig UTF-8-kodiert. Damit können zum Beispiel auch deutsche Umlaute oder das Eszett gespeichert werden. Ein Konstruktor der Klasse `File` erwartet eine Zeichenkette, die den Namen einer Datei beinhaltet. Wird kein Pfad angegeben, befindet sich die Datei im Hauptverzeichnis des Kotlin-Projekts.

Nach der Erstellung des Objekts der Klasse `File` können Sie mithilfe der Methode `writeText()` Daten in die Datei schreiben. Die Methode erwartet eine Zeichenkette. Existiert die Datei bereits, wird der alte Inhalt vollständig entfernt und mit den neuen Daten überschrieben. Im Unterschied dazu hängt die Methode `appendText()` Daten ans Ende der Datei an. Die bereits vorhandenen Daten werden in diesem Fall ergänzt.

Hier werden folgende Daten in der Datei gespeichert:

- Text mit Buchstaben, Ziffern, deutschen Umlauten und dem Eszett
- Zeilenumbrüche

- eine Reihe von ganzen Zahlen
- eine Reihe von Zahlen mit Nachkommastellen

In Abbildung 4.1 sehen Sie den Inhalt der Datei in einem Texteditor.

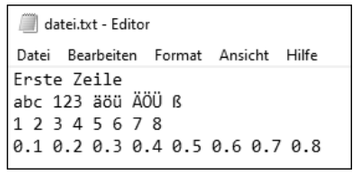

Abbildung 4.1 Inhalt der Textdatei

4.3.2 Lesen aus Textdateien

Es folgt Teil 2 des Programms:

```
fun main() {
    ...
    println("Gesamte Datei:")
    val inhalt = datei.readText()
    println(inhalt)

    println("Liste mit einzelnen Zeilen:")
    val liste = datei.readLines()
    println("Anzahl: " + liste.size)
    for(x in liste.indices)
        println("$x: ${liste[x]}")
}
```

Listing 4.8 Projekt »KonsoleSpeichern«, Teil 2 von 2

Die Methode `readText()` liest den gesamten Inhalt der Datei und speichert sie inklusive der Zeilenumbrüche in einer einzelnen `String`-Variablen. Diese wird hier anschließend ausgegeben.

Die Methode `readLines()` liest ebenfalls alle Zeilen der Datei und speichert sie in einer Datenstruktur des Typs `List<String>`, also einer Liste von Strings, und zwar ohne die Zeilenumbrüche. Die Eigenschaft `size` beinhaltet die Anzahl der Zeilen. Hier wird jede Zeile mit Index mithilfe einer `for-in`-Schleife und der Eigenschaft `indices` ausgegeben.

Die Ausgabe dieses Programmteils:

```
Gesamte Datei:
Erste Zeile
abc 123 äöü ÄÖÜ ß
1 2 3 4 5 6 7 8
0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8

Liste mit einzelnen Zeilen:
Anzahl: 4
0: Erste Zeile
1: abc 123 äöü ÄÖÜ ß
2: 1 2 3 4 5 6 7 8
3: 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8
```

Kapitel 10
Entwicklung von Android-Apps

Mit der ersten eigenen Android-App lernen Sie die Entwicklungsumgebung Android Studio kennen.

Sie installieren das Android Studio und lernen seine wichtigsten Elemente anhand des ersten Android-Projekts kennen. Sie erstellen ein virtuelles Gerät und testen Ihre Apps darauf. Zudem schließen Sie Ihr Smartphone an Ihren Entwicklungs-PC an und testen Ihre Apps auf diesem realen Gerät.

10.1 Installation von Android Studio

Bei *Android Studio* handelt es sich um die frei verfügbare Entwicklungsumgebung von Google zur Erstellung von Android-Apps. Sie basiert auf der Entwicklungsumgebung IntelliJ IDEA, die Sie bisher in diesem Buch verwendet haben.

Nachfolgend beschreibe ich die Installation von Android Studio in der im August 2021 aktuellen Version 2020.3.1 für Windows. Die Installationen unter Ubuntu Linux und macOS werden in Anhang C beziehungsweise Anhang D erläutert.

Über die Adresse *https://developer.android.com/studio* können Sie das Android Studio herunterladen. Dort finden Sie den Button Download Android Studio (siehe Abbildung 10.1). Speichern Sie die Installationsdatei auf Ihrem PC (siehe Abbildung 10.2). Rufen Sie die heruntergeladene Installationsdatei auf.

Es folgen mehrere Set-up-Schritte, bei denen die vorgegebenen Einstellungen bestätigt werden können. Unter anderem kann wie in Abbildung 10.3 eingestellt werden, dass zusätzlich zum Android Studio ein Android Virtual Device (AVD) eingerichtet wird. Dabei handelt es sich um ein virtuelles Gerät, auf dem Sie Ihre Apps testen können. Sie können weitere virtuelle Geräte einrichten (siehe Abschnitt 10.4) und Ihre Apps zudem auf realen Geräten testen (siehe Abschnitt 10.5).

Abbildung 10.1 Android Studio herunterladen

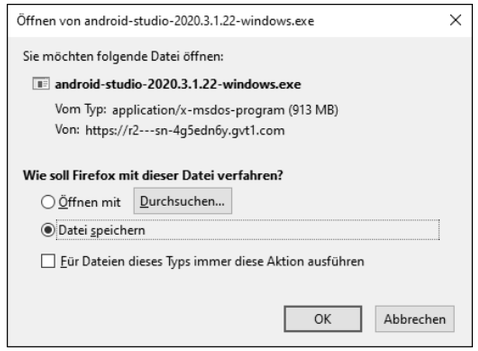

Abbildung 10.2 Installationsdatei speichern

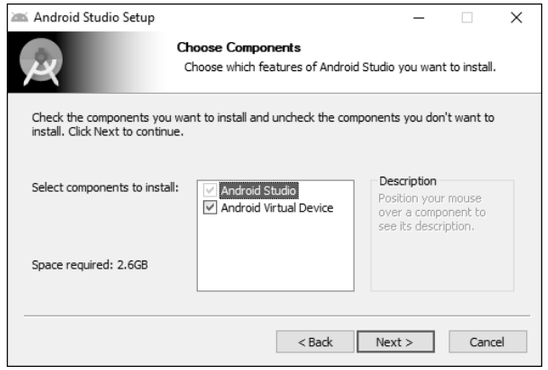

Abbildung 10.3 AVD einrichten lassen

Am Ende der Installation können Sie die Startseite von Android Studio aufrufen (siehe Abbildung 10.4).

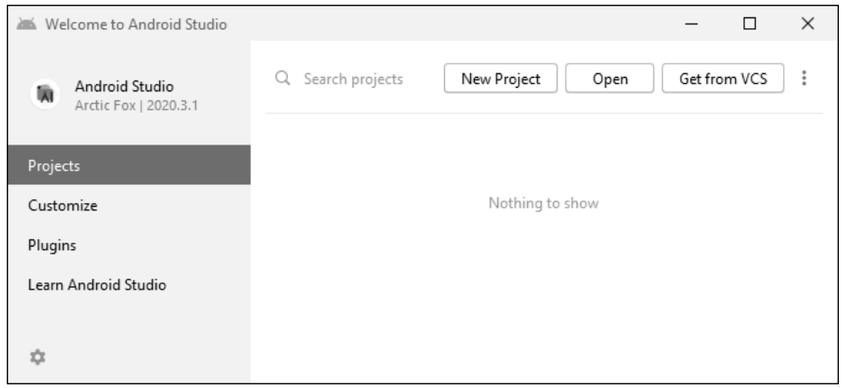

Abbildung 10.4 Startseite von Android Studio

Links unten können Sie über das Menü EINSTELLUNGEN, erkennbar am Zahnradsymbol, den Menüpunkt CHECK FOR UPDATES aufrufen, um Ihr Android Studio auf den neuesten Stand zu bringen. Das sollten Sie unmittelbar nach der Installation auf jeden Fall durchführen. Nach einem Update starten Sie das Android Studio neu.

10.2 Die erste App

Die erste einfache Android-App hat den Namen *HalloWelt* und zeigt in der Mitte des Bildschirms den Text »Hello World!« an. Ähnlich wie in IntelliJ IDEA dient ein solches einfaches Projekt häufig als erstes Beispiel zum Kennenlernen einer Entwicklungsumgebung.

Zur Erstellung eines neuen Projekts beginnen wir auf dem Startbildschirm von Android Studio, wie er nach der Installation erscheint (siehe Abbildung 10.4). Wären bereits fertige Projekte vorhanden, würden sie auf der linken Seite in einer Liste angezeigt werden. Zur Erstellung eines neuen Projekts betätigen Sie den Button NEW PROJECT.

10.2.1 Projekttyp und Activity auswählen

Es erscheint das Dialogfeld mit dem Namen NEW PROJECT (siehe Abbildung 10.5). Hier wählen Sie den Typ des Zielgeräts und das *Template* (deutsch: Vorlage) für Ihr Projekt

aus. In diesem Buch werden Projekte für Smartphones und Tablets erstellt. Daher bleiben wir auf der Registerkarte PHONE AND TABLET.

Eine *Activity* beinhaltet das Aussehen und den Programmcode einer einzelnen Benutzeroberfläche der App. Android-Apps bestehen aus einer Haupt-Activity und ggf. weiteren Activitys. Das Dialogfeld bietet Ihnen bereits mehrere Activity-Vorlagen, die ihrerseits schon viele Elemente enthalten. Die Vorlage EMPTY ACTIVITY beinhaltet nur diejenigen vorgefertigten Elemente, die notwendig sind, um eine lauffähige App zu erzeugen. Sie ist daher zum Kennenlernen der Entwicklungsumgebung gut geeignet.

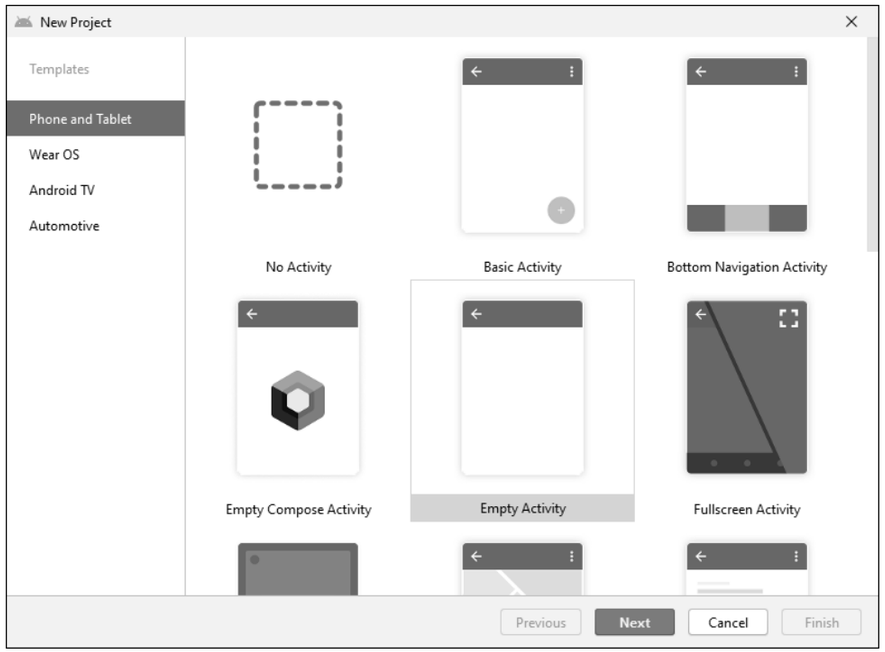

Abbildung 10.5 Projekttyp und Activity auswählen

10.2.2 Projekt konfigurieren

Nach der Betätigung des Buttons NEXT erscheint das Dialogfeld NEW PROJECT • EMPTY ACTIVITY. Hier konfigurieren Sie das neue Projekt (siehe Abbildung 10.6). Das erste Projekt trägt den Namen `HalloWelt` (ohne Leerzeichen). Diesen Namen tragen Sie in das Feld NAME ein.

Der Eintrag im Feld PACKAGE NAME sorgt für eine eindeutige Bezeichnung des Pakets, das das neue Projekt beinhaltet. Er wird automatisch mithilfe des Eintrags im Feld NAME erzeugt und beinhaltet nur Kleinbuchstaben.

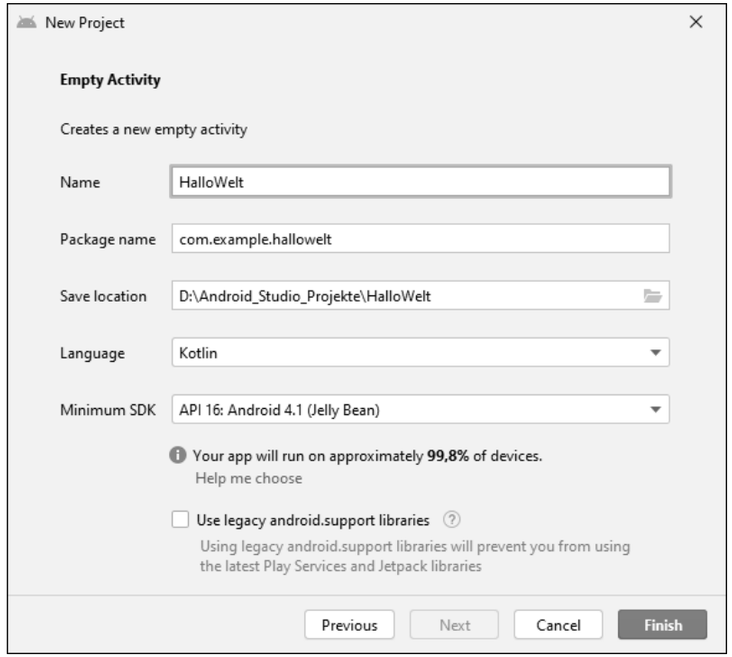

Abbildung 10.6 Projekt konfigurieren

Der Eintrag im Feld SAVE LOCATION für das Projektverzeichnis wird ebenfalls automatisch mithilfe des Eintrags im Feld NAME erzeugt. Sie können ihn übernehmen oder an Ihre Anforderungen anpassen.

Der Eintrag lautet bei mir *D:\Android_Studio_Projekte\HalloWelt*.

Im Feld LANGUAGE wählen Sie die Sprache `Kotlin` aus, damit die Unterstützung und die Prüfung für diese Sprache aktiviert wird.

Beim Android SDK handelt es sich um das *Android Software Development Kit*. Das ist eine Sammlung von Werkzeugen und Bibliotheken zur Entwicklung von Android-Software.

Im Feld MINIMUM SDK müssen Sie diejenige Android-Version mit dem zugehörigen API-Level auswählen, für die Ihre App mindestens geeignet sein soll. Der Begriff *API* steht

für *Application Programming Interface* (dt.: *Schnittstelle für die Anwendungsprogrammierung*). Für die Projekte dieses Buchs habe ich standardmäßig die Android-Version 4.1 mit der API 16 ausgewählt. Aktuell (im August 2021) läuft Ihre App damit auf 99,8 % der verwendeten Geräte.

Die Programmierung für Android wird ständig weiterentwickelt. Bestimmte Sprachelemente stehen erst ab neueren APIs zur Verfügung. Daher hat eine Änderung im Feld MINIMUM SDK sowohl Vorteile als auch Nachteile. Wählen Sie zum Beispiel die API 21 statt der API 16, können Sie in Ihrem Projekt modernere Elemente nutzen, die erst mit einer der APIs von 17 bis 21 eingeführt wurden. Allerdings erreichen Sie mit der dann entstandenen App zurzeit nur 94,1 % aller Geräte. Die Statistik, aus der die Anteile berechnet werden, wird ständig aktualisiert.

10.2.3 Synchronisierung, Fehler und Updates

Nach der Betätigung des Buttons FINISH wird das Projekt erstellt. Nach kurzer Zeit erscheint der Hauptbildschirm von Android Studio mit dem geöffneten Projekt *HalloWelt* (siehe Abbildung 10.7).

Zunächst werden einige automatisierte Vorgänge zur *Synchronisierung* der verschiedenen Elemente des Projekts durchgeführt. Das erkennen Sie an den sich laufend verändernden Anzeigen unten im Android Studio. Beachten Sie besonders die Anzeige rechts unten in der Statuszeile. Diese Synchronisierung wird Ihnen im Verlauf der Projektentwicklung immer wieder begegnen. Ich empfehle Ihnen, bei unerwarteten Verzögerungen die genannten Anzeigen zu beachten.

Trotz einer vollständigen Installation kann es bei der Erstellung von Projekten gelegentlich zu Fehlermeldungen kommen, weil dem Android Studio bestimmte Komponenten fehlen. Zusammen mit der Meldung des Fehlers erscheint aber unmittelbar ein Link zur Behebung des Fehlers. Nach Betätigen des Links kann eine erneute Synchronisierung folgen.

Die Entwicklungsumgebung bietet gelegentlich Updates ihrer verschiedenen Komponenten an. Nach dem Betätigen der betreffenden Links werden diese Updates durchgeführt. Auch in diesem Fall kann eine erneute Synchronisierung folgen.

10.3 Ein Projekt im Android Studio

Betrachten wir die Anordnung einiger wichtiger Elemente von Android Studio mithilfe des soeben erstellten Projekts (siehe Abbildung 10.7).

Am oberen Rand werden Menüs und Symbole zur Ausführung von Aktionen angeboten. Darunter sehen Sie links das *Projektfenster* zur Verwaltung der Projektdateien, die in einem aufklappbaren Verzeichnisbaum mit dem Hauptelement *app* liegen. Die Ansicht des Projektfensters kann über eine Liste am oberen Rand eingestellt werden. Wir arbeiten meist in der Ansicht ANDROID.

Rechts ist das *Codefenster* zu sehen, in dem der Code der Dateien in einzelnen Editorfenstern bearbeitet wird. Im unteren Bereich erhalten Sie in verschiedenen Fenstern Meldungen zum Verlauf des Projekts.

Weitere Einzelheiten werden später an geeigneter Stelle besprochen.

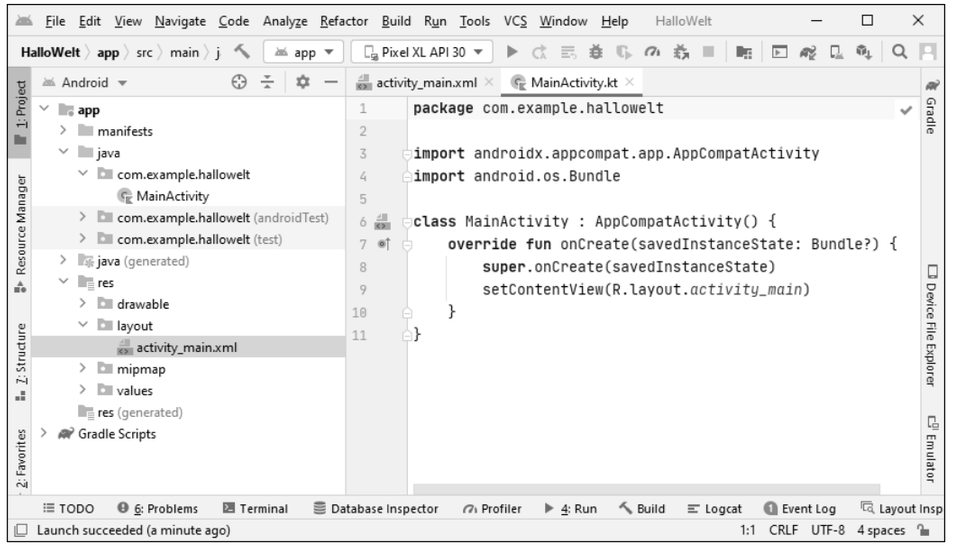

Abbildung 10.7 Projekt »HalloWelt«

10.3.1 Die Datei MainActivity.kt

Im Codefenster rechts erreichen Sie die Datei *MainActivity.kt* über die gleichnamige Registerkarte. Sie beinhaltet den Code der Haupt-Activity in der Programmiersprache Kotlin, auch erkennbar an der Dateiendung *kt* (siehe Abbildung 10.8).

Wird die Datei nicht im Codefenster angezeigt, wählen Sie im Projektfenster auf der linken Seite zunächst die Ansicht ANDROID. Anschließend klappen Sie den Pfad bis zu dem Eintrag *MainActivity* auf (siehe Abbildung 10.9) und führen einen Doppelklick auf diesem Eintrag aus.

MainActivity.kt

```
1  package com.example.hallowelt
2
3  import androidx.appcompat.app.AppCompatActivity
4  import android.os.Bundle
5
6  class MainActivity : AppCompatActivity() {
7      override fun onCreate(savedInstanceState: Bundle?) {
8          super.onCreate(savedInstanceState)
9          setContentView(R.layout.activity_main)
10     }
11 }
```

Abbildung 10.8 Inhalt der Datei »MainActivity.kt«

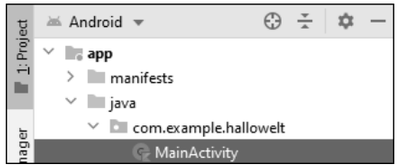

Abbildung 10.9 Datei »MainActivity.kt« anzeigen

10.3.2 Die Datei activity_main.xml

Über das Codefenster rechts erreichen Sie auch die Datei *activity_main.xml*, ebenfalls mithilfe der gleichnamigen Registerkarte. Die Datei beinhaltet das Layout der Benutzeroberfläche der Activity in Form von Code in der Auszeichnungssprache XML in zwei Ansichten. Nach einem Wechsel der Registerkarte dauert es einen Moment, bis sich die Anzeige aktualisiert hat.

Wird die Datei nicht im Codefenster angezeigt, wählen Sie im Projektfenster auf der linken Seite zunächst die Ansicht ANDROID. Anschließend klappen Sie den Pfad bis zu dem Eintrag *activity_main.xml* auf (siehe Abbildung 10.10) und führen einen Doppelklick auf diesem Eintrag aus.

Der Inhalt der Datei *activity_main.xml* kann in drei Ansichten betrachtet werden: DESIGN, CODE oder SPLIT. Zwischen diesen Ansichten schalten Sie über die zugehörigen Register um, die sich oben rechts am Codefenster befinden. In der Ansicht CODE sehen Sie den XML-Code der Datei (siehe Abbildung 10.11).

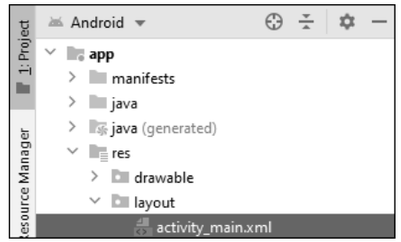

Abbildung 10.10 Datei »activity_main.xml« anzeigen

activity_main.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      tools:context=".MainActivity">
9
10     <TextView
11         android:layout_width="wrap_content"
12         android:layout_height="wrap_content"
13         android:text="Hello World!"
14         app:layout_constraintBottom_toBottomOf="parent"
15         app:layout_constraintLeft_toLeftOf="parent"
16         app:layout_constraintRight_toRightOf="parent"
17         app:layout_constraintTop_toTopOf="parent" />
18
19 </androidx.constraintlayout.widget.ConstraintLayout>
20
```

Abbildung 10.11 Ansicht »Code«

Die Ansicht DESIGN bietet eine vereinfachte Vorschau der App, in Abbildung 10.12 auf der rechten Seite zu sehen. Links unten werden die bereits eingefügten Elemente der App im Bereich COMPONENT TREE in einer hierarchischen Ansicht abgebildet. Links oben stehen Ihnen weitere Elemente zum Einfügen im Bereich PALETTE zur Verfügung.

Das bereits automatisch eingefügte Element TEXTVIEW ist für die Abbildung markiert und wird daher im Bereich COMPONENT TREE und in der Vorschau hervorgehoben.

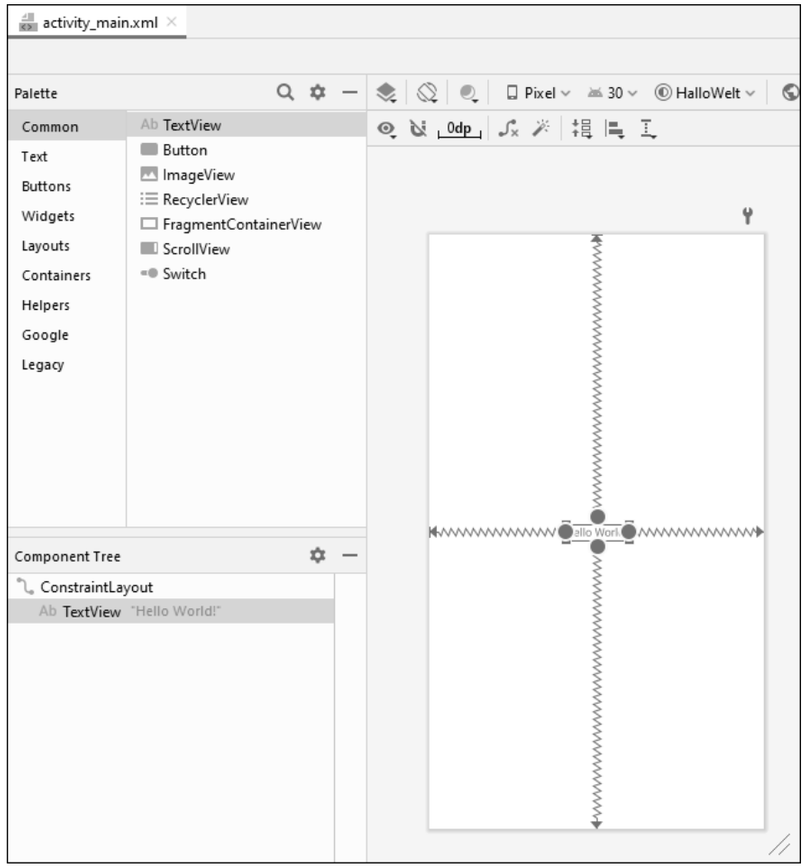

Abbildung 10.12 Ansicht »Design«

Durch das Verschieben oder Entfernen von Elementen im Bereich COMPONENT TREE sowie durch das Hinzufügen von weiteren Elementen aus dem Bereich PALETTE ändert sich das Aussehen der App.

Wichtige Elemente sind zum Beispiel *Views* und *Layouts*. Der Benutzer der App sieht und bedient die Views. Die Views werden mithilfe von Layouts auf der Benutzeroberfläche angeordnet.

In dem einfachen Projekt *HalloWelt* sehen Sie eine View des Typs `TextView` zur Anzeige des Texts »Hello World«. Diese TextView ist in einem Layout des Typs `ConstraintLayout` angeordnet.

Nachdem Sie in der Ansicht DESIGN eines der Elemente ausgewählt haben, ob nun im Bereich COMPONENT TREE oder in der Vorschau der App, sehen Sie die Eigenschaften des betreffenden Elements ganz rechts im Bereich ATTRIBUTES. In Abbildung 10.13 wird der obere Teil dieses umfangreichen Bereichs gezeigt, und zwar für die TextView, in der der Text »Hello World!« angezeigt wird. Die Werte der Eigenschaften können im Bereich ATTRIBUTES verändert werden.

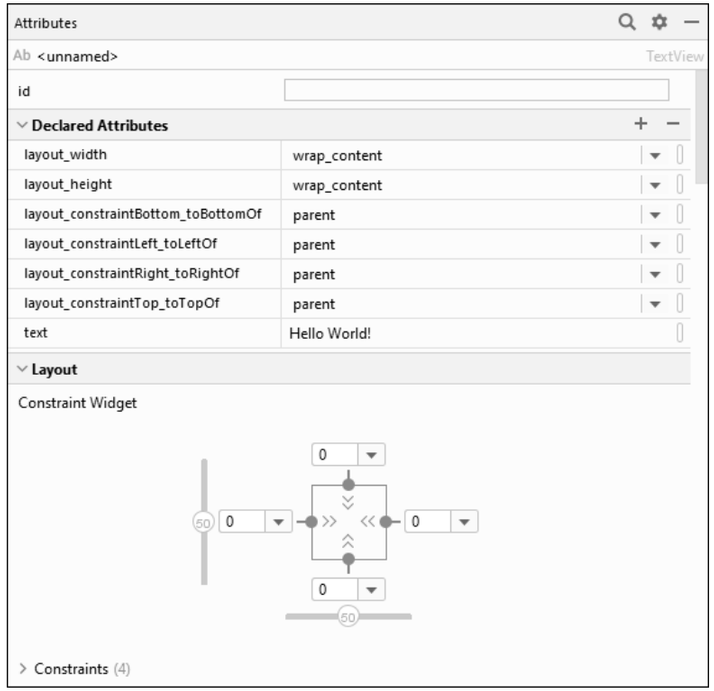

Abbildung 10.13 Bereich »Attributes«

Sobald Sie in der Ansicht DESIGN eine Änderung vornehmen, ändert sich der XML-Code der Datei *activity_main.xml* in der Ansicht CODE. Umgekehrt gilt: Nehmen Sie in der Ansicht CODE eine Änderung im XML-Code vor, ändert sich sofort das Aussehen der App in der Ansicht DESIGN. Es handelt sich nur um zwei unterschiedliche Ansichten derselben Elemente.

Es ist auch möglich, für mehrere Views gleichzeitig Eigenschaftswerte einzustellen. Markieren Sie dazu alle gewünschten Views im Bereich COMPONENT TREE gemeinsam und tragen Sie den neuen oder geänderten Wert im Bereich ATTRIBUTES ein. Eine gemein-

same Markierung von benachbarten Views nehmen Sie mithilfe der zusätzlich gedrückten Taste ⇧ vor. Bei nicht benachbarten Views drücken Sie stattdessen zusätzlich die Taste Strg (Mac: Taste Cmd).

10.3.3 Die Arbeit mit Projekten

Während der Entwicklung können Sie alle geänderten Projektdateien mithilfe des Menüpunkts FILE • SAVE ALL (Tastenkombination Strg + S, Mac: Cmd + S) speichern. Über den Menüpunkt FILE • CLOSE PROJECT schließen Sie das Projekt. Dabei werden alle geänderten Projektdateien automatisch gespeichert.

Ein bereits vorhandenes Projekt können Sie vom Android-Studio-Startbildschirm aus öffnen (siehe Abbildung 10.14). Dazu wählen Sie einfach den betreffenden Eintrag auf der linken Seite aus.

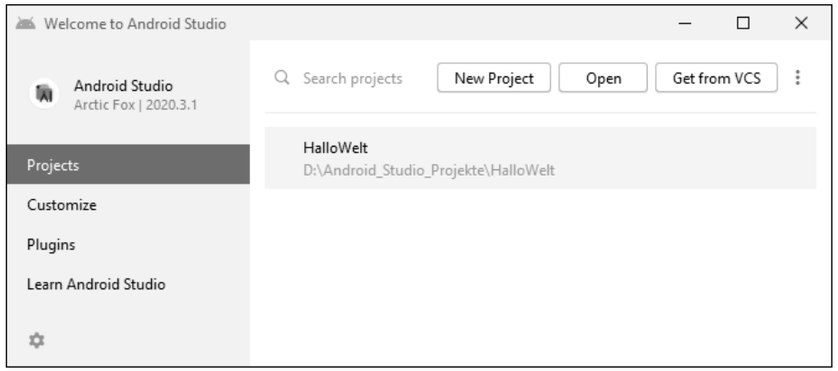

Abbildung 10.14 Vorhandenes Projekt öffnen

Alternativ betätigen Sie auf der rechten Seite den Link OPEN AN EXISTING PROJECT. Im darauf erscheinenden Dialogfeld OPEN FILE OR PROJECT wählen Sie das Projektverzeichnis aus dem Feld SAVE LOCATION aus (siehe Abschnitt 10.2.2).

Bei mir ist das *D:\Android_Studio_Projekte\HalloWelt*

Die Auswahl wird vorgenommen, indem Sie die Verzeichnishierarchie Ebene für Ebene über den Pfeil neben dem jeweiligen Verzeichnisnamen aufklappen. Zum Öffnen des Projekts markieren Sie das betreffende Projektverzeichnis (siehe Abbildung 10.15) und betätigen den Button OK. Ein Doppelklick auf das Projektverzeichnis würde nur das Verzeichnis aufklappen, aber nicht das Projekt öffnen.

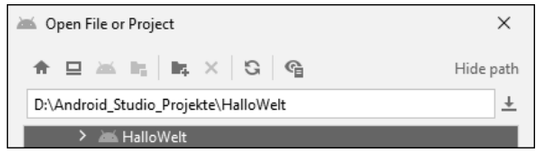

Abbildung 10.15 Auswahl des Projektverzeichnisses

Nach dem Öffnen synchronisiert sich das Projekt wieder. Zur Anzeige der beiden Dateien *MainActivity.kt* und *activity_main.xml* verweise ich auf Abschnitt 10.3.1 und Abschnitt 10.3.2.

Hinweise

In Abschnitt 11.2 werden weitere Fragen zu der Arbeit mit Projekten geklärt:

- Wie öffnet man mehrere Projekte parallel, um Elemente von einem in ein anderes Projekt zu übernehmen?
- Wie kopiert man ein Projekt, um ein neues Projekt auf Basis eines bereits vorhandenen Projekts aufzubauen?
- Wie benennt man ein Projekt im Nachhinein um, falls der ursprünglich gewählte Name eines Projekts nicht mehr passt?
- Wie repariert man ein Projekt, das sich nicht mehr öffnen oder nicht mehr starten lässt?

10.4 App auf virtuellem Gerät starten

Sie haben die Möglichkeit, die neu erstellte App *HalloWelt* und Ihre weiteren Apps sowohl auf virtuellen als auch auf realen Geräten zu testen. In diesem Abschnitt geht es zunächst um ein virtuelles Gerät (engl.: *android virtual device*, kurz: AVD).

Es läuft neben dem Android Studio in einer eigenen Anwendung auf Ihrem Entwicklungs-PC und ähnelt einem realen Gerät in Aussehen, Bedienung und Verhalten. Haben Sie noch kein virtuelles Gerät erstellt, muss dies zunächst erledigt werden. Der AVD Manager von Android Studio ermöglicht Ihnen die Erstellung und Verwaltung von virtuellen Geräten.

Allgemein empfehle ich Ihnen, mithilfe des AVD Manager mehrere virtuelle Geräte zu erstellen, die jeweils individuell konfiguriert sind. Dadurch haben Sie die Möglichkeit, Ihre Apps in verschiedenen Android-Versionen beziehungsweise auf Bildschirmen mit unterschiedlichen Größen und Auflösungen zu testen.

10.4.1 Ein virtuelles Gerät erstellen

Es gibt zwei Möglichkeiten, den AVD Manager aufzurufen:

- Haben Sie kein Projekt geöffnet, rufen Sie ihn über den Eintrag AVD Manager in der Liste Configure auf. Diese Liste finden Sie rechts unten auf dem Startbildschirm von Android Studio.
- Haben Sie bereits ein Projekt geöffnet, rufen Sie ihn über den Menüpunkt Tools • AVD Manager auf.

Zur Erstellung eines virtuellen Geräts betätigen Sie im Dialogfeld Android Virtual Device Manager den Button Create Virtual Device. Es erscheint das Dialogfeld Virtual Device Configuration • Select Hardware. Darin steht Ihnen eine Liste von vorkonfigurierten virtuellen Geräten zur Verfügung (siehe Abbildung 10.16).

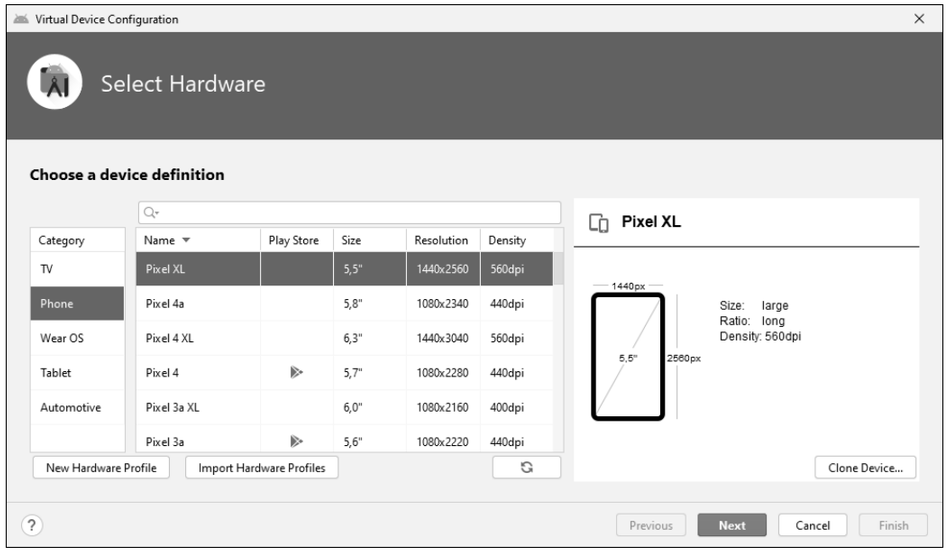

Abbildung 10.16 Hardware auswählen

Im Beispiel habe ich aus der Kategorie Phone das virtuelle Gerät Pixel XL mit einer Bildschirmgröße von 5,5 Zoll, einer Bildschirmauflösung von 1.440 × 2.560 Pixeln und einer Punktdichte von 560 dpi gewählt.

Über die beiden Buttons New Hardware Profile und Import Hardware Profiles könnten Sie auch ein individuell konfiguriertes virtuelles Gerät erzeugen oder ein bereits vorkonfiguriertes virtuelles Gerät importieren.

Nach Anklicken des Buttons Next erscheint das Dialogfeld Virtual Device Configuration • System Image (siehe Abbildung 10.17). Hier wählen Sie die Android-Version für Ihr virtuelles Gerät.

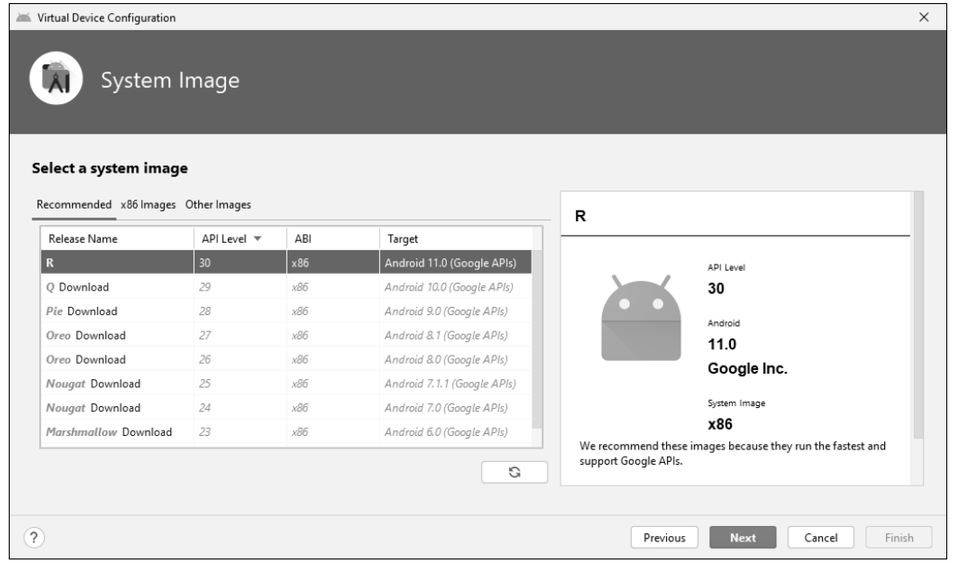

Abbildung 10.17 Software auswählen

Ich habe auf der Registerkarte Recommended die Android-Version 11.0 mit dem API-Level 30 gewählt. Erscheint neben einer Android-Version der Link Download, müssen Sie zunächst ein Komponenten-Update über diesen Link durchführen. Erst anschließend können Sie diese Android-Version für Ihr neues Gerät auswählen.

Nach der erneuten Betätigung des Buttons Next erscheint das Dialogfeld Virtual Device Configuration • Android Virtual Device (AVD) (siehe Abbildung 10.18).

Hier können Sie die eingestellte Konfiguration und den Namen des virtuellen Geräts bestätigen oder verändern. Über die beiden Buttons mit der Bezeichnung Change... könnten Sie Änderungen an der gewählten Hardware oder an der gewählten Android-Version vornehmen.

Nach Betätigung des Buttons Finish erscheint das Dialogfeld Android Virtual Device Manager • Your Virtual Devices. Es beinhaltet eine Liste Ihrer virtuellen Geräte, inklusive des soeben erzeugten Geräts (siehe Abbildung 10.19).

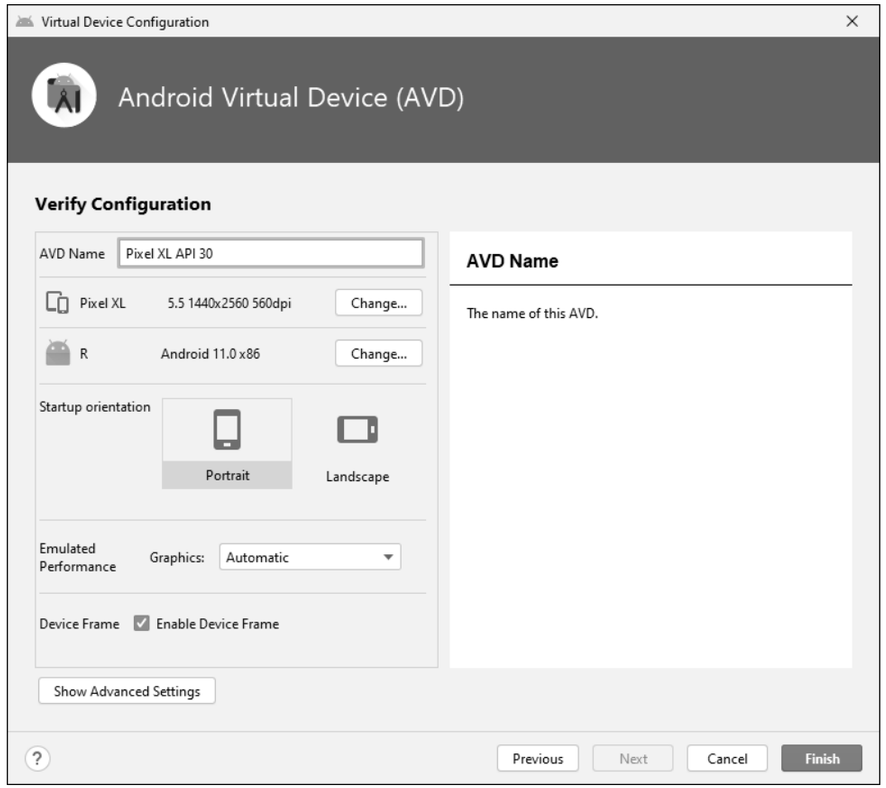

Abbildung 10.18 Konfiguration

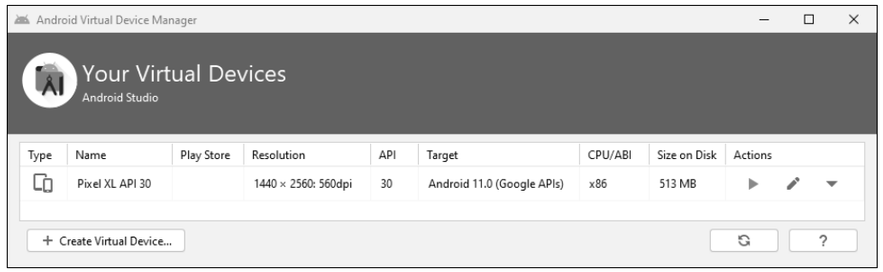

Abbildung 10.19 Virtuelle Geräte, Liste

Nach der Betätigung des grünen Pfeils startet das betreffende virtuelle Gerät. Beim allerersten Start dauert es ein wenig länger. Ähnlich wie ein reales Gerät wird das virtuelle Gerät zunächst konfiguriert. Sie können den AVD Manager nach dem Start des virtuellen Geräts schließen.

Nach dem Start des virtuellen Geräts finden Sie darin viele der üblichen Bedienungselemente. Neben dem virtuellen Gerät wird eine zusätzliche Bedienleiste abgebildet, die Ihnen weitere Möglichkeiten eines realen Geräts bietet. Sie können damit zum Beispiel das virtuelle Gerät aus- und einschalten, die Lautstärke regeln und das Gerät virtuell drehen.

In Abbildung 10.20 sehen Sie das virtuelle Gerät mit der bereits gestarteten App *HalloWelt*, daneben die Bedienleiste. Der Start der App wird in Abschnitt 10.4.2 beschrieben.

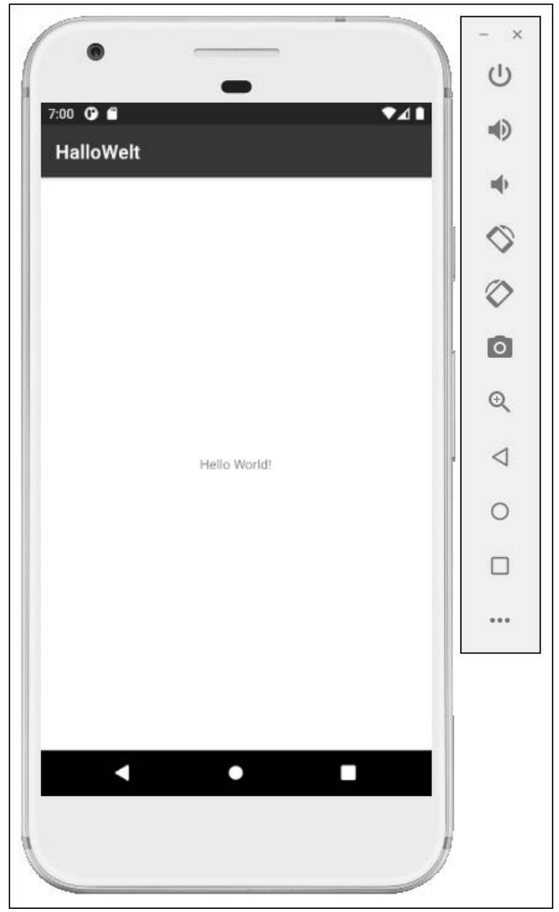

Abbildung 10.20 Virtuelles Gerät, Bedienleiste

Sie beenden die Anwendung, die das virtuelle Gerät beinhaltet, auf Ihrem Entwicklungs-PC wie üblich über das Kreuz oben rechts in der Bedienleiste. Ich empfehle Ihnen allerdings, das virtuelle Gerät während der Arbeit an Ihren Projekten nicht auszuschalten, damit nicht jedes Mal der Startvorgang durchlaufen wird.

Ganz unten auf der Bedienleiste gelangen Sie über die drei Punkte zum Dialogfeld EXTENDED CONTROLS (siehe Abbildung 10.21). Darin haben Sie erweiterte Bedienmöglichkeiten. Sie können unter anderem den Standort und die Umgebungsbedingungen des virtuellen Geräts wählen.

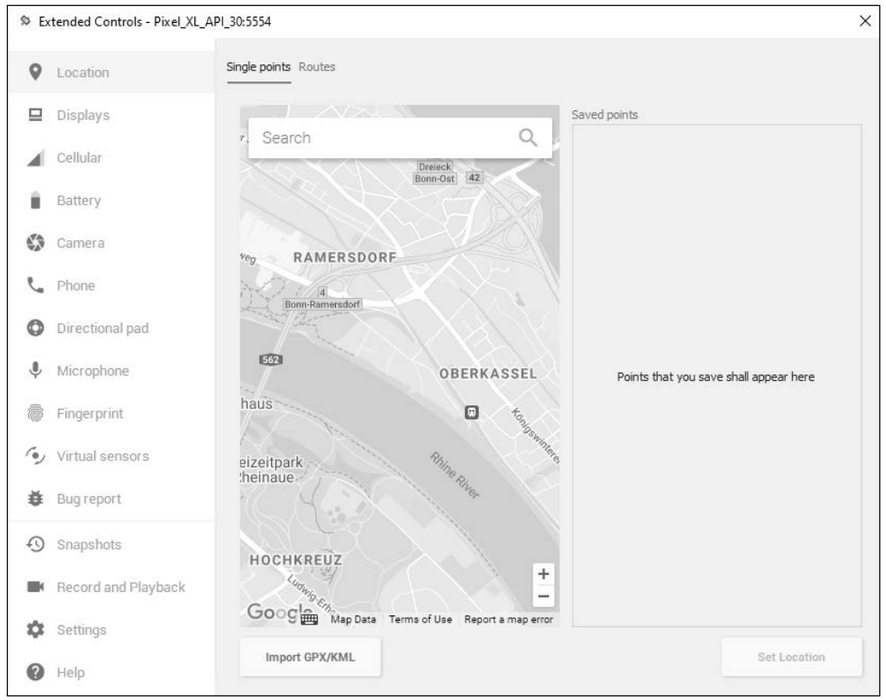

Abbildung 10.21 Erweiterte Bedienung

Die Änderung der Konfiguration eines vorhandenen virtuellen Geräts wird in Anhang A erläutert.

10.4.2 App starten und beenden

Sie starten die App eines geöffneten Projekts über den Menüpunkt RUN • RUN APP (Tastenkombination ⇧ + F10). Alternativ können Sie auch auf den grünen Pfeil neben der Liste der Geräte klicken (siehe Abbildung 10.22).

Abbildung 10.22 App starten, virtuelles Gerät

Ist das virtuelle Gerät noch nicht gestartet, wird es zunächst gestartet. In diesem Fall dauert der Start der App entsprechend länger. In Abbildung 10.20 ist die gestartete App *HalloWelt* abgebildet.

Beim ersten Start Ihrer App wird sie auf dem virtuellen Gerät installiert. Nach einer Weiterentwicklung Ihres Projekts wird die neue Version der App anstelle der alten Version auf dem virtuellen Gerät installiert.

Sie beenden die App über den Menüpunkt RUN • STOP APP (Tastenkombination Strg + F2). Alternativ können Sie auch auf das rote Quadrat klicken, das Sie ganz rechts in Abbildung 10.23 sehen.

Abbildung 10.23 App beenden, virtuelles Gerät

Startet eine App nicht, wird in Android Studio eine Fehlermeldung angezeigt. Die Ursache kann darin liegen, dass die verschiedenen Teile eines Projekts noch nicht auf die richtige Weise zusammenarbeiten. In diesem Fall erweist es sich als hilfreich, das Projekt mithilfe des Menüpunkts FILE • SYNC PROJECT WITH GRADLE FILES zu synchronisieren und es mithilfe des Menüpunkts BUILD • CLEAN PROJECT zu bereinigen.

In seltenen Fällen kann es vorkommen, dass zwar keine Fehlermeldung angezeigt wird, eine App aber dennoch nicht startet. In einem solchen Fall könnten Sie das virtuelle Gerät in seinen Startzustand zurückversetzen, indem Sie in der Spalte ACTIONS den schwarzen Pfeil betätigen und im anschließend erscheinenden Menü den Menüpunkt WIPE DATA ausführen, siehe auch Anhang A. Hilft auch das nicht, könnten Sie das virtuelle Gerät mithilfe des Menüpunkts DELETE löschen und es anschließend mit wenigen Klicks wieder neu erzeugen.

10.5 App auf realem Gerät starten

Sie haben die Möglichkeit, die neu erstellte App *HalloWelt* und Ihre weiteren Apps sowohl auf virtuellen Geräten als auch auf realen Geräten zu testen. In diesem Abschnitt geht es um ein reales Gerät. Dazu sind bestimmte Vorbereitungen notwendig.

10.5.1 Reales Gerät vorbereiten

Verbinden Sie das reale Gerät, also Ihr Smartphone oder Ihr Tablet, mithilfe eines USB-Kabels mit dem Entwicklungs-PC, auf dem Sie mithilfe von Android Studio Ihre Apps entwickeln. Rufen Sie auf dem realen Gerät die EINSTELLUNGEN auf.

Geben Sie dort in der SUCHE den Suchbegriff `Build` ein. Als Ergebnis der Suche wird, eventuell erst nach einigen Sekunden, der Eintrag BUILDNUMMER angezeigt. Tippen Sie diesen Eintrag sieben Mal nacheinander an. Anschließend erhalten Sie die Information, dass nunmehr der Entwicklermodus auf Ihrem Gerät freigeschaltet ist.

Geben Sie anschließend in der Suche den Suchbegriff `Entwickler` ein. Als Ergebnis der Suche wird das Menü ENTWICKLEROPTIONEN angezeigt. Wählen Sie dieses Menü aus und aktivieren Sie den Schalter, der anschließend ganz oben erscheint. Sie werden aufgefordert, die Verwendung der Entwicklungseinstellungen zu bestätigen. Stimmen Sie dem Dialog mit OK zu.

Im Menü ENTWICKLEROPTIONEN finden Sie den Eintrag USB-DEBUGGING. Sollte sich dieser Eintrag auf Ihrem Gerät in einem anderen Menü befinden, finden Sie ihn über die Eingabe des Suchbegriffs `USB` in den Einstellungen. Aktivieren Sie den Schalter bei dem Eintrag USB-DEBUGGING. Sie werden aufgefordert, das *USB-Debugging* zuzulassen. Bestätigen Sie den Dialog mit OK.

Anschließend wird Ihnen zum Legitimieren der Verbindung zwischen dem realen Gerät und Ihrem Entwicklungsrechner der Fingerabdruck des RSA-Schlüssels des Rechners angezeigt. Setzen Sie durch Antippen das Häkchen im Feld VON DIESEM COMPUTER IMMER ZULASSEN und bestätigen Sie den Dialog mit OK.

Ab jetzt können Sie Ihre Apps auch auf dem realen Gerät testen.

Es kann vorkommen, dass Sie nach einem Update im Android Studio gemäß einem entsprechenden Hinweis die Einrichtung des realen Geräts erneut ausführen müssen.

10.5.2 App starten und beenden

Standardmäßig ist Ihr reales Gerät nach dem erfolgreichen Anschluss in der Liste der Geräte ausgewählt, ähnlich wie in Abbildung 10.24 zu sehen. In meinem Fall teste ich

die Apps mit einem älteren Smartphone des Typs *Samsung Galaxy S6 active* mit Android 6.0.1 und der API 23.

Sie starten die App eines geöffneten Projekts über den Menüpunkt RUN • RUN APP (Tastenkombination ⇧ + F10). Alternativ können Sie auch auf den grünen Pfeil neben der Liste der Geräte klicken (siehe Abbildung 10.24).

Abbildung 10.24 App starten, reales Gerät

Beim ersten Start Ihrer App wird sie auf dem realen Gerät installiert. Nach einer Weiterentwicklung Ihres Projekts wird die neue Version der App anstelle der alten Version auf dem realen Gerät installiert.

Sie beenden die App über den Menüpunkt RUN • STOP APP (Tastenkombination Strg + F2). Alternativ können Sie auch auf das rote Quadrat klicken, das Sie ganz rechts in Abbildung 10.25 sehen.

Abbildung 10.25 App beenden, reales Gerät