

LERNEN EINFACH GEMACHT



# Android-Apps programmieren lernen

für  
**dummies**<sup>®</sup>



App-Programmierung  
erlernen ohne  
Programmierkenntnisse

Von ersten Schritten  
über ein Quiz bis zur  
GPS-gesteuerten Karte

Grafikprogrammierung  
für Bildschirmspiele

**Arnold Willemer**

# Android-Apps programmieren lernen für Dummies

## Schummelseite

---

### SHORTCUTS AUSFÜHRUNG

Android-Studio startet die Apps über die folgenden Tastaturkürzel:

- ✓  +  : App starten
- ✓  +  : Änderungen übernehmen und App neu starten
- ✓  +  : Debug starten

### SHORTCUTS DEBUG-MODUS

Im Debug-Modus wirken folgende Shortcuts:

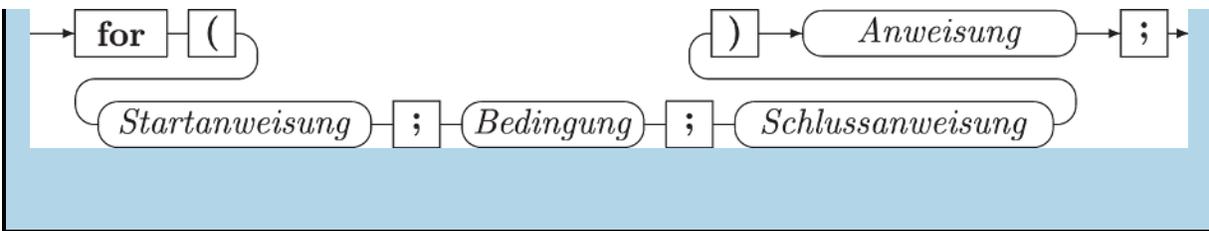
- ✓  : Step-over. Springt über die aktuelle Zeile, auch wenn sie einen Methodenaufruf enthält.
- ✓  : Step-into. Springt in die Methode, über deren Aufruf der Debugger gerade steht.
- ✓  +  : Step-out. Verlässt die aktuelle Methode, in der der Debugger steht.
- ✓  : Weiterlaufen (Resume). Die schrittweise Ausführung wird aufgelöst. Das Programm läuft bis zum nächsten Breakpoint.
- ✓  +  : Beende das Programm.

### SHORTCUTS EDITOR

Im Editor gibt es

- ✓  +  : Suche Verwendung





## DIE GRAFIKPRIMITIVE DES CANVAS

Der Canvas ist die Zeichenfläche, die typischerweise als Parameter der Methode `onDraw` übergeben wird. Die Klasse bietet folgende Methoden zum Zeichnen.

✓ `void drawLine(float xStart, float yStart, float xEnde, float yEnde, Paint paint);`

`drawLine` zieht eine Linie von der Koordinate (xStart, yStart) zur Koordinate (xEnde, yEnde).

✓ `void drawRect(float xStart, float yStart, float xEnde, float yEnde, Paint paint);`

`drawRect` zeichnet ein gefülltes Rechteck von der Koordiante links oben zur zweiten Koordinate rechts unten.

✓ `void drawText(float xStart, float yStart, String text, Paint paint);`

`drawText` zeichnet den Inhalt von `text` als Text an die Koordinate (xStart, yStart). Die Position ist die linke untere Ecke des Strings. Ist `yStart` 0, ist die Schrift nicht zu sehen, da sie oberhalb des Canvas landet, der seinen Nullpunkt links oben hat.

✓ `void drawCircle(float xStart, float yStart, float radius, Paint paint);`

`drawCircle` zeichnet einen Kreis mit dem Mittelpunkt an der Koordinate (xStart, yStart). Als weiterer Parameter wird der Radius, also der halbe Durchmesser angegeben.

## LEBENSZYKLUS EINER ACTIVITY

- ✓ onCreate - onDestroy: Die Activity wird erzeugt beziehungsweise entsorgt.
- ✓ onStart - onStop: Die Activity wird sichtbar beziehungsweise verschwindet.
- ✓ onResume - onPause: Die Activity kommt in den Vordergrund, nachdem sie sichtbar wurde, beziehungsweise wird aus den Vordergrund verdrängt.

## LISTENER UND IHRE METHODEN

- ✓ onTouchListener (onTouch) wird bei der Berührung einer selbstgeschriebenen View ausgelöst.
- ✓ onClickListener (onClick) reagiert auf die Berührung eines Buttons.
- ✓ onItemClickListener (onItemClick) reagiert auf die Berührung eines Listenelements bei einer ListView.
- ✓ onItemSelectedListener (onItemSelected) reagiert auf die Berührung eines Listenelements bei einem Spinner.
- ✓ onCheckedChangeListener (onCheckedChange) reagiert auf den Wechsel der Auswahl eines Hakenelements (CheckBox, RadioButton, RadioGroup).
- ✓ onFocusChangeListener (onFocusChange) reagiert, wenn ein Element den Fokus erhält oder verliert.
- ✓ onDateSetListener (onDateSet) reagiert auf Änderungen im DatePicker.
- ✓ locationListener (onLocationChanged, ...) reagiert auf das Verändern einer Standortposition (GPS).
- ✓ onChronometerTickListener (onChronometerTick) reagiert auf das Ticken eines Timers (Chronometer).
- ✓ onCompletionListener (onCompletion) wird aktiviert, wenn der MediaPlayer ein Stück fertig abgespielt hat.
- ✓ sensorEventListener (onSensorChanged) reagiert auf neue Events eines Sensors.



Arnold Willemer

# Android-Apps programmieren lernen

für  
**dummies**<sup>®</sup>



**WILEY**

WILEY-VCH GmbH

## **Android-Apps programmieren lernen für Dummies**

### **Bibliografische Information der Deutschen Nationalbibliothek**

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

© 2022 Wiley-VCH GmbH, Boschstraße 12, 69469  
Weinheim, Germany

Wiley, the Wiley logo, Für Dummies, the Dummies Man logo, and related trademarks and trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries. Used by permission.

Wiley, die Bezeichnung »Für Dummies«, das Dummies-Mann-Logo und darauf bezogene Gestaltungen sind Marken oder eingetragene Marken von John Wiley & Sons, Inc., USA, Deutschland und in anderen Ländern.

Das vorliegende Werk wurde sorgfältig erarbeitet. Dennoch übernehmen Autoren und Verlag für die Richtigkeit von Angaben, Hinweisen und Ratschlägen sowie eventuelle Druckfehler keine Haftung.

**Coverfoto:** © elenabsl - [stock.adobe.com](https://www.stock.adobe.com)

**Korrektur:** Matthias Delbrück, Dossenheim/Bergstraße

**Print ISBN:** 978-3-527-71880-1

**ePub ISBN:** 978-3-527-83458-7

# Über den Autor

---

**Arnold V. Willemer** hat an der Universität Frankfurt/Main sein Diplom in Informatik abgelegt. Anschließend hat er viele Jahre als Software-Entwickler gearbeitet. Seit 2016 ist er an der Hochschule Flensburg tätig, vor allem in den Bereichen Programmierung und Netzwerke.

Seit 2001 schreibt er Bücher zu den verschiedensten Informatikthemen, etwa Programmiersprachen wie Java, C++ oder Python. Aber er befasst sich auch gern mit Betriebssystemen und Netzwerken. Vor allem interessieren ihn da UNIX und Linux. Leider kam er nie in die engere Wahl für den Literatur-Nobelpreis und auch auf den Friedenspreis des Deutschen Buchhandels wartet er seit Jahren vergeblich.

Tatsächlich motiviert ihn die Aussicht auf Preise weniger als die Herausforderung, komplexe Themen Anfängern möglichst leicht verständlich in leicht zu bewältigenden Schritten nahezubringen.

## ***Danksagung***

Meine Lektorin Andrea Baulig musste sehr viel Geduld aufbringen, weil ich mehrere Anläufe benötigte, bis ich es schaffte, alles verständlich in die richtige Reihenfolge zu bringen. Und obwohl ich einen Abgabetermin gerissen habe, blieb sie erstaunlich ruhig.

# Inhaltsverzeichnis

[Cover](#)

[Titelblatt](#)

[Impressum](#)

[Über den Autor](#)

[Einleitung](#)

[Konventionen in diesem Buch](#)

[Törichte Annahmen über den Leser](#)

[Wie dieses Buch aufgebaut ist](#)

[Symbole, die in diesem Buch verwendet werden](#)

[Wie es weitergeht](#)

[\*\*Teil I: Werkzeugbeschaffung und Einrichtung\*\*](#)

[\*\*Kapitel 1: Entwicklungsumgebung\*\*](#)

[Anforderungen an den Entwickler-PC](#)

[Smartphone oder Tablet](#)

[Gestatten: Android Studio](#)

[Weitere Hilfen aus dem Internet](#)

[\*\*Kapitel 2: Das »leere« Projekt von Android Studio\*\*](#)

[Die erste App](#)

[Übersicht über die IDE](#)

[Android Virtual Device Manager](#)

[Emulator starten](#)

[Falsche SDK-Version](#)

[Nichts anfassen, nur gucken](#)

[\*\*Teil II: Wir programmieren\*\*](#)

[\*\*Kapitel 3: Eine Mini-App für die ersten Programmierschritte\*\*](#)

[Erste eigene Programmschritte](#)

[Variablen und Typen](#)

[Ausgabe von Ergebnissen](#)

[Eingaben zu Ausgaben verrechnen](#)

[Eigene Projekte](#)

## **Kapitel 4: BMI: Abfragen und Schleifen**

[Das Layout für Ihre BMI-App](#)

[Abfragen in Java](#)

[Java bringt Leben in den BMI](#)

[Texte statt BMI-Werte](#)

[Automatische Neuberechnung bei Eingabe](#)

[Eine neue Methode zur Berechnung des BMI](#)

[Eigene Projekte](#)

## **Kapitel 5: Lotto: Zufall, Arrays und Schleifen**

[Arrays in Java](#)

[Zufällige Lottozahlen](#)

[Eine Würfel-App](#)

[Java-Schleifen](#)

[Sortieren](#)

[Darstellen der Lottozahlen in der ListView](#)

[Eigene Projekte](#)

## **Kapitel 6: Brüche in Klassen**

[Klasse](#)

[Die Klasse als Datenstruktur](#)

[Methoden zur Multiplikation](#)

[Konstruktoren der Klasse Bruch](#)

[Der Destruktor und die Müllabfuhr](#)

[Private Attribute](#)

[Erweiterte Erbschaft](#)

[Abstrakte Klassen und Interfaces](#)

[Packages und Importe](#)

[Eigene Projekte](#)

## **Teil III: Ein eigenes grafisches Spiel**

### **Kapitel 7: Das Spiel Minesweeper als View**

[Das Spiel Minesweeper](#)

[Das Minesweeper-Projekt](#)

[Eine eigene View](#)

[Grafik in der View mit onDraw](#)

[Wir zeichnen ein Spielfeld](#)

[OnTouchListener reagieren auf Tatschen](#)

[Die Spiellogik](#)

[Ein Button für ein neues Spiel](#)

[Markieren](#)

[MineSweeperView meldet Spielende](#)

[Das gedrehte Display](#)

[Eigene Projekte](#)

### **Kapitel 8: Die Highscore-Liste**

[Chronometer: Die tickende TextView](#)

[Der Name des Spielers](#)

[Zeit und Datum des Triumphs](#)

[Die Highscore-Liste in der Datenbank](#)

[Mit dem Menü zur Highscore-Liste](#)

[Anzeige der Highscore-Liste](#)

[Eigene Projekte](#)

## **Teil IV: Wechselnde Displays und ihre Daten**

### **Kapitel 9: Wechselspiel der Bildschirme**

[Ein Quiz](#)

[Android-eigene Activitys](#)

[Daten sichern: SharedPreferences](#)

[Lebenszyklus einer Activity](#)

[Ein Datenmodell für ein Quiz](#)

[Die App stellt Fragen](#)

[Sicherung des Quiz-Zustands](#)

[Ein Menü für das Quizspiel](#)

[Neue Fragen für das Quiz](#)

[Änderung einer Aufgabe](#)

[Eigene Projekte](#)

## **Kapitel 10: Fragmente einer Activity**

[Mehrere Fragmente in einer Activity](#)

[Bildschirmnavigation mit Fragmenten](#)

## **Kapitel 11: Zugriff auf Dateien und das Internet**

[JSON](#)

[Externe Dateien](#)

[Daten aus dem Internet lesen](#)

[Wie ein Netzwerk tickt](#)

[Eigene Projekte](#)

## **Teil V: Ortskenntnis und Sensoren**

### **Kapitel 12: Wo bin ich?**

[Der Spion in der Tasche](#)

[Positionsbestimmung per GPS](#)

[Die Berechtigungen anfordern](#)

[Location: Der Manager und sein Listener](#)

[Zusammenfassung: Anzeige von GPS-Daten](#)

[Einen Kartenausschnitt beschaffen und integrieren](#)

[In OMS-Droid Karten online navigieren](#)

### **Kapitel 13: Sensoren**

[Ein Sensor-Projekt](#)

[Sensor-Manager](#)

[Sensor-Ereignissen nachspüren](#)

[Eigene Projekte](#)

### **Kapitel 14: Multimedia: Video und Audio**

[Berechtigungen](#)

[Beispiel-App für ein Foto](#)

[Auslaufmodell startActivityForResult](#)

Video

Audioaufnahme und -wiedergabe

Abspielen von Klangressourcen

Eigene Projekte

## **Teil VI: Der Top-Ten-Teil**

### **Kapitel 15: Die Top 10 der Emulator-Gemeinheiten**

Der Emulator benötigt das Virtual Flag der CPU

Zickig im Umgang mit virtuellen Kollegen

Laaaaaangsam

Wenig Speicher

Fettleibigkeit

Stürzt bei Nichtbeachtung ab

Startet manchmal kommentarlos nicht

Mangel an Fähigkeiten

Eine Sperrdatei

Unverzichtbar

### **Kapitel 16: Die Top-10-Strategie für benutzbare Apps**

Planen Sie Ihre Funktionalität

Befragen Sie potentielle Benutzer

Gruppieren und Trennen

Prototypen erstellen und verwerfen

Testen Sie die Bedienung zuerst selbst

Programmierer sind keine Anwender (und umgekehrt)

Erstbenutzer suchen

Benutzertest mit lautem Denken

Fragen stellen

Tester freundlich behandeln

## **Abbildungsverzeichnis**

## **Stichwortverzeichnis**

## **End User License Agreement**

# Tabellenverzeichnis

## Kapitel 4

[Tabelle 4.1: BMI-Werte und ihre Bedeutung](#)

[Tabelle 4.2: Spannungsabfall an LEDs](#)

## Kapitel 9

[Tabelle 9.1: Der Lebenszyklus einer Activity](#)

# Illustrationsverzeichnis

## Kapitel 1

[Abbildung 1.1: Webseite zum Download von Android Studio](#)

[Abbildung 1.2: Installation auf dem Mac nach dem Download](#)

## Kapitel 2

[Abbildung 2.1: Auswahl des Projektstils](#)

[Abbildung 2.2: Konfiguration des Projekts](#)

[Abbildung 2.3: Android Studio mit geladenem Projekt](#)

[Abbildung 2.4: Zugang zum AVD-Manager](#)

[Abbildung 2.5: Der AVD-Manager mit den Emulatoren](#)

[Abbildung 2.6: Auswahl einer neuen Emulator-Hardware im AVD-Manager](#)

[Abbildung 2.7: Ein erster Gruß des Emulators](#)

[Abbildung 2.8: Das Projekt kann nicht übersetzt werden.](#)

[Abbildung 2.9: Auswahl der SDK im Dialog Project Structure](#)

[Abbildung 2.10: Modi des Editors](#)

[Abbildung 2.11: TextView im Designer](#)

## Kapitel 3

[Abbildung 3.1: Die Werkzeugleiste mit dem Käfer als Debug-Symbol](#)

[Abbildung 3.2: Die App wartet auf den Debugger.](#)

[Abbildung 3.3: Die Werkzeugleiste zur Steuerung des Debuggers](#)

[Abbildung 3.4: Das Listing unter Debug-Kontrolle](#)

[Abbildung 3.5: Ergebnisse im Logcat](#)

[Abbildung 3.6: Ergebnisse im Toast](#)

[Abbildung 3.7: Eintrag der id im Designer](#)

[Abbildung 3.8: Die Tischfläche in voller Schönheit](#)

[Abbildung 3.9: Zentrieren eines Elements](#)

[Abbildung 3.10: Alle Elemente werden oben auf dem Display angeordnet.](#)

[Abbildung 3.11: Elemente markieren](#)

[Abbildung 3.12: Vertikale Kette](#)

[Abbildung 3.13: Fehlermeldung!](#)

[Abbildung 3.14: Android Studio schlägt Ihnen eine Methode vor.](#)

[Abbildung 3.15: App zur Berechnung von Fläche und Umfang eines Rechtecks](#)

## **Kapitel 4**

[Abbildung 4.1: Beschriftung und Eingabe auf dieselbe Höhe bringen.](#)

[Abbildung 4.2: In der Höhe zentrieren.](#)

[Abbildung 4.3: Die BMI-Activity im Design](#)

[Abbildung 4.4: Der BMI-Screen](#)

[Abbildung 4.5: Der Dialog für eine weitere Sprachdatei](#)

## **Kapitel 5**

[Abbildung 5.1: Eine App voller Zufälle](#)

[Abbildung 5.2: Lottozahlen im Debugger](#)

[Abbildung 5.3: Die Anzeige der Lottozahlen, links vor der ersten Ziehung](#)

[Abbildung 5.4: Ein Wechselgeldautomat](#)

## **Kapitel 6**

[Abbildung 6.1: Der Klasse einen Namen geben](#)

[Abbildung 6.2: Ein Vererbungsbaum](#)

## **Kapitel 7**

[Abbildung 7.1: Ein Minesweeper-Spiel](#)

[Abbildung 7.2: Eine Klasse erstellen](#)

[Abbildung 7.3: Der Ort für die Klasse](#)

[Abbildung 7.4: Der Name für die Klasse](#)

[Abbildung 7.5: Erster View-Test mit kaum erkennbarer Diagonale](#)

[Abbildung 7.6: Eine gerasterte View](#)

[Abbildung 7.7: Das Raster mit X gefüllt](#)

[Abbildung 7.8: Quadratische Felder mit Überhang](#)

[Abbildung 7.9: Quadratische Felder ohne Überhang](#)

[Abbildung 7.10: Das Spiel merkt sich Berührungen als Kreuze.](#)

[Abbildung 7.11: Das Spiel zeigt die Nachbarminen.](#)

[Abbildung 7.12: Das Spiel senkrecht und waagrecht](#)

[Abbildung 7.13: Ausrichtung einer weiteren Layout-Datei](#)

[Abbildung 7.14: Im Querformat](#)

## **Kapitel 8**

[Abbildung 8.1: Der Spielname im Portrait-Modus](#)

[Abbildung 8.2: Drei Punkte verweisen auf das Optionsmenü.](#)

[Abbildung 8.3: Das aufgeklappte Optionsmenü](#)

[Abbildung 8.4: Erstellen eines neuen Layouts](#)

[Abbildung 8.5: Die fertige Highscore-Tabelle](#)

[Abbildung 8.6: Der DatePicker im Rädchenmodus](#)

[Abbildung 8.7: Der DatePicker im Kalendermodus als Dialog](#)

## **Kapitel 9**

[Abbildung 9.1: Der Begrüßungsbildschirm im Layout-Designer](#)

[Abbildung 9.2: Wähl-Aktion in der Emulation](#)

[Abbildung 9.3: Die Quiz-App stellt eine Aufgabe.](#)

[Abbildung 9.4: Der Menü-Designer](#)

[Abbildung 9.5: Eingabe einer Quizfrage](#)

## **Kapitel 10**

[Abbildung 10.1: Einfügen der FragmentContainerView](#)

[Abbildung 10.2: Fragment in Activity](#)

[Abbildung 10.3: Zwei Fragmente in einer Activity](#)

[Abbildung 10.4: Ausrichtung einer weiteren Layout-Datei](#)

[Abbildung 10.5: Im Querformat](#)

[Abbildung 10.6: Zwei Fragmente nebeneinander in einer Activity](#)

[Abbildung 10.7: Projektstruktur](#)

[Abbildung 10.8: Auswählen eines Fragments](#)

[Abbildung 10.9: Der Navigationsgraph](#)

## **Kapitel 12**

[Abbildung 12.1: Der Dialog für die Berechtigung zur Standortbestimmung](#)

[Abbildung 12.2: Die Anzeige der Location-Bestandteile](#)

[Abbildung 12.3: Erzeugen einer PNG-Datei aus OSM](#)

[Abbildung 12.4: Links mehrere Welten, rechts Deutschland](#)

[Abbildung 12.5: Die Karte wird auf die GPS-Position positioniert.](#)

## **Kapitel 13**

[Abbildung 13.1: Die Sensorsimulatoren des Emulators](#)

[Abbildung 13.2: Weitere Sensoren, darunter auch der Lichtsensor](#)

## **Kapitel 14**

[Abbildung 14.1: Ein minimales Aufnahmestudio](#)

[Abbildung 14.2: Das Aufnahmestudio im Record-Modus](#)

# Einleitung

---

Dieses Buch richtet sich an Menschen, die gern programmieren lernen wollen und dabei vor allem Anwendungen für ihr Android-Smartphone im Blick haben. Auch wenn das Buch keine Kenntnisse in der Programmierung voraussetzt, wird es deutlich einfacher werden, wenn Sie bereits programmieren können.

Aber selbst wenn Sie schon Erfahrungen in der Programmierung haben, könnte dieses Buch vielleicht genau das Buch sein, das Sie sich wünschen. Die Programmierung von Android-Apps greift ausgiebig auf die Techniken der objektorientierten Programmierung zurück. Sollten Sie da noch die eine oder andere Lücke haben, wird sich diese hier schließen.

Das Buch erläutert also die ersten Schritte, die objektorientierte Programmierung und die Programmierung in einer Android-Umgebung. Damit es nicht langweilig wird, habe ich versucht, die theoretischen Grundlagen dort zu erläutern, wo sie in der Android-Programmierung gebraucht werden.

## ***Konventionen in diesem Buch***

Für Schlüsselwörter, Bezeichner und ähnliche Dinge, verwende ich die nichtproportionale Schrift. Wenn ich Begriffe erläutere, werden Sie *kursive Schrift* sehen.

Dateien und Pfade werden ebenfalls in *kursiver Schrift* gesetzt.

Menüs, Buttons oder Beschriftungen von Programmen oder Webseiten werden in KAPITÄLCHEN dargestellt.

Auch URLs werden in nichtproportionaler Schrift gesetzt. Im Falle von eBooks mag es sogar funktionieren, diese direkt anzuklicken. Ob das bei Ihnen klappt, hängt aber von Ihrer technischen Umgebung ab.

Ich werde Sie solange siezen, bis wir uns persönlich kennenlernen und Sie mir erklären, von mir geduzt werden zu wollen. Danach dürfen Sie gern mit einem Rotstift durch das Buch gehen und jedes »Sie« durch ein »Du« ersetzen.

## ***Törichte Annahmen über den Leser***

Sie müssen nicht programmieren können, um dieses Buch zu verstehen. Gewisse Vorerfahrungen wären hilfreich, werden aber nicht vorausgesetzt. Ich erkläre alles.

Eine törichte Annahme ist, dass Ihnen der Umgang mit Android-Geräten nicht fremd ist. Warum sollten Sie Apps für etwas programmieren wollen, mit dem Sie selbst nicht umgehen können?

Die Programmierung von Android-Apps können Sie nicht auf einem Smartphone durchführen. Sie benötigen einen Computer. Daran führt kein Weg vorbei. Dieser Computer kann Linux, Windows oder macOS nutzen. Er sollte nicht an Altersschwäche leiden und reichlich Platz auf der Festplatte haben. Wenn der Hauptspeicher knapp und der Internetzugang langsam ist, wird es vielleicht etwas zäh. Dann müssen Sie halt etwas Geduld mitbringen.

# ***Wie dieses Buch aufgebaut ist***

Wir haben viel vor. Immerhin hat alles zwischen die Buchdeckel gepasst. Ihre Aufgabe ist es, zu lesen, zu verstehen und vor allem, selbst nachzuvollziehen. Denn mit dem Programmieren ist es wie mit dem Gitarrespielen. Sie lernen es nicht durch Zuschauen und Zuhören, sondern nur dadurch, dass Sie es selbst versuchen.

## ***Teil I: Werkzeugbeschaffung und Einrichtung***

In [Kapitel 1](#) werde ich Ihnen erzählen, wie Sie Ihren Arbeitsplatz für die App-Programmierung herrichten. So viel kann ich schon verraten: Die Entwicklungsumgebung ist kostenlos.

Im [Kapitel 2](#) werden Sie bereits eine erste App einrichten. Die kann allerdings nur »Guten Tag« sagen.

## ***Teil II: Wir programmieren***

In diesem Grundgerüst werden Sie in [Teil II](#) Ihre ersten Schritte in Java machen. Sie werden in [Kapitel 3](#) erfahren, wie man Zahlen und Texte verarbeitet und wie man unter Android Ein- und Ausgaben realisiert.

Anhand einer BMI-App werden Sie in [Kapitel 4](#) Abfragen und weitere Eingabelemente wie Radio-Buttons kennenlernen. Diese App kann nicht nur das Übergewicht des Autors beziffern, sondern auch zeigen, wie man mehrsprachige Apps organisiert.

Nicht nur dem Zufall, sondern auch den Zahlenkolonnen sind Sie in [Kapitel 5](#) auf der Spur. Sie werden Schleifen und Listen kennenlernen.

Die Grundlagen der objektorientierten Programmierung stehen in [Kapitel 6](#) auf dem Menü, und zwar am Beispiel der guten alten Bruchrechnung.

### ***Teil III: Ein eigenes grafisches Spiel***

Wenn Sie Spiele programmieren wollen, werden Sie Grafiken erzeugen wollen und auf die Aktionen des Anwenders reagieren müssen. In [Kapitel 7](#) werden Sie ein Spiel erstellen und dabei die Grundlagen von Java und objektorientierter Programmierung noch einmal vertiefen. In [Kapitel 8](#) werden wir für das Spiel eine Highscore-Liste erstellen, um Datenbanken und Listenanzeigen kennenzulernen.

### ***Teil IV: Wechselnde Displays und ihre Daten***

Smartphone-Apps müssen aufgrund des kleinen Bildschirms immer wieder die Displays wechseln. In [Kapitel 9](#) erfahren Sie, wie man eine solche App erstellt und wie sie die Daten übergibt. In [Kapitel 10](#) schauen wir kurz auf Fragmente und in [Kapitel 11](#) betrachten wir, wie man Dateien exportiert oder aus dem Internet lädt.

### ***Teil V: Ortskenntnis und Sensoren***

Fast jedes Smartphone hat einen GPS-Empfänger an Bord und in [Kapitel 12](#) werden Sie sehen, wie man dieses anspricht und damit Landkarten ansteuert. Da das GPS-System auch ein prima Spitzel ist, müssen Sie den Anwender dafür um Erlaubnis bitten.

In [Kapitel 13](#) schauen wir uns Sensoren an, welche Licht und Beschleunigung registrieren.

Für den Schluss habe ich mir in [Kapitel 14](#) das Thema Multimedia aufgehoben. Sie nutzen die Kamera für Fotos und Filme und setzen Mikrophon und Audiowiedergabe mit einer kleinen Diktaphon-App ein.

## ***Teil VI: Der Top-Ten-Teil***

Im Top-Ten-Teil finden Sie Trost, wenn Sie sich über den Emulator ärgern, und gute Strategien, wie Sie darauf Rücksicht nehmen, dass ein Smartphone-Display naturgemäß keine Großbildleinwand ist.

# ***Symbole, die in diesem Buch verwendet werden***

Damit Sie bestimmte Informationen schneller wiederfinden, habe ich das Buch mit Symbolen verziert:



Was hinter so einem Symbol steht, sollten Sie sich merken. Es ist quasi die Aufforderung, einen Spickzettel anzulegen.



Dieses Symbol zeigt an, dass es etwas technischer wird. Sie sollten es sich nicht gar so zu Herzen nehmen, wenn Sie nicht auf Anhieb verstehen, was bei so einem Symbol steht. Stellen Sie sich vor, dass der Informatiker in mir durchgegangen ist.

Übergehen Sie es aber nicht von vornherein! Ich hatte ja schließlich meinen Grund, es aufzuschreiben.



Manchmal kann man sich das Leben vereinfachen. Die Glühbirne weist Ihnen den Weg.



Mit diesem Symbol möchte ich Sie darauf hinweisen, dass Ihre Aufmerksamkeit gefragt ist. Vielleicht sollten Sie den Text dazu sicherheitshalber zwei Mal lesen.



Wenn das Warndreieck erscheint, mache ich mir etwas Sorgen. Schauen Sie bitte genauer hin. Sie könnten in eine Situation geraten, die Ihnen nicht gefällt.



Hier plaudere ich mal aus dem Informatikerkästchen. Das müssen Sie sich nicht merken und es auch nicht unbedingt lesen, um an Ihr Ziel zu gelangen.



Hier stelle ich Ihnen eine Aufgabe, die Sie versuchen sollten, selbst zu lösen, bevor Sie weiterlesen. Ich stelle Ihnen die Aufgabe, weil ich denke, dass Sie alles für die Lösung Nötige parat haben müssten. Programmieren lernt man durch eigenes Handeln. Wenn Sie die Aufgabe nicht allein lösen können, verlieren Sie nicht den Mut! Ich werde alles anschließend erläutern. Aber den meisten Gewinn haben Sie, wenn Sie selbst auf die Lösung kommen.

## ***Wie es weitergeht***

Sie werden hoffentlich das Buch lesen. Dafür habe ich es nämlich geschrieben. Und in diesem besonderen Fall ist es vielleicht gar nicht schlecht, wenn Sie es zwei Mal lesen. Ein noch so kleines Android-Programm enthält so

viel theoretischen Hintergrund, dass ich erst im Laufe des Buches alles erklären kann. Ich rechne also damit, dass manche Zusammenhänge erst beim zweiten Lesen richtig klar werden. Gönnen Sie sich die Zeit!



Ich werde Ihnen unter diesem Symbol immer wieder einmal vorschlagen, etwas selbst zu versuchen, wenn ich das Gefühl habe, Sie müssten nun alle notwendigen Informationen bereits zusammenhaben. Das sollten Sie dann auch wahrnehmen. Gleich anschließend werde ich eine Lösung vorschlagen, anhand derer Sie Ihre Ideen kontrollieren können.

In einigen Kapiteln unterbreite ich Ihnen Vorschläge für eigene Projekte. Damit schlage ich Ihnen etwa vor, wie Sie die Beispiel-App noch erweitern und das gerade Gelernte umsetzen könnten. Diese Projekte sind durchaus etwas anspruchsvoller und es ist völlig in Ordnung, wenn Sie daran einige Tage herumbasteln. Selbst ein Scheitern und späteres Wiederaufnehmen gehört zum Plan. Hier ist wirklich der Weg das Ziel. Darum halte ich in diesen Fällen eine Musterlösung für kontraproduktiv.

Ich habe für dieses Buch eine Webseite erstellt: <http://willemer.de/app4d>. Dort finden Sie Korrekturen und Verweise, die bei der Drucklegung noch nicht zur Verfügung standen.

Auch die im Buch beschriebenen Projekte habe ich dort zum Download bereitgestellt. Sie finden sie ebenso auf der Webseite des Verlags zum Buch: <http://www.wiley-vch.de/ISBN9783527718801>

# Teil I

## Werkzeugbeschaffung und Einrichtung



## **IN DIESEM TEIL...**

Bevor wir unsere erste App stricken, sollten wir uns Stricknadeln und Wolle beschaffen. Dazu werden wir sowohl Hardware als auch Software brauchen.

# Kapitel 1

## Entwicklungsumgebung

---

### IN DIESEM KAPITEL

Entwicklungsumgebung AndroidStudio

Virtuelle Smartphones

Echte Smartphones einbinden

---

Ich bin sicher, Sie besitzen bereits einen Computer. Falls nicht, wird es bitter. Kaufen Sie einen.

Überraschenderweise können Sie durchaus auch ohne Smartphone oder Tablet Android-Apps programmieren. Aber da Sie ja offenbar Apps für diese Geräte entwickeln wollen, werden Sie vermutlich auch (mindestens) eines davon besitzen.

## *Anforderungen an den Entwickler-PC*

Sie werden in diesem Buch zwar lernen, Programme für Smartphones und Tablets unter Android zu programmieren, aber entwickelt werden die Programme auf einem Computer. Und Sie werden auf lange Sicht froh darüber sein, die Quelltexte nicht mit der virtuellen Tastatur auf dem kleinen Display Ihres Smartphones bearbeiten zu müssen.

Das Betriebssystem Ihres PCs ist dabei weitgehend egal. Ich entwickle auf einem Linux-System, die meisten Programmierer werden vermutlich ein Windows-System

verwenden. Aber es ist auch möglich, Android-Apps auf einem Mac zu erstellen.

Die Entwicklungsumgebung wird Ihrem PC nicht sehr viel mehr abverlangen als ein gängiges Office-Paket, aber Sie werden die Beispiele ausprobieren wollen. Das geht zwar auch mit einem angekoppelten Smartphone, aber etwas einfacher wird es, wenn Sie den Emulator der Umgebung verwenden. Dieser Emulator tut so, als wäre er ein Smartphone. Ein komplettes Smartphone in einem PC zu simulieren, stellt allerdings Anforderungen an Ihre Hardware:

- ✓ Das Virtual-Flag Ihres Prozessors muss eingeschaltet sein. Das benötigt der Emulator, weil er eine sogenannte virtuelle Maschine ist. Gehen Sie erst einmal davon aus, dass das Virtual-Flag in Ordnung ist. Wenn nicht, wird sich der Emulator schon melden. Reparieren Sie nichts, was nicht kaputt ist!  
Bei einigen, vor allem älteren PCs ist das virtuelle Flag im BIOS beziehungsweise im UEFI abgeschaltet. Leider ist die Konfiguration bei jedem Computer etwas anders. Falls Sie diese Einstellung nicht selbst finden können, vertrauen Sie Ihrer Suchmaschine den Typ Ihres PCs sowie die Stichworte »UEFI« und »Virtual« an und Sie werden sicherlich eine Seite finden, die Ihnen weiterhilft.
- ✓ Der Hauptspeicher (RAM) sollte nicht auf Kante gestrickt sein. Der Emulator beherbergt schließlich ein komplettes Android-System. Das benötigt natürlich auch das eine oder andere Gigabyte extra.
- ✓ Auf der Festplatte sollten Sie auch noch ordentlich Platz haben. Allein für jede Android-Version, die im Emulator getestet werden soll, geht davon deutlich mehr als ein Gigabyte stiften.

- ✓ Diese Speichermengen müssen Sie herunterladen und sich darum durch Ihre Internetleitungen quälen. Dementsprechend sollten Sie auch in dieser Beziehung gut ausgestattet sein – oder mit großzügigen Pausen rechnen.

## *Smartphone oder Tablet*

Da Sie vor allem am Anfang hauptsächlich den Emulator nutzen werden, ist ein Android-Gerät nicht zwingend erforderlich. Aber es wäre natürlich sehr schade, wenn Sie die erstellten Apps nicht auch mal in freier Wildbahn testen könnten.

Dazu müssen Sie Ihr Smartphone per USB an den Entwicklungs-PC anschließen. Anschließend wird Ihre Entwicklungsumgebung auf das Gerät zugreifen wollen, um Programme installieren, starten und überwachen zu können. Diese Berechtigungen müssen Sie für Ihren PC aber zunächst Ihrem Smartphone abringen.

1. Starten Sie Ihr Smartphone und gehen Sie dort in die Einstellungen.
2. Wechseln Sie auf den Punkt SYSTEM.
3. Dort gibt es einen Punkt ÜBER DAS TELEFON.
4. Gegen Ende der Liste finden Sie die Build-Nummer.
5. Diese tippen Sie sieben Mal an. Kein Scherz. Das haben sich die Entwickler bei den Gebrüder Grimm abgeschaut.
6. Anschließend erscheint eine Meldung, welche den Entwicklermodus bestätigt.

Bei manchen Geräten gibt es einen Eintrag ENTWICKLEROPTIONEN. Dort schalten Sie die Option USB-DEBUGGING ein.



Das Umschalten in den Entwicklermodus erlaubt den Zugang über USB. Sie schalten damit die sogenannte Android Debug Bridge (ADB) frei. Die ADB ermöglicht die Ausführung von Kommandozeilenprogrammen auf Ihrem Smartphone über USB. Dies nutzt die Entwicklungsumgebung zum Ausführen Ihrer Programme.



Die ADB kann aber auch nützlich sein, wenn Ihr Display zerbrochen ist und Sie Ihr Smartphone auf direktem Weg nicht mehr steuern können. Wenn es zuvor im Entwicklermodus war, bleibt Ihnen immerhin die Möglichkeit über die ADB an Ihre Daten zu kommen. Für eine genaue Beschreibung fehlt hier der Platz. Falls Sie einmal dafür Verwendung haben, weiß Ihre Suchmaschine mehr darüber.

Sie müssen Ihrem Smartphone noch erlauben, Apps aus unbekannter Quelle zu installieren. Andere Installationsquellen als Google lehnt ein Android-Gerät aus Sicherheitsgründen und zum Schutz der Einnahmen von Google ab. Wenn Sie allerdings Ihre eigenen Programme entwickeln, stammt Ihr Programm zwangsläufig nicht aus Google Play. Auch diesen Schalter finden Sie in den Einstellungen Ihres Smartphones.



Es kann sein, dass Ihr Android-Gerät sich etwas anders verhält als hier beschrieben. Hersteller haben manchmal putzige Ideen. Falls sich Ihr Gerät nicht an dieses Buch hält, rügen Sie es dafür ordentlich und bitten Sie eine Suchmaschine um Hilfe.