

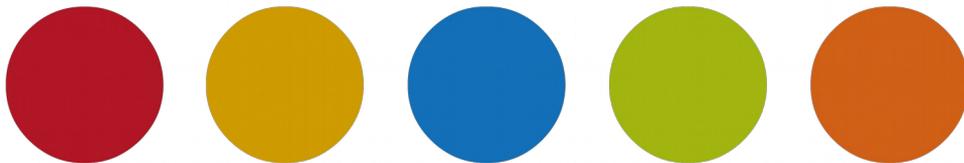
Android Schulung

Grundlagen Skript 2016

Konfigurations- und Schulungsunterlagen

Schulung:	Einführung in die Programmierung von Android Apps anhand klassischer Unterrichtsbeispiele
Referent:	Christine Janischek

Stand: 6. Jun 2016



© Christine Janischek

Inhaltsverzeichnis

1 Allgemeines.....	3
1.1 Überblick Projektthemen.....	4
1.2 Starten der Entwicklungsumgebung.....	5
2 Grundlagen: Projekte erstellen.....	8
2.1 Das Projekt BMI-Rechner 1.0.....	8
2.2 Das Projekt Taschenrechner 1.0.....	12
2.3 Das Projekt Währungsrechner 1.0.....	16
3 Modell: Implementierung der Fachklassen.....	21
3.1 Die Fachklasse des BMI-Rechner 1.0.....	21
3.2 Die Fachklasse des Taschenrechner 1.0.....	25
3.3 Die Fachklasse des Währungsrechner 1.0.....	29
4 View: Layouts, Komponenten & XML.....	36
4.1 Die Benutzeroberfläche des BMI-Rechner 1.0.....	36
4.2 Die Benutzeroberfläche des Taschenrechner 1.0.....	58
4.3 Die Benutzeroberfläche des Währungsrechner 1.0.....	72
5 Controller: Implementierung der Ereignissteuerung.....	85
5.1 Ereignissteuerung des BMI-Rechners 1.0.....	85
5.2 Ereignissteuerung des Taschenrechners 1.0.....	95
5.3 Ereignissteuerung des Währungsrechners 1.0.....	105
6 Projekte und Erweiterungen.....	116
6.1 Erweiterung des BMI-Rechners 2.0.....	116
6.2 Erweiterung des Taschenrechners 2.0.....	117
6.2.1 DecimalFormat zu Anzeige gerundeter Ergebnisse.....	118
6.2.2 Effizienter Quellcode: Hilfsmethoden.....	119
6.3 Variante des Währungsrechners 2.0.....	123
7 Installation und Konfiguration der Entwicklungsumgebung.....	131
7.1 Installation.....	131
7.2 Einstellungen.....	140
7.3 Hinweise.....	147
7.4 Fehler.....	154
7.5 Top 10 der Hilfestellungen.....	158
7.6 Gradle.....	159
7.7 UML-Modelling PlugIn.....	160

1 Allgemeines



Das Skript schildert den Umgang mit Android Studio anhand von konkreten Beispielen die unter Umständen auch in den Unterricht im Fachbereich Wirtschaftsinformatik respektive im Fachbereich Informatik einbetten lassen.

Aktuelle Versionen des Skriptes selbst und die im Skript behandelten Quellcodes können Sie online herunterladen und testen:

Skript & Sources:

Download: Skript Android App Schulung

Download: Skript Android App Schulung (ODT)

Quellcodes: Android Studio Projekte mit den Lösungen

→ [Hier in Chrissis Edublog herunterladen](#)



Für alle Inhalte gilt natürlich das Urheberrecht. Ich selber achte auch darauf. Um Details zur Creative-Commons-Lizenz für die von mir selbst verfassten Texte und Quellcodes zu erhalten, klicken Sie links auf das CC-BY-NC-SA-Logo. Für Ergänzungs- und/oder Verbesserungsvorschläge schreiben Sie mir bitte eine E-Mail: cjanischek@gmx.de

Alle Quellangaben wurden nach bestem Gewissen genannt und aufgeführt. Permanent begleitende Literatur waren:

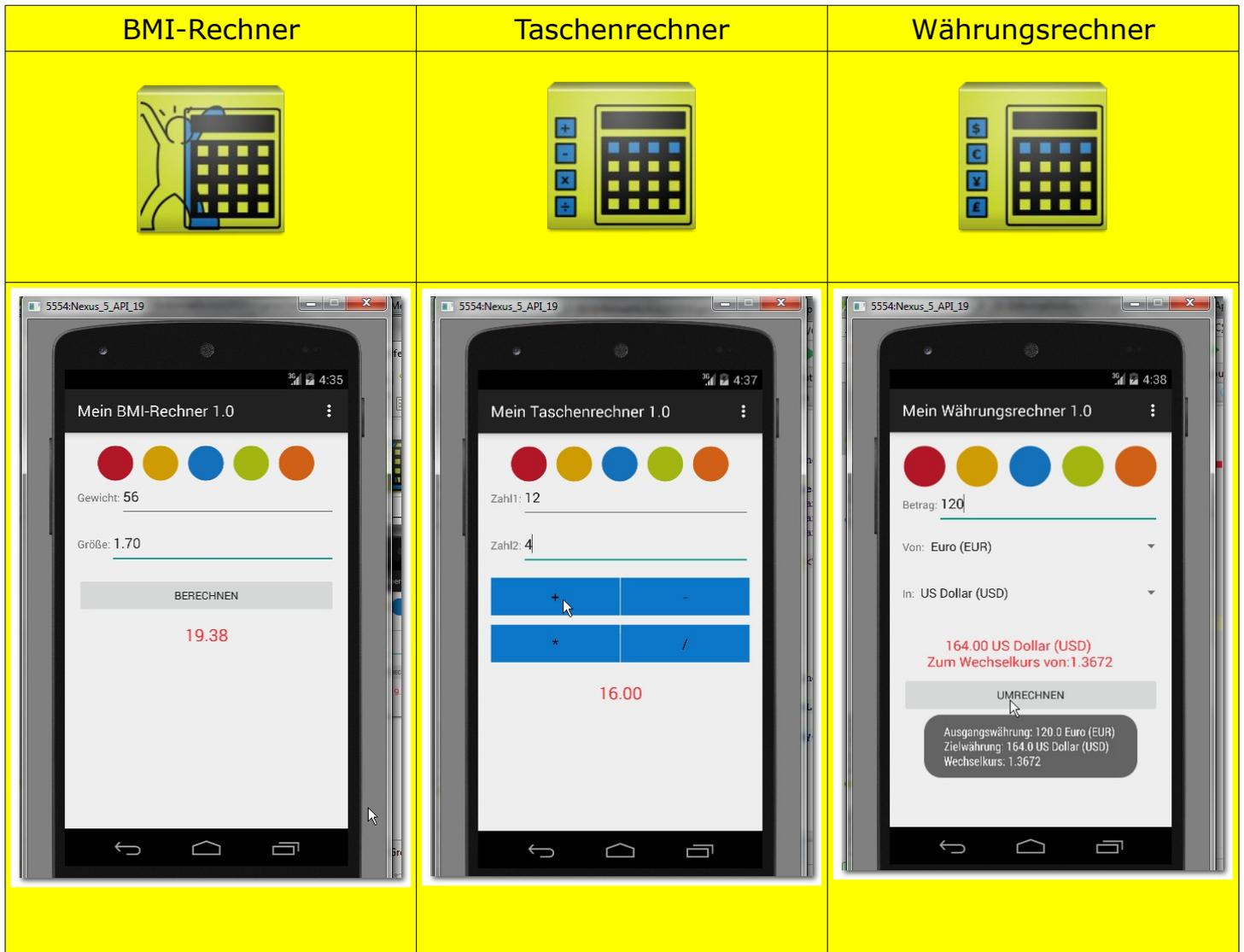
[KUE00]

Künneht, Thomas "Android 4 – Apps entwickeln mit dem Android SDK", 978-3-8362-1948-8, 2013, Galileo Computing

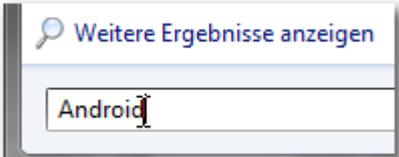
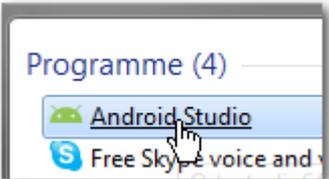
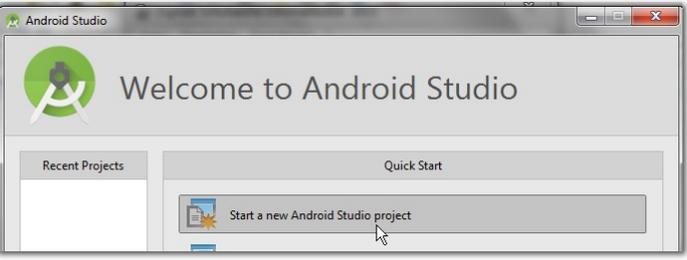
[KUE01]

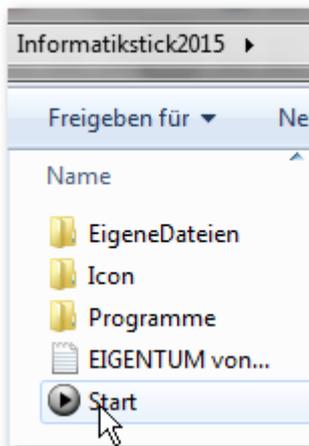
Künneht, Thomas "Android 5 – Apps entwickeln mit Android Studio", 978-3-8362-2665-3, 2015, Galileo Computing

1.1 Überblick Projektthemen



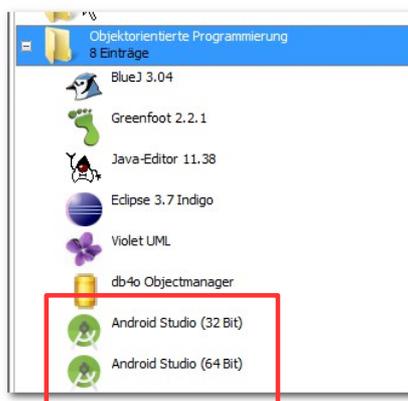
1.2 Starten der Entwicklungsumgebung

 <p>A screenshot of the Windows Start menu search bar. The search bar is labeled 'Alle Programme' and contains the text 'Programme/Dateien durchsuchen'. Below the search bar, several application icons are visible, including the Windows logo, Internet Explorer, and others.</p>	<p>Variante 1: Starten der lokal installierten Anwendung.</p> <p>Klicken Sie auf die Schaltfläche → Start und geben Sie den Begriff → Android in das Suchfeld ein.</p>  <p>A screenshot of the search results for 'Android'. The search bar contains the text 'Android' and the button 'Weitere Ergebnisse anzeigen' is visible above it.</p>
 <p>A screenshot of the Windows Start menu search results. The search bar is labeled 'Programme (4)' and shows the results 'Android Studio' and 'Free Skype voice and video calls'.</p>	<p>Starten Sie die Entwicklungsumgebung „Android Studio“.</p> <p>Im Bereich → Programme erscheint daraufhin die Option → Android Studio</p> <p>Klicken Sie diese Option an, um die Entwicklungsumgebung zu starten.</p>
 <p>A screenshot of the Android Studio Welcome dialog. The dialog is titled 'Welcome to Android Studio' and shows the 'Quick Start' menu with the option 'Start a new Android Studio project' selected.</p>	<p>Ein Neues Projekt erzeugen.</p> <p>Der angezeigte Dialog öffnet sich für den Fall, dass zuvor alle Projekte geschlossen wurden bzw. die Entwicklungsumgebung erstmalig geöffnet wurde.</p> <p>Um ein neues Projekt zu erzeugen, wählen Sie im Quick Start-Menü die Option → Start a new Android Studio project.</p>



Variante 2:
Starten der Anwendung aus der Digitalen Tasche (Informatikstick).

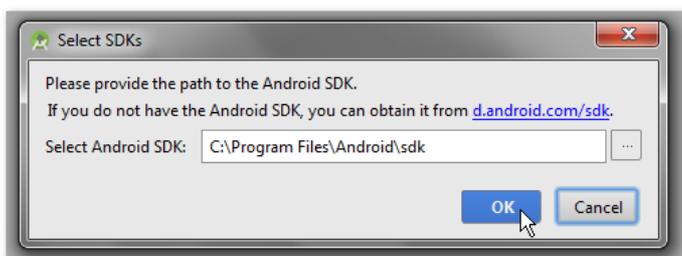
Starten Sie die Digitale Tasche mit einem Klick auf die Schaltfläche → Start.



Starten Sie die Entwicklungsumgebung „Android Studio“.

Je nachdem welche Hardwarevoraussetzungen gegeben sind, besteht hier die Wahl zwischen der 32 Bit und 64 Bit-Variante.

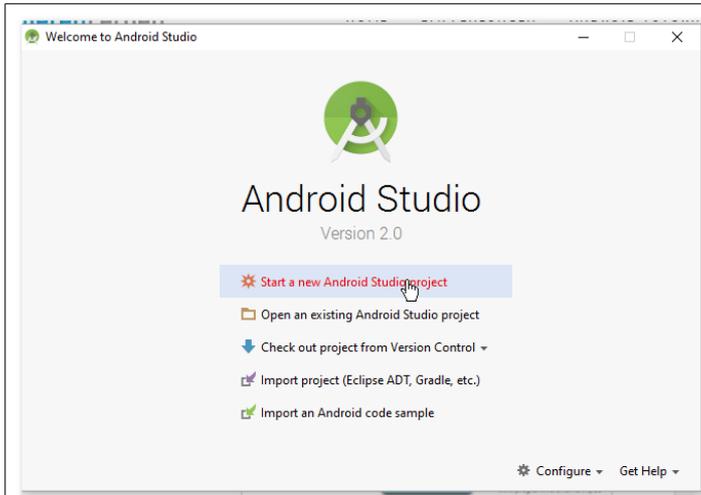
Wählen Sie entsprechend die geeignete Variante und klicken Sie die Schaltfläche doppelt an.



Aktueller Pfad zur SDK herstellen.

Für den Fall, dass die Entwicklungsumgebung bei erstmaligen öffnen die Angabe des Ortes der installierten SDK anfordert, müssen Sie den entsprechenden Pfad angeben.

Der Software Development Kit (SDK). Der Umfang der SDK nimmt entsprechend viel Speicher in Anspruch. Mittlerweile beträgt der Umfang >20 GB.



Android Studio: Start Menü

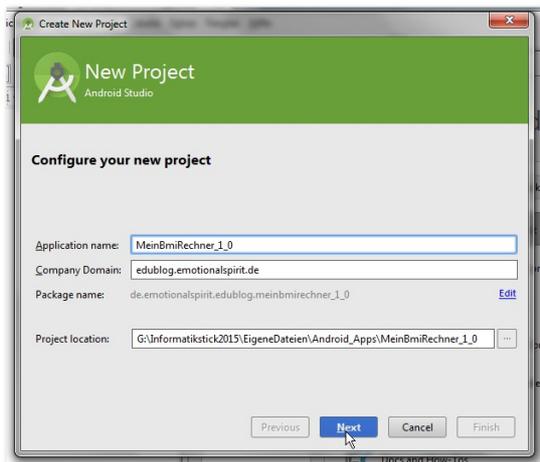
Ein Neues Projekt erzeugen.

Der angezeigte Dialog öffnet sich für den Fall, dass zuvor alle Projekte geschlossen wurden bzw. die Entwicklungsumgebung erstmals geöffnet wurde.

Um ein neues Projekt zu erzeugen, wählen Sie im Quick Start-Menü die Option → Start a new Android Studio project.

2 Grundlagen: Projekte erstellen

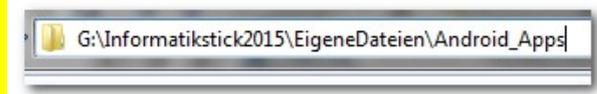
2.1 Das Projekt BMI-Rechner 1.0



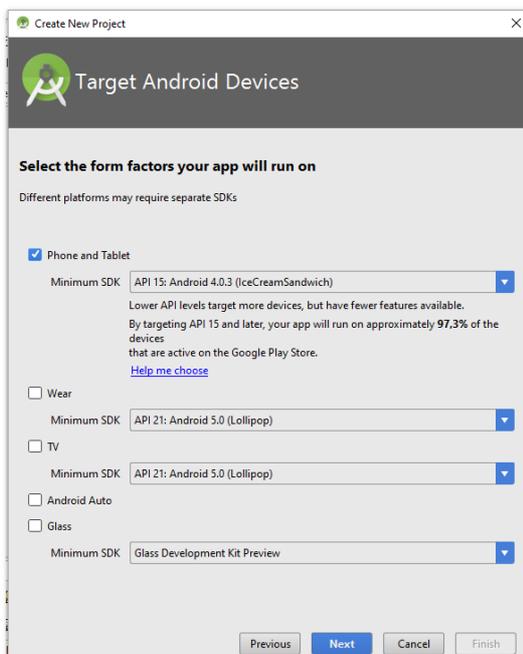
Legen Sie nun schrittweise die Eigenschaften für Ihr neues Android-Projekt fest.

Geben Sie dazu die nebenstehend angezeigten Angaben für

1. Application name:
Der Anwendungsname.
2. Company Domain:
Ihre Internetadresse oder die Ihrer Schule.
3. Project location:
Wir nutzen bestenfalls den bereits vorhandenen Arbeitsbereich in
→ EigeneDateien\Android_Apps der Digitalen Tasche auf dem USB-Stick.



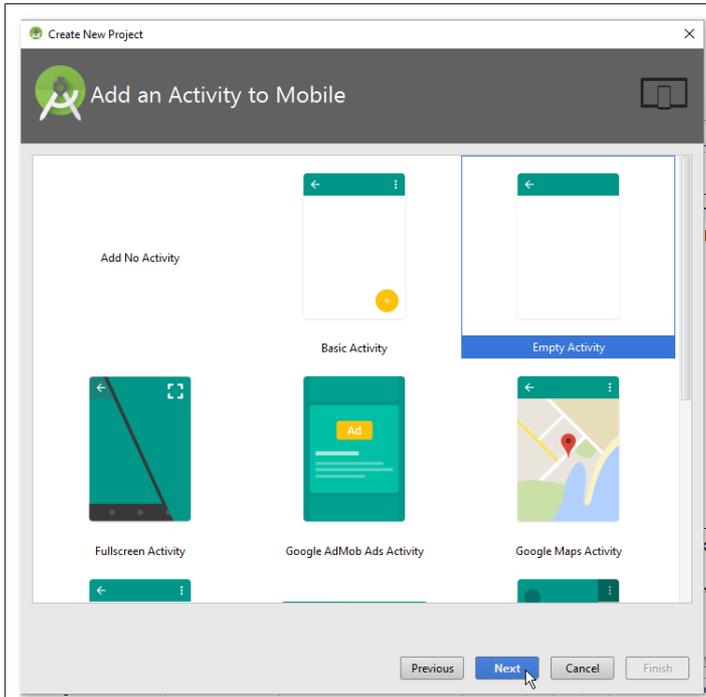
Je nach Konfiguration kann der Buchstabe des Laufwerks variieren.



Wir wählen als Ziel unserer Anwendung das API Level, mit der höchsten Abdeckung für die Lauffähigkeit auf verfügbaren Android Geräten, aus.

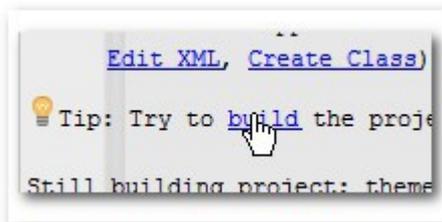
Der Assistent macht uns dazu einen Vorschlag für Telefone und Tablets.

Wir nehmen den Vorschlag an und klicken auf die Schaltfläche → Next.



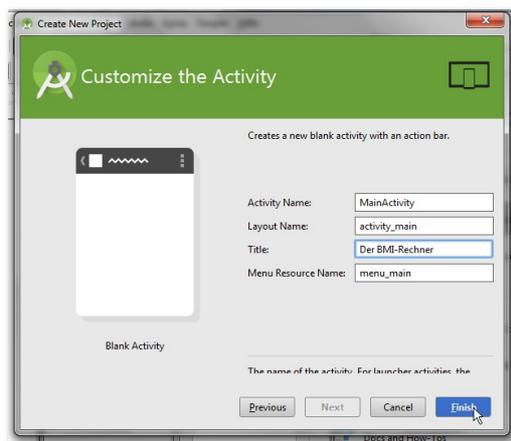
Im ersten Schritt nutzen wir die einfachste Form zur Steuerung von Ereignissen. Die → Empty Activity.

Built → Rebuilt Project
bei Rendering Problemen:



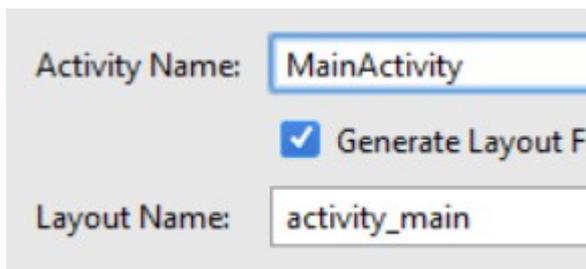
Aktuelle Hinweise – Projekt erstellen

Beim Erstellen eines neuen Projektes hat man aktuell die Wahl zwischen einer Blank und Empty Activity. Wenn man wie bisher eine Blank Activity erzeugt werden auch zunächst Rendering Probleme im Previewer angezeigt, die verschwinden aber wenn man ein rebuilt macht.



Hinweis:

Seit Android Studio 2.0 ist an dieser Stelle nur die Angabe des Activity Namens und des Layout Namens zwingend erforderlich.

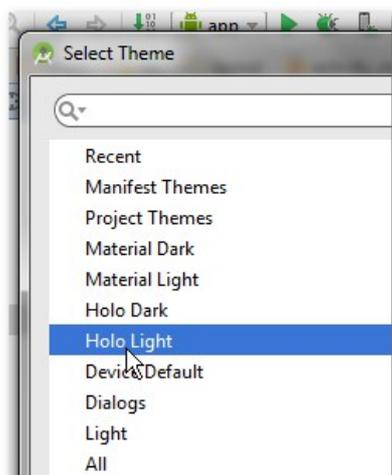


Unsere Anwendung wird vorerst mit einer einzigen Activity Klasse auskommen. Wir signalisieren mit dem Signalwort → Main, dass es sich um den Startpunkt der Anwendung handelt.

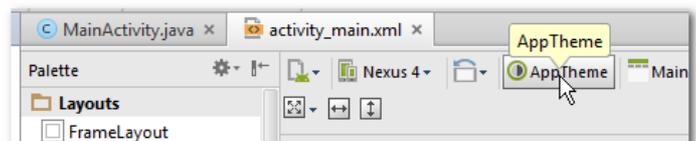
Zu den benötigten Angaben:

1. **Activity Name:**
Name der ereignissteuernden Klasse (*Controller* → *Klasse*).
2. **Layout Name:**
Name der XML-Datei. Sie beschreibt die Darstellung der zugehörigen Benutzeroberfläche (*View* → *XML-Datei*).
3. **Title:**
Wir legen mit dem Titel den angezeigten Text in der Titel-Leiste fest. Dementsprechend variieren wir hier die Angabe, wie nebenstehend angezeigt.
4. **Menu Resource Name:**
Jede Aktivität hat ein eigenes Menü. Die Zusammensetzung und Darstellung wird in einer eigens dafür erzeugten XML-Datei definiert. Wir nutzen in unserem Fall das Hauptmenü → menu_main

Klicken Sie auf die Schaltfläche → Finish, um den Konfigurationsvorgang abzuschließen.



Wählen Sie dann das geeignete „AppTheme“. Sie finden diese Schaltfläche oberhalb der Design-Bühne in der Symbolleiste.



Themes sind vorgeschlagene Farbgebungsvarianten für unsere Benutzeroberflächen.

Hierzu sollten die Standards berücksichtigt werden. Google gibt dazu folgenden Rat:

1. **App Design Theme: Holo Light**
Schwarze Schrift auf weißem Grund. Die meisten Apps nutzen diese Vorgabe.

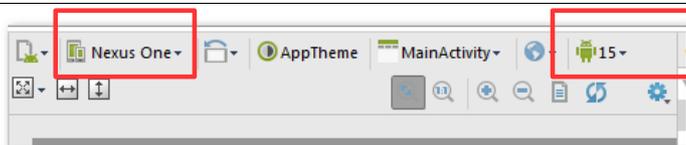
2. Setting Design Theme: Holo Dark

Weißer Schrift auf schwarzem Grund. Kommt meistens nur für die Einstellungsmöglichkeiten (Settings) der App zum Einsatz.

Wählen Sie für die App das Theme „Holo Light“. Bestätigen Sie die Angabe im Fenster „Select Theme“ mit einem Klick auf die Schaltfläche → OK.

Hinweis:

Die Wahl dieses Themes hindert uns nachher nicht daran, unsere App farblich individuell zu gestalten. Sie trifft nur die Grundsatzentscheidung „Dunkel auf Hell“ bzw. „Hell auf Dunkel“. Dort sollten wir sinnvollerweise den Gewohnheiten der vielen App-Nutzer Folge leisten.



Exkurs: Geräte (AVD) und API Level.

Geräte (AVD).

Das Android Virtual Device entspricht dem mobilen Endgerät das emuliert, also vom Emulator erzeugt wird, um Anwendungen darauf testen zu können. Wichtig zu wissen ist, dass der Emulator und die Auswahl an Geräten abhängig ist von der Hardware-Ausstattung des Testrechners.

API Level.

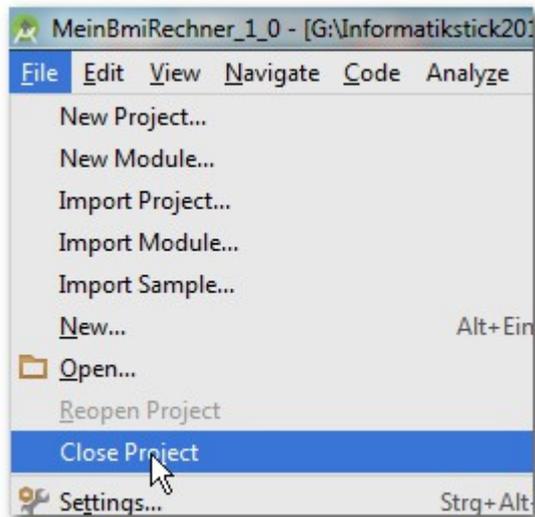
Das API Level bestimmt die verwendete Betriebssystemversion für das emulierte mobile Endgerät.

Die Verwaltung installierter Geräte und Betriebssystemversionen übernimmt der SDK Manager.

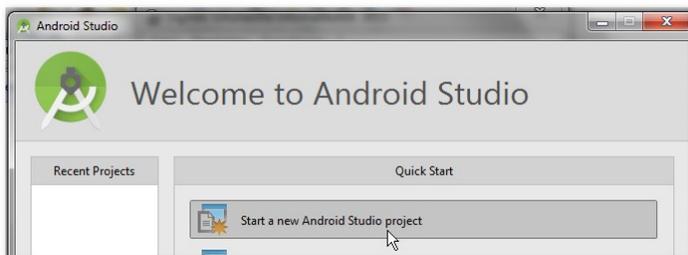
Der Software Development Kit (SDK).

Der Umfang der SDK nimmt entsprechend viel Speicher in Anspruch. Mittlerweile beträgt der Umfang nahezu 25 GB und ist damit in den meisten Fällen zu umfangreich für die Installation auf einem USB-Stick.

2.2 Das Projekt Taschenrechner 1.0

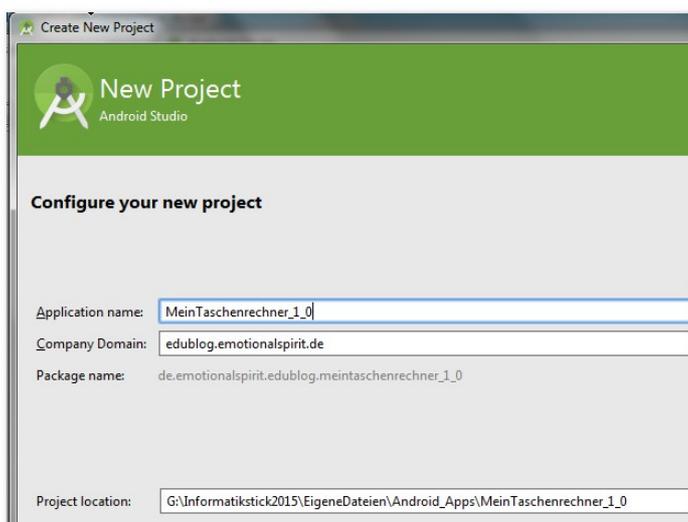


Klicken Sie in der Menü-Leiste die Option File → Close Project, um das noch geöffnete Projekt zu schließen.



Der angezeigte Dialog öffnet sich für den Fall, dass zuvor alle Projekte geschlossen wurden bzw. die Entwicklungsumgebung erstmals geöffnet wird.

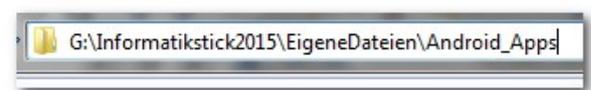
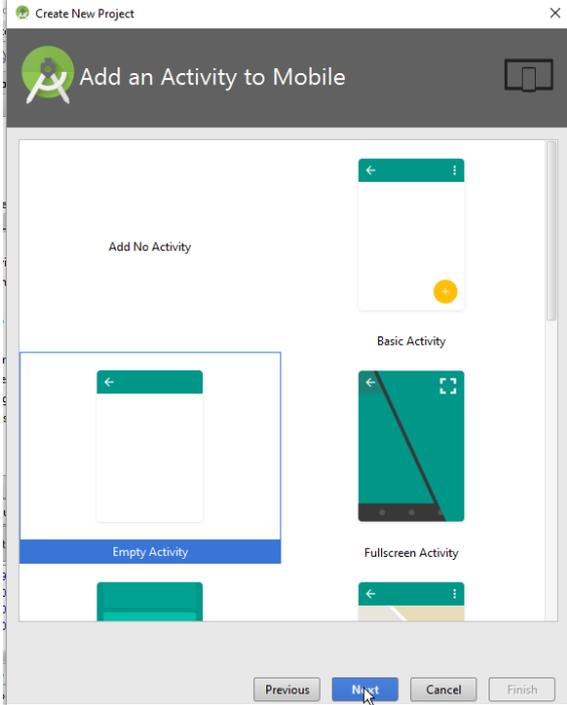
Um ein neues Projekt zu erzeugen, wählen Sie im Quick Start-Menü die Option → Start a new Android Studio project.

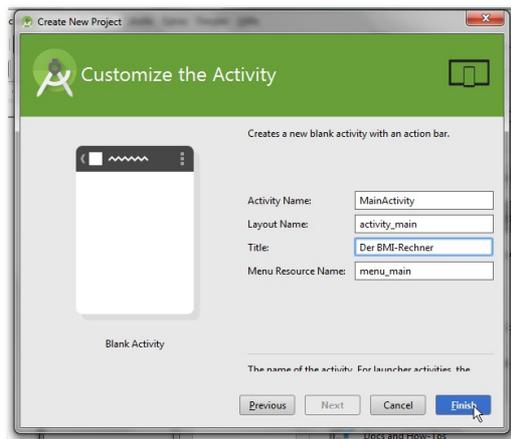


Legen Sie nun schrittweise die Eigenschaften für Ihr neues Android-Projekt fest.

Geben Sie dazu die nebenstehend angezeigten Angaben für

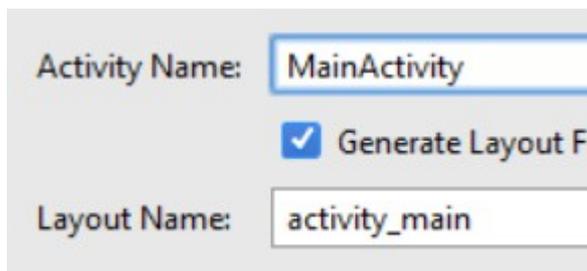
5. Application name:
Der Anwendungsname.
6. Company Domain:
Ihre Internetadresse oder die Ihrer Schule.
7. Project location:
Wir nutzen bestenfalls den bereits vorhandenen Arbeitsbereich in → EigeneDateien\Android_Apps der Digitalen Tasche auf dem USB-Stick.

	 <p>Je nach Konfiguration kann der Buchstabe des Laufwerks variieren.</p>
	<p>Wir wählen als Ziel unserer Anwendung das API Level, mit der höchsten Abdeckung für die Lauffähigkeit auf verfügbaren Android Geräten, aus.</p> <p>Der Assistent macht uns dazu einen Vorschlag für Telefone und Tablets.</p> <p>Wir nehmen den Vorschlag an und klicken auf die Schaltfläche → Next.</p>
	<p>Im ersten Schritt nutzen wir die einfachste Form zur Steuerung von Ereignissen. Die → Empty Activity.</p>



Hinweis:

Seit Android Studio 2.0 ist an dieser Stelle nur die Angabe des Activity Namens und des Layout Namens zwingend erforderlich.

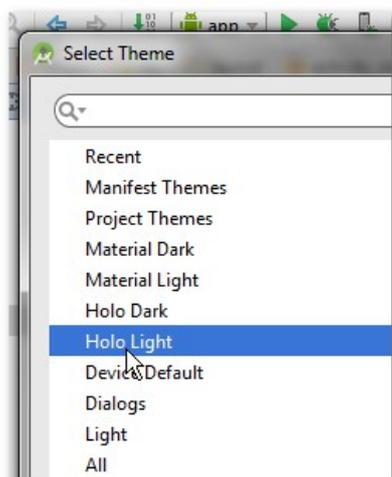


Unsere Anwendung wird vorerst mit einer einzigen Activity Klasse auskommen. Wir signalisieren mit dem Signalwort → Main, dass es sich um den Startpunkt der Anwendung handelt.

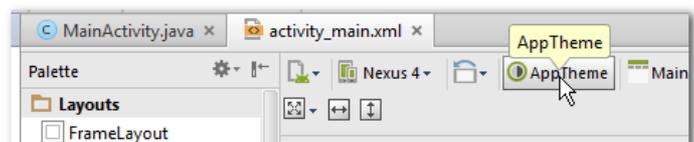
Zu den benötigten Angaben:

1. Activity Name:
Name der ereignissteuernden Klasse (*Controller* → *Klasse*).
2. Layout Name:
Name der XML-Datei. Sie beschreibt die Darstellung der zugehörigen Benutzeroberfläche (*View* → *XML-Datei*).
3. Title:
Wir legen mit dem Titel den angezeigten Text in der Titel-Leiste fest. Dementsprechend variieren wir hier die Angabe, wie nebenstehend angezeigt.
4. Menu Resource Name:
Jede Aktivität hat ein eigenes Menü. Die Zusammensetzung und Darstellung wird in einer eigens dafür erzeugten XML-Datei definiert. Wir nutzen in unserem Fall das Hauptmenü → menu_main

Klicken Sie auf die Schaltfläche → Finish, um den Konfigurationsvorgang abzuschließen.



Wählen Sie dann das geeignete „AppTheme“. Sie finden diese Schaltfläche oberhalb der Design-Bühne in der Symbolleiste.



Themes sind vorgeschlagene Farbgebungsvarianten für unsere Benutzeroberflächen.

Hierzu sollten die Standards berücksichtigt werden. Google gibt dazu folgenden Rat:

3. App Design Theme: Holo Light
Schwarze Schrift auf weißem Grund. Die meisten Apps nutzen diese Vorgabe.

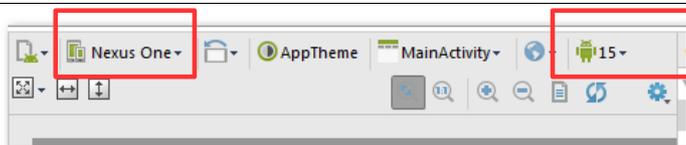
4. Setting Design Theme: Holo Dark

Weißer Schrift auf schwarzem Grund. Kommt meistens nur für die Einstellungsmöglichkeiten (Settings) der App zum Einsatz.

Wählen Sie für die App das Theme „Holo Light“. Bestätigen Sie die Angabe im Fenster „Select Theme“ mit einem Klick auf die Schaltfläche → OK.

Hinweis:

Die Wahl dieses Themes hindert uns nachher nicht daran, unsere App farblich individuell zu gestalten. Sie trifft nur die Grundsatzentscheidung „Dunkel auf Hell“ bzw. „Hell auf Dunkel“. Dort sollten wir sinnvollerweise den Gewohnheiten der vielen App-Nutzer Folge leisten.



Exkurs: Geräte (AVD) und API Level.

Geräte (AVD).

Das Android Virtual Device entspricht dem mobilen Endgerät das emuliert, also vom Emulator erzeugt wird, um Anwendungen darauf testen zu können. Wichtig zu wissen ist, dass der Emulator und die Auswahl an Geräten abhängig ist von der Hardware-Ausstattung des Testrechners.

API Level.

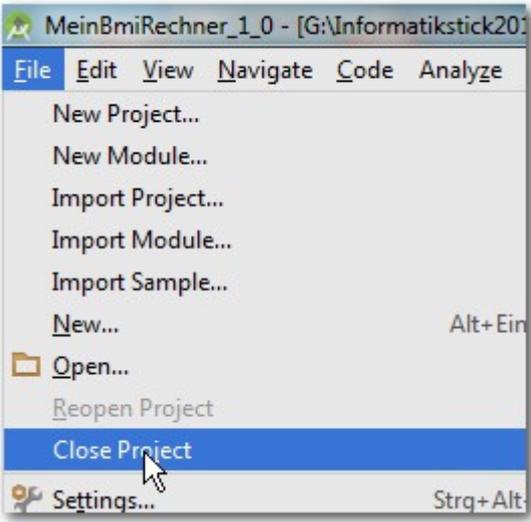
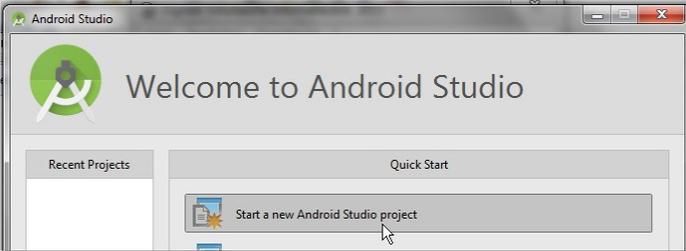
Das API Level bestimmt die verwendete Betriebssystemversion für das emulierte mobile Endgerät.

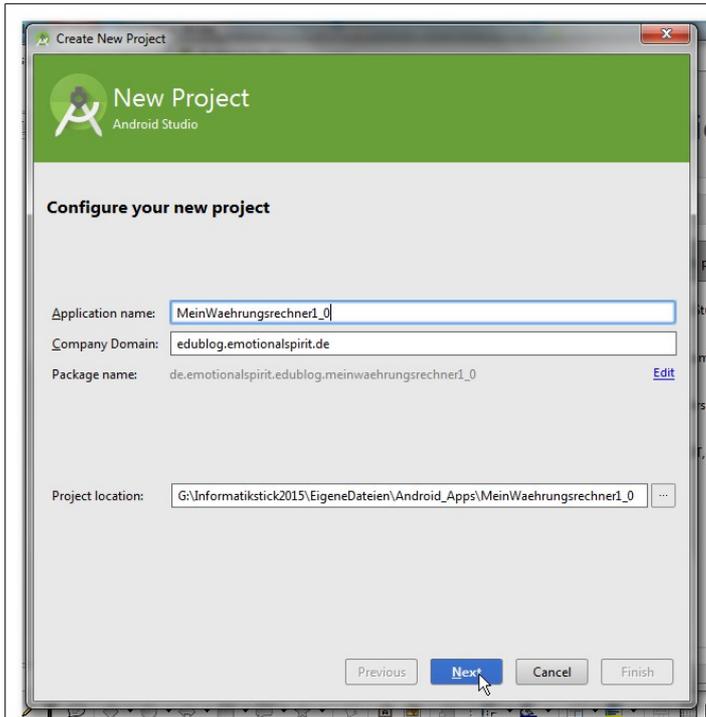
Die Verwaltung installierter Geräte und Betriebssystemversionen übernimmt der SDK Manager.

Der Software Development Kit (SDK).

Der Umfang der SDK nimmt entsprechend viel Speicher in Anspruch. Mittlerweile beträgt der Umfang nahezu 25 GB und ist damit in den meisten Fällen zu umfangreich für die Installation auf einem USB-Stick.

2.3 Das Projekt Währungsrechner 1.0

 <p>The screenshot shows the 'File' menu of an Android Studio project named 'MeinBmiRechner_1_0'. The menu items are: New Project..., New Module..., Import Project..., Import Module..., Import Sample..., New... (with Alt+Ein shortcut), Open..., Reopen Project, Close Project (highlighted with a mouse cursor), and Settings... (with Strg+Alt shortcut).</p>	<p>Klicken Sie in der Menü-Leiste die Option File → Close Project, um das noch geöffnete Projekt zu schließen.</p>
 <p>The screenshot shows the 'Welcome to Android Studio' dialog. It features a 'Recent Projects' list on the left and a 'Quick Start' section on the right. In the 'Quick Start' section, the option 'Start a new Android Studio project' is highlighted with a mouse cursor.</p>	<p>Der angezeigte Dialog öffnet sich für den Fall, dass zuvor alle Projekte geschlossen wurden bzw. die Entwicklungsumgebung erstmalig geöffnet wird.</p> <p>Um ein neues Projekt zu erzeugen, wählen Sie im Quick Start-Menü die Option „Start a new Android Studio project“.</p>



Legen Sie nun schrittweise die Eigenschaften für Ihr neues Android-Projekt fest.

Geben Sie dazu die nebenstehend angezeigten Angaben für

5. Application name:
Der Anwendungsname.
6. Company Domain:
Ihre Internetadresse oder die Ihrer Schule.
7. Project location:
Wir nutzen bestenfalls den bereits vorhandenen Arbeitsbereich in
→ EigeneDateien\Android_Apps der Digitalen Tasche auf dem USB-Stick.

G:\Informatikstick2015\EigeneDateien\Android_Apps

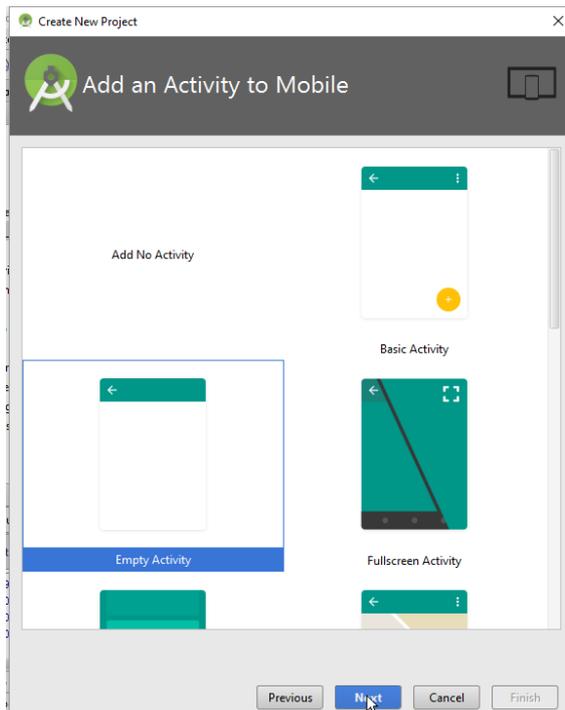
Je nach Konfiguration kann der Buchstabe des Laufwerks variieren.



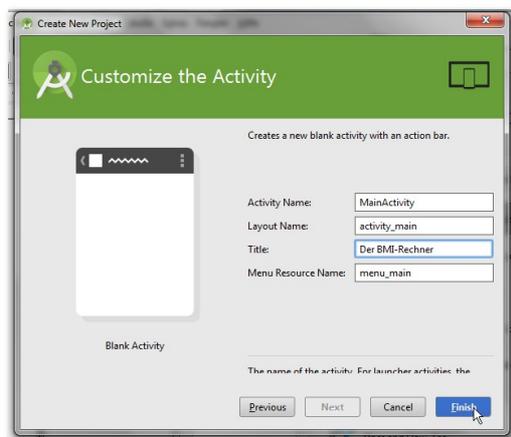
Wir wählen als Ziel unserer Anwendung das API Level, mit der höchsten Abdeckung für die Lauffähigkeit auf verfügbaren Android Geräten, aus.

Der Assistent macht uns dazu einen Vorschlag für Telefone und Tablets.

Wir nehmen den Vorschlag an und klicken auf die Schaltfläche → Next.



Im ersten Schritt nutzen wir die einfachste Form zur Steuerung von Ereignissen. Die → Empty Activity.



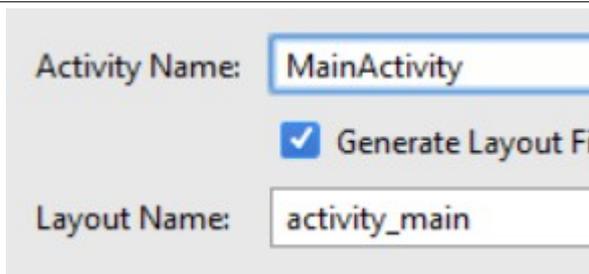
Unsere Anwendung wird vorerst mit einer einzigen Activity Klasse auskommen. Wir signalisieren mit dem Signalwort → Main, dass es sich um den Startpunkt der Anwendung handelt.

Zu den benötigten Angaben:

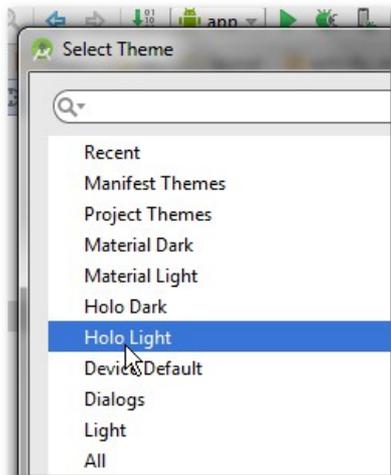
1. Activity Name:
Name der ereignissteuernden Klasse (*Controller* → *Klasse*).
2. Layout Name:
Name der XML-Datei. Sie beschreibt die Darstellung der zugehörigen Benutzeroberfläche (*View* → *XML-Datei*).
3. Title:
Wir legen mit dem Titel den angezeigten Text in der Titel-Leiste fest. Dementsprechend variieren wir hier die Angabe, wie nebenstehend angezeigt.
4. Menu Resource Name:
Jede Aktivität hat ein eigenes Menü. Die Zusammensetzung und Darstellung wird in einer eigens dafür erzeugten XML-Datei definiert. Wir nutzen in unserem Fall das Hauptmenü → menu_main

Hinweis:

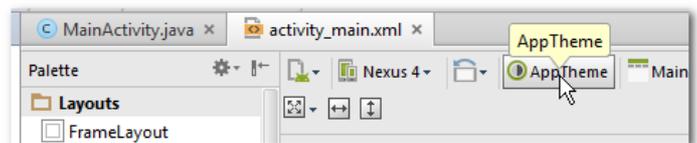
Seit Android Studio 2.0 ist an dieser Stelle nur die Angabe des Activity Namens und des Layout Namens zwingend erforderlich.



Klicken Sie auf die Schaltfläche → Finish, um den Konfigurationsvorgang abzuschließen.



Wählen Sie dann das geeignete „AppTheme“. Sie finden diese Schaltfläche oberhalb der Design-Bühne in der Symbolleiste.



Themes sind vorgeschlagene Farbgebungsvarianten für unsere Benutzeroberflächen.

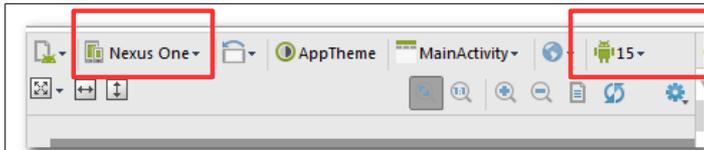
Hierzu sollten die Standards berücksichtigt werden. Google gibt dazu folgenden Rat:

5. App Design Theme: Holo Light
Schwarze Schrift auf weißem Grund. Die meisten Apps nutzen diese Vorgabe.
6. Setting Design Theme: Holo Dark
Weiße Schrift auf schwarzem Grund. Kommt meistens nur für die Einstellungsmöglichkeiten (Settings) der App zum Einsatz.

Wählen Sie für die App das Theme „Holo Light“. Bestätigen Sie die Angabe im Fenster „Select Theme“ mit einem Klick auf die Schaltfläche → OK.

Hinweis:

Die Wahl dieses Themes hindert uns nachher nicht daran, unsere App farblich individuell zu gestalten. Sie trifft nur die Grundsatzentscheidung „Dunkel auf Hell“ bzw. „Hell auf Dunkel“. Dort sollten wir sinnvollerweise den Gewohnheiten der vielen App-Nutzer Folge leisten.



Exkurs: Geräte (AVD) und API Level.

Geräte (AVD).

Das Android Virtual Device entspricht dem mobilen Endgerät das emuliert, also vom Emulator erzeugt wird, um Anwendungen darauf testen zu können. Wichtig zu wissen ist, dass der Emulator und die Auswahl an Geräten abhängig ist von der Hardware-Ausstattung des Testrechners.

API Level.

Das API Level bestimmt die verwendete Betriebssystemversion für das emulierte mobile Endgerät.

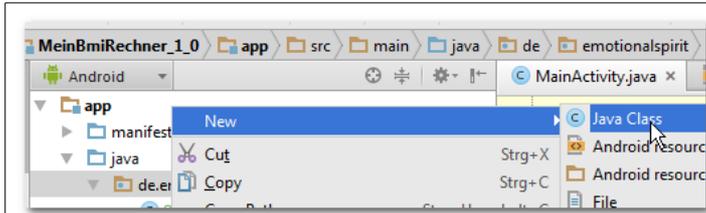
Die Verwaltung installierter Geräte und Betriebssystemversionen übernimmt der SDK Manager.

Der Software Development Kit (SDK).

Der Umfang der SDK nimmt entsprechend viel Speicher in Anspruch. Mittlerweile beträgt der Umfang nahezu 25 GB und ist damit in den meisten Fällen zu umfangreich für die Installation auf einem USB-Stick.

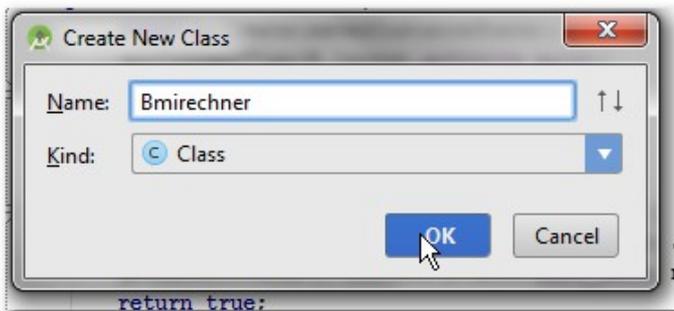
3 Modell: Implementierung der Fachklassen

3.1 Die Fachklasse des BMI-Rechner 1.0



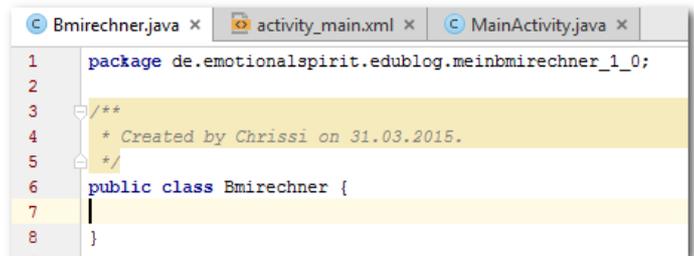
Neue Fachklasse erstellen.

Klicken Sie im „app“-Verzeichnis mit der rechten Maustaste auf das Package und wählen Sie die Option New → Java Class.



Klassenname festlegen.

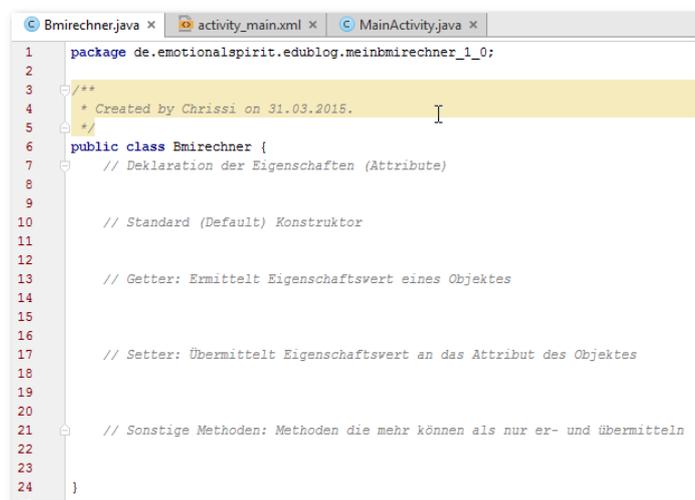
Geben Sie als Klassenname → Bmirechner ein und klicken Sie auf die Schaltfläche → OK.



Grundgerüst einer Klasse festlegen.

1. Deklaration der Attribute
2. Deklaration des Konstruktors
3. Get-Methoden (Getter) deklarieren und implementieren.
4. Set-Methode (Setter) deklarieren und implementieren.
5. Sonstige Methoden deklarieren und implementieren

Übernehmen Sie die nebenstehend angezeigten Kommentare.



Deklariere:

In der objektorientierten Programmierung ist mit der Deklaration die

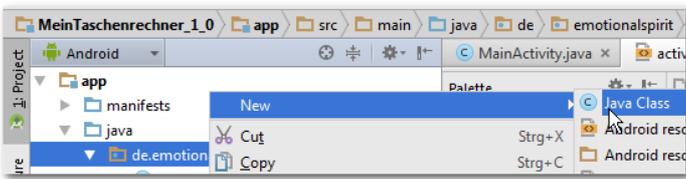
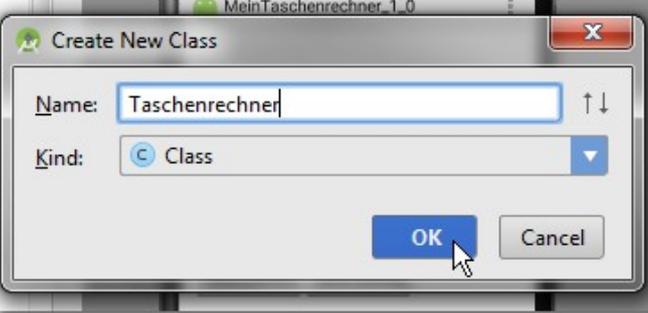
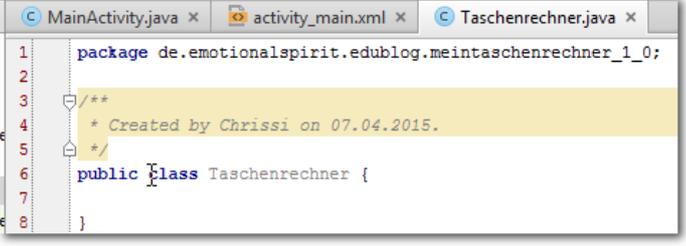
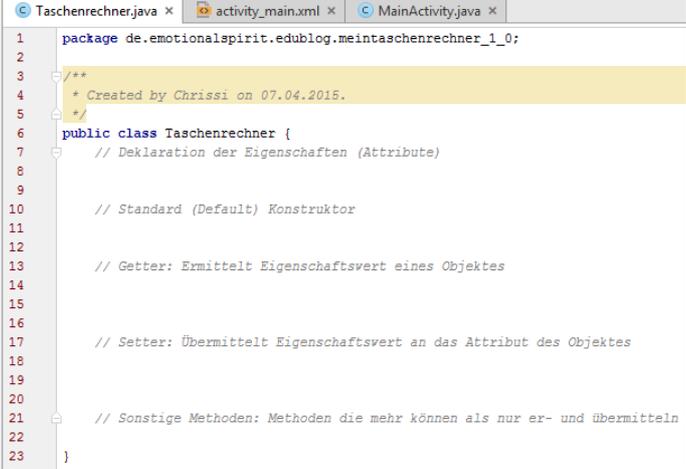
1. Festlegung einer Dimension, eines Bezeichners,
2. eines Datentyp und

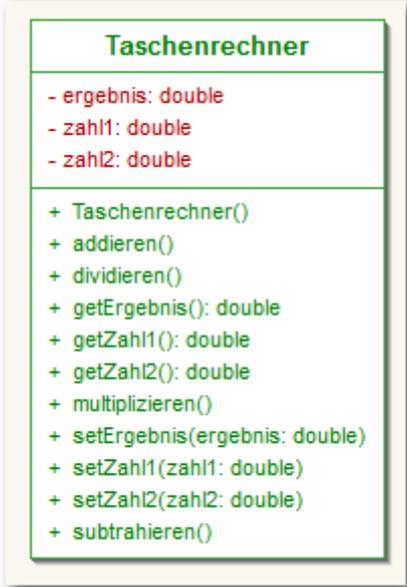
	<p>3. weiterer Aspekte einer Klasse, eines Konstruktors, einer Eigenschaft (Attribut) oder einer Verhaltensweise (Methode und Signatur), gemeint.</p> <p>Implementieren: In der objektorientierten Programmierung ist mit der Implementation die Einbettung bzw. Umsetzung konkreter Programmstrukturen gemeint. Die sogenannte Umsetzung vom „Business Logic“ (automatisierte Prozesse) in Programmcode (Quellcode) einer bestimmten Programmiersprache. Zumeist handelt es sich um das Anfüllen der Methoden mit dem benötigten Quellcode, also Inhalt einer Methode. Dabei dient der Quellcode dazu, die gewünschten Verhaltensweisen eines Systems (Programms) zu realisieren.</p>
 <p style="text-align: center;">UML-Klasse: <i>Bmirechner</i></p>	<p><i>Fachklasse implementieren.</i></p> <p>Wir implementieren die Fachklasse → Bmirechner, indem wir sie mit dem benötigten Quellcode ausstatten.</p> <p>Entsprechend den Vorgaben (Anforderungen) der nebenstehend angezeigten UML-Klasse, werden wir das in den kommenden Schritten tun.</p>
<pre> 7 // Deklaration der Eigenschaften (Attribute) 8 private double groesse; 9 private double gewicht; 10 private double bmi; </pre>	<p><i>Deklaration der Attribute.</i></p> <p><code>private double groesse;</code></p> <p>Der Zugriffsmodifikator → private stellt sicher, dass nur die Objekte der Klasse selbst auf die Eigenschaftswerte direkt zugreifen können.</p> <p>Der primitive Datentyp → double bestimmt den Wertebereich und das Zahlenformat für eine Gleitkommazahl mit doppelter Genauigkeit. Sobald in Java eine Gleitkommazahl (Darstellung einer reellen Zahl) verarbeitet werden soll grei-</p>

	<p>fen die meisten Programmierer zum Datentyp → double. Mit der Bestimmung des geeigneten Datentyps für ein Attribut wird gleichzeitig der maximal benötigte Speicherplatz vorab reserviert.</p> <p>→ grosse ist der Attributname. Attribute werden in Java kleingeschrieben und enthalten keine Umlaute und/oder Sonderzeichen.</p> <p>Hinweis: Leerzeichen sind auch Sonderzeichen!</p> <p>Deklarieren Sie auch die übrigen Attribute.</p>
<pre> 12 // Standard (Default) Konstruktor 13 public Bmirechner() { 14 15 }</pre>	<p><i>Deklaration des Konstruktors.</i></p> <p>Der Konstruktor einer Klasse sorgt dafür, dass beliebig viele Objekte der Klasse erzeugt „konstruiert“ werden können.</p> <p>Jeder Benutzer erzeugt damit sein eigenes Bmirechner-Objekt.</p> <p>Wir nutzen den Standard Konstruktor, ohne Parameter und ohne Initialisierung von Anfangswerten. Neu erzeugte Bmirechner-Objekte sind also am Anfang ihrer Entstehung „wertelos“.</p>
<p><i>Beispiel Attribut „bmi“:</i></p> <p>Get-Methode</p> <pre> public double getBmi() { return bmi; }</pre> <p>Set-Methode</p> <pre> public void setBmi(double pBmi) { this.bmi = pBmi; }</pre> <p>Kapselung: Ein Prinzip der Informatik bei dem der Zugriff auf Daten bewusst durch den Programmierer reguliert</p>	<p><i>Deklaration und Implementierung der Get- und Set-Methoden.</i></p> <p>Berücksichtigen Sie, dass wir auf die Eigenschaftswerte der Bmirechner-Objekte von außerhalb der Klasse (z.B. von der Benutzeroberfläche aus) zugreifen müssen. Jedes Attribut benötigt deshalb eine Get- und Set-Methode.</p> <p>Implementieren Sie außerdem nach dem gleichen Muster die Get- und Set-Methoden für die übrigen Attribute.</p> <p>Hinweis: Mit Sicherheit könnten wir die Architektur auch anderweitig gestalten z. B. könnten wir die Eigenschaften und Verhaltensweisen auf die Fach-</p>

<p>wird. In der OOP werden dazu sog. Zugriffsmodifikatoren (z.B. → private, → public, → protected) genutzt. Damit verhindert der Programmierer der Klasse, dass ein anderer Programmierer durch den Zugriff aus seiner Klasse unfreiwillige Manipulierungen der Daten durchführen kann. Mit den Get- und Set-Methoden kann der Programmierer den bewussten Zugriff auf Daten ermöglichen.</p>	<p>klasse Person und Rechner verteilen. Wir haben es uns in dem obigen Beispiel also sehr einfach gemacht, eine Benutzeroberfläche, eine Fachklasse.</p>
<pre>53 public void berechnen(){ 54 55 this.bmi = this.gewicht / (this.groesse * this.groesse); 56 }</pre>	<p><i>Deklaration und Implementierung sonstiger Methoden.</i></p> <p>Die Methode für die Berechnung des BMIs.</p> <p>BMI-Formel:</p> <p>BMI = gewicht/ (groesse*groesse)</p> <p>In dieser einfachen Variante wird es nur eine Funktionalität geben, nämlich die Möglichkeit den BMI zu berechnen. Die Erweiterung um eine Interpretationskomponente erfolgt zu einem späteren Zeitpunkt.</p> <p>Hinweis: Unsere Software wird nie optimal sein → Softwareentwicklungszyklus.</p>

3.2 Die Fachklasse des Taschenrechner 1.0

	<p><i>Neue Fachklasse erstellen.</i></p> <p>Klicken Sie im „app“-Verzeichnis mit der rechten Maustaste auf das Package und wählen Sie die Option New → Java Class.</p>
	<p><i>Klassenname festlegen.</i></p> <p>Geben Sie als Klassenname → Taschenrechner ein und klicken Sie auf die Schaltfläche → OK.</p> 
	<p><i>Grundgerüst einer Klasse festlegen.</i></p> <ol style="list-style-type: none"> 1. Deklaration der Attribute 2. Deklaration des Konstruktors 3. Get-Methoden (Getter) deklarieren und implementieren. 4. Set-Methode (Setter) deklarieren und implementieren. 5. Sonstige Methoden deklarieren und implementieren <p>Übernehmen Sie die nebenstehend angezeigten Kommentare.</p>
	<p>Deklarieren: In der objektorientierten Programmierung ist mit der Deklaration die</p> <ol style="list-style-type: none"> 1. Festlegung einer Dimension, eines Bezeichners, 2. eines Datentyp und 3. weiterer Aspekte einer Klasse, eines Konstruktors, einer Eigenschaft (Attribut) oder einer Verhaltensweise (Methode und Signatur),

	<p>gemeint.</p> <p>Implementieren: In der objektorientierten Programmierung ist mit der Implementation die Einbettung bzw. Umsetzung konkreter Programmstrukturen gemeint. Die sogenannte Umsetzung vom „Business Logic“ (automatisierte Prozesse) in Programmcode (Quellcode) einer bestimmten Programmiersprache. Zumeist handelt es sich um das Anfüllen der Methoden mit dem benötigten Quellcode, also Inhalt einer Methode. Dabei dient der Quellcode dazu, die gewünschten Verhaltensweisen eines Systems (Programms) zu realisieren.</p>
 <p>The diagram shows a class named 'Taschenrechner'. It has three private attributes: '- ergebnis: double', '- zahl1: double', and '- zahl2: double'. It has a constructor '+ Taschenrechner()' and several methods: '+ addieren()', '+ dividieren()', '+ getErgebnis(): double', '+ getZahl1(): double', '+ getZahl2(): double', '+ multiplizieren()', '+ setErgebnis(ergebnis: double)', '+ setZahl1(zahl1: double)', '+ setZahl2(zahl2: double)', and '+ subtrahieren()'. Brackets on the right side of the diagram group these elements into three categories: 'Klasse' (the class name), 'Attribute' (the private attributes), and 'Konstruktor & Methoden' (the constructor and methods).</p> <p>UML-Klasse: <i>Taschenrechner</i></p>	<p><i>Fachklasse implementieren.</i></p> <p>Wir implementieren die Fachklasse → Taschenrechner, indem wir sie mit dem benötigten Quellcode ausstatten.</p> <p>Entsprechend den Vorgaben (Anforderungen) der nebenstehend angezeigten UML-Klasse, werden wir das in den kommenden Schritten tun.</p>
<pre> 7 // Deklaration der Eigenschaften (Attribute) 8 private double zahl1; 9 private double zahl2; 10 private double ergebnis; 11 </pre>	<p><i>Deklaration der Attribute.</i></p> <p>private double zahl1;</p> <p>Der Zugriffsmodifikator → private stellt sicher, dass nur die Objekte der Klasse selbst auf die Eigenschaftswerte direkt zugreifen können.</p> <p>Der primitive Datentyp → double bestimmt den Wertebereich und das Zahlenformat für eine Gleitkommazahl mit doppelter Genauigkeit. Sobald in Java eine Gleitkommazahl (Darstellung einer reellen Zahl) verarbeitet werden soll greifen die meisten Programmierer zum Datentyp → double. Mit der Bestimmung des geeigneten</p>

	<p>Datentyps für ein Attribut wird gleichzeitig der maximal benötigte Speicherplatz vorab reserviert.</p> <p>→ zahl1 ist der Attributname. Attribute werden in Java kleingeschrieben und enthalten keine Umlaute und/oder Sonderzeichen.</p> <p>Hinweis: Leerzeichen sind auch Sonderzeichen!</p> <p>Deklarieren Sie auch die übrigen Attribute.</p>
<pre> 12 // Standard (Default) Konstruktor 13 public Taschenrechner() { 14 15 } </pre>	<p><i>Deklaration des Konstruktors.</i></p> <p>Der Konstruktor einer Klasse sorgt dafür, dass beliebig viele Objekte der Klasse erzeugt „konstruiert“ werden können.</p> <p>Jeder Benutzer erzeugt damit sein eigenes Taschenrechner-Objekt.</p> <p>Wir nutzen den Standard Konstruktor, ohne Parameter und ohne Initialisierung von Anfangswerten. Neu erzeugte Taschenrechner-Objekte sind also am Anfang ihrer Entstehung „wertelos“.</p>
<p><i>Beispiel Attribut „ergebnis“:</i></p> <p>Get-Methode (Ermittlung)</p> <pre> public double getErgebnis() { return ergebnis; } </pre> <p>Set-Methode (Übermittlung)</p> <pre> public void setErgebnis(double ergebnis) { this.ergebnis = ergebnis; } </pre>	<p><i>Deklaration und Implementierung der Get- und Set-Methoden.</i></p> <p>Berücksichtigen Sie, dass wir auf die Eigenschaftswerte der Taschenrechner-Objekte von außerhalb der Klasse (z.B. von der Benutzeroberfläche aus) zugreifen müssen. Jedes Attribut benötigt deshalb eine Get- und Set-Methode.</p> <p>Implementieren Sie außerdem nach dem gleichen Muster die Get- und Set-Methoden für die übrigen Attribute.</p> <p>Hinweis: Mit Sicherheit könnten wir die Architektur auch anderweitig gestalten z. B. könnten wir die Eigenschaften und Verhaltensweisen auf die Fachklasse Rechner, Rechenoperation und die erbbenden Klassen Addition, Subtraktion, Multiplikation und Division verteilen. Wir haben es uns in dem obigen Beispiel also sehr einfach gemacht, eine Benutzero-</p>

Kapselung:

Ein Prinzip der Informatik bei dem der Zugriff auf Daten bewusst durch den Programmierer reguliert wird.

In der OOP werden dazu sog. Zugriffsmodifikatoren (z.B. → private, → public, → protected) genutzt. Damit verhindert der Programmierer der Klasse, dass ein anderer Programmierer durch den Zugriff aus seiner Klasse unfreiwillige Manipulierungen der Daten durchführen kann.

Mit den Get- und Set-Methoden kann der Programmierer den bewussten Zugriff auf Daten ermöglichen.

berfläche, eine Fachklasse.

```
50
51 // Sonstige Methoden: Methoden die mehr können als nur er- und übermitteln
52 public void addieren() {
53
54     this.ergebnis = this.zahl1 + this.zahl2;
55 }
56
```

Deklaration und Implementierung sonstiger Methoden.

Die Methode für die Addition.

Formel:

Ergebnis = zahl1 + zahl1

In dieser einfachen Variante werden nur vier Funktionalitäten implementiert, nämlich die Möglichkeit zu addieren, subtrahieren, multiplizieren und dividieren.

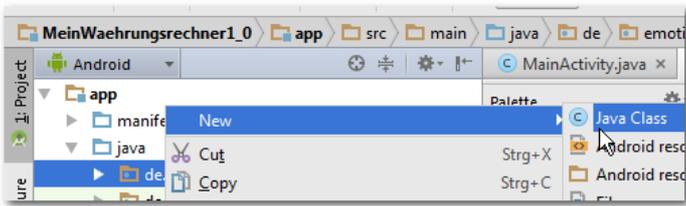
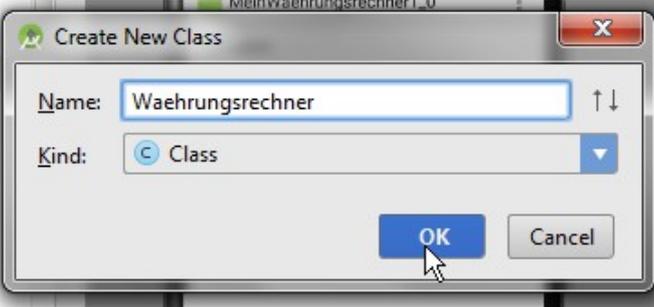
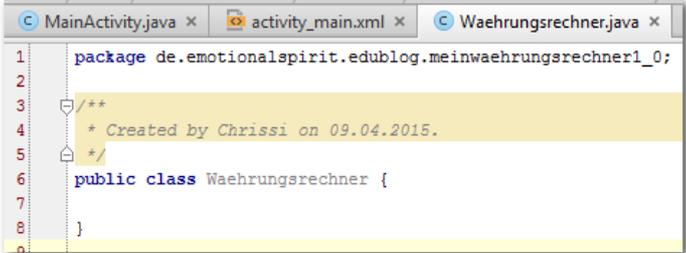
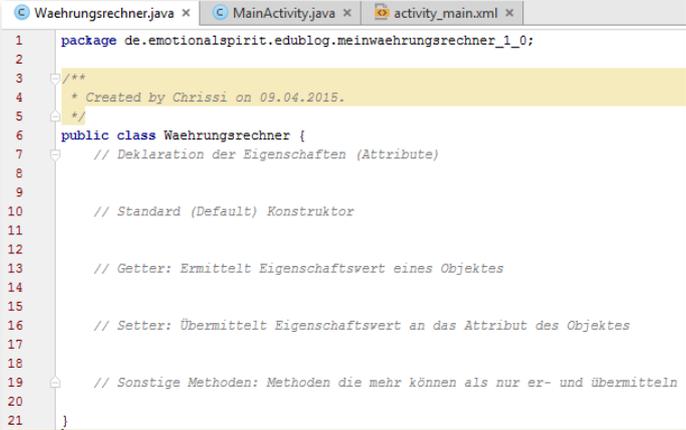
Implementieren Sie auch die übrigen Grundrechenarten.

Die Erweiterung um zwei weitere Rechenoperationen erfolgt zu einem späteren Zeitpunkt.

Hinweis:

Unsere Software wird nie optimal sein → Softwareentwicklungszyklus.

3.3 Die Fachklasse des Währungsrechner 1.0

	<p><i>Neue Fachklasse erstellen.</i></p> <p>Klicken Sie im „app“-Verzeichnis mit der rechten Maustaste auf das Package und wählen Sie die Option New → Java Class.</p>
	<p><i>Klassenname festlegen.</i></p> <p>Geben Sie als Klassennamen → Taschenrechner ein und klicken Sie auf die Schaltfläche → OK.</p> 
	<p><i>Grundgerüst einer Klasse festlegen.</i></p> <ol style="list-style-type: none"> 1. Deklaration der Attribute 2. Deklaration des Konstruktors 3. Get-Methoden (Getter) deklarieren und implementieren. 4. Set-Methode (Setter) deklarieren und implementieren. 5. Sonstige Methoden deklarieren und implementieren <p>Übernehmen Sie die nebenstehend angezeigten Kommentare.</p> <p>Deklarieren: In der objektorientierten Programmierung ist mit der Deklaration die</p> <ol style="list-style-type: none"> 1. Festlegung einer Dimension, eines Bezeichners, 2. eines Datentyp und

	<p>3. weiterer Aspekte einer Klasse, eines Konstruktors, einer Eigenschaft (Attribut) oder einer Verhaltensweise (Methode und Signatur), gemeint.</p> <p>Implementieren: In der objektorientierten Programmierung ist mit der Implementation die Einbettung bzw. Umsetzung konkreter Programmstrukturen gemeint. Die sogenannte Umsetzung vom „Business Logic“ (automatisierte Prozesse) in Programmcode (Quellcode) einer bestimmten Programmiersprache. Zumeist handelt es sich um das Anfüllen der Methoden mit dem benötigten Quellcode, also Inhalt einer Methode. Dabei dient der Quellcode dazu, die gewünschten Verhaltensweisen eines Systems (Programms) zu realisieren.</p>
<div data-bbox="114 900 571 1384"> <pre> classDiagram class Waehrungsrechner { - betrag: double - ergebnis: double - in: String - von: String - wechsellkurs: double + Waehrungsrechner() + getBetrag(): double + getErgebnis(): double + getIn(): String + getVon(): String + getWechsellkurs(): double + setBetrag(pBetrag: double) + setErgebnis(pErgebnis: double) + setIn(pIn: String) + setVon(pVon: String) + setWechsellkurs(pWechsellkurs: double) + umrechnen(pWaehrung1: String, pWaehrung2: String) } </pre> </div> <p data-bbox="619 922 778 958">Klasse</p> <p data-bbox="619 1034 778 1070">Attribute</p> <p data-bbox="619 1191 794 1258">Konstruktor & Methoden</p> <p data-bbox="82 1438 561 1473">UML-Klasse: <i>Waehrungsrechner</i></p>	<p><i>Fachklasse implementieren.</i></p> <p>Wir implementieren die Fachklasse → Waehrungsrechner, indem wir sie mit dem benötigten Quellcode ausstatten.</p> <p>Entsprechend den Vorgaben (Anforderungen) der nebenstehend angezeigten UML-Klasse, werden wir das in den kommenden Schritten tun.</p>

<pre> 15 // Deklaration der Eigenschaften (Attribute) 16 private double betrag; 17 private String von; 18 private String in; 19 private double ergebnis; 20 private double wechselkurs; </pre>	<p>Deklaration der Attribute.</p> <p>private double betrag;</p> <p>Der Zugriffsmodifikator → private stellt sicher, dass nur die Objekte der Klasse selbst auf die Eigenschaftswerte direkt zugreifen können.</p> <p>Der primitive Datentyp → double bestimmt den Wertebereich und das Zahlenformat für eine Gleitkommazahl mit doppelter Genauigkeit. Sobald in Java eine Gleitkommazahl (Darstellung einer reellen Zahl) verarbeitet werden soll greifen die meisten Programmierer zum Datentyp → double. Mit der Bestimmung des geeigneten Datentyps für ein Attribut wird gleichzeitig der maximal benötigte Speicherplatz vorab reserviert.</p> <p>→ betrag ist der Attributname. Attribute werden in Java kleingeschrieben und enthalten keine Umlaute und/oder Sonderzeichen.</p> <p>Hinweis: Leerzeichen sind auch Sonderzeichen!</p> <p>Deklarieren Sie auch die übrigen Attribute.</p>
<pre> 15 // Standard (Default) Konstruktor 16 public Waehrungsrechner() { 17 18 } </pre>	<p>Deklaration des Konstruktors.</p> <p>Der Konstruktor einer Klasse sorgt dafür, dass beliebig viele Objekte der Klasse erzeugt „konstruiert“ werden können.</p> <p>Jeder Benutzer erzeugt damit sein eigenes Waehrungsrechner-Objekt.</p> <p>Wir nutzen den Standard Konstruktor, ohne Parameter und ohne Initialisierung von Anfangswerten. Neu erzeugte Waehrungsrechner-Objekte sind also am Anfang ihrer Entstehung „wertelos“.</p>
<p>Beispiel Attribut „ergebnis“:</p> <p>Get-Methode (Ermittlung)</p>	<p>Deklaration und Implementierung der Get- und Set-Methoden.</p> <p>Berücksichtigen Sie, dass wir auf die Eigenschaftswerte der Waehrungsrechner-Objekte von außerhalb der Klasse (z.B. von der Benut-</p>

```
public double getErgebnis() {
    return ergebnis;
}
```

Set-Methode (Übermittlung)

```
public void setErgebnis(double ergebnis) {
    this.ergebnis = ergebnis;
}
```

Kapselung:

Ein Prinzip der Informatik bei dem der Zugriff auf Daten bewusst durch den Programmierer reguliert wird.

In der OOP werden dazu sog. Zugriffsmodifikatoren (z.B. → private, → public, → protected) genutzt. Damit verhindert der Programmierer der Klasse, dass ein anderer Programmierer durch den Zugriff aus seiner Klasse unfreiwillige Manipulierungen der Daten durchführen kann.

Mit den Get- und Set-Methoden kann der Programmierer den bewussten Zugriff auf Daten ermöglichen.

Die ELSE IF-Kontrollstruktur :
 Eine Fallunterscheidung mit mehreren Bedingungen.

```
if(Bedingung){
    Anweisung01;
    Anweisung02;
}else if(Bedingung){
    Anweisung03;
    Anweisung04;
}else if(Bedingung){
    Anweisung05;
    Anweisung06;
}else if(Bedingung){
    Anweisung07;
    Anweisung08;
}else{
    Anweisung09;
    Anweisung10;
}
```

zeroberfläche aus) zugreifen müssen. Jedes Attribut benötigt deshalb eine Get- und Set-Methode.

Implementieren Sie außerdem nach dem gleichen Muster die Get- und Set-Methoden für die übrigen Attribute.

Hinweis:

Mit Sicherheit könnten wir die Architektur auch anderweitig gestalten z. B. könnten wir die Eigenschaften und Verhaltensweisen auf die Fachklasse Rechner, Waehrung und Kurs verteilen. Wir haben es uns in dem obigen Beispiel also sehr einfach gemacht, eine Benutzeroberfläche, eine Fachklasse.

Deklaration und Implementierung sonstiger Methoden.

Methode → umrechnen(von, in)
 Abhängig von der Wahl des Benutzers (von, in) soll der Wechselkurs bestimmt werden.

Für die vier Währungen:

1. Euro (EUR)
2. Britisches Pfund (GBP)
3. US Dollar (USD)
4. Japanischer Yen (JPY)

ergeben sich 16 Kombinationen die der Benutzer auswählen kann. All diese Fälle müssen wir berücksichtigen:

von	Euro (EUR)
in	Euro (EUR)



Die Methode umrechnen(String pVon, String pIn) nutzt die Kontrollstruktur zur Behandlung der 16 möglichen Fälle:

```

63 // Sonstige Methode: Methoden die mehr können als nur er- und übermitteln
64 public void umrechnen(String pWahrung1, String pWahrung2){
65     if(pWahrung1.equals("Euro (EUR)") &&
66        pWahrung2.equals("Euro (EUR)")){
67         //Euro (EUR) - Euro (EUR)
68         this.wechselkurs = 1.00000;
69         this.setVon(pWahrung1);
70         this.setIn(pWahrung2);
71     }else if(pWahrung1.equals("Euro (EUR)") &&
72            pWahrung2.equals("Britisches Pfund (GBP)")){
73         //Euro (EUR) - Britisches Pfund (GBP)
74         this.wechselkurs = 0.72085;
75         this.setVon(pWahrung1);
76         this.setIn(pWahrung2);
77     }else if(pWahrung1.equals("Britisches Pfund (GBP)") &&
78            pWahrung2.equals("Britisches Pfund (GBP)")){
79         //Britisches Pfund (GBP) - Britisches Pfund (GBP)
80         this.wechselkurs = 1.00000;
81         this.setVon(pWahrung1);
82         this.setIn(pWahrung2);

```

wechselkurs	1.00000
-------------	---------

von	Euro (EUR)
in	Britisches Pfund (GBP)
wechselkurs	0.72085

von	Britisches Pfund (GBP)
in	Britisches Pfund (GBP)
wechselkurs	1.00000

von	Euro (EUR)
in	US Dollar (USD)
wechselkurs	1.05987

von	US Dollar (USD)
in	US Dollar (USD)
wechselkurs	1.00000

von	Euro (EUR)
in	Japanischer Yen (JPY)
wechselkurs	126.86000

von	Japanischer Yen (JPY)
in	Japanischer Yen (JPY)
wechselkurs	1.00000

von	Britisches Pfund (GBP)
in	Euro (EUR)
wechselkurs	1.38690

...

```

155     }else if(pWahrung1.equals("Japanischer Yen (JPY)") &&
156            pWahrung2.equals("US Dollar (USD)")){
157         //Japanischer Yen (JPY) - US Dollar (USD)
158         this.wechselkurs = 0.00835;
159         this.setVon(pWahrung1);
160         this.setIn(pWahrung2);
161     }else{
162         this.wechselkurs = 0.00000;
163     }
164     this.ergebnis = Math.round((this.betrag * this.wechselkurs)*100) / 100;
165 }

```

Abschließend soll dann der Betrag in die Zielwährung umgerechnet werden:

```
zielwährung = ausgangswährung * wechselkurs;
```

Hier sind viele Lösungsansätze möglich!

Eine relativ einfache Lösungsvariante mittels einer ELSE IF-Kontrollstruktur wird hier vorgestellt.



von	Britisches Pfund (GBP)
in	US Dollar (USD)
wechselkurs	1.47011
von	Britisches Pfund (GBP)
in	Japanischer Yen (JPY)
wechselkurs	175.96000
von	US Dollar (USD)
in	Euro (EUR)
wechselkurs	0.94337
von	US Dollar (USD)
in	Britisches Pfund (GBP)
wechselkurs	0.68012
von	US Dollar (USD)
in	Japanischer Yen (JPY)
wechselkurs	119.69000
von	Japanischer Yen (JPY)
in	Euro (EUR)
wechselkurs	0.00788
von	Japanischer Yen (JPY)
in	Britisches Pfund (GBP)
wechselkurs	0.00568
von	Japanischer Yen (JPY)
in	US Dollar (USD)
wechselkurs	0.00835

kurs

Exkurs

Kontrollstrukturen

Fallunterscheidungen

Wiederholstrukturen
Schleifen (Loops)

IF ELSE

ELSE IF

TRY CATCH

SWITCH CASE

For-Schleife

While-Schleife

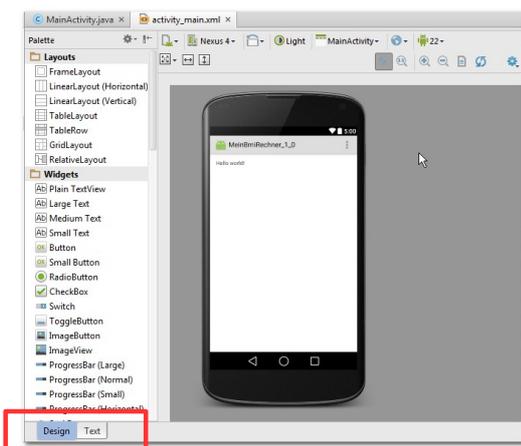
Do-While-Schleife

4 View: Layouts, Komponenten & XML

4.1 Die Benutzeroberfläche des BMI-Rechner 1.0



Wir werden nun die Benutzeroberfläche für unseren BMI-Rechner erzeugen.



Benutzeroberflächen erzeugen.

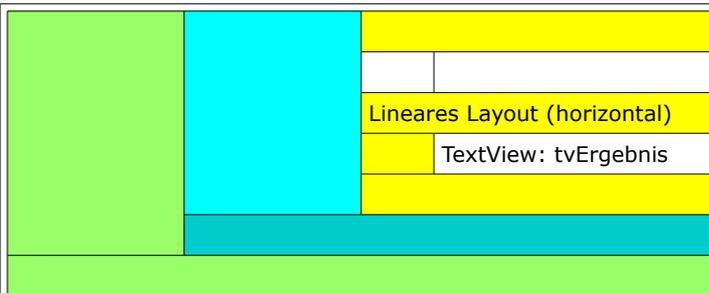
Das Android Studio bietet hier zwei mögliche Varianten an:

1. Den Designer:
Ein Oberflächen-Generator, ähnlich dem Swing-Designer in Eclipse. Es gibt jedoch wesentliche Unterschiede und in der Praxis werden wir sehen, dass wir ohne grundlegende XML-Kenntnisse nicht auskommen werden.
2. Den Text-Editor
Ein XML-Editor für die Beschreibung der Benutzeroberflächen in XML-Quellcode.

Arbeitsweise:

Die beiden Varianten sind unabdingbar miteinander

	<p>verbunden. Das heißt konkret, dass Veränderungen die im Text-Editor vorgenommen wurden im Design ersichtlich werden, umgekehrt erzielt man denselben Effekt. Der Wechsel zwischen den Varianten erfolgt über die zwei Reiter „Design“ und „Text“ unterhalb der Komponenten-Palette.</p>
<div data-bbox="172 501 699 770" data-label="Image"> </div> <p data-bbox="140 819 734 857">onCreate-Methode im HalloWelt-Projekt</p> <div data-bbox="92 869 778 1128" data-label="Code-Block"> <pre> @Override protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity_main); Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar); setSupportActionBar(toolbar); FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab); fab.setOnClickListener(new View.OnClickListener() { @Override public void onClick(View view) { Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG) .setAction("Action", null).show(); } }); } </pre> </div>	<p data-bbox="804 450 1289 488">Aktueller Hinweis: Blank Activity</p> <p data-bbox="804 521 1241 591">In der onCreate-Methode der → MainActivity.java</p> <p data-bbox="804 629 1516 763">wird wie üblich die View → activity_main gesetzt. Neu ist, dass die → activity_main eine weitere xml-Datei, die → content_main, inkludiert.</p> <p data-bbox="804 801 1516 1016">Ähnlich wie beim dynamischen Weblayout bleibt das Grundgerüst (Menüs) gleich nur der Inhalt (Content) und die Ereignissteuerung variieren. Das Start-Layout (GUI) für die Anwendung sollte also in die content_main. (siehe Projekt HalloWelt).</p> <p data-bbox="804 1055 1516 1189">Sie können auf diese Architektur verzichten, wenn Sie bei Erzeugung eines neuen Projektes anstelle dessen eine → Empty Activity verwenden.</p>
<div data-bbox="86 1211 788 1939" data-label="Diagram"> </div>	<p data-bbox="804 1211 1324 1249"><i>Vorgehensweise: Component Tree.</i></p> <ol data-bbox="852 1283 1468 1417" style="list-style-type: none"> 1. AppTheme „Holo Light“ wählen 2. Layoutschachtelung erzeugen 3. Komponenten im Layout platzieren 4. Komponenteneigenschaften definieren <p data-bbox="804 1458 1516 1682">Das Relative Layout: Die in einem relativen Layout enthaltenen Komponenten werden immer in Abhängigkeit seiner direkt benachbarten Komponenten betrachtet. Deshalb erfolgt die Beschreibung der Platzierung auch in Abhängigkeit der direkt benachbarten Komponenten.</p> <p data-bbox="804 1720 1516 1944">Praxis-Hinweis: Das ist für die Darstellung von Benutzeroberflächen auf unterschiedlichen Displaygrößen sehr sinnvoll. Während der Entwicklung verändern wir die Platzierung jedoch ständig. Bei jeder kleinen Änderung müssten wir die gesamte Benutzeroberfläche überarbeiten. Die Lösung des Problems ist die</p>



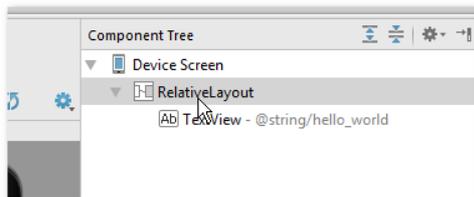
Schachtelung von Layouts.

Das Lineare Layout (vertikal):

Die in einem vertikalen Linearen Layout platzierten Komponenten werden untereinander angeordnet.

Das Lineare Layout (horizontal):

Die in einem horizontalen Linearen Layout platzierten Komponenten werden nebeneinander angeordnet.



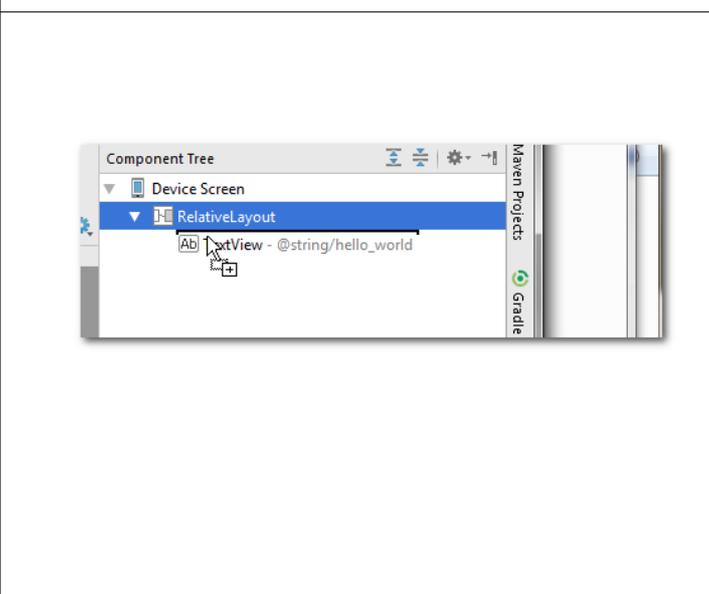
Der Komponenten-Baum.

Im oberen, rechten Frame-Fenster wird der Komponenten-Baum (Component Tree) angezeigt.

Als Komponenten werden alle Elemente einer Benutzeroberfläche bezeichnet.

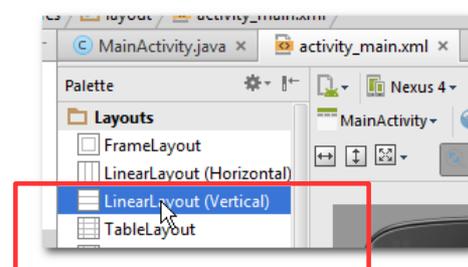
Die Grundlage jeder Benutzeroberfläche sind die Layouts. Diese werden, wie wir gleich lernen, geschachtelt, um die gewünschten Ergebnisse zu erzielen.

Das Standard-Layout ist das „Relative Layout“. In der Vergangenheit wurden häufig absolute Layouts eingesetzt, diese haben sich auf mobilen Endgeräten aufgrund der unterschiedlichen Displaygrößen als sehr unflexibel erwiesen.

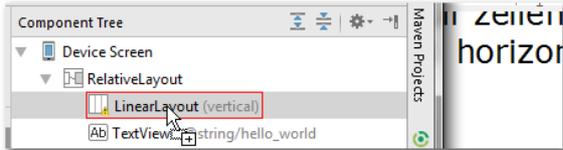
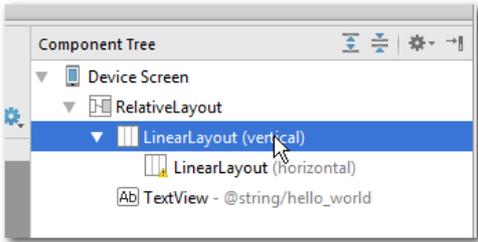
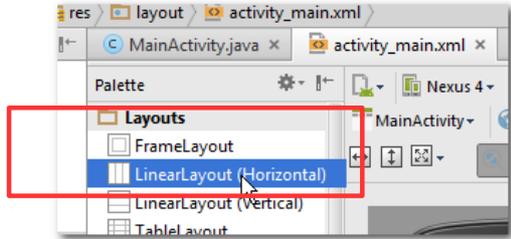
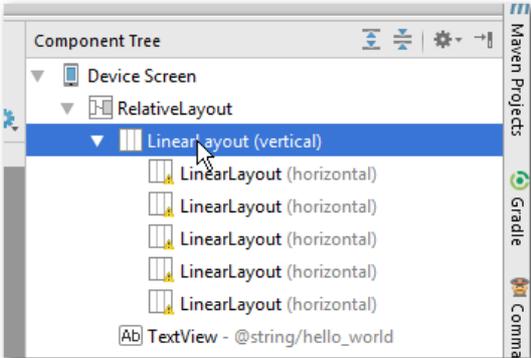


LinearesLayout (Vertical) verwenden.

Klicken Sie dazu im linken Frame-Fenster „Palette“ neben der Design-Bühne auf die Option „LinearLayout (Vertical)“.



Ziehen Sie dazu diese Komponente mit ge-

	<p>drückter linker Maustaste in das rechte, obere Frame-Fenster „Component Tree“, wie nebenstehend angezeigt.</p> <p>Fügen Sie diese Komponente unterhalb des „Relativen Layouts“ ein.</p>
 <p>Sollte danach so angezeigt werden:</p> 	<p><i>Lineares Layout (Horizontal) verwenden.</i></p> <p>Die Anordnung für die Komponenten innerhalb des vertikalen Linearen Layouts soll zeilenweise erfolgen, dazu nutzen wir das horizontale Lineare Layout:</p>  <p>Ziehen Sie dazu diese Komponente mit gedrückter linker Maustaste in das rechte, obere Frame-Fenster „Component Tree“, wie nebenstehend angezeigt.</p> <p>Fügen Sie diese Komponente in das „Lineare Layout (vertikal)“ ein.</p>
	<p><i>Schachtelung umsetzen.</i></p> <p>Wie wir eingangs im Überblick (Vorgehensweise) sehen, werden die Komponenten innerhalb der horizontalen LinearLayouts platziert und mit den benötigten Eigenschaften versehen.</p> <p>Innerhalb des vertikalen Linearen Layouts platzieren wir dazu nun, auf die gleiche Weise, vier weitere horizontale Lineare Layouts für die benötigten Komponenten:</p>

1. das Logo

Komponente	Präfix	@id/
ImageView	iv	ivLogo

2. die Einabe des Gewichts

Komponente	Präfix	@id/
TextView	tv	tvGewicht
EditText	et	etGewicht

3. die Eingabe der Größe

Komponente	Präfix	@id/
TextView	tv	tvGroesse
EditText	et	etGroesse

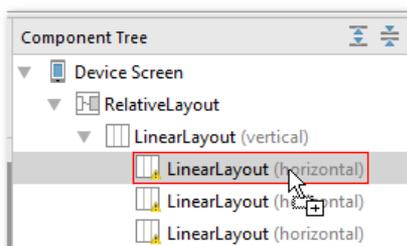
4. die Schaltfläche Berechnen

Komponente	Präfix	@id/
Button	bt	btBerechnen

5. die Anzeige des Ergebnisses (BMI)

Komponente	Präfix	@id/
TextView	tv	tvErgebnis

Realisieren Sie dazu die Layout-Schachtelung, wie nebenstehend angezeigt.

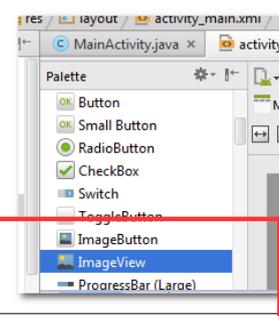


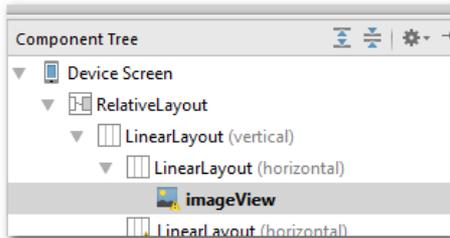
Sollte danach so angezeigt werden:

Platz für das Logo schaffen.

Um zu einem späteren Zeitpunkt ein Logo angezeigt zu bekommen, fügen wir die ImageView Komponente ein.

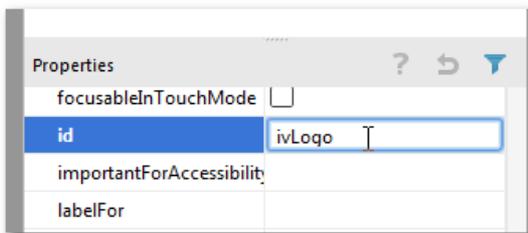
Wählen Sie dazu die ImageView Komponente im linken Frame-Fenster → Palette aus:





Ziehen Sie dazu diese Komponente mit gedrückter linker Maustaste in das rechte, obere Frame-Fenster → Component Tree, wie nebenstehend angezeigt.

Fügen Sie dazu diese Komponente in das → Lineare Layout (horizontale) ein.



Die Eigenschaft „id“ für das Logo definieren.

Im rechten, unteren Frame-Fenster unterhalb des → Component Tree werden die Eigenschaften (Properties) der Aktuell angeklickten Komponente angezeigt. Um die Eigenschaften für die gerade eingefügte ImageView-Komponente zu verändern müssen Sie diese im „Component Tree“ anklicken.

Nutzen Sie dann die vertikale Bildlaufleiste im Fenster → Properties, um die Eigenschaft für die → id wie nebenstehend angezeigt ändern zu können.

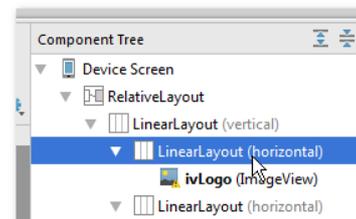
Die Anpassung der restlichen Eigenschaften werden wir zu einem späteren Zeitpunkt durchführen.



Anzeige danach:

Regulierung des Linearen Layouts (horizontal).

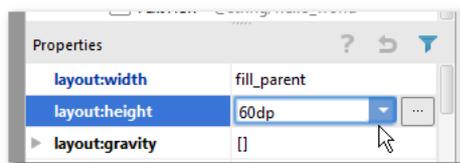
Im Moment nimmt das erste horizontale Lineare Layout den ganzen Platz auf dem Display ein. Klicken Sie das erste → Lineare Layout (horizontal) an, um es zu verkleinern:



Ziehen Sie dazu mit gedrückter linker Maustaste den zuerst blauen Rahmen am unteren Rand nach oben bis auf → 50dp.



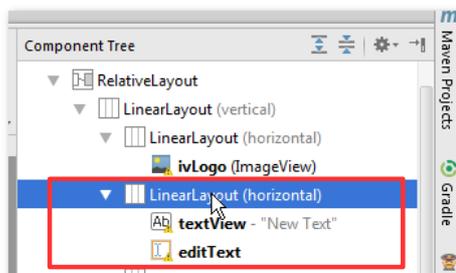
Eigenschaft „Layout:height“



Density-independent pixel (dp):
Eine virtuelle Pixel-Maßeinheit (optisch unabhängige Dichte). Wird genutzt, um die Größenangaben für Layouts zu definieren. Im übrigen ist es aufgrund der Anpassungsfähigkeit in vielen Fällen besser auf „statische Größenangaben“ gänzlich zu verzichten.

Regulieren Sie anschließend auf gleiche Weise die restlichen vier horizontalen Linearen Layouts auf eine Höhe von jeweils → 60dp.

Alternativ können Sie die Änderung auch, wie nebenstehend angezeigt, im Eigenschaftsfenster → Properties an entsprechender Stelle vornehmen.

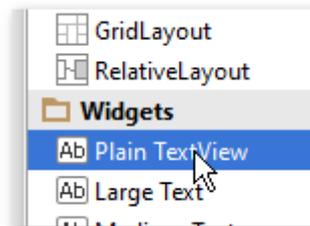


Ergebnis:

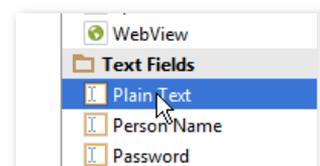


Die Eingabe des Gewichts.

Hier sind zwei Komponenten nötig. Eine → TextView-Komponente (Plain TextView, Label):

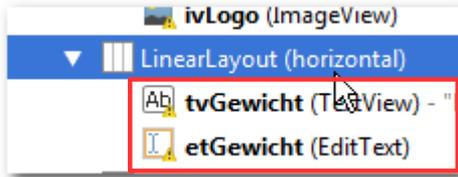


Und eine EditText-Komponente (Plain Text, Texteingabefeld):



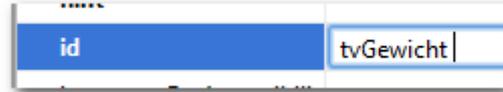
Ziehen Sie dazu diese beiden Komponenten mit gedrückter linker Maustaste in das rechte, obere Frame-Fenster „Component Tree“, wie nebenstehend angezeigt.

Fügen Sie diese Komponenten in das „Lineare Layout (horizontale)“ ein.

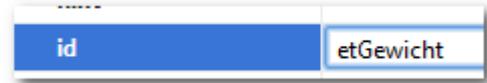


Definieren Sie die Eigenschaften „id“ für die TextView und EditText Komponente (Eingabe des Gewichts).

TextView (Lable):



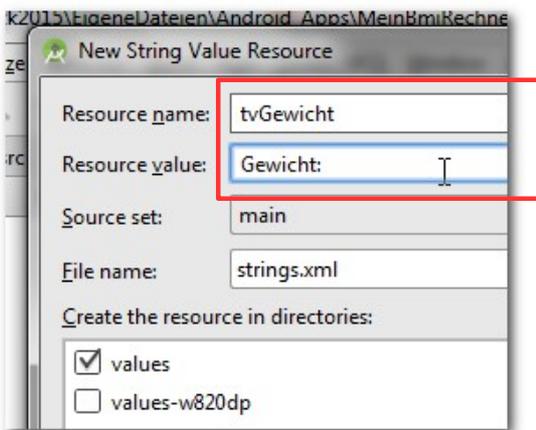
EditText (Texteingabefeld):



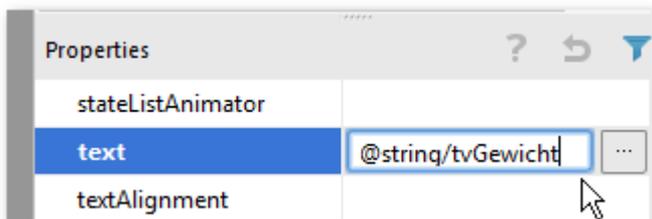
Bezeichnungen von Komponenten.

Klicken Sie auf die TextView-Komponente um die Bezeichnung „Gewicht:“ als → String-Resource zu definieren.

Wählen Sie dann im rechten, unteren Frame-Fenster → Properties die Eigenschaft → Text aus:



Danach:



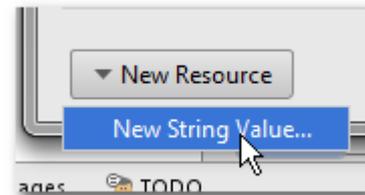
Davor:



Klicken Sie dann auf die → ...-Schaltfläche am rechten Rand.

Klicken Sie nun im Fenster → Resource auf die

Schaltfläche → New Resource:



Verändern Sie die Angaben im Fenster → New String Value Resource, wie nebenstehend angezeigt.

Bestätigen sie die Eingabe mit einem Klick auf die Schaltfläche → OK



Hinweis:

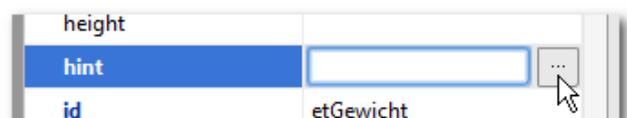
Aus gutem Grund werden alle Bezeichner der Benutzeroberfläche in eine String-Resource (res/values/strings.xml) ausgelagert. Für den Fall, dass Apps in anderen Sprachen verfügbar gemacht werden sollen. Findet der Übersetzer alle benötigten Begriffe in genau einer Datei.

Ein Hint (Hinweis) als Bezeichner.

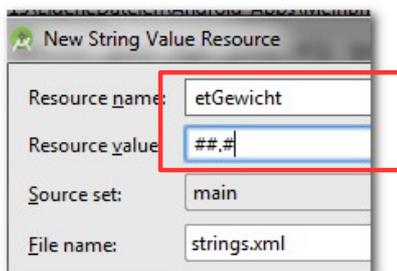
Für die Komponente EditText benötigen wir eine hinweisende Bezeichnung, die für den Benutzer eine Eingabehilfe darstellt.

Definieren Sie für die Eigenschaft → hint im rechten, unteren Frame-Fenster Properties eine weitere String-Resource.

Davor:

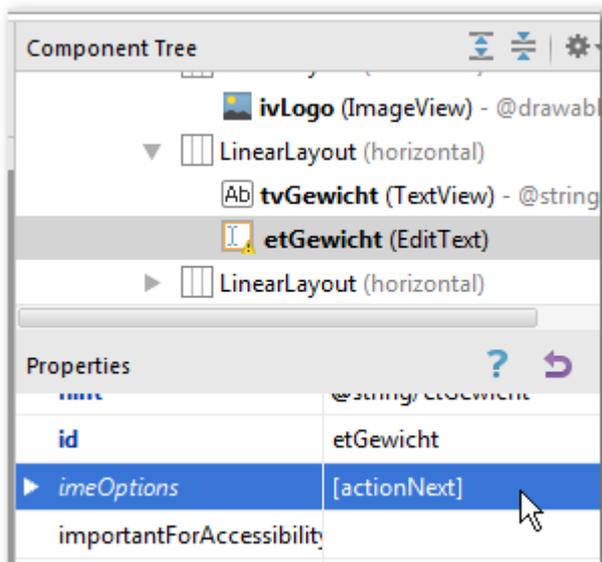
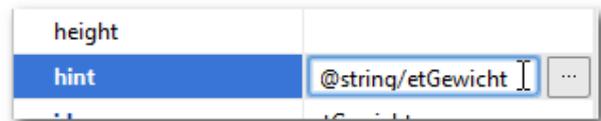


Ergebnis:





Danach:

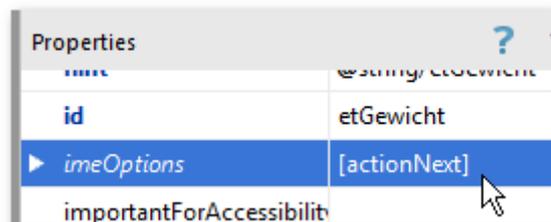


Navigation (imeOptions) über die Tastatur für etGewicht.

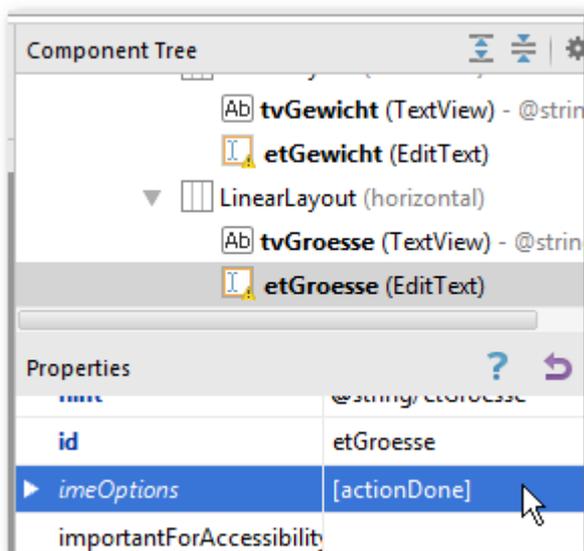
Für das Feld etGewicht:



In den Eigenschaften:



Mit einem Klick springt der Cursor in das nächste Texteingabefeld.

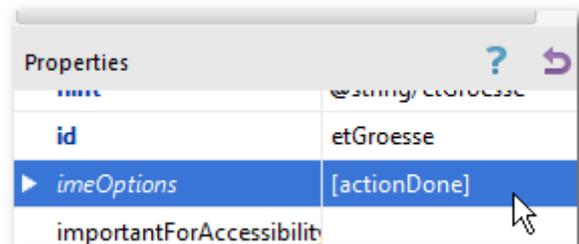


Navigation (imeOptions) über die Tastatur für etGroesse.

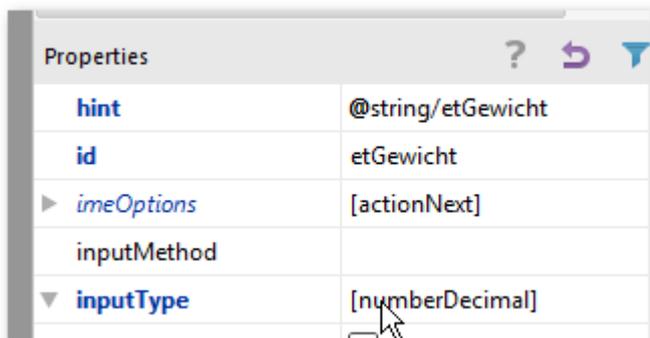
Für das Feld etGroesse:



In den Eigenschaften:



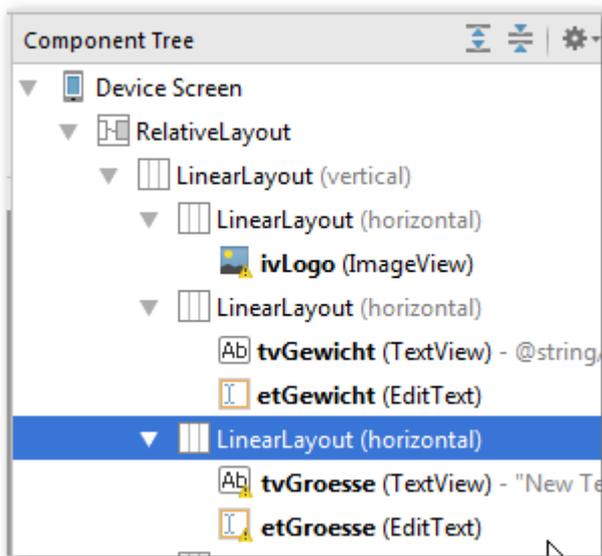
Mit einem Klick wird die Tastatur ausgeblendet!



Numerische Felder definieren.

Die EditView-Komponenten → etGewicht und → etGroesse sollen numerische dezimal Werte sein.

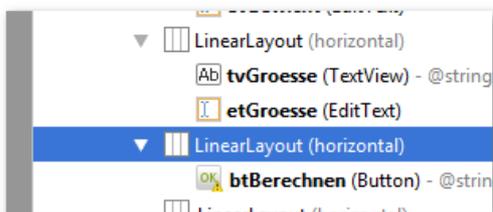
Aktivieren Sie für die Einstellung im rechten, unteren Frame-Fenster → Properties die Eigenschaft → inputType → numberDecimal. Damit stellen Sie gleichzeitig sicher, dass der Nutzer bei der Eingabe die numerische Tastatur angezeigt bekommt.



Eingabe der Größe.

Definieren Sie, wie zuvor für das Gewicht, die Komponenten und deren Eigenschaften. Achten Sie auf die entsprechenden Präfixe für die Komponenten.

Ergebnis:

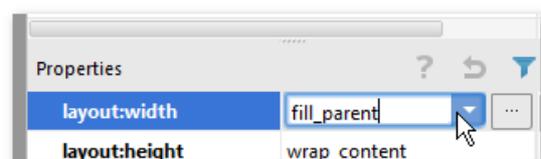


Die Schaltfläche Berechnen.

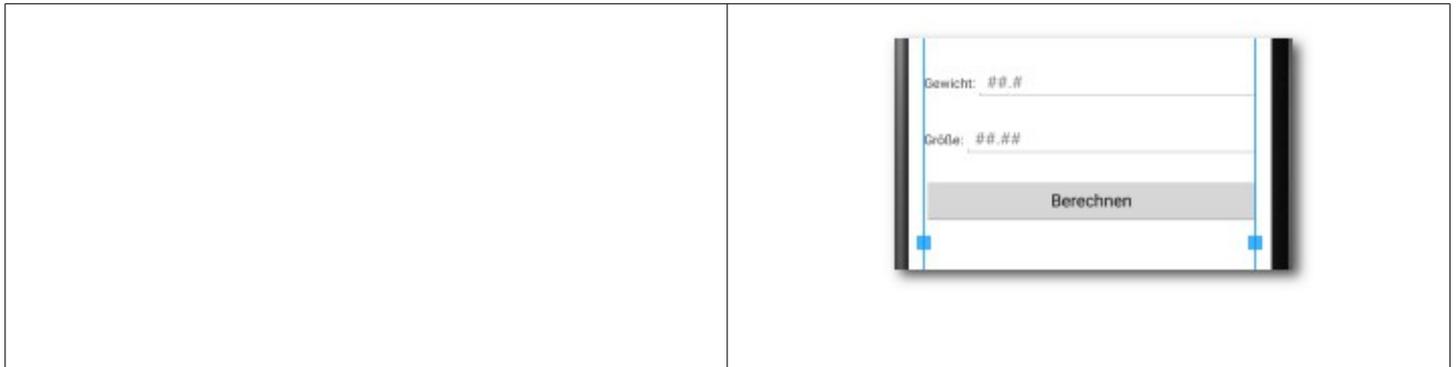
Definieren Sie die Eigenschaften → id und → text für die Schaltfläche Berechnen. Verwenden Sie den Präfix → bt für Buttons.

Damit die Schaltfläche über die gesamte Breite angezeigt wird, ändern wir die Eigenschaft → layout:width wie angezeigt.

Ergebnis:



Verändern Sie nachträglich diese Eigenschaft für die → EditText-Komponenten (etGewicht und etGroesse):



Im Component Tree:

Die Anzeige des Ergebnisses.

Nutzen Sie die → TextView-Komponente → Large Text“.

Definieren Sie die Eigenschaften → id und → hint für diese Komponente. Verwenden Sie den Präfix → tv für TextView.

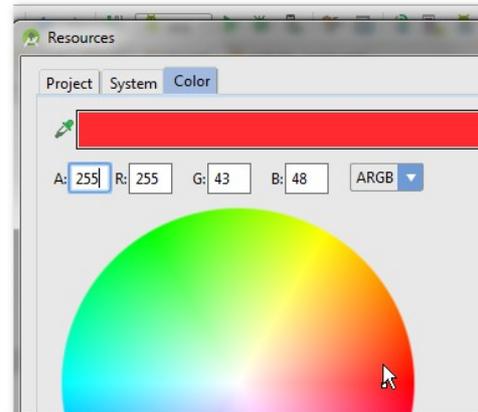
Ändern Sie außerdem die Eigenschaft für die Ausrichtung von Komponenten innerhalb des Layouts → gravity:

Ändern Sie die Eigenschaft Textfarbe → textColor. Klicken Sie dazu auf die Schaltfläche „...“ am rechten Rand.

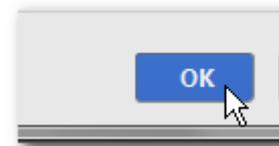
Ergebnis:



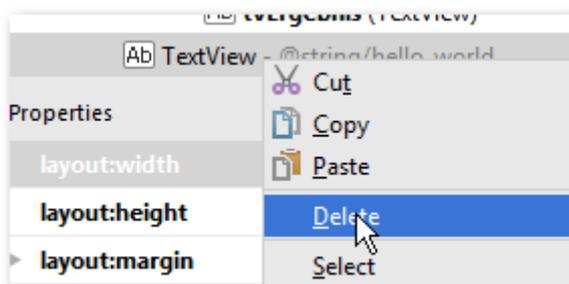
Wechseln Sie weiter auf den Reiter → Color im Fenster → Resource und wählen Sie einen beliebigen Rot-Ton.



Bestätigen Sie die Eingabe mit einem Klick auf die Schaltfläche → OK:

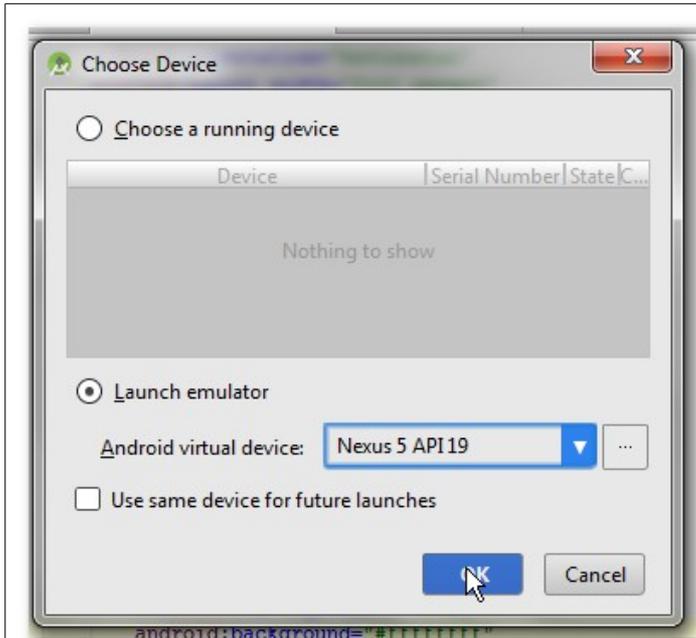


Entfernen Sie abschließend den Text → Large Text aus der Eigenschaft → text.



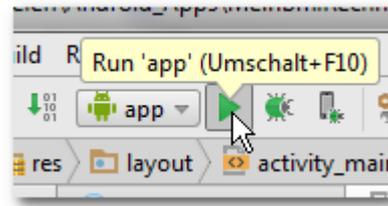
Entfernen Sie abschließend die noch enthaltene Standard-TextView-Komponente → hello_world.

Klicken Sie dazu mit der rechten Maustaste im rechten, oberen Frame-Fenster → Component Tree auf die Standard-TextView-Komponente „hello_world“ und wählen Sie im Kontext-Menü die Option → Delete.



Testen der View.

Wir starten nun den Emulator.

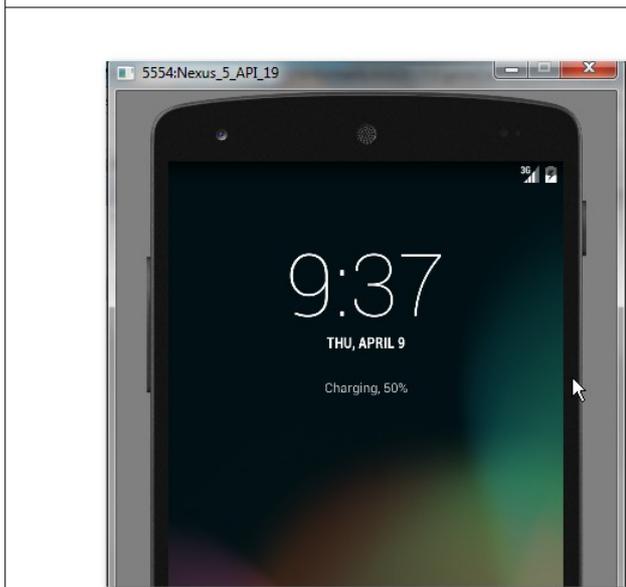
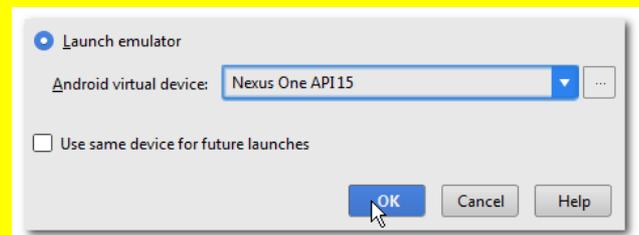


Emulator:

Der Emulator simuliert in unserem Fall ein virtuelles Mobiltelefon vom Typ „Nexus 5“.

Wir nutzen die API 19 (KitKat) oder die API 15 (SanwichIceCream) mit einem Nexus 4. Lollipop ist an einigen Stellen noch fehlerbehaftet und das würde bei den Tests stören.

Für wenig leistungsfähige Rechner empfiehlt sich ein Nexus One Device mit API 15 (SanwichIceCream).



Der Emulator öffnet sich.

Beim ersten öffnen kann das einen Moment dauern.

Ziehen Sie dann das auf dem Display erscheinende Schlösschen mit gedrückter linken Maustaste senkrecht nach oben.

Im Ergebnis sollte die Benutzeroberfläche erscheinen:



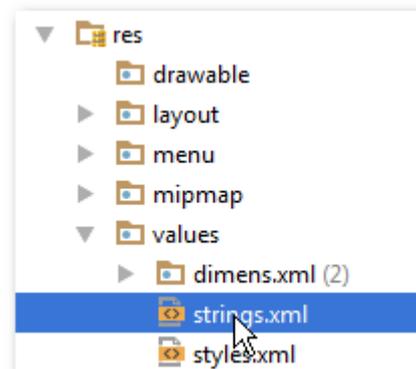
```

Edit translations for all locales in the translations editor.
<resources>
  <string name="app_name">MeinBmiRechner_1_0</string>
  <string name="hello_world">Hello world!</string>
  <string name="action_settings">Settings</string>
  <string name="tvGewicht">Gewicht:</string>
  <string name="etGewicht">###.#</string>
  <string name="tvGroesse">Größe:</string>
  <string name="etGroesse">###.##</string>
  <string name="btBerechnen">Berechnen</string>
  <string name="tfErgebnis">Anzeige des berechneten BMI</string>
</resources>

```

Betrachtung der XML-Datei „strings.xml“.

Die Datei befindet sich im Unterverzeichnis:



Hier sind alle definierten Strings aufgeführt.

Verändern Sie den Wert für das Attribut „app_name“, wie folgt:

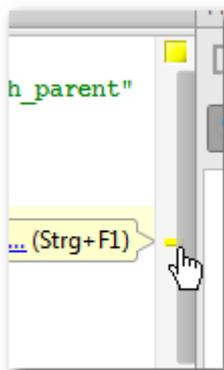
```
<string name="app_name">Mein BMI-Rechner 1.0</string>
```

```

1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2   xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
3   android:layout_height="match_parent" android:paddingLeft="16dp"
4   android:paddingRight="16dp"
5   android:paddingTop="16dp"
6   android:paddingBottom="16dp" tools:context=".MainActivity">
7
8   <LinearLayout
9     android:orientation="vertical"
10    android:layout_width="fill_parent"
11    android:layout_height="fill_parent">
12
13    <LinearLayout
14      android:orientation="horizontal"
15      android:layout_width="fill_parent"
16      android:layout_height="50dp">
17
18      <ImageView
19        android:layout_width="wrap_content"
20        android:layout_height="wrap_content"
21        android:id="@+id/ivLogo" />
22
23    </LinearLayout>

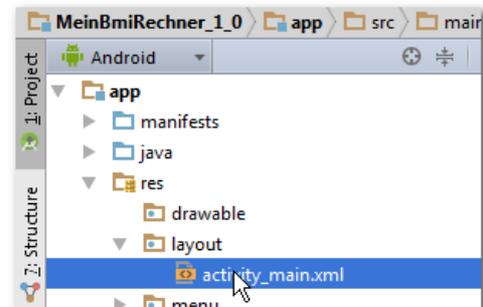
```

Auf der rechten Seite finden Sie den Hinweis auf die noch fehlende Angaben für die Bildeigenschaften innerhalb der ImageView. Wir kümmern uns später darum!

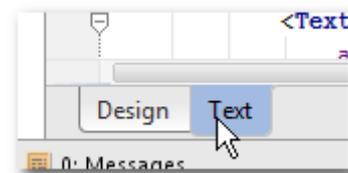


Betrachtung der XML-Datei „activity_main.xml“.

Die Datei befindet sich im Unterverzeichnis:

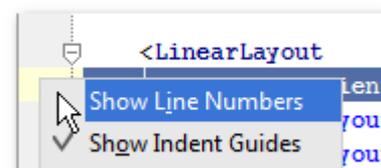


Um die Datei mit dem Text-Editor zu öffnen können Sie unterhalb des Designers auf den Reiter „Text“ klicken.

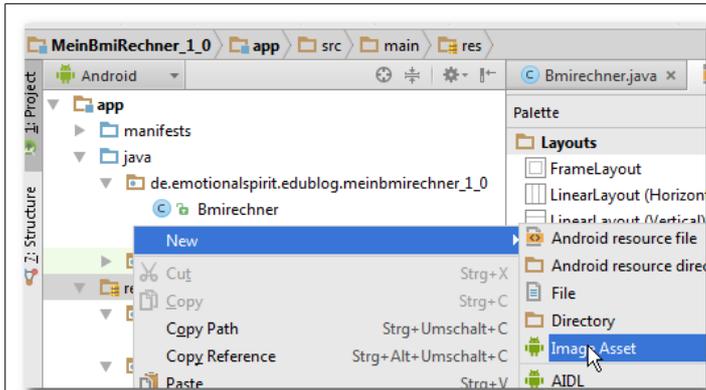


Hier sehen Sie das Ergebnis Ihrer geschachtelten Layouts.

Lassen Sie sich die Zeilennummern anzeigen:



Klicken Sie dazu mit der rechten Maustaste auf den linken Rand des XML-Text-Editors und wählen Sie im Kontext-Menü die Option „Show Line Numbers“.



App Icon ändern.

Klicken Sie dazu mit der rechten Maustaste in Ihrem Projekt auf das Verzeichnis `app` → `src` → `res` wählen Sie im Kontext-Menü die Option → `New Image Asset`.

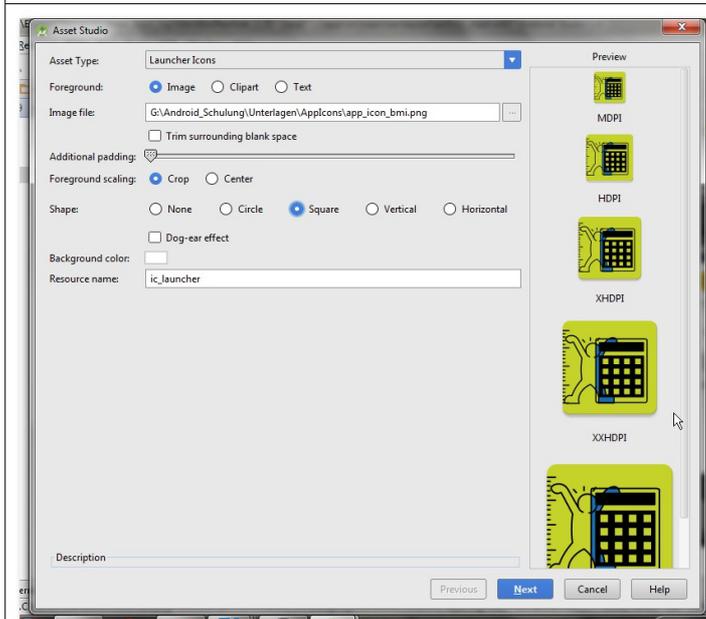
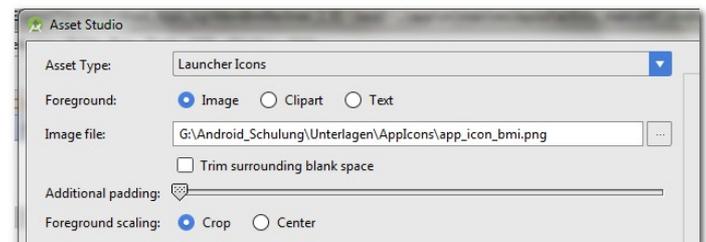


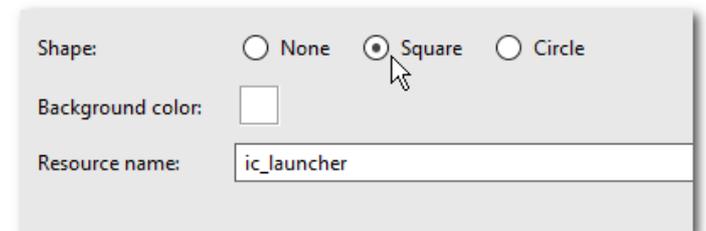
Image Icon definieren.

Wählen Sie dazu für den Image-File-Pfad die Bild-Datei aus:

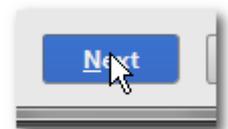
`AppIcons\app_icon_bmi.png`

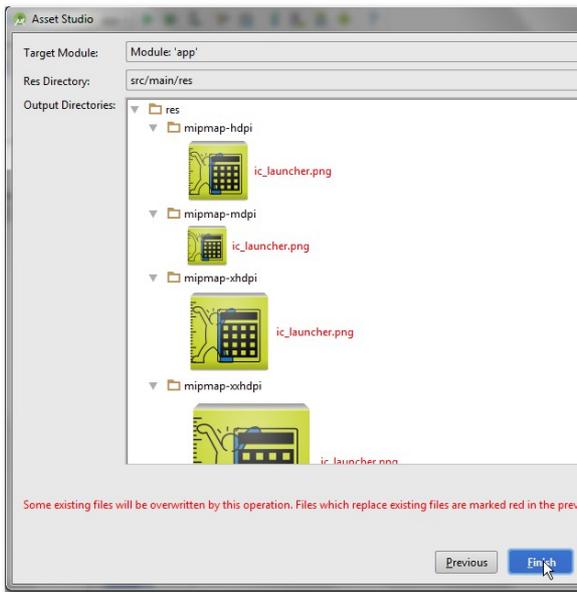


Aktivieren Sie für die Eigenschaft „Shape“ die Option „Square“ aus



Klicken Sie auf die Schaltfläche → `Next`.





Icon Konfiguration abschließen.

Klicken Sie auf Finish. Dabei wird das vorhandene Icon überschrieben.

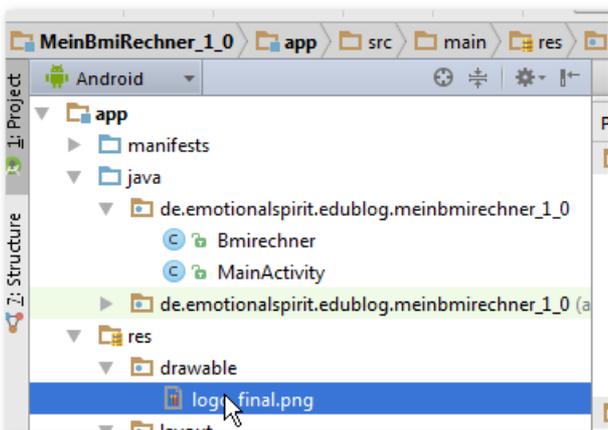
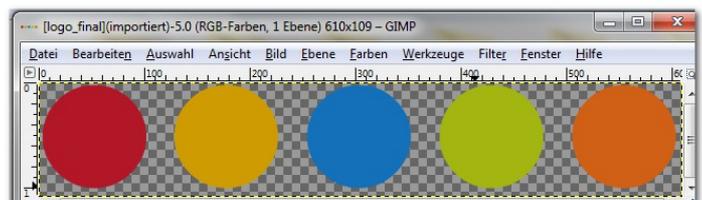
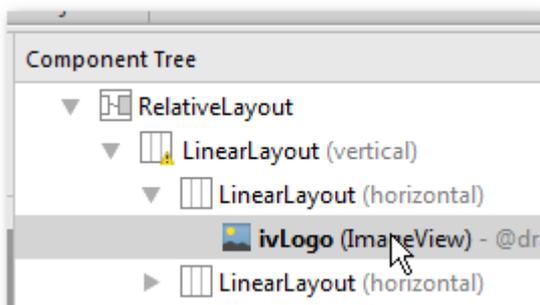


Bild (Banner) einfügen.



AppIcons\logo_final.png

Bilddatei „logo_final.png“ in das Verzeichnis → res → drawable kopieren.



Bildquelle definieren.

Dazu im Fenster Component Tree auf die ImageView-Komponente klicken.

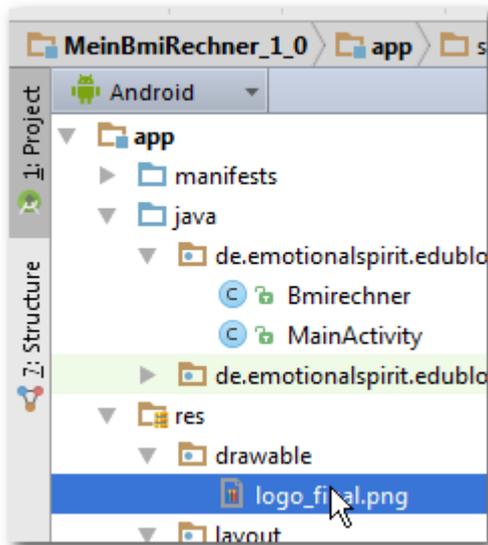


Bild kopieren und einfügen.

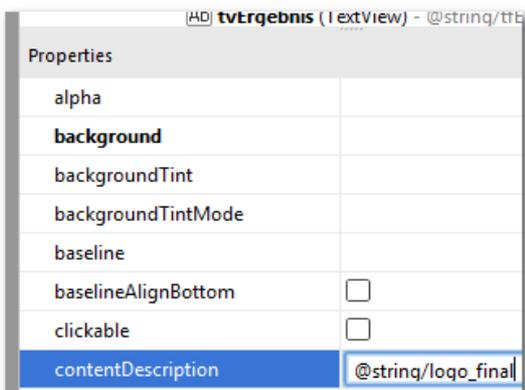


Im Fenster Properties die Bildquelle eingeben.

Dazu für die Eigenschaft „src“ die Quelle:

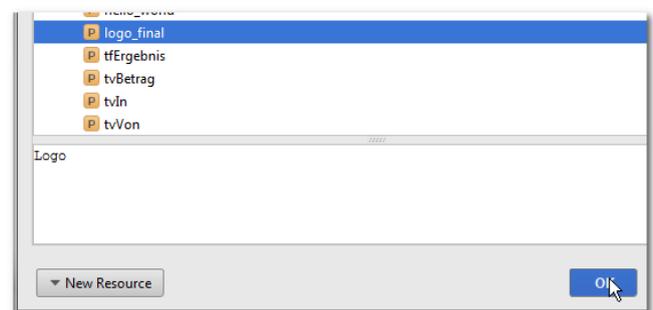
`@drawable/logo_final`

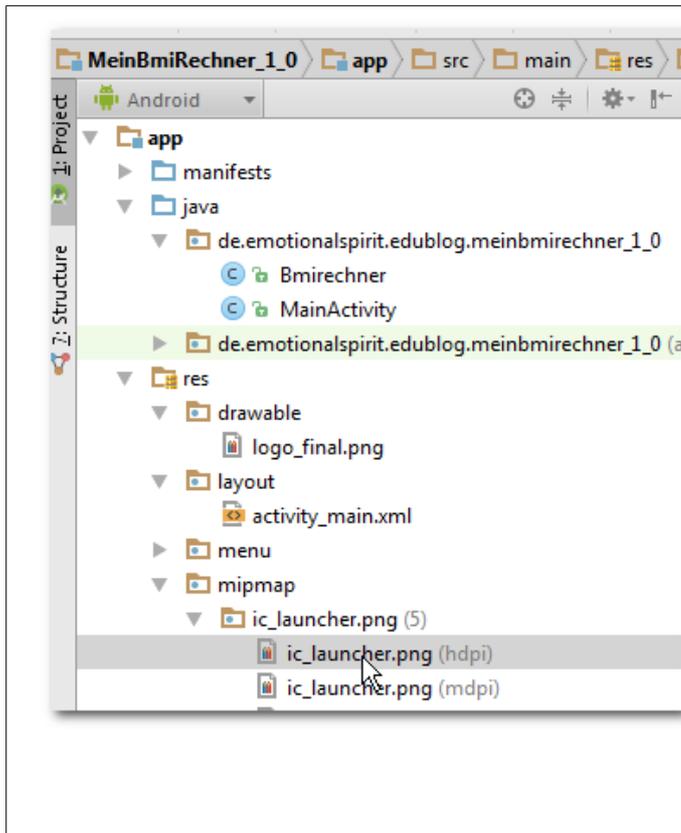
definieren.



Content Description definieren.

Dazu einen neuen String definieren. Dazu wie zuvor vorgehen:

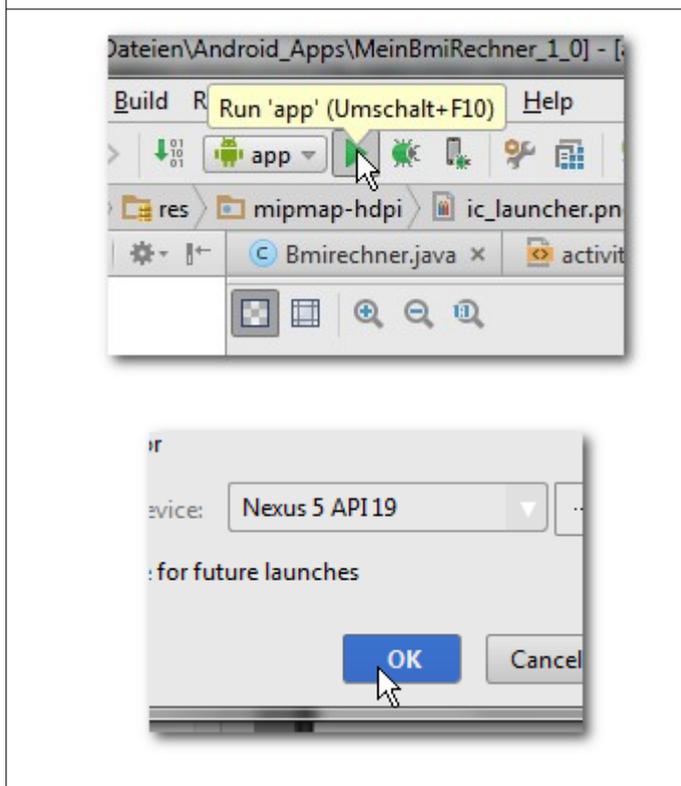
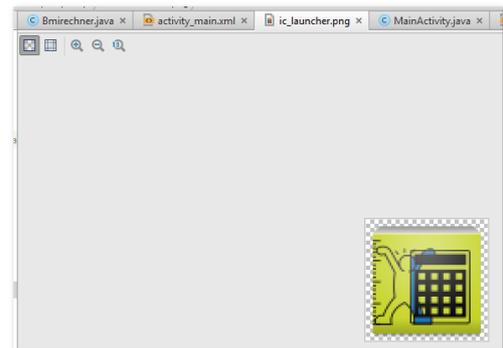




Kontrolle.

Dazu im „app“-Verzeichnis auf das Unterverzeichnis „mipmap“ klicken.

Im Verzeichnis „ic_launcher.png“ sind die erzeugten Icons in den unterschiedlichen Größen aufgeführt. Mit einem Doppelklick auf einer der Grafiken können Sie diese öffnen.



Icon und Logo Testen.

Testen Sie wie gewohnt die Anwendung. Klicken Sie dazu in der Symbol-Leiste auf die Schaltfläche „Run“.

Starten Sie die AVD mit einem Klick auf die Schaltfläche „OK“.



Logo (Banner) anzeigen.

Mit dem Öffnen der AVD sollte sich auf die Anwendung öffnen, wie nebenstehend angezeigt.

Um das App Icon zu sehen wechseln Sie in das App-Menü. Klicken Sie dazu diese Schaltfläche auf dem Display:



Icon anzeigen.

Auf dem Display ist das App Icon nun aufgeführt.



Gratulation das Layout ist erstellt!

4.2 Die Benutzeroberfläche des Taschenrechner 1.0



Wir möchten die nebenstehende Benutzeroberfläche des Taschenrechners erzeugen.

Aufgabe: Benutzeroberflächenlayout erstellen.

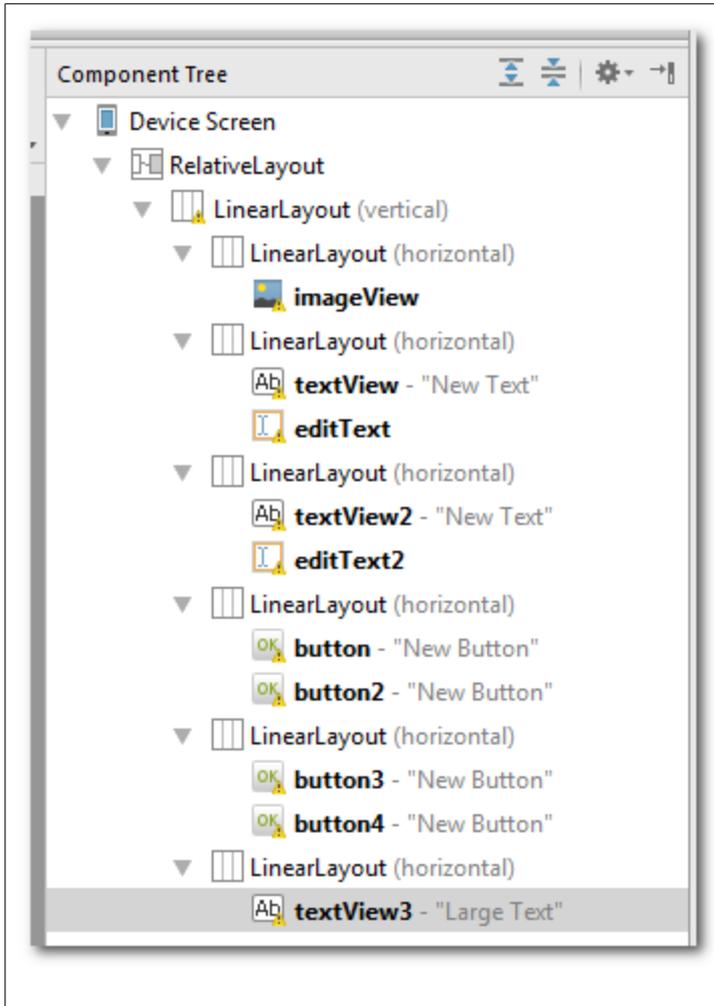
Ihre Aufgabe ist es nun das nebenstehend angezeigte Benutzeroberflächenlayout für den Taschenrechner exakt umzusetzen.

Nutzen Sie Ihre erworbenen Kenntnisse! Gehen Sie auf die selbe Art vor, wie in Kapitel für den „BMI-Rechner 1.0“ beschrieben wurde.

Nutzen Sie die nachfolgenden Vorgaben.

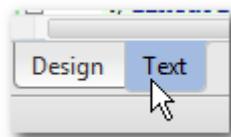
Vorgehensweise: Component Tree.

5. AppTheme „Holo Light“ wählen
6. Layoutschachtelung erzeugen
7. Komponenten im Layout platzieren
8. Komponenteneigenschaften definieren



Realisieren Sie dazu die Layout-Schachtelung, wie nebenstehend angezeigt.

Hinweis:
 Falls sich die Komponenten „ids“ über das Fenster „Properties“ im Design-Editor nicht ändern lassen, wechseln Sie bitte in den XML-Text-Editor um die notwendigen Anpassungen durchzuführen.



Komponenteneigenschaften: Properties.

1. das Logo

Komponente	@id/	text	gravity
ImageView	ivLogo	--	center

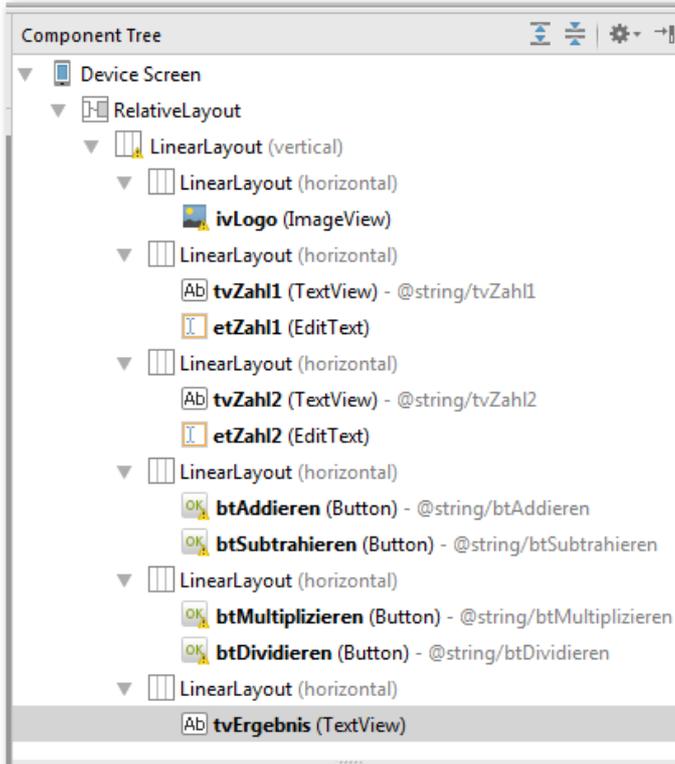
2. die Einabe der ersten Zahl

Komponente	@id/	text	Layout: width	numeric	hint
TextView	tvZahl1	tvZahl1	wrap_content	decimal	--
EditText	etZahl1	--	fill_parent	decimal	##.##

3. die Einabe der zweiten Zahl

Komponente	@id/	text	Layout: width	numeric	hint
TextView	tvZahl2	tvZahl2	wrap_content	decimal	--
EditText	etZahl2	--	fill_	decimal	##.##

Component Tree:



View:



			parent		
--	--	--	--------	--	--

4. die Schaltfläche Addieren

Komponente	@id/	
Button	btAddieren	

5. die Schaltfläche Subtrahieren

Komponente	@id/	
Button	btSubtrahieren	

6. die Schaltfläche Multiplizieren

Komponente	@id/	
Button	btMultiplizieren	

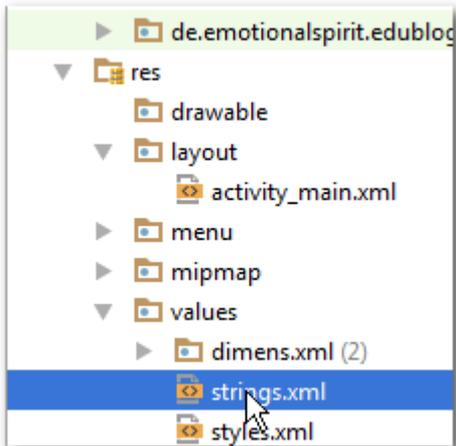
7. die Schaltfläche Dividieren

Komponente	@id/	
Button	btDividieren	

8. die Anzeige des Ergebnisses

Komponente	@id/	hint	gravity	textColor
TextView → LargeText	tv Ergebnis	Anzeige des be- rechneten Ergebnis- ses	center	#ffff2b30

Definieren Sie im Fenster „Properties“ die jeweils angegebenen Eigenschaften der Komponenten.

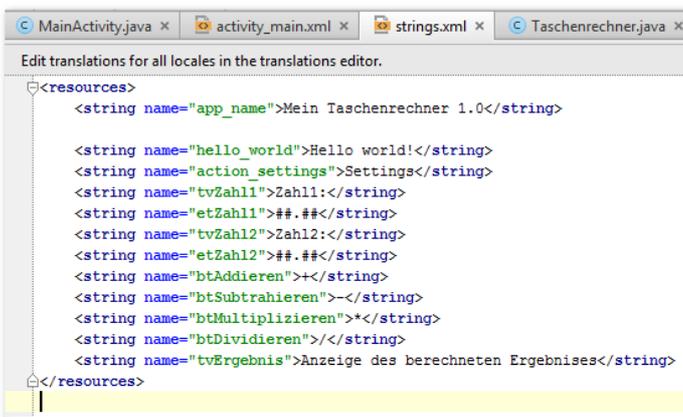


Selbstkontrolle: strings.xml.

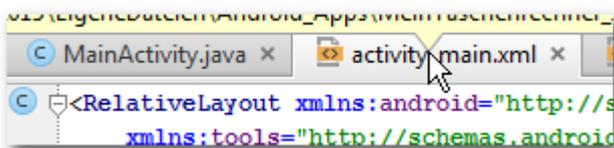
Zur Kontrolle prüfen wir, ob alle notwendigen Bezeichnungen als Strings definiert wurden.

Öffnen Sie aus Ihrem Verzeichnis
app → res → values

die Datei „strings.xml“.



Vergleichen, ergänzen und verbessern Sie ggf. Ihre Angaben.

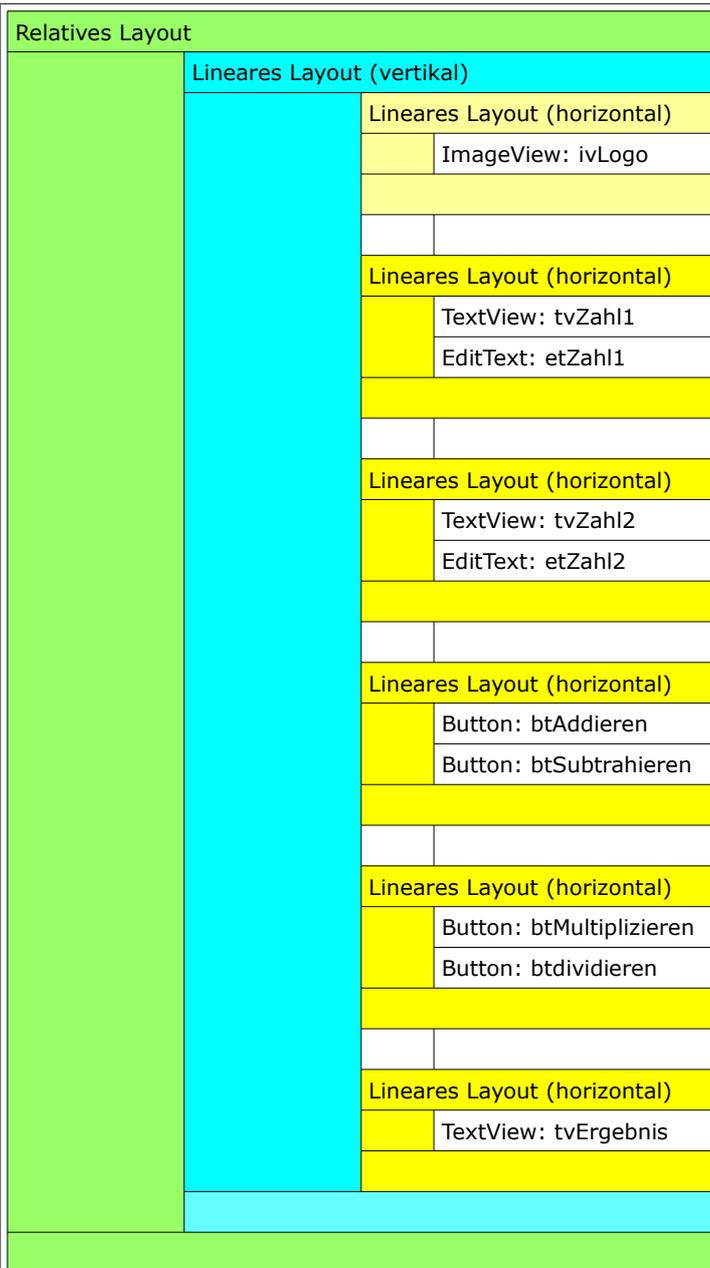


Selbstkontrolle: activity_main.xml.

Zur Kontrolle prüfen wir, ob die geforderte Layoutschichtung, die Komponenten und deren Eigenschaften definiert wurden.

Öffnen Sie aus Ihrem Verzeichnis
app → layout

die Datei „activity_main.xml“.



Prüfen Sie:

1. Schachtelung der Layouts
2. Komponenten ids
3. Sonstige Eigenschaften (siehe oben)

Für die EditView-Komponente: etZahl1

```

35  <EditText
36      android:layout_width="fill_parent"
37      android:layout_height="wrap_content"
38      android:id="@+id/etZahl1"
39      android:hint="@string/etZahl1"
40      android:numeric="decimal"
41      android:focusable="true"
42      android:singleLine="true"
43      android:imeOptions="actionNext"/>
    
```

Erweiterung der Editview-Komponenten im XML-Editor.

Prüfen Sie Ihre Angaben und fügen Sie fehlende XML-Anweisungen ein.

Hinweise:

```

41      android:focusable="true"
    
```

Der Benutzer kann in das Texteingabefeld (Edit-View) Eingaben tätigen, ohne die Komponente zuvor

Für die EditText-Komponente: etZahl2

```

57 | <EditText
58 |     android:layout_width="fill_parent"
59 |     android:layout_height="wrap_content"
60 |     android:id="@+id/etZahl2"
61 |     android:hint="@string/etZahl2"
62 |     android:numeric="decimal"
63 |     android:focusable="true"
64 |     android:singleLine="true"
65 |     android:imeOptions="actionDone"/>

```

explizit mit dem Finger ausgewählt zu haben. Wenn der Nutzer die Oberfläche berührt, wechselt die Anwendung automatisch in den Eingabemodus, wenn das bisher noch nicht der Fall war.

```
42 |     android:singleLine="true"
```

Wir beschränken die Eingabe auf eine Zeile.

```
43 |     android:imeOptions="actionNext",
```

Wir stellen sicher, dass die Tastur im Eingabemodus die Schaltfläche Next anzeigt. Damit kann der Nutzer in nächste Texteingabefeld springen.

```
65 |     android:imeOptions="actionDone",
```

Wir stellen sicher, dass die Tastur im Eingabemodus die Schaltfläche Done anzeigt. Damit kann der Nutzer den Eingabemodus verlassen. Die Tastatur verschwindet damit „auf Klick“.

Weitere Optionen sind ggf. möglich:

```

android:imeOptions="action"/>
rLayout>
Layout
roid:orientation="ho
roid:layout_width="f
roid:layout_height="
xtView
android:layout widt

```

- actionDone
- actionGo
- actionNext
- actionNone
- actionPrevious
- actionSearch
- actionSend
- actionUnspecified

Description

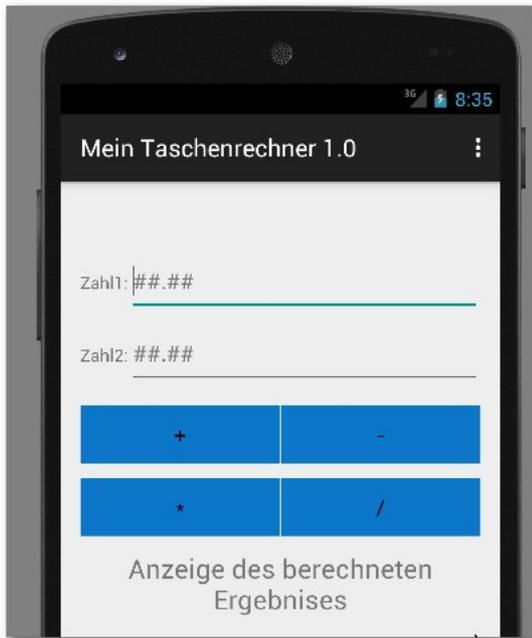
Button should be borderless

Button bars typically use a borderless style for the buttons. Set the `style="?android:attr/buttonBarButtonStyle"` attribute on each of the buttons, and set `style="?android:attr/buttonBarStyle"` on the parent layout

<http://developer.android.com/design/building-blocks/buttons.html>

Hinweise auf Design-Standards.

Immer wieder werden wir von der Entwicklungsumgebung an die geltenden Design-Standards erinnert. Bitte ignorieren Sie diese nicht, immerhin hat Google mitunter die besten Möglichkeiten das Verhalten von App-Nutzern zu analysieren. Die Präferenzen eines Nutzers sollten wir also in unserem Design berücksichtigen.



Der Style für Buttons.

Prüfen Sie Ihre Angaben und fügen Sie fehlende XML-Anweisungen ein. Ergänzen Sie auch die restlichen Buttons.

Hinweise zu den Buttons:

Komponente wird innerhalb des Layouts mit 1 gewichtet. → Hier: Alle Buttons sollen gleich groß sein:

```
82 | <Button android:layout_weight="1"
```

Der Text innerhalb des Buttons wird zentriert:

```
83 | android:gravity="center"
```

Entspricht dem durch Google empfohlenem Design-Standard-Style:

```
94 | style="?android:attr/buttonBarButtonStyle"
```

Individuelle Farbangabe (Hexadezimalcode) für den Hintergrund der Schaltfläche:

```
95 | android:background="#ff0c77c9"
```

Für die Schriftfarbe legen wir Schwarz fest:

```
86 | android:textColor="#ff000000"
```

Die Viewkomponente dient lediglich als Abstandhalter zwischen den Komponenten (Buttons):

```
<View
  android:layout_width="10dp"
  android:layout_height="wrap_content"
  android:background="#ffffff"
/>
```

Hinweise zu den Lineares Layout (horizontal):

Buttons:

```
77 | <Button
78 |     android:layout_width="wrap_content"
79 |     android:layout_height="wrap_content"
80 |     android:text="+"
81 |     android:id="@+id/btAddieren"
82 |     android:layout_weight="1"
83 |     android:gravity="center"
84 |     style="?android:attr/buttonBarButtonStyle"
85 |     android:background="#ff0c77c9"
86 |     android:textColor="#ff000000" />
```

View: als Separator zwischen den Buttons

```
88 | <View
89 |     android:layout_width="10dp"
90 |     android:layout_height="wrap_content"
91 |     android:background="#ffffff"
92 | />
```

Lineares Layout (horizontal):

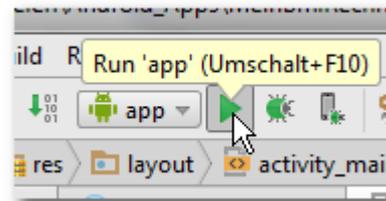
```
70 | <LinearLayout
71 |     android:orientation="horizontal"
72 |     android:layout_width="fill_parent"
73 |     android:layout_height="60dp"
74 |     style="?android:attr/buttonBarStyle"
75 | >
```

```
74 | style="?android:attr/buttonBarStyle"
```

Layouts, die Buttons enthalten, werden mit zusätzlichen Style-Angaben versehen. Damit wird u.a. sichergestellt, dass ein kleiner Raum zwischen den Button-Komponenten frei bleibt (optische Trennung).

Testen der View.

Wir starten nun den Emulator.

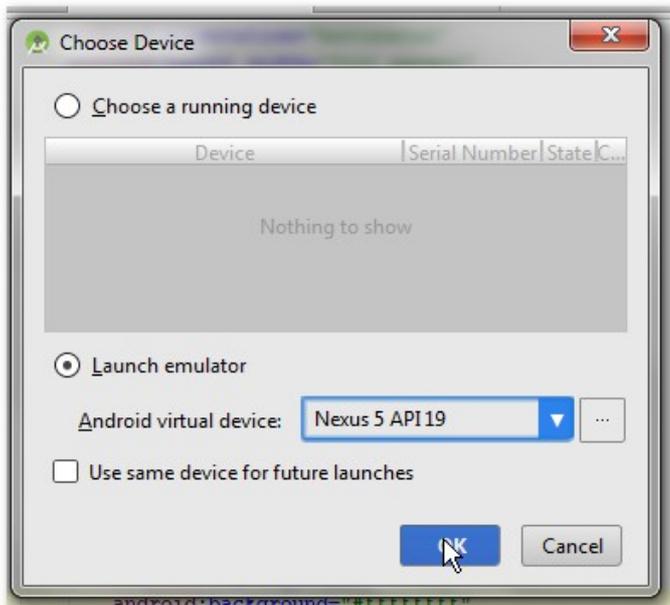


Emulator:

Der Emulator simuliert in unserem Fall ein virtuelles Mobiltelefon vom Typ „Nexus 5“.

Wir nutzen die API 19 (KitKat) oder die API 15 (SanwichIceCream) mit einem Nexus 4. Lollipop ist an einigen Stellen noch fehlerbehaftet und das würde bei den Tests stören.

Für wenig leistungsfähige Rechner empfiehlt sich ein Nexus One Device mit API 15 (SanwichIceCream).



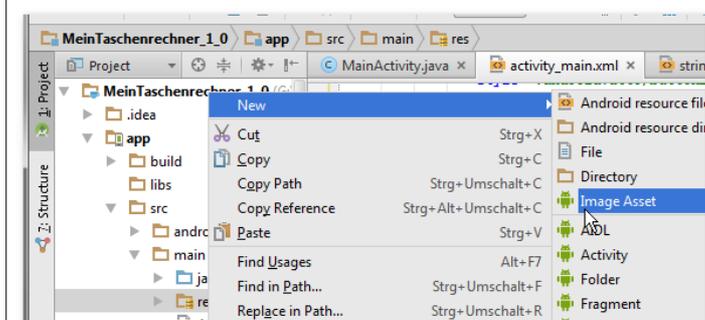


Der Emulator öffnet sich.

Beim ersten öffnen kann das einen Moment dauern.

Ziehen Sie dann das auf dem Display erscheinende Schlösschen mit gedrückter linken Maustaste senkrecht nach oben.

Im Ergebnis sollte die Benutzeroberfläche erscheinen:



App Icon ändern.

Klicken Sie dazu mit der rechten Maustaste in Ihrem Projekt auf das Verzeichnis app → src → res wählen Sie im Kontext-Menü die Option → New Image Asset.

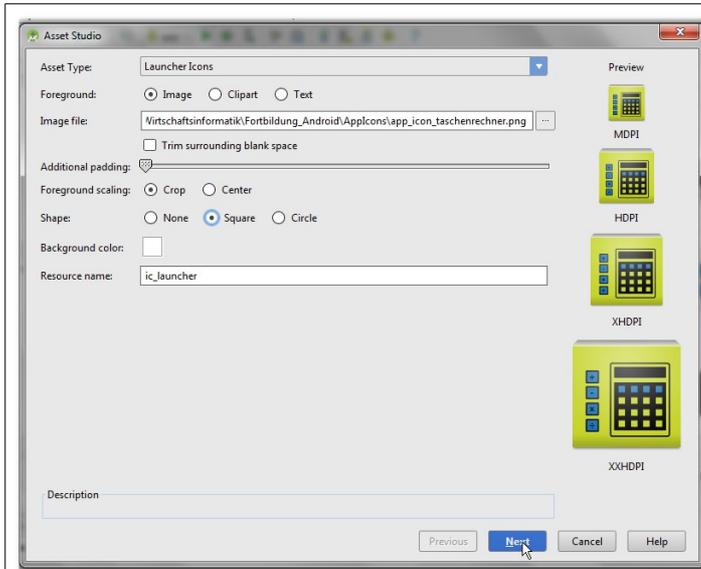
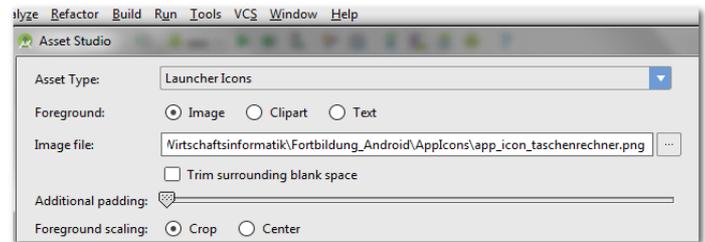


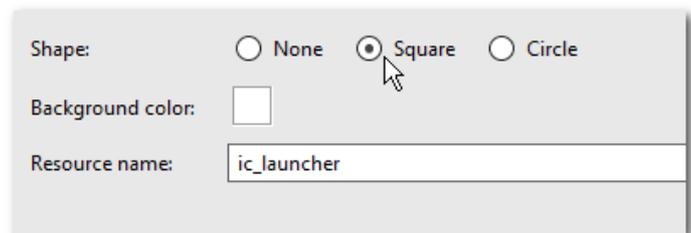
Image Icon definieren.

Wählen Sie dazu für den Image-File-Pfad die Bild-Datei aus:

AppIcons\app_icon_taschenrechner.png



Aktivieren Sie für die Eigenschaft „Shape“ die Option „Square“ aus



Klicken Sie auf die Schaltfläche Next.



Icon Konfiguration abschließen.

Klicken Sie auf Finish. Dabei wird das vorhandene Icon überschrieben.

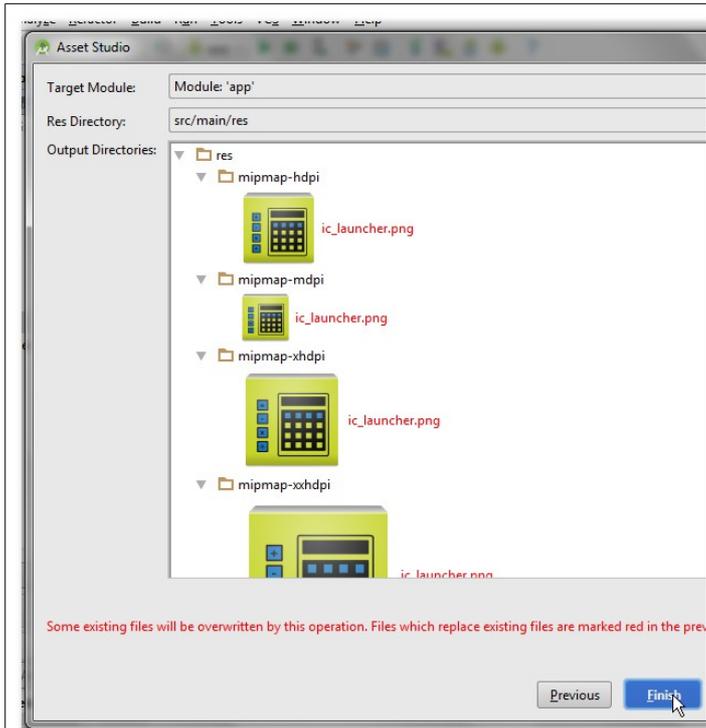
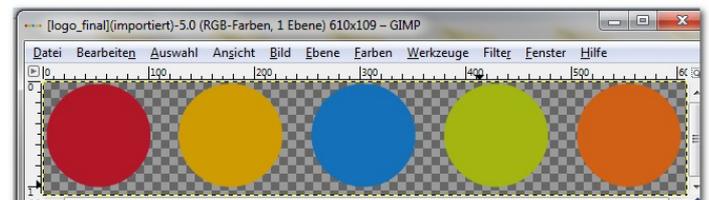
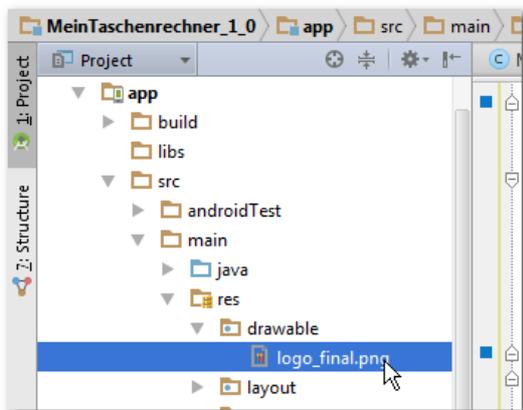
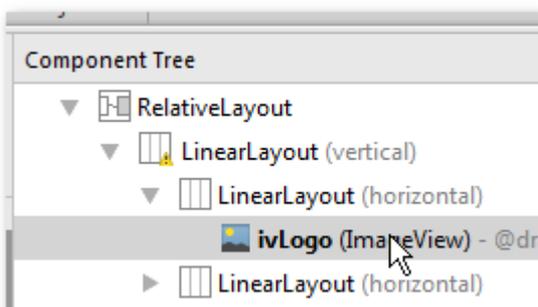


Bild (Banner) einfügen.



Bilddatei „logo_final.png“ in das Verzeichnis res
→ drawable kopieren.



Bildquelle definieren.

Dazu im Fenster Component Tree Auf die
ImageView-Komponente klicken.

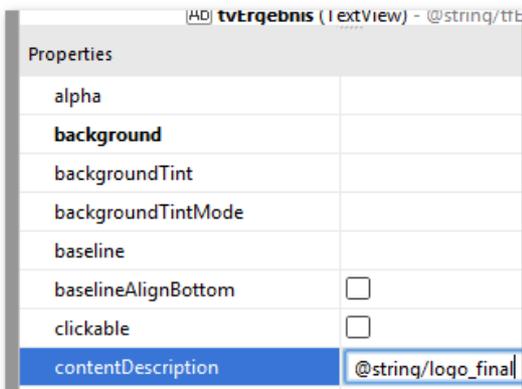


Im Fenster Properties die Bildquelle eingeben.

Dazu für die Eigenschaft „src“ die Quelle:

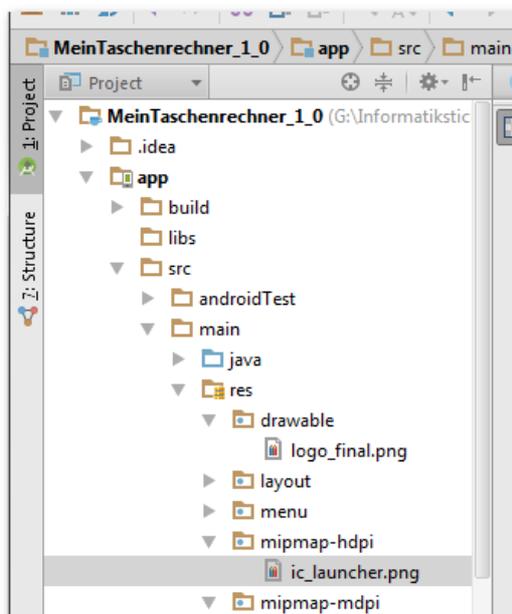
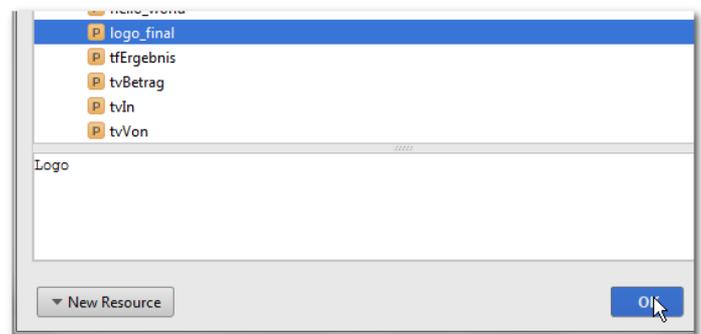
`@drawable/logo_final`

definieren.



Content Description definieren.

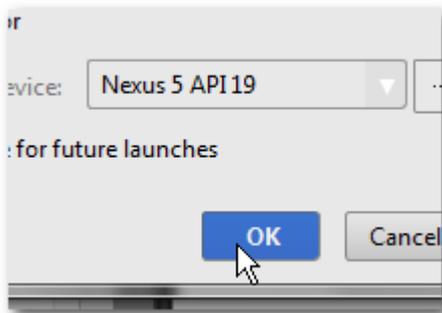
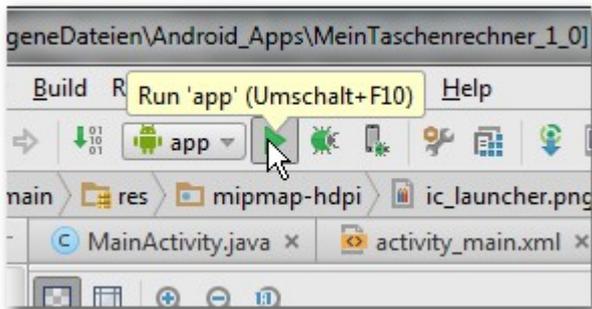
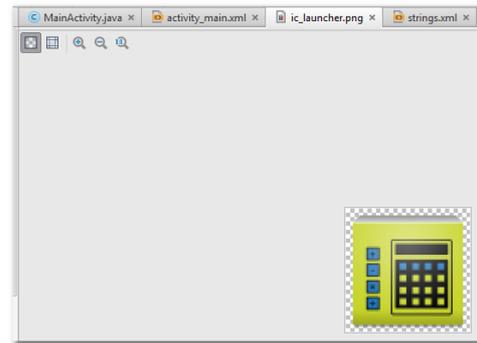
Dazu einen neuen String definieren. Dazu wie zuvor vorgehen:



Kontrolle.

Dazu im „app“-Verzeichnis auf das Unterverzeichnis „mipmap“ klicken.

Im Verzeichnis „ic_launcher.png“ sind die erzeugten Icons in den unterschiedlichen Größen aufgeführt. Mit einem Doppelklick auf einer der Grafiken können Sie diese öffnen.



Icon und Logo Testen.

Testen Sie wie gewohnt die Anwendung. Klicken Sie dazu in der Symbol-Leiste auf die Schaltfläche „Run“.

Starten Sie die AVD mit einem Klick auf die Schaltfläche „OK“.



Logo (Banner) anzeigen.

Mit dem Öffnen der AVD sollte sich auf die Anwendung öffnen, wie nebenstehend angezeigt.

Um das App Icon zu sehen wechseln Sie in das App-Menü. Klicken Sie dazu diese Schaltfläche auf dem Display:



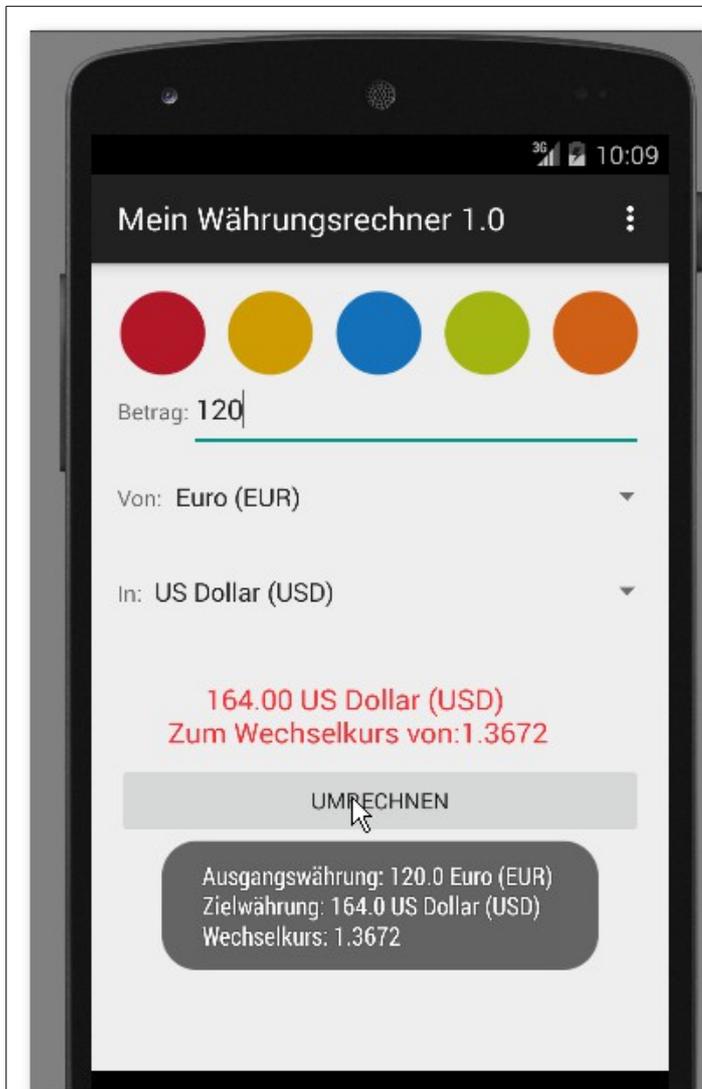
Icon anzeigen.

Auf dem Display ist das App Icon nun aufgeführt.



Gratulation das Layout ist erstellt!

4.3 Die Benutzeroberfläche des Währungsrechner 1.0



Wir möchten die nebenstehende Benutzeroberfläche des Währungsrechners erzeugen.

Besonderheit sind die Spinner-Komponenten, für die Auswahl der Ausgangs- und Zielwährung.

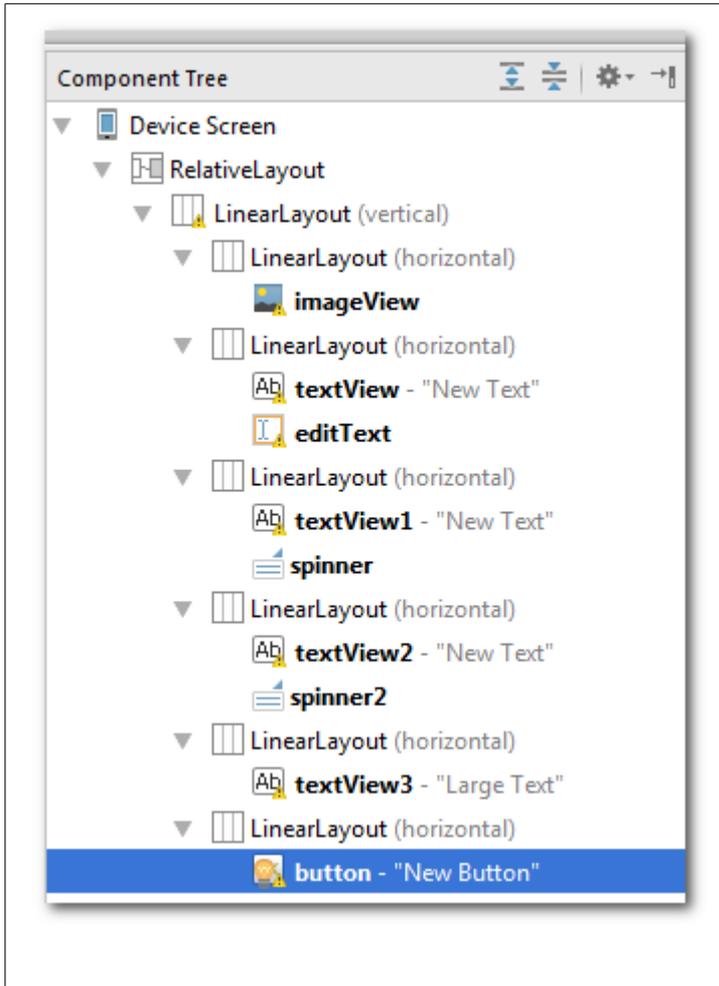


Aufgabe: Benutzeroberflächenlayout erstellen.

Ihre Aufgabe ist es nun das nebenstehend angezeigte Benutzeroberflächenlayout für den Währungsrechners exakt umzusetzen.

Nutzen Sie Ihre erworbenen Kenntnisse! Gehen Sie auf die selbe Art vor, wie bereits in Kapitel für den „Taschenrechner 1.0“ beschrieben wurde.

Nutzen Sie die nachfolgenden Vorgaben.



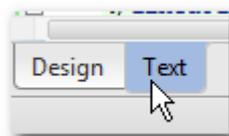
Vorgehensweise: Component Tree.

1. AppTheme „Holo Light“ wählen
2. Layoutschachtelung erzeugen
3. Komponenten im Layout platzieren
4. Komponenteneigenschaften definieren



Realisieren Sie dazu die Layout-Schachtelung, wie nebenstehend angezeigt.

Hinweis:
 Falls sich die Komponenten „ids“ über das Fenster „Properties“ im Design-Editor nicht ändern lassen, wechseln Sie bitte in den XML-Text-Editor um die notwendigen Anpassungen durchzuführen.



Komponenteneigenschaften: Properties.

1. das Logo

Komponente	@id/	text	layout:gravity
ImageView	ivLogo	--	center

2. die Einabe des Betrages

Komponente	@id/	text	layout_width	hint
TextView	tvBetrag	Betrag:	wrap_content	--
EditText	etBetrag	--	fill_parent	####.##

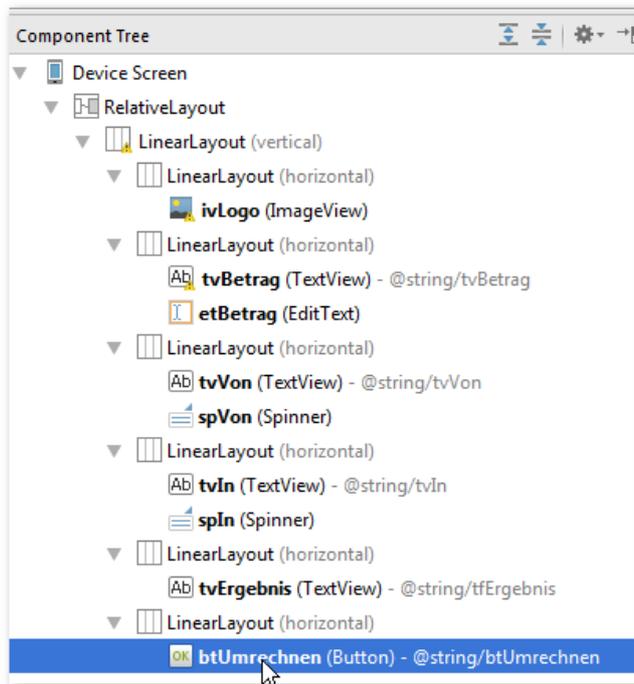
3. die Auswahl der Ausgangswährung

Komponente	@id/	text	layout_width		
TextView	tvVon	Von:	wrap_content		
spinner	spVon	--	fill_parent		

4. die Auswahl der Zielwährung

Komponente	@id/	text	layout_width		

Component Tree:



View:



TextView	tvZahl2	In:	wrap_content		
spinner	spIn	--	fill_parent		

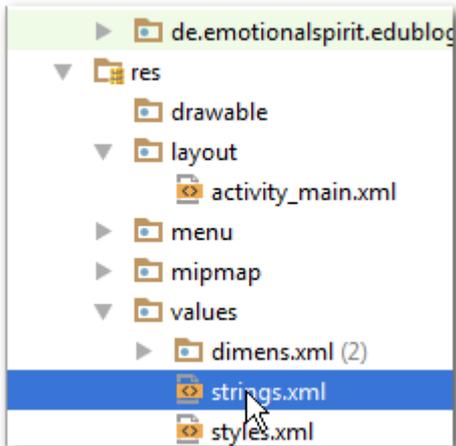
5. die Anzeige des Ergebnisses

Komponente	@id/	hint	gravity	textColor
TextView → LargeText	tv Ergebnis	Anzeige des be- rechneten Ergebnis- ses	center	#ffff2b30

6. die Schaltfläche Dividieren

Komponente	@id/	
Button	btUmrechnen	

Definieren Sie im Fenster „Properties“ die jeweils angegebenen Eigenschaften der Komponenten.



Selbstkontrolle: strings.xml.

Zur Kontrolle prüfen wir, ob alle notwendigen Bezeichnungen als Strings definiert wurden.

Öffnen Sie aus Ihrem Verzeichnis

app → res → values

die Datei „strings.xml“.

```

1  <resources>
2    <string name="app_name">MeinWahrungsrechner1_0</string>
3
4    <string name="hello_world">Hello world!</string>
5    <string name="action_settings">Settings</string>
6    <string name="tvVon">Von:</string>
7    <string name="tvIn">In:</string>
8    <string name="tfErgebnis">Anzeige des berechneten Ergebnisses</string>
9    <string name="btUmrechnen">Umrechnen</string>
10   <string name="tvBetrag">Betrag:</string>
11   <string name="etBetrag">#####</string>
12
13   <string-array name="von_array">
14     <item>Euro (EUR)</item>
15     <item>Britische Pfund (GBP)</item>
16     <item>US Dollar (USD)</item>
17     <item>Japanischer Yen (JPY)</item>
18   </string-array>
19
20   <string-array name="in_array">
21     <item>Euro (EUR)</item>
22     <item>Britische Pfund (GBP)</item>
23     <item>US Dollar (USD)</item>
24     <item>Japanischer Yen (JPY)</item>
25   </string-array>
26 </resources>

```

Vergleichen, ergänzen und verbessern Sie ggf. Ihre Angaben.

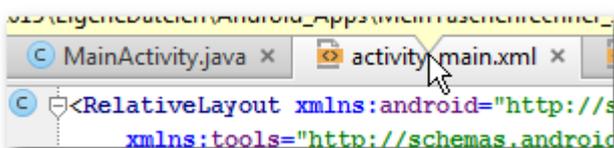
Für die Spinner-Komponenten (Dropdown-Menüs) benötigen wir jeweils eine einfache Liste (array).

Ergänzen Sie die beiden arrays:

```

13   <string-array name="von_array">
14     <item>Euro (EUR)</item>
15     <item>Britische Pfund (GBP)</item>
16     <item>US Dollar (USD)</item>
17     <item>Japanischer Yen (JPY)</item>
18   </string-array>
19
20   <string-array name="in_array">
21     <item>Euro (EUR)</item>
22     <item>Britische Pfund (GBP)</item>
23     <item>US Dollar (USD)</item>
24     <item>Japanischer Yen (JPY)</item>
25   </string-array>

```



Selbstkontrolle: activity_main.xml.

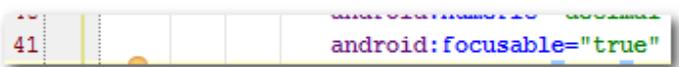
Zur Kontrolle prüfen wir, ob die geforderte Layoutschachtelung, die Komponenten und deren Eigenschaften definiert wurden.

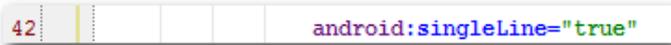
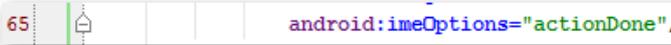
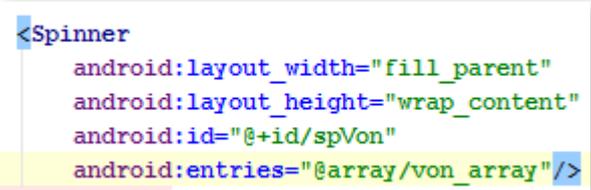
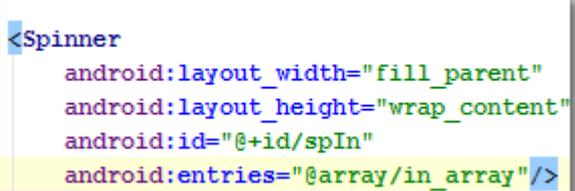
Öffnen Sie aus Ihrem Verzeichnis

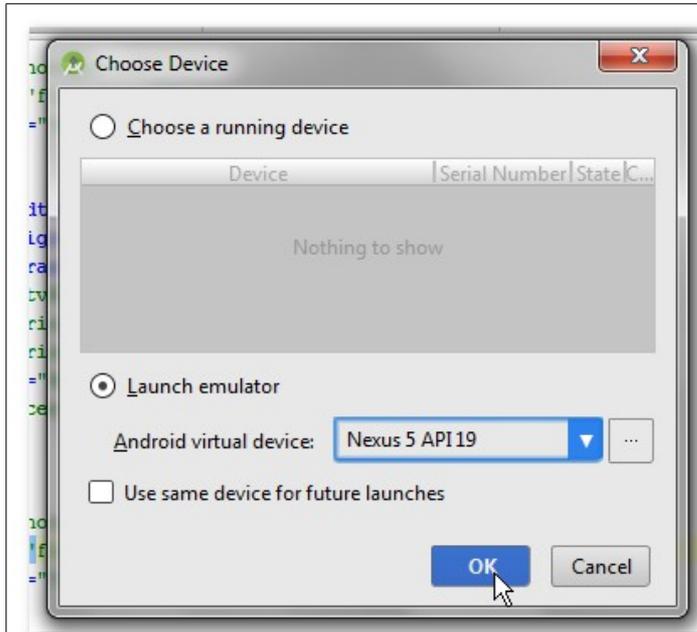
app → layout

<div style="background-color: #90EE90; padding: 5px;">Relatives Layout</div> <div style="background-color: #00FFFF; padding: 5px;">Lineares Layout (vertikal)</div> <div style="background-color: #FFFF00; padding: 5px;">Lineares Layout (horizontal)</div> <div style="background-color: #FFFF00; padding: 5px;">ImageView: ivLogo</div> <div style="background-color: #FFFF00; padding: 5px;">Lineares Layout (horizontal)</div> <div style="background-color: #FFFF00; padding: 5px;">TextView: tvBetrag</div> <div style="background-color: #FFFF00; padding: 5px;">EditText: etBetrag</div> <div style="background-color: #FFFF00; padding: 5px;">Lineares Layout (horizontal)</div> <div style="background-color: #FFFF00; padding: 5px;">TextView: tvVon</div> <div style="background-color: #FFFF00; padding: 5px;">Spinner: spVon</div> <div style="background-color: #FFFF00; padding: 5px;">Lineares Layout (horizontal)</div> <div style="background-color: #FFFF00; padding: 5px;">TextView: tvIn</div> <div style="background-color: #FFFF00; padding: 5px;">Spinner: spIn</div> <div style="background-color: #FFFF00; padding: 5px;">Lineares Layout (horizontal)</div> <div style="background-color: #FFFF00; padding: 5px;">TextView: tvErgebnis</div> <div style="background-color: #FFFF00; padding: 5px;">Lineares Layout (horizontal)</div> <div style="background-color: #FFFF00; padding: 5px;">Button: btUmrechnen</div>	<p>die Datei „activity_main.xml“.</p> <p><i>Prüfen Sie:</i></p> <ol style="list-style-type: none"> 1. Schachtelung der Layouts 2. Komponenten ids 3. Sonstige Eigenschaften (siehe oben)
--	---

<p>Für die EditText-Komponente: etBetrag</p>  <pre> 40 <EditText 41 android:layout_width="fill_parent" 42 android:layout_height="wrap_content" 43 android:id="@+id/etBetrag" 44 android:hint="@string/etBetrag" 45 android:focusable="true" 46 android:singleLine="true" 47 android:imeOptions="actionDone"/> </pre>
--

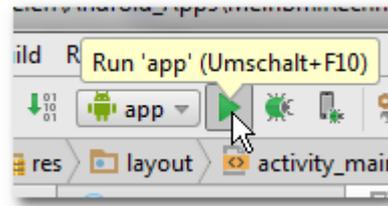
<p><i>Erweiterung der Editview-Komponenten im XML-Editor.</i></p> <p>Prüfen Sie Ihre Angaben und fügen Sie fehlende XML-Anweisungen ein.</p> <p>Hinweise:</p>  <pre> 41 android:focusable="true" </pre>
--

	<p>Der Benutzer kann in das Texteingabefeld (Edit-View) Eingaben tätigen, ohne die Komponente zuvor explizit mit dem Finger ausgewählt zu haben. Wenn der Nutzer die Oberfläche berührt, wechselt die Anwendung automatisch in den Eingabemodus, wenn das bisher noch nicht der Fall war.</p>  <p>Wir beschränken die Eingabe auf eine Zeile.</p>  <p>Wir stellen sicher, dass die Tastatur im Eingabemodus die Schaltfläche Done anzeigt. Damit kann der Nutzer den Eingabemodus verlassen. Die Tastatur verschwindet damit „auf Klick“.</p>
<p>Für den spVon:</p>  <p>Für den spIn:</p> 	<p><i>Erweiterung der Spinner-Komponenten im XML-Editor.</i></p> <p>Wir weisen dem Spinner die String-Elemente aus dem jeweiligen Array zu.</p>



Testen der View.

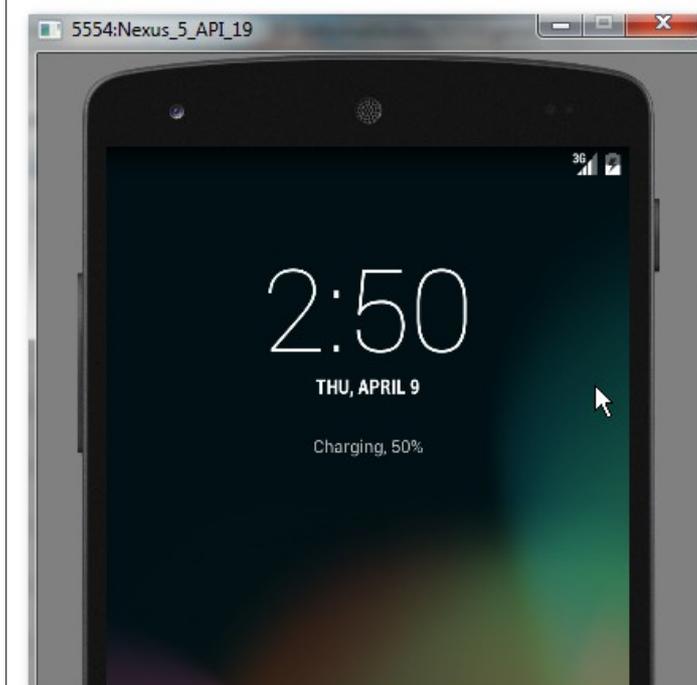
Wir starten nun den Emulator.



Emulator:

Der Emulator simuliert in unserem Fall ein virtuelles Mobiltelefon vom Typ „Nexus 5“.

Überprüfen Sie Ihre Einstellungen und ändern Sie diese ggf. wie nebenstehend angezeigt ab.

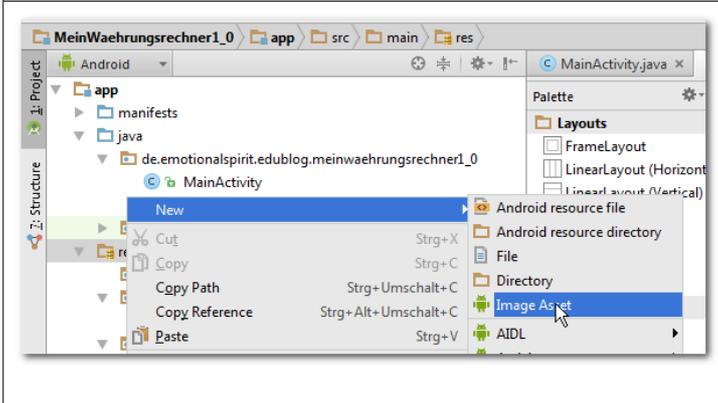
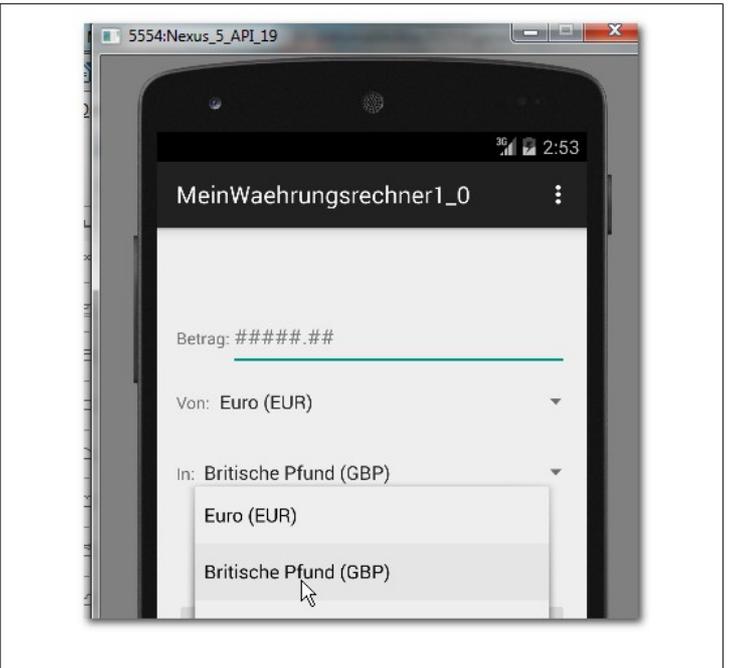
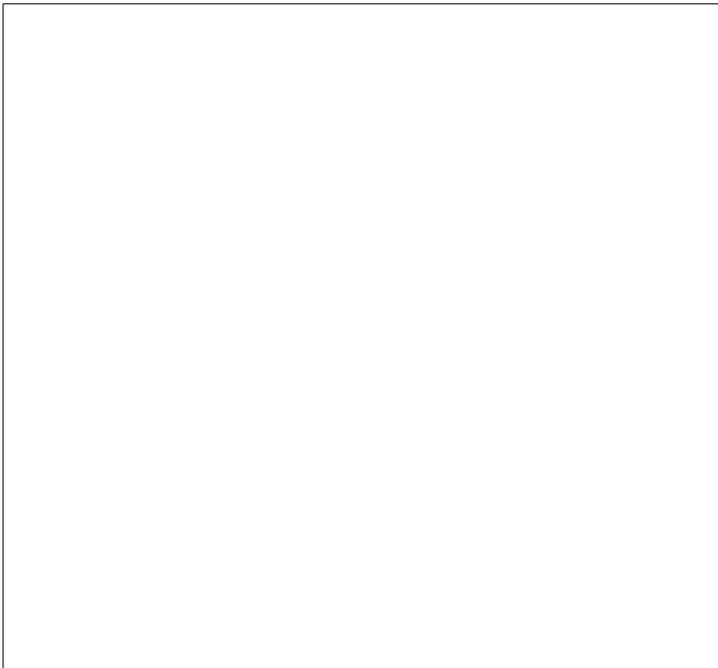


Der Emulator öffnet sich.

Beim ersten öffnen kann das einen Moment dauern.

Ziehen Sie dann das auf dem Display erscheinende Schlösschen mit gedrückter linken Maustaste senkrecht nach oben.

Im Ergebnis sollte die Benutzeroberfläche erscheinen:



App Icon ändern.

Klicken Sie dazu mit der rechten Maustaste in Ihrem Projekt auf das Verzeichnis → res wählen Sie im Kontext-Menü die Option → New Image Asset.

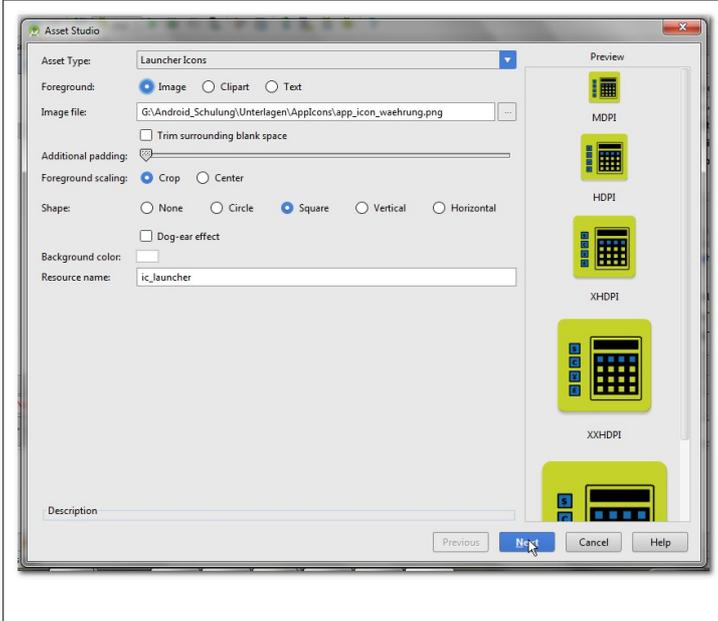


Image Icon definieren.

Wählen Sie dazu für den Image-File-Pfad die Bild-Datei aus:

AppIcons\app_icon_waehrung.png

Aktivieren Sie für die Eigenschaft „Shape“ die Option „Square“ aus

Shape: None Square Circle

Background color:

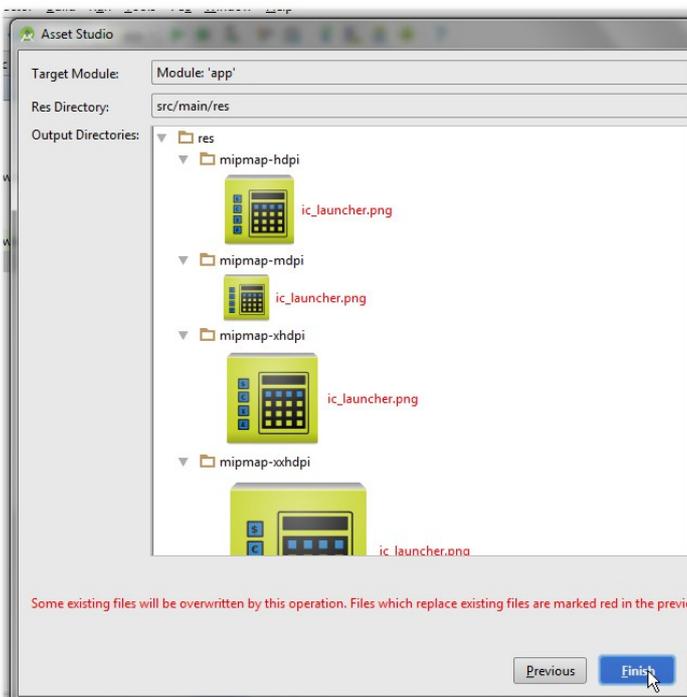
Resource name:

Klicken Sie auf die Schaltfläche Next.



Icon Konfiguration abschließen.

Klicken Sie auf Finish. Dabei wird das vorhandene Icon überschrieben.



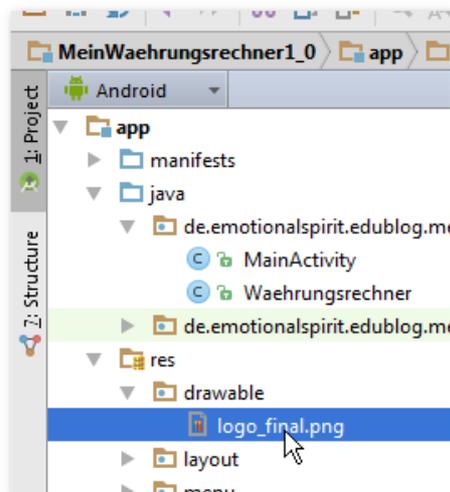
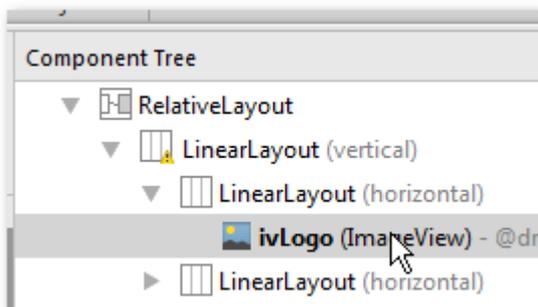


Bild (Banner) einfügen.

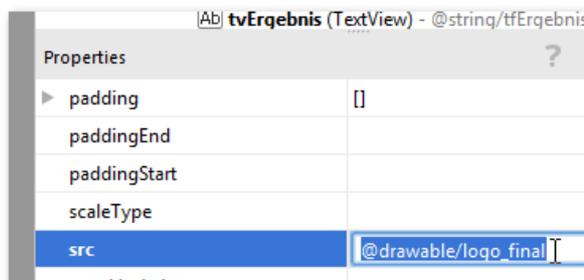


Bilddatei „logo_final.png“ in das Verzeichnis res
→ drawable kopieren.



Bildquelle definieren.

Dazu im Fenster Component Tree Auf die
ImageView-Komponente klicken.

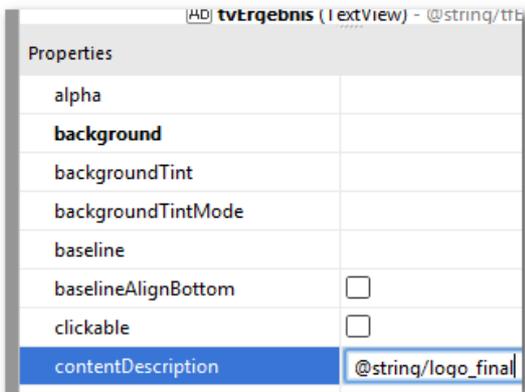


Im Fenster Properties die Bildquelle eingeben.

Dazu für die Eigenschaft „src“ die Quelle:

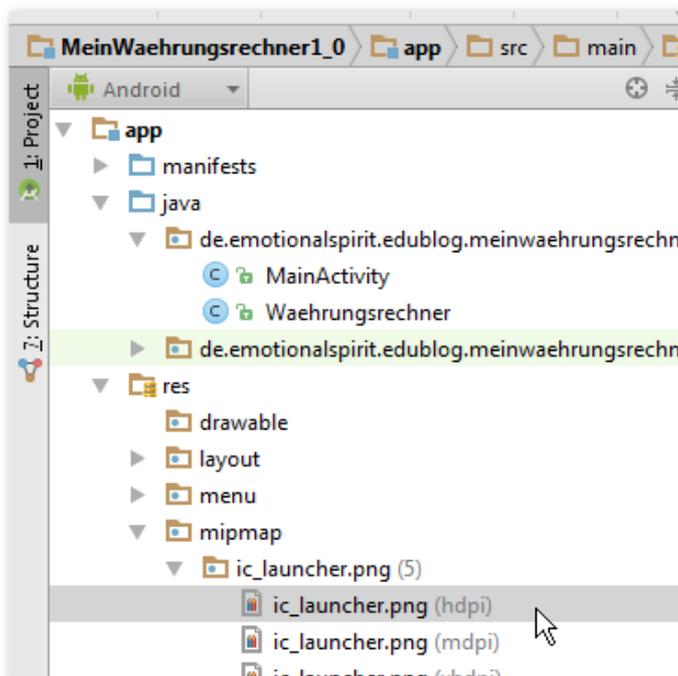
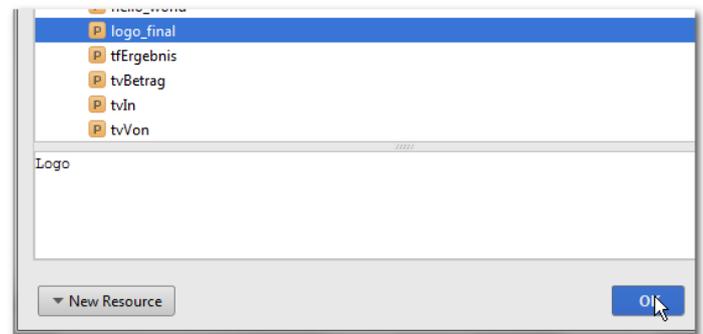
`@drawable/logo_final`

definieren.



Content Description definieren.

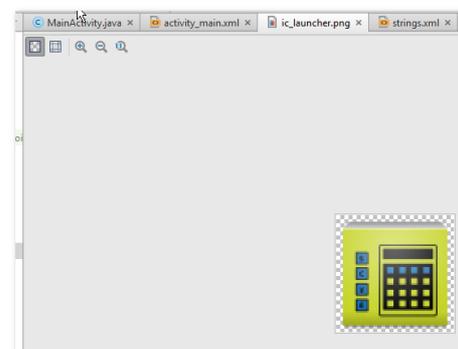
Dazu einen neuen String definieren. Dazu wie zuvor vorgehen:

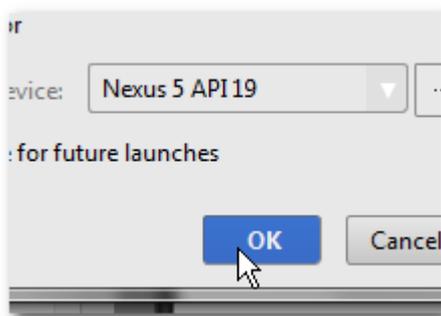
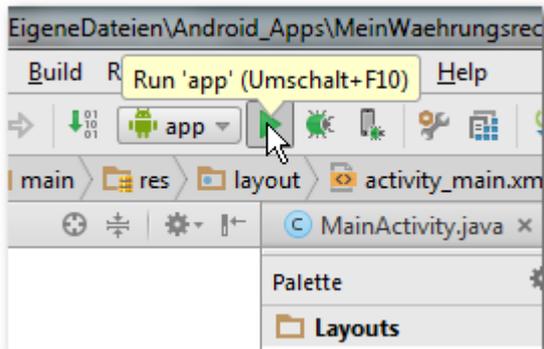


Kontrolle.

Dazu im „app“-Verzeichnis auf das Unterverzeichnis „mipmap“ klicken.

Im Verzeichnis „ic_launcher.png“ sind die erzeugten Icons in den unterschiedlichen Größen aufgeführt. Mit einem Doppelklick auf einer der Grafiken können Sie diese öffnen.





Icon und Logo Testen.

Testen Sie wie gewohnt die Anwendung. Klicken Sie dazu in der Symbol-Leiste auf die Schaltfläche „Run“.

Starten Sie die AVD mit einem Klick auf die Schaltfläche „OK“.



Logo (Banner) anzeigen.

Mit dem Öffnen der AVD sollte sich auf die Anwendung öffnen, wie nebenstehend angezeigt.

Um das App Icon zu sehen wechseln Sie in das App-Menü. Klicken Sie dazu diese Schaltfläche auf dem Display:





Icon anzeigen.

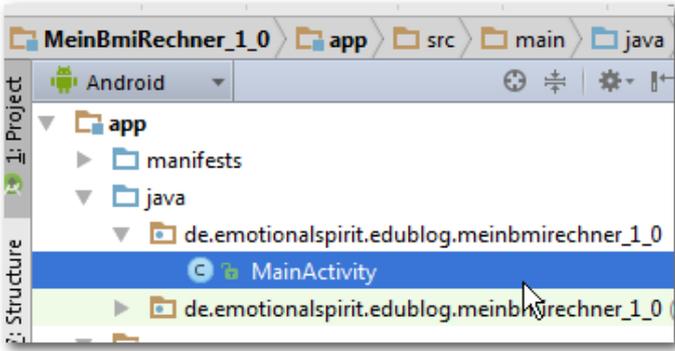
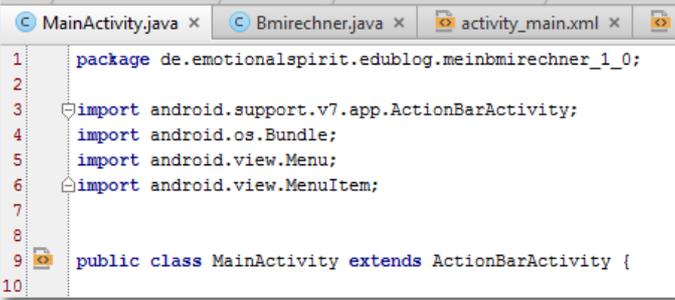
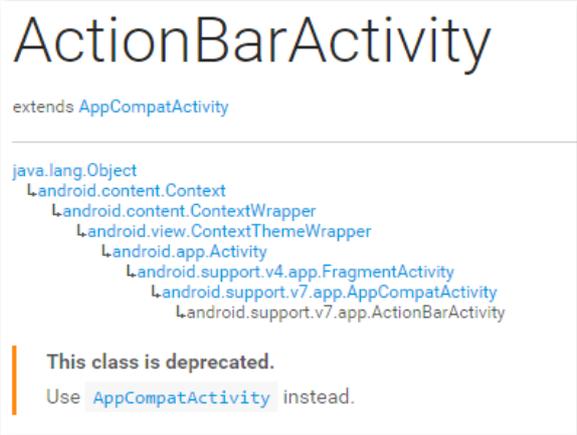
Auf dem Display ist das App Icon nun aufgeführt.



Gratulation das Layout ist erstellt!

5 Controller: Implementierung der Ereignissteuerung

5.1 Ereignissteuerung des BMI-Rechners 1.0

	<p>Öffnen Sie die Klasse <i>MainActivity.java</i></p> <p>Activity: Bei Anwendungen auf Android Betriebssystemen erfolgt die Zerlegung aufgabenorientiert. Konkret bedeutet das, dass der Quellcode für die Steuerung einer Funktionalität in eine Activity-Klasse ausgelagert wird. Vielfach erkennt man die Aktivitäten (Activities) schon auf der Benutzeroberfläche, denn u.a. repräsentieren Schaltflächen solche Funktionalitäten.</p>
 <pre> 1 package de.emotionalspirit.edublog.meinbmirechner_1_0; 2 3 4 import android.support.v7.app.AppCompatActivity; 5 import android.os.Bundle; 6 import android.view.Menu; 7 import android.view.MenuItem; 8 9 public class MainActivity extends AppCompatActivity { 10 </pre>	<p>Pakete und Importe.</p> <p>Zeile 1 beinhaltet die Angabe des Package. Die Angabe setzt sich zusammen aus den eingangs definierten Projekteigenschaften (→ Domain und → App name).</p> <p>Die Klasse → MainActivity stellt eine Erweiterung der Klasse → AppCompatActivity* dar.</p>
<p>→ AppCompatActivity*</p>  <p>ActionBarActivity extends AppCompatActivity</p> <p>java.lang.Object ↳ android.content.Context ↳ android.content.ContextWrapper ↳ android.view.ContextThemeWrapper ↳ android.app.Activity ↳ android.support.v4.app.FragmentActivity ↳ android.support.v7.app.AppCompatActivity ↳ android.support.v7.app.AppCompatActivity</p> <p>This class is deprecated. Use AppCompatActivity instead.</p>	<p>Im Gegensatz zu anderen Java-Anwendungen benötigen Android Apps die MainActivity, um eine Instanz der Anwendung zu erzeugen, außerdem stellt sie den Lebenszyklus der Instanz sicher und ergreift ggf. alle lebenserhaltenden Maßnahmen. Im Prinzip übernimmt das Objekt der MainActivity-Klasse u.a. die Funktionalität der Main-Methode einer konventionellen Java-Anwendung.</p> <p>Die vererbten standardmäßig vorhandenen Verhaltensweisen (Methoden) einer → Activity erfordern die im oberen Teil der Klasse angegebenen Import-Anweisungen der Klassen → ActionBarActivity, → Bundle, → Menu und → MenuItem.</p> <p>Aktuelle Hinweise – Ereignissteuerung</p> <p>Die MainActivity erbt zwischenzeitlich standardmäßig von der Klasse AppCompatActivity:</p>

	<p>MainActivity extends AppCompatActivity</p> <p>Bei den meisten älteren Projekten erbt die MainActivity von der ActionBarActivity</p> <p>MainActivity extends ActionBarActivity</p> <p>Die Verwendung der Klasse ActionBarActivity ist hinfällig (depreceated).</p> <p>Eine Anpassung älterer Projekte an die neue Architektur ist empfehlenswert.</p>
<pre> 19 //Komponenten deklarieren 20 private EditText gewicht; 21 private EditText groesse; 22 private TextView bmi; 23 private Button berechnen; </pre> <p>Nachher:</p> <pre> 17 public class MainActivity extends ActionBarActivity { 18 19 //Komponenten deklarieren 20 private EditText gewicht; 21 private EditText groesse; 22 private TextView bmi; 23 private Button berechnen; </pre>	<p><i>Komponenten deklarieren.</i></p> <p>Deklarieren Sie im Anschluss an die Klassende-klaration die Komponenten der Benutzeroberfläche.</p> <pre> 17 public class MainActivity extends 18 ? android.widget.EditText? Alt+Eingabe 19 //Komponenten deklarieren 20 private EditText gewicht; </pre> <p>Klicken Sie auf die rot gekennzeichneten Klassen-amen für die Komponente und folgen Sie der Empfehlung mit der Tastenkombination ALT + Eingabe (Enter) die Klasse zu importieren:</p> <pre> 7 import android.widget.EditText; </pre> <p>Wiederholen Sie den Vorgang für die TextView und Button-Komponente.</p>
<pre> 11 @Override 12 protected void onCreate(Bundle savedInstanceState) { 13 super.onCreate(savedInstanceState); 14 setContentView(R.layout.activity_main); 15 } </pre>	<p><i>Die onCreate-Methode.</i></p> <p>In der onCreate-Methode sollte das beim Aufruf des Activity-Objektes benötigte Layout (XML-Datei) übermitteln und in einem Objektbaum entfalten.</p> <p>Genau das geschieht mit dem Methodenauf- setContentView(...)</p>

	<p>R ist eine Klasse deren Aufgabe es ist, alle Elemente der Layouts und anderer XML-Dateien zu verwalten, u.a. um diese in Java verfügbar zu machen.</p>
<pre> 30 @Override 31 protected void onCreate(Bundle savedInstanceState) { 32 super.onCreate(savedInstanceState); 33 setContentView(R.layout.activity_main); 34 35 //Initialisierung der Komponenten 36 gewicht = (EditText) findViewById(R.id.etGewicht); 37 groesse = (EditText) findViewById(R.id.etGroesse); 38 bmi = (TextView) findViewById(R.id.tvErgebnis); 39 berechnen = (Button) findViewById(R.id.btBerechnen); </pre>	<p><i>Ausstattung der onCreate-Methode.</i></p> <p>Wir müssen sicherstellen, dass Komponenten, deren Inhalte gelesen bzw. in die geschrieben werden soll, zuvor initialisiert werden. Wir ergänzen dazu den Quellcode, wie nebenstehend angezeigt.</p> <p>Erklärung:</p> <pre>gewicht = (EditText) findViewById(R.id.etGewicht);</pre> <ul style="list-style-type: none"> • gewicht: Ist u.a. ein Klassenattribut der Activity-Klasse vom Typ EditText (siehe Deklaration). • (EditText): Der Cast stellt sicher, dass die zugewiesene Komponente dem Typ entspricht. • findViewById(int) Sucht den Parameterwert anhand der id. Als Parameter wird ein int-Wert erwartet. • R.id.tvErgebnis R liefert zum String tvErgebnis den entsprechenden int-Wert zurück. Den entsprechenden Schlüsselwert.
<pre> 41 //Text-Listener für die EditText Komponenten 42 gewicht.addTextChangedListener(43 new TextWatcher() { 44 45 }); </pre> <p>Auszug aus der API:</p>	<p><i>Listener in der onCreate-Methode.</i></p> <p>Ein Listener ist wie ein Fühler der Veränderungen auf der Benutzeroberfläche registriert und in Form eines Impulses an das System weiterreicht.</p> <p>Wir fügen dem editierbaren Objekt → gewicht mit dem Methodenaufruf</p> <pre>gewicht.addTextChangedListener(TextWatcher watcher)</pre>

public interface Summary: Methods | [Expand All]
Added in API level 1

TextWatcher

implements [NoCopySpan](#)

android.text.TextWatcher

► Known Indirect Subclasses
[AbsListView](#), [ExpandableListView](#), [GridView](#), [ListView](#), [PasswordTransformationMethod](#), [PhoneNumberFormattingTextWatcher](#)

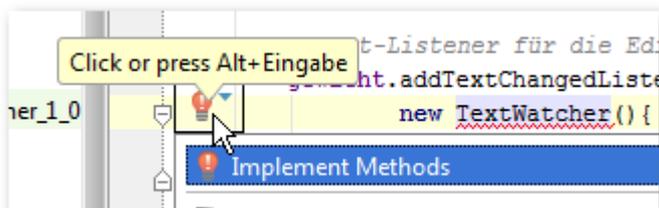
Class Overview

When an object of a type is attached to an Editable, its methods will be called when the text is changed.

```
new TextWatcher() {
```

den Listener hinzu. Als Parameter wird ein neu erzeugtes TextWatcher-Objekt übergeben.

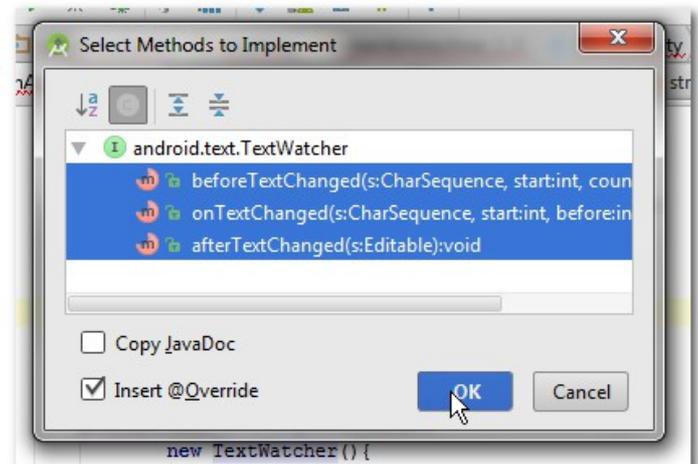
Implementieren Sie den Methodenaufruf wie nebenstehend angezeigt.



TextWatcher-Methoden-Deklaration einfügen.

Klicken Sie auf den Klassennamen TextWatcher. Mit einem Klick auf die kleine rote

Glühbirne am linken Rand und der Tastenkombination ALT+ Eingabe (Enter) werden die fehlenden Methoden implementiert.



```
public void onTextChanged(CharSequence s, int start, int before, int count)
```

Die Methode wird aufgerufen, um uns Veränderungen innerhalb der eingegebenen Zeichenkette „s“ anzuzeigen.

```
public void beforeTextChanged(CharSequence s, int
```

TextWatcher-Methoden implementieren.

Dieses Objekt bringt von sich aus drei Verhaltensweisen (Methoden) mit sich. Diese Methoden sind nun deklariert. Die Implementierung müssen wir bei Bedarf selbst vornehmen.

start, int count, int after)

Die Methode wird aufgerufen, um uns vorab über die Veränderungen innerhalb der eingegebenen Zeichenkette „s“ zu informieren.

```
public void afterTextChanged(Editable s)
```

Die Methode wird aufgerufen, um über die Veränderungen innerhalb der eingegebenen Zeichenkette „s“ zu informieren, nachdem sie bereits vorgenommen wurde.

@Override

Der Vermerk signalisiert uns, dass es sich um eine vererbte Methode handelt. Wir werden diese Methoden bei Bedarf überschreiben.

Implementierung:

Wir möchten den Fühler für die Zeichenkette aus der EditText-Komponente → gewicht freigeben, sodass wir jede Eingabeänderung nachträglich angezeigt bekommen.

Dafür implementieren wir die Methode wie folgt:

```
public void afterTextChanged(Editable s){
    gewicht.setEnabled(s.length() >= 0);
}
```

```
61     groesse.addTextChangedListener(
62
63         new TextWatcher() {
64             //Hier Quellcode ergänzen
65
66         });
```

Listener für EditText Komponenten.

Implementieren Sie wie zuvor für das → gewicht den vollständigen Listener für die → groesse.

```
berechnen.setOnClickListener(new View.OnClickListener() {
});
```

Listener für die Button Komponente.

Auch der Button braucht einen Fühler der Aktivitäten registriert.

Wählen Sie im Dropdown-Menü die Option

OnClickListener{...}

[View.OnClickListener](#)

Ist eine Interface-Klasse. Ein Interface ist so etwas wie eine Vorlage. Verhaltensweisen die im Interface deklariert sind, sollten implementiert werden, da sie eine zwingende Verhaltensweise eines Objektes darstellen.

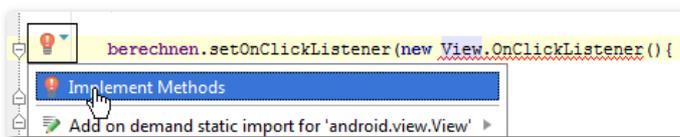
Auszug der API:

```
public static interface
View.OnClickListener

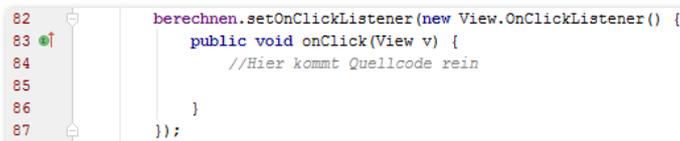
android.view.View.OnClickListener
▶ Known Indirect Subclasses
CharacterPickerDialog, KeyboardView, QuickContactBadge, SearchOrbView, SpeechOrbView

Class Overview

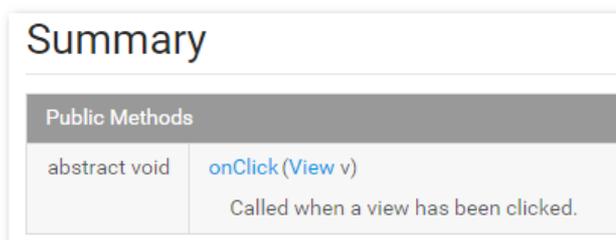
Interface definition for a callback to be invoked when a view is clicked.
```



Ergebnis:



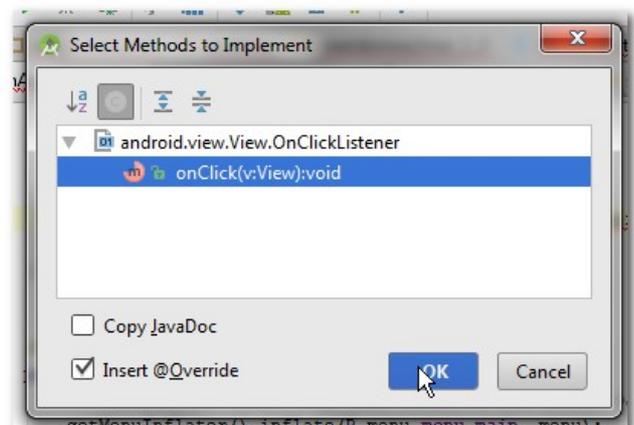
Auszug der API:



View.OnClickListener-Methoden deklarieren.

Gehen Sie auf die gleiche Weise vor wie zuvor für den TextWatcher.

Klicken Sie dazu auf den Klassennamen → View. Mit einem Klick auf die kleine rote Glühbirne am linken Rand und der Tastenkombination ALT+ Eingabe (Enter) werden die fehlenden Methoden implementiert.



Die Implementierung der Methode:

```

public void onClick(View v){
    /*hier fehlt Quellcode*/
}

```

Wir wenden bei der Umsetzung drei weitere Prinzipien der Softwareentwicklung an. Unser Fokus: Die Prinzipien „Zerlegung“ und „Wiederverwendung“.

Zerlegung:

Ist eine der wichtigsten Hilfen in der Informatik bei der Lösung komplexer Probleme. Man unterteilt große, komplexe Probleme in kleine, strukturierte Teilprobleme (→ Hilfsmethoden) und setzt diese in Quellcode um.

Wenn alle Teilprobleme umgesetzt sind, ist damit auch das große, komplexe Problem gelöst. → devide

```

if(v == berechnen){
    //Eingabe

    //Verarbeitung

    //Ausgabe
}else{
    finish();
}

```

	<p>and conquer (→ teile und herrsche)</p> <p>Wiederverwendung: Aus der o.g. Zerlegung ergibt sich ein weiterer Vorteil. Die Auslagerung von Quellcode in Methoden und Hilfsmethoden ermöglicht die Wiederverwendung des Quellcodes an anderer Stelle.</p> <p>Wir zerlegen also im ersten Schritt unser logisches Problemchen:</p> <p>Für den Fall, dass die Schaltfläche → berechnen angeklickt wurde, soll wie folgt vorgegangen werden:</p> <p>//Eingabe</p> <ol style="list-style-type: none"> 1. Lesen des → gewichts und Übergabe des → gewichts an das Objekt der Fachklasse. 2. Lesen der → groesse und Übergabe der → groesse an das Objekt der Fachklasse. <p>//Verarbeitung</p> <ol style="list-style-type: none"> 3. Berechnung des → bmis am Objekt der Fachklasse. <p>//Ausgabe</p> <ol style="list-style-type: none"> 4. Ermittlung des berechneten → bmis aus dem Objekt der Fachklassen und Anzeige des Wertes auf der Benutzeroberfläche im Ergebnisfeld. <p>Ansonsten soll die Aktivität geschlossen werden.</p>
<pre> 21 private Button berechne, 22 23 //Assoziation 24 private Bmirechner derRechner = new Bmirechner(); 25 </pre>	<p><i>Assoziation. MainActivity → Bmirechner</i></p> <p>Alle Schritte im EVA-Prinzip erfolgen am Objekt der Fachklasse (Modell).</p> <p>Damit brauchen wir in unserer Activity ein Bmirechner-Objekt das wir nutzen können.</p> <p>Deklarieren und initialisieren Sie dieses Objekt unterhalb der bereits deklarierten Komponenten in der Klasse → MainActivity.java ganz oben.</p>

Lese-Methode für das Gewicht:

```

128 //Meine Hilfsmethoden
129 private void leseGewicht(){
130     double mGewicht =
131         Double.valueOf(gewicht.getText().toString());
132     derRechner.setGewicht(mGewicht);
133 }

```

Erläuterung: *Von Innen nach Außen*

`gewicht.getText().toString()`

Der Wert für das → `gewicht` wird ermittelt und in einen String umgewandelt.

`Double.valueOf(...);`

Mit Hilfe der Wrapper-Klasse → `Double` wird sichergestellt, dass das Parameterattribut in einen `Double` umgewandelt wird.

`double mGewicht =`

Der umgewandelte Wert wird einem lokalen Attribut → `mGewicht` zugewiesen.

`derRechner.setGewicht(mGewicht);`

Übermitteln des Wertes an das Objekt der Fachklasse.

EVA-Prinzip. Die Eingabe.

Wir implementieren dazu die Lese-Methoden am unteren Rand der MainActivity-Klasse. Erzeugen Sie einen Kommentar

```

---
127
128 //Meine Hilfsmethoden
129

```

damit Sie die Methoden künftig schnell finden.

Implementieren Sie die Lese-Methode für das Gewicht, wie nebenstehend angezeigt.

Erzeugen Sie dann die Lese-Methode für die Größe nach dem gleichen Muster.

Rufen Sie dann an entsprechender Stelle der `onClick(View v)`-Methode, die Lese-Methoden auf.

Methodenaufrufe in der `onClick(View v)`-Methode

```

83 ↑
84
85 public void onClick(View v) {
86     //Hier kommt Quellcode rein
87     if (v == berechnen) {
88         //Eingabe
89         leseGewicht();
90         leseGroesse();
91     }
92 }

```

```

90 //Verarbeitung
91 derRechner.berechnen();
92

```

EVA-Prinzip. Die Verarbeitung.

Die Anweisung für die Berechnung entspricht genau einer Zeile. Am Objekt der Fachklasse wird dazu die Methode → `berechnen()` aufgerufen.

Implementieren Sie den Methodenaufruf an entsprechender Stelle der `onClick(View v)`-Methode.

Schreibe-Methode für das Ergebnis:

```

141 private void schreibeErgebnis(){
142     bmi.setText(String.valueOf(
143         f.format(derRechner.getBmi())));
144 }

```

Erläuterung: *Von Innen nach Außen*

`derRechner.getBmi()`

Ermitteln des berechneten Wertes aus dem Objekt der Fachklasse.

`f.format(...)`

Am Objekt `f` der Klasse `DecimalFormat` wird der ermittelte Wert anschließend formatiert.*

`String.valueOf(...)`

Dann wird der Wert in einen String (Zeichenkette) umgewandelt.

`bmi.setText(...);`

Der Wert wird in das Ergebnisfeld (TextView) der Benutzeroberfläche übermittelt.

```

*private DecimalFormat f =
    new DecimalFormat("#0.00");

```

EVA-Prinzip. Die Ausgabe.

Wir schreiben die benötigten Anweisungen zur Ermittlung und Ausgabe des Ergebnisses in die Hilfsmethode → `schreibeErgebnis()`.

Implementieren Sie die Hilfsmethode → `schreibeErgebnis()`.

Rufen Sie dann an entsprechender Stelle der `onClick(View v)`-Methode, die `schreibe`-Methoden auf.

Methodenaufwurf in der `onClick(View v)`-Methode:

```

93 //Ausgabe
94 schreibeErgebnis();

```



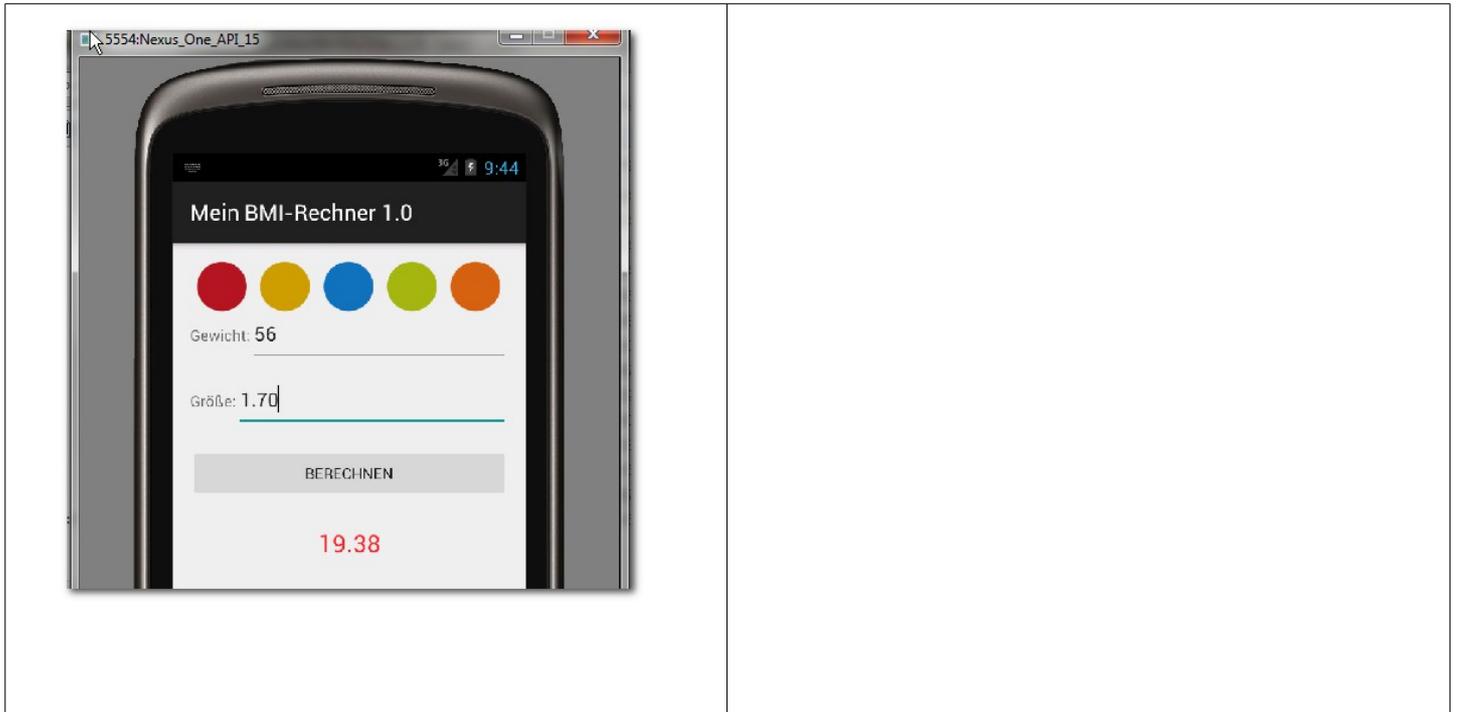
Prototyp testen.

So nun sollte unser kleiner, einfacher BMI-Rechner funktionieren. Nutzen Sie nun erneut den Emulator, dieses mal um die Funktionalität „berechnen“ zu testen.

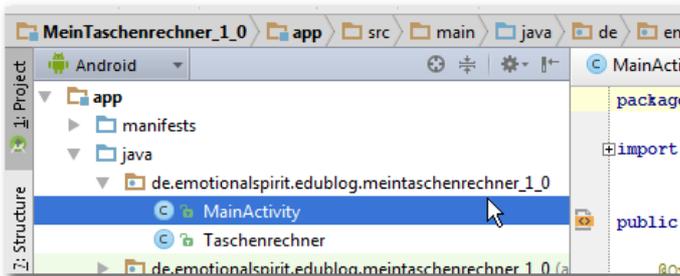
Klicken Sie auf den grünen Pfeil in der Symbolleiste oberhalb des Designers.

Gratulation!

Ihre erste kleine App ist funktionstüchtig.



5.2 Ereignissteuerung des Taschenrechners 1.0



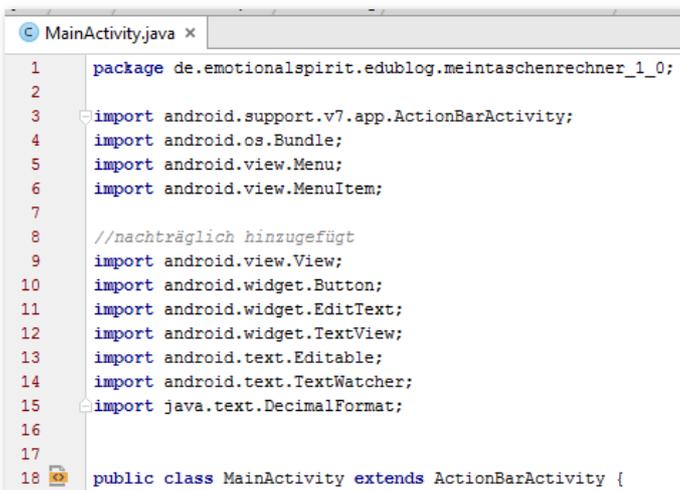
Öffnen Sie die Klasse MainActivity.java

Activity:

Bei Anwendungen auf Android Betriebssystemen erfolgt die Zerlegung aufgabenorientiert.

Konkret bedeutet das, dass der Quellcode für die Steuerung einer Funktionalität in eine Activity-Klasse ausgelagert wird.

Vielfach erkennt man die Aktivitäten (Activities) schon auf der Benutzeroberfläche, denn u.a. repräsentieren Schaltflächen solche Funktionalitäten.



Pakete und Importe.

Zeile 1 beinhaltet die Angabe des Package. Die Angabe setzt sich zusammen aus den eingangs definierten Projekteigenschaften (→ Domain und → App name).

Die Klasse → MainActivity stellt eine Erweiterung der Klasse → ActionBarActivity* dar.

Im Gegensatz zu anderen Java-Anwendungen benötigen Android Apps die MainActivity, um eine Instanz der Anwendung zu erzeugen, außerdem stellt sie den Lebenszyklus der Instanz sicher und ergreift ggf. alle lebenserhaltenden Maßnahmen. Im Prinzip übernimmt das Objekt der MainActivity-Klasse u.a. die Funktionalität der Main-Methode einer konventionellen Java-Anwendung.

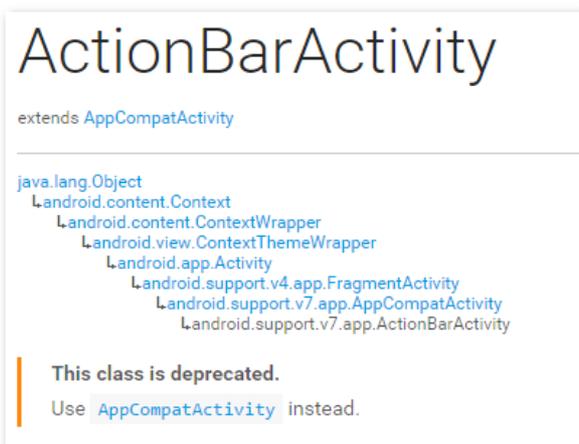
Die vererbten standardmäßig vorhandenen Verhaltensweisen (Methoden) einer → Activity erfordern die im oberen Teil der Klasse angegebenen Import-Anweisungen der Klassen → ActionBarActivity, → Bundle, → Menu und → MenuItem.

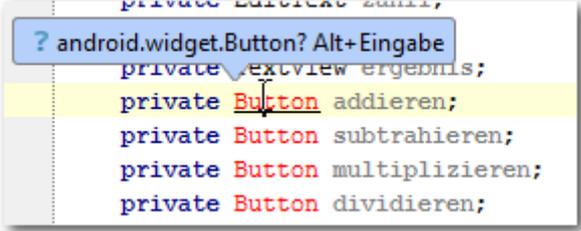
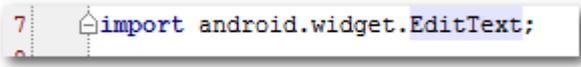
Die nachträglich hinzugefügten import-Anweisungen ergeben sich aus den nun folgenden notwendigen Implementierungen.

Aktuelle Hinweise – Ereignissteuerung

Die MainActivity erbt zwischenzeitlich standardmäßig von der Klasse AppCompatActivity:

→ ActionBarActivity*



	<p>MainActivity extends AppCompatActivity</p> <p>Bei den meisten älteren Projekten erbt die MainActivity von der ActionBarActivity</p> <p>MainActivity extends ActionBarActivity</p> <p>Die Verwendung der Klasse ActionBarActivity ist hinfällig (depreceated).</p> <p>Eine Anpassung älterer Projekte an die neue Architektur ist empfehlenswert.</p>
<pre> 11 public class MainActivity extends ActionBarActivity { 12 //Komponenten deklarieren 13 private EditText zahl1; 14 private EditText zahl2; 15 private TextView ergebnis; 16 private Button addieren; 17 private Button subtrahieren; 18 private Button multiplizieren; 19 private Button dividieren; 20 </pre>	<p><i>Komponenten deklarieren.</i></p> <p>Deklarieren Sie im Anschluss an die Klassende-klaration die Komponenten der Benutzer-oberfläche.</p>
<pre> 13 //Komponenten deklarieren 14 private EditText zahl1; 15 private EditText zahl2; 16 private TextView ergebnis; 17 private Button addieren; 18 private Button subtrahieren; 19 private Button multiplizieren; 20 private Button dividieren; </pre>	<p></p> <p>Klicken Sie auf die rot gekennzeichneten Klassen- namen für die Komponente und folgen Sie der Empfehlung mit der Tastenkombination ALT + Eingabe (Enter) die Klasse zu importieren:</p> <p></p>
<pre> 34 @Override 35 protected void onCreate(Bundle savedInstanceState) { 36 super.onCreate(savedInstanceState); 37 setContentView(R.layout.activity_main); 38 } </pre>	<p><i>Die onCreate-Methode.</i></p> <p>In der onCreate-Methode sollte das beim Auf- ruf des Activity-Objektes benötigte Layout (XML-Datei) übermitteln und in einem Objekt- baum entfalten.</p>

	<p>Genau das geschieht mit dem Methodenaufruf</p> <pre>setContentView(...)</pre> <p>R ist eine Klasse deren Aufgabe es ist, alle Elemente der Layouts und anderer XML-Dateien zu verwalten, u.a. um diese in Java verfügbar zu machen.</p>
<pre> 34 @Override 35 protected void onCreate(Bundle savedInstanceState) { 36 super.onCreate(savedInstanceState); 37 setContentView(R.layout.activity_main); 38 39 //Initialisierung der Komponenten 40 zahl1 = (EditText) findViewById(R.id.etZahl1); 41 zahl2 = (EditText) findViewById(R.id.etZahl2); 42 ergebnis = (TextView) findViewById(R.id.tvErgebnis); 43 addieren = (Button) findViewById(R.id.btAddieren); 44 subtrahieren = (Button) findViewById(R.id.btSubtrahieren); 45 multiplizieren = (Button) findViewById(R.id.btMultiplizieren); 46 dividieren = (Button) findViewById(R.id.btDividieren); 47 </pre>	<p><i>Ausstattung der onCreate-Methode.</i></p> <p>Wir müssen sicherstellen, dass Komponenten, deren Inhalte gelesen bzw. in die geschrieben werden soll, zuvor initialisiert werden. Wir ergänzen dazu den Quellcode, wie nebenstehend angezeigt.</p> <p>Erklärung:</p> <pre> 39 //Initialisierung der Komponenten 40 zahl1 = (EditText) findViewById(R.id.etZahl1); </pre> <ul style="list-style-type: none"> • zahl1: Ist ein Klassenattribut der Activity-Klasse vom Typ EditText (siehe Deklaration). • (EditText): Der Cast stellt sicher, dass die zugewiesene Komponente dem Typ entspricht. • findViewById(int) Sucht den Parameterwert anhand der id. Als Parameter wird ein int-Wert erwartet. • R.id.tvErgebnis R liefert zum String tvErgebnis den entsprechenden int-Wert zurück. Den entsprechenden Schlüsselwert.
<pre> 39 zahl1.addTextChangedListener(40 new TextWatcher() { 41 42 } 43 44); </pre>	<p><i>Listener in der onCreate-Methode.</i></p> <p>Ein Listener ist wie ein Fühler der Veränderungen auf der Benutzeroberfläche registriert und in Form eines Impulses an das System weiterreicht.</p> <p>Wir fügen dem editierbaren Objekt → zahl1 mit dem Methodenaufruf</p> <pre>zahl1.addTextChangedListener(TextWatcher watcher)</pre>

Auszug aus der API:

public interface Summary: Methods | [Expand All]
Added in API level 1

TextWatcher

implements [NoCopySpan](#)

android.text.TextWatcher

► Known Indirect Subclasses

[AbsListView](#), [ExpandableListView](#), [GridView](#), [ListView](#), [PasswordTransformationMethod](#), [PhoneNumberFormattingTextWatcher](#)

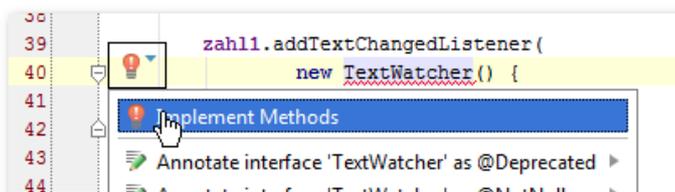
Class Overview

When an object of a type is attached to an Editable, its methods will be called when the text is changed.

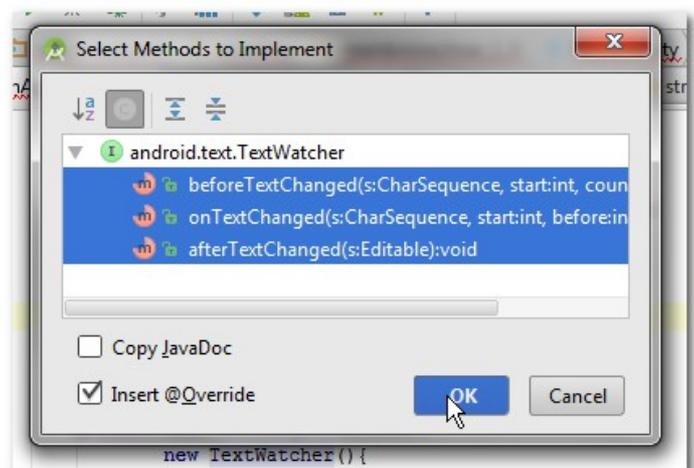
```
new TextWatcher() {
```

den Listener hinzu. Als Parameter wird ein neu erzeugtes TextWatcher-Objekt übergeben.

Implementieren Sie den Methodenaufruf wie nebenstehend angezeigt.

*TextWatcher-Methoden-Deklaration einfügen.*

Klicken Sie auf den Klassennamen TextWatcher. Mit einem Klick auf die kleine rote Glühbirne am linken Rand und der Tastenkombination ALT+ Eingabe (Enter) werden die fehlenden Methoden implementiert.



```
public void onTextChanged(CharSequence s, int start, int before, int count)
```

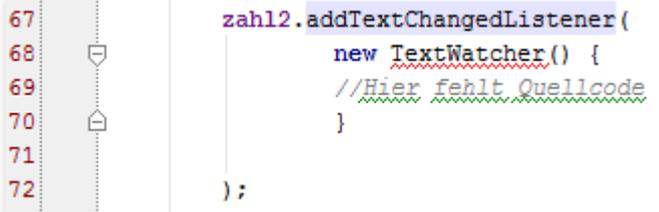
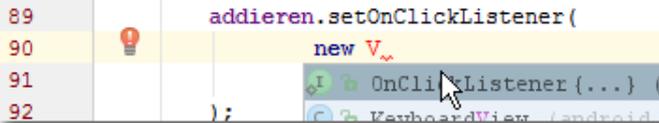
Die Methode wird aufgerufen, um uns Veränderungen innerhalb der eingegebenen Zeichenkette „s“ anzuzeigen.

```
public void beforeTextChanged(CharSequence s, int start, int count, int after)
```

TextWatcher-Methoden implementieren.

Dieses Objekt bringt von sich aus drei Verhaltensweisen (Methoden) mit sich. Diese Methoden sind nun deklariert. Die Implementierung müssen wir bei Bedarf selbst vornehmen.

@Override

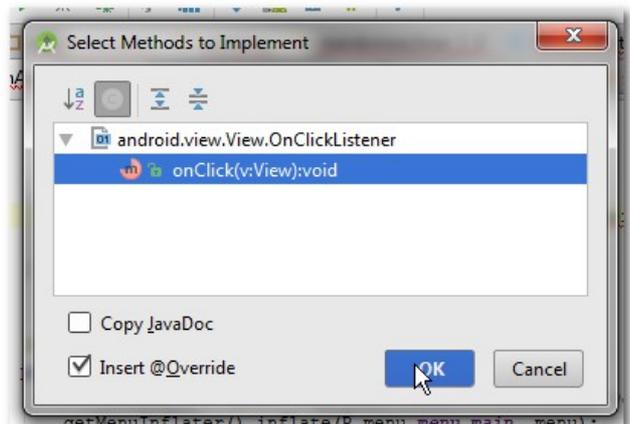
<p>Die Methode wird aufgerufen, um uns vorab über die Veränderungen innerhalb der eingegebenen Zeichenkette „s“ zu informieren.</p> <pre>public void afterTextChanged(Editable s)</pre> <p>Die Methode wird aufgerufen, um über die Veränderungen innerhalb der eingegebenen Zeichenkette „s“ zu informieren, nachdem sie bereits vorgenommen wurde.</p>	<p>Der Vermerk signalisiert uns, dass es sich um eine vererbte Methode handelt. Wir werden diese Methoden bei Bedarf überschreiben.</p> <p>Wir möchten den Fühler für die Zeichenkette aus der EditText-Komponente → zahl1 freigeben, sodass wir jede Eingabeänderung nachträglich angezeigt bekommen.</p> <p>Dafür implementieren wir die Methode wie folgt:</p> <pre>public void afterTextChanged(Editable s){ zahl1.setEnabled(s.length() >= 0); }</pre>
	<p><i>Listener für EditText Komponenten.</i></p> <p>Implementieren Sie wie zuvor für das → zahl1 den vollständigen Listener für die → zahl2.</p>
	<p><i>Listener für die Button Komponente.</i></p> <p>Auch der Button braucht einen Fühler der Aktivitäten registriert.</p> <p>Wählen Sie im Dropdown-Menü die Option</p> <pre>OnClickListener{...}</pre> <p>View.OnClickListener Ist eine Interface-Klasse. Ein Interface ist so etwas wie eine Vorlage. Eigenschaften und Verhaltensweisen die im Interface deklariert sind, müssen implementiert werden, da sie eine zwingende Verhaltensweise eines Objektes darstellen.</p>
<p>Ergebnis:</p>	<p><i>View.OnClickListener-Methode deklarieren.</i></p> <p>Gehen Sie auf die gleiche Weise vor wie zuvor für den TextWatcher.</p>

```

89      addieren.setOnClickListener(
90          new View.OnClickListener() {
91              @Override
92              public void onClick(View v) {
93              }
94          }
95      );
96
97

```

Alternativ klicken Sie auf den Klassennamen View. Mit einem Klick auf die kleine rote Glühbirne am linken Rand und der Tastenkombination ALT+ Eingabe (Enter) werden die fehlenden Methoden implementiert.



Implementieren Sie die Listener auch für die übrigen Schaltflächen.

```

@Override
public void onClick(View v) {
    if( v == addieren){
        //Eingabe

        //Verarbeitung

        //Ausgabe
    }else{
        finish();
    }
}

```

Die Implementierung der Methode:

```

public void onClick(View v){
    /*hier fehlt Quellcode*/
}

```

Wir wenden bei der Umsetzung drei weitere Prinzipien der Softwareentwicklung an. Unser Fokus: Die Prinzipien „Zerlegung“ und „Wiederverwendung“.

Zerlegung:

Ist eine der wichtigsten Hilfen in der Informatik bei der Lösung komplexer Probleme. Man unterteilt große, komplexe Probleme in kleine, strukturierte Teilprobleme (→ Hilfsmethoden) und setzt diese in Quellcode um.

Wenn alle Teilprobleme umgesetzt sind, ist damit auch das große, komplexe Problem gelöst. → divide and conquer (→ teile und herrsche)

Wiederverwendung:

Aus der o.g. Zerlegung ergibt sich ein weiterer Vorteil. Die Auslagerung von Quellcode in Methoden und Hilfsmethoden ermöglicht die Wiederverwen-

	<p>derung des Quellcodes an anderer Stelle.</p> <p>Wir zerlegen also im ersten Schritt unser logisches Problemchen:</p> <p>Für den Fall, dass die Schaltfläche → berechnen angeklickt wurde, soll wie folgt vorgegangen werden:</p> <p>//Eingabe</p> <ol style="list-style-type: none"> 1. Lesen des → zahl1 und Übergabe des → zahl1 an das Objekt der Fachklasse. 2. Lesen der → zahl2 und Übergabe der → zahl2 an das Objekt der Fachklasse. <p>//Verarbeitung</p> <ol style="list-style-type: none"> 3. Berechnung des → ergebnisses am Objekt der Fachklasse. <p>//Ausgabe</p> <ol style="list-style-type: none"> 4. Ermittlung des berechneten → ergebnisses aus dem Objekt der Fachklassen und Anzeige des Wertes auf der Benutzeroberfläche im Ergebnisfeld. <p>Ansonsten soll die Aktivität beendet werden.</p>
<pre> 25 //Assoziation 26 private Taschenrechner derRechner = new Taschenrechner(); 27 </pre>	<p><i>Assoziation. MainActivity → Taschenrechner</i></p> <p>Alle Schritte im EVA-Prinzip erfolgen am Objekt der Fachklasse (Modell).</p> <p>Damit brauchen wir in unserer Activity ein Taschenrechner-Objekt das wir nutzen können.</p> <p>Deklarieren und initialisieren Sie dieses Objekt unterhalb der bereits deklarierten Komponenten in der Klasse → MainActivity.java ganz oben.</p>
<p>Lese-Methode für das Zahl1:</p> <pre> 208 private void leseZahl1(){ 209 double mZahl1 = 210 Double.valueOf(211 zahl1.getText().toString()); 212 derRechner.setZahl1(mZahl1); 213 } </pre>	<p><i>EVA-Prinzip. Die Eingabe.</i></p> <p>Wir implementieren dazu die Lese-Methoden am unteren Rand der MainActivity-Klasse. Erzeugen Sie einen Kommentar</p>

Erläuterung: Von Innen nach Außen

```
zahl1.getText().toString()
```

Der Wert für das → gewicht wird ermittelt und in einen String umgewandelt.

```
Double.valueOf(...);
```

Mit Hilfe der Wapper-Klasse → Double wird sichergestellt, dass das Parameterattribut in einen Double umgewandelt wird.

```
double mZahl1 =
```

Der umgewandelte Wert wird einem lokalen Attribut → mZahl1 zugewiesen.

```
derRechner.setZahl1(mZahl1);
```

Übermitteln des Wertes an das Objekt der Fachklasse

```
186
187 //Meine Hilfsmethoden
188
```

damit Sie die Methoden künftig schnell finden.

Implementieren Sie die Lese-Methode für die → zahl1, wie nebenstehend angezeigt.

Erzeugen Sie dann die Lese-Methode für die → zahl1 nach dem gleichen Muster.

Rufen Sie dann an entsprechender Stelle der onClick(View v)-Methode, die Lese-Methoden auf.

Methodenaufrufe in der onClick(View v)-Methode

```
92
93
94
95
96
97
public void onClick(View v) {
    if (v == addieren){
        //Eingabe
        leseZahl1();
        leseZahl2();
    }
}
```

```
98
99 //Verarbeitung
100 derRechner.addieren();
```

EVA-Prinzip. Die Verarbeitung.

Die Anweisung für die Berechnung entspricht genau einer Zeile. Am Objekt der Fachklasse wird dazu die Methode → addieren() aufgerufen.

Implementieren Sie den Methodenaufruf an entsprechender Stelle der onClick(View v)-Methode.

Schreibe-Methode für das Ergebnis:

```
220 private void schreibeErgebnis(){
221     ergebnis.setText(
222         String.valueOf(
223             f.format(
224                 derRechner.getErgebnis()));
225 }
```

Erläuterung: Von Innen nach Außen**EVA-Prinzip. Die Ausgabe.**

Wir schreiben die benötigten Anweisungen zur Ermittlung und Ausgabe des Ergebnisses in die Hilfsmethode → schreibeErgebnis().

Implementieren Sie die Hilfsmethode → schreibeErgebnis().

`derRechner.getErgebnis()`
Ermitteln des berechneten Wertes aus dem Objekt der Fachklasse.

`f.format(...)`
Am Objekt `f` der Klasse `DecimalFormat` wird der ermittelte Wert anschließend formatiert.*

`String.valueOf(...)`
Dann wird der Wert in einen String (Zeichenkette) umgewandelt.

`ergebnis.setText(...);`
Der Wert wird in das Ergebnisfeld (TextView) der Benutzeroberfläche übermittelt.

```
* private DecimalFormat f =
    new DecimalFormat("#0.00");
```

Rufen Sie dann an entsprechender Stelle der `onClick(View v)`-Methode, die schreibe-Methode auf.

Methodenaufruf in der `onClick(View v)`-Methode:

```
120 //Ausgabe
121 schreibeErgebnis();
```

```
89 addieren.setOnClickListener(
90     new View.OnClickListener() {
91         @Override
92         public void onClick(View v) {
93             if (v == addieren){
94                 //Eingabe
95                 leseZahl1();
96                 leseZahl2();
97
98                 //Verarbeitung
99                 derRechner.addieren();
100
101                 //Ausgabe
102                 schreibeErgebnis();
103             }else{
104                 finish();
105             }
106         }
107     }
108 );
```

Deklarieren und implementieren von Ereignisgesteuerten `onClick(View v)`-Methode.

Alle Schaltflächen sollen funktionieren!

Deklarieren und implementieren Sie auf gleiche Weise die Funktionalität für die restlichen Schaltflächen.



Prototyp testen.

So nun sollte unser kleiner, einfacher Taschenrechner funktionieren.

Klicken Sie auf den grünen Pfeil in der Symbolleiste oberhalb des Designers.



Gratulation!

Ihre zweite kleine App ist funktionstüchtig.

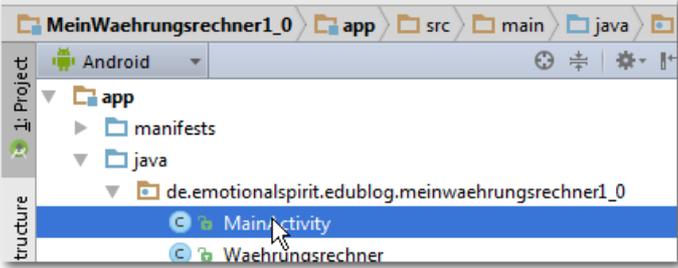
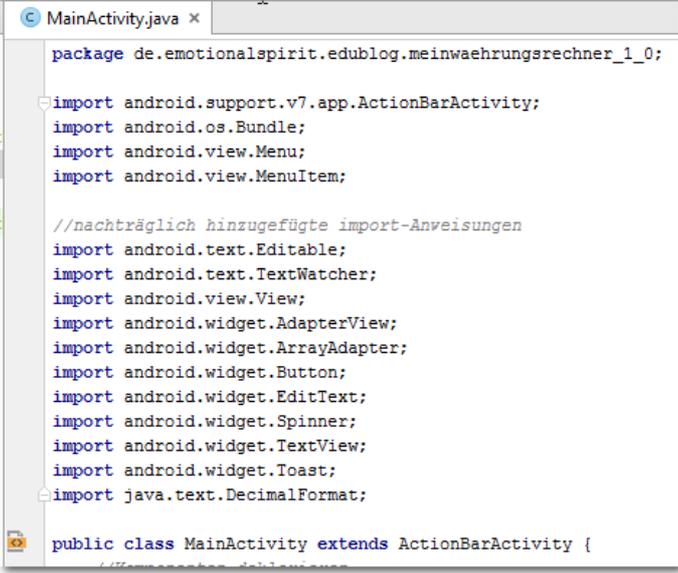
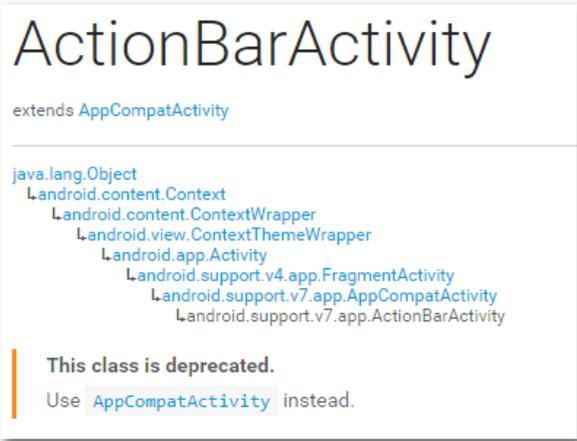
Gut!
Aber es geht immer besser!

Optimierung.

Sie haben vielleicht schon gemerkt, dass sich der Quellcode in der MainActivity-Klasse an mehreren Stellen unnötiger Weise wiederholt.

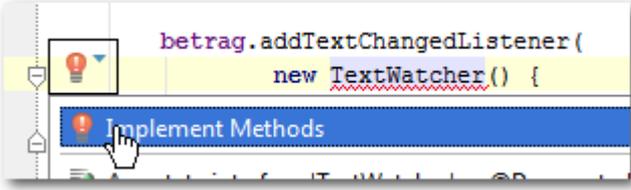
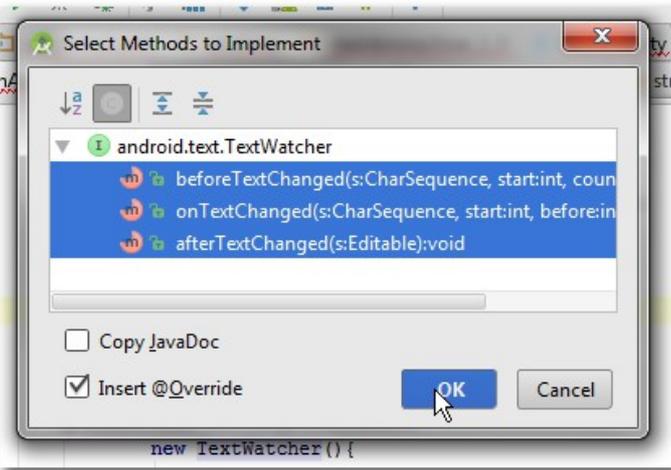
Im letzten Kapitel werden wir u.a. einige Maßnahmen ergreifen, um den Quellcode effizienter zu gestalten!

5.3 Ereignissteuerung des Währungsrechners 1.0

	<p>Öffnen Sie die Klasse MainActivity.java</p>
 <pre> package de.emotionalspirit.edublog.meinwaehrungsrechner_1_0; import android.support.v7.app.AppCompatActivity; import android.os.Bundle; import android.view.Menu; import android.view.MenuItem; //nachträglich hinzugefügte import-Anweisungen import android.text.Editable; import android.text.TextWatcher; import android.view.View; import android.widget.AdapterView; import android.widget.AdapterView.OnItemClickListener; import android.widget.ArrayAdapter; import android.widget.Button; import android.widget.EditText; import android.widget.Spinner; import android.widget.TextView; import android.widget.Toast; import java.text.DecimalFormat; public class MainActivity extends AppCompatActivity { </pre>	<p><i>Pakete und Importe.</i></p> <p>Zeile 1 beinhaltet die Angabe des Package. Die Angabe setzt sich zusammen aus den eingangs definierten Projekteigenschaften (→ Domain und → App name).</p> <p>Die Klasse → MainActivity stellt eine Erweiterung der Klasse → AppCompatActivity* dar.</p> <p>Im Gegensatz zu anderen Java-Anwendungen benötigen Android Apps die MainActivity, um eine Instanz der Anwendung zu erzeugen, außerdem stellt sie den Lebenszyklus der Instanz sicher und ergreift ggf. alle lebenserhaltenden Maßnahmen. Im Prinzip übernimmt das Objekt der MainActivity-Klasse u.a. die Funktionalität der Main-Methode einer konventionellen Java-Anwendung.</p>
<p>→ AppCompatActivity*</p> 	<p>Die vererbten standardmäßig vorhandenen Verhaltensweisen (Methoden) einer → Activity erfordern die im oberen Teil der Klasse angegebenen Import-Anweisungen der Klassen → AppCompatActivity, → Bundle, → Menu und → MenuItem.</p> <p>Die nachträglich hinzugefügten import-Anweisungen ergeben sich aus den nun folgenden notwendigen Implementierungen.</p> <p><i>Aktuelle Hinweise – Ereignissteuerung</i></p> <p>Die MainActivity erbt zwischenzeitlich stan-</p>

	<p>dardmäßig von der Klasse AppCompatActivity:</p> <p>MainActivity extends AppCompatActivity</p> <p>Bei den meisten älteren Projekten erbt die MainActivity von der ActionBarActivity</p> <p>MainActivity extends ActionBarActivity</p> <p>Die Verwendung der Klasse ActionBarActivity ist hinfällig (deprecated).</p> <p>Eine Anpassung älterer Projekte an die neue Architektur ist empfehlenswert.</p>
<pre> 13 public class MainActivity extends AppCompatActivity { 14 //Komponenten deklarieren 15 private EditText betrag; 16 private Spinner von; 17 private Spinner in; 18 private TextView ergebnis; 19 private Button umrechnen; </pre>	<p><i>Komponenten deklarieren.</i></p> <p>Deklarieren Sie im Anschluss an die Klassendeclaration die Komponenten der Benutzeroberfläche.</p>
<pre> 14 //Komponenten deklarieren 15 private EditText betrag; 16 private Spinner von; 17 private Spinner in; 18 private TextView ergebnis; 19 private Button umrechnen; </pre>	<pre> 10 ? android.widget.EditText? Alt+Eingabe 11 //Komponenten deklarieren 12 private EditText betrag; 13 private Spinner von; 14 private Spinner in; 15 private TextView ergebnis; 16 private Button umrechnen; 17 </pre> <p>Klicken Sie auf die rot gekennzeichneten Klassennamen für die Komponente und folgen Sie der Empfehlung mit der Tastenkombination ALT + Eingabe (Enter) die Klasse zu importieren:</p> <pre> 7 import android.widget.EditText; </pre>
<pre> 22 @Override 23 protected void onCreate(Bundle savedInstanceState) { 24 super.onCreate(savedInstanceState); 25 setContentView(R.layout.activity_main); 26 } </pre>	<p><i>Die onCreate-Methode.</i></p> <p>In der onCreate-Methode sollte das beim Aufruf des Activity-Objektes benötigte Layout (XML-Datei) übermitteln und in einem Objektbaum entfalten.</p>

	<p>Genau das geschieht mit dem Methodenaufruf</p> <pre>setContentView(...)</pre> <p>R ist eine Klasse deren Aufgabe es ist, alle Elemente der Layouts und anderer XML-Dateien zu verwalten, u.a. um diese in Java verfügbar zu machen.</p>
<pre> 23 protected void onCreate(Bundle savedInstanceState) { 24 super.onCreate(savedInstanceState); 25 setContentView(R.layout.activity_main); 26 27 //Initialisierung der Komponenten 28 betrag = (EditText) findViewById(R.id.etBetrag); 29 von = (Spinner) findViewById(R.id.spVon); 30 in = (Spinner) findViewById(R.id.spIn); 31 ergebnis = (TextView) findViewById(R.id.tvErgebnis); 32 umrechnen = (Button) findViewById(R.id.btUmrechnen); </pre>	<p><i>Ausstattung der onCreate-Methode.</i></p> <p>Wir müssen sicherstellen, dass Komponenten, deren Inhalte gelesen bzw. in die geschrieben werden soll, zuvor initialisiert werden. Wir ergänzen dazu den Quellcode, wie nebenstehend angezeigt.</p> <p>Erklärung:</p> <pre> 28 betrag = (EditText) findViewById(R.id.etBetrag); </pre> <ul style="list-style-type: none"> • betrag: Ist ein Klassenattribut der Activity-Klasse vom Typ EditText (siehe Deklaration). • (EditText): Der Cast stellt sicher, dass die zugewiesene Komponente dem Typ entspricht. • findViewById(int) Sucht den Parameterwert anhand der id. Als Parameter wird ein int-Wert erwartet. • R.id.tvErgebnis R liefert zum String tvErgebnis den entsprechenden int-Wert zurück. Den entsprechenden Schlüsselwert.
<pre> 36 betrag.addTextChangedListener(37 new TextWatcher() { 38 39 } 40 41); </pre>	<p><i>Listener in der onCreate-Methode.</i></p> <p>Ein Listener ist wie ein Fühler der Veränderungen auf der Benutzeroberfläche registriert und in Form eines Impulses an das System weiterreicht.</p> <p>Wir fügen dem editierbaren Objekt → betrag mit dem Methodenaufruf</p> <pre>betrag.addTextChangedListener(TextWatcher watcher)</pre>

	<div data-bbox="1002 264 1315 331" style="border: 1px solid gray; padding: 5px; width: fit-content; margin: 0 auto;"> <pre>new TextWatcher() {</pre> </div> <p>den Listener hinzu. Als Parameter wird ein neu erzeugtes TextWatcher-Objekt übergeben.</p> <p>Implementieren Sie den Methodenaufruf wie nebenstehend angezeigt.</p>
	<p><i>TextWatcher-Methoden-Deklaration einfügen.</i></p> <p>Klicken Sie auf den Klassennamen TextWatcher. Mit einem Klick auf die kleine rote Glühbirne am linken Rand und der Tastenkombination ALT+ Eingabe (Enter) werden die fehlenden Methoden implementiert.</p> 
<pre>public void onTextChanged(CharSequence s, int start, int before, int count)</pre> <p>Die Methode wird aufgerufen, um uns Veränderungen innerhalb der eingegebenen Zeichenkette „s“ anzuzeigen.</p> <pre>public void beforeTextChanged(CharSequence s, int start, int count, int after)</pre> <p>Die Methode wird aufgerufen, um uns vorab über die Veränderungen innerhalb der eingegebenen Zeichenkette „s“ zu informieren.</p>	<p><i>TextWatcher-Methoden implementieren.</i></p> <p>Dieses Objekt bringt von sich aus drei Verhaltensweisen (Methoden) mit sich. Diese Methoden sind nun deklariert. Die Implementierung müssen wir bei Bedarf selbst vornehmen.</p> <p>@Override Der Vermerk signalisiert uns, dass es sich um eine vererbte Methode handelt. Wir werden diese Methoden bei Bedarf überschreiben.</p> <p>Wir möchten den Fühler für die Zeichenkette</p>

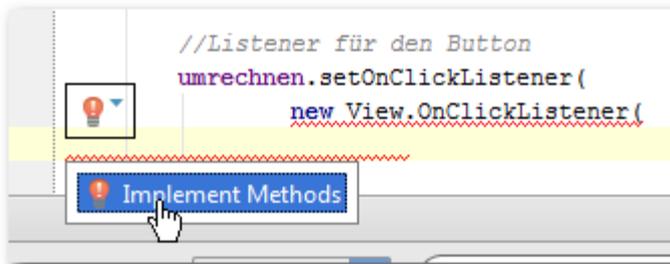
```
public void afterTextChanged(Editable s)
```

Die Methode wird aufgerufen, um über die Veränderungen innerhalb der eingegebenen Zeichenkette „s“ zu informieren, nachdem sie bereits vorgenommen wurde.

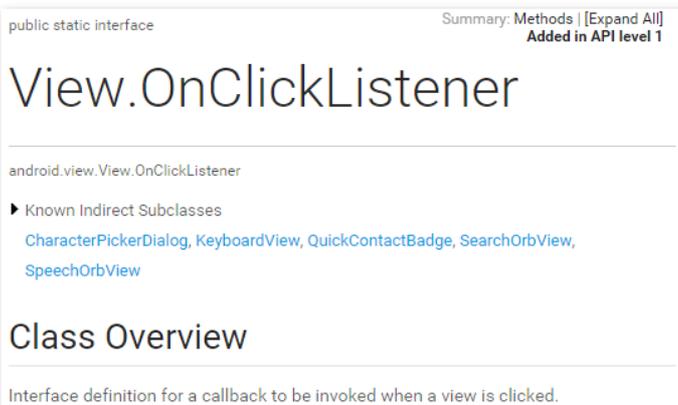
aus der EditText-Komponente → betrag freigeben, sodass wir jede Eingabeänderung nachträglich angezeigt bekommen.

Dafür implementieren wir die Methode wie folgt:

```
public void afterTextChanged(Editable s){
    betrag.setEnabled(s.length() >= 0);
}
```



Auszug der API:



Listener für die Button Komponente.

Auch der Button braucht einen Fühler der Aktivitäten registriert.

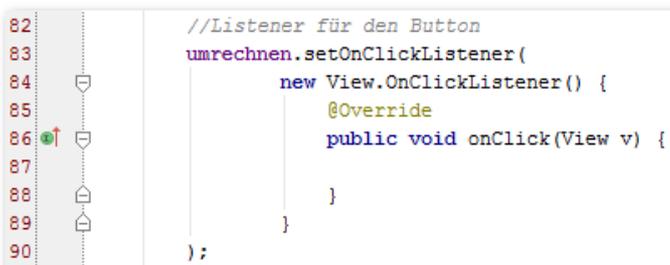
Wählen Sie im Dropdown-Menü die Option

```
OnClickListener{...}
```

[View.OnClickListener](#)

Ist eine Interface-Klasse. Ein Interface ist so etwas wie eine Vorlage. Eigenschaften und Verhaltensweisen die im Interface deklariert sind, müssen implementiert werden, da sie eine zwingende Verhaltensweise eines Objektes darstellen.

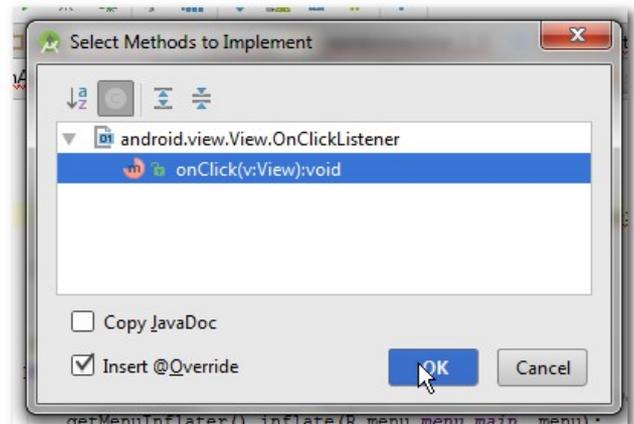
Ergebnis:



View.OnClickListener-Methode deklarieren.

Gehen Sie auf die gleiche Weise vor wie zuvor für den TextWatcher.

Alternativ klicken Sie auf den Klassennamen View. Mit einem Klick auf die kleine rote Glühbirne am linken Rand und der Tastenkombination ALT+ Eingabe (Enter) werden die fehlenden Methoden implementiert.



Implementieren Sie den Listener wie nebenstehend angezeigt.

```

95 //ADAPTER FÜR DIE SPINNER
96
97 /*Für spVon: Erzeuge einen ArrayAdapter
98 spVon_adapter der das String array von_array
99 und das standard spinner Layout nutzt*/
100 ArrayAdapter<CharSequence> spVon_adapter
101     = ArrayAdapter.createFromResource(this,
102         R.array.von_array,
103         android.R.layout.simple_spinner_item);
104
105 // Spezifiziere das Layout für das Auswahlmnü
106 spVon_adapter.setDropDownViewResource (
107     android.R.layout.simple_spinner_dropdown_item);
108
109 // Das Adapter-Objekt wird gesetzt
110 von.setAdapter(spVon_adapter);

```

Zeile 100:

```

ArrayAdapter<CharSequence>
spVon_adapter=
ArrayAdapter.createFromResource(this,
R.array.von_array,
android.R.layout.
    simple_spinner_item);

```

Zeile 106:

```

spVon_adapter.setDropDownViewResource (
    android.R.layout
        .simple_spinner_dropdown_item);

```

Zeile 110:

```

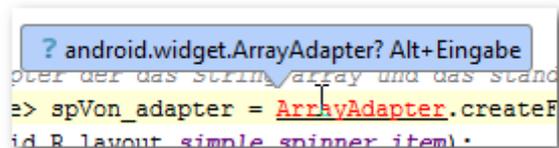
von.setAdapter(spVon_adapter);

```

Adapter für die Spinner-Komponenten.

Unterhalb der Listener-Aufrufe erzeugen wir die Array Adapter.

Importieren Sie der Tastenkombination ALT+Eingabe (Enter) die Klasse ArrayAdapter.



Für → spVon

Implementieren Sie den Adapter für den Spinner → spVon, wie nebenstehend angezeigt.

Für → spIn

Implementieren Sie auch den Adapter für den Spinner → spIn.

```

112     von.setOnItemSelectedListener(
113         new AdapterView.OnItemSelectedListener() {
114             public void onItemSelected(
115                 AdapterView<?> parent, View view,
116                 int position, long id) {
117                 showToast("Spinner1: position="
118                     + position + " id=" + id);
119             }
120
121             public void onNothingSelected(AdapterView<?> parent) {
122                 showToast("Spinner1: unselected");
123             }
124         });

```

Zeile 112:

```

von.setOnItemSelectedListener(
    new AdapterView.OnItemSelectedListener() {
        public void onItemSelected(
            AdapterView<?> parent, View view,
            int position, long id) {...}

        public void onNothingSelected(
            AdapterView<?> parent) {...}
    });

```

Listener für die Spinner-Komponenten.

Toasts

Erfüllen in der Android Anwendungen den Zweck klassischer PopUps. Man verwendet Toasts allerdings hauptsächlich zur Ausgabe von kleinen textuellen Hinweisen für den Benutzer.

Im nebenstehenden Beispiel werden sie zur Anzeige einer Testausgabe verwendet.

Zeile 117:

Anzeige der Position und Id des Spinners für den Fall, dass der Benutzer ein Objekt aus dem spinner (Drop-Down-Menü) → von gewählt hat.

```

showToast("Spinner1: position="
    + position + " id=" + id);

```

Zeile 122:

Anzeige der Meldung „unselected“ für den Fall, dass der Benutzer kein Objekt aus dem spinner (Drop-Down-Menü) → von gewählt hat.

```

showToast("Spinner1: unselected");

```

```

@Override
public void onClick(View v) {
    if(v == umrechnen){
        //Eingabe

        //Verarbeitung

        //Ausgabe

    }else{
        finish();
    }
}

```

Die Implementierung der Methode:

```

public void onClick(View v){
    /*hier fehlt Quellcode*/
}

```

Wir wenden bei der Umsetzung drei weitere Prinzipien an. Unser Fokus: Die Prinzipien „Zerlegung“, Kapselung und „Wiederverwendung“.

Wir Zerlegen also im ersten Schritt unser logisches Problemchen:

Für den Fall, dass die Schaltfläche → umrechnen angeklickt wurde soll, wie folgt vorgegangen werden:

//Eingabe

1. Lesen des → betrages und Übergabe des → betrages an das Objekt der Fachklasse
2. Lesen der Ausgangswährung (von) und Übergabe des Wertes → von an ein temporäres Attribut → mVon
3. Lesen der Zielwährung (in) und Übergabe des Wertes → in an ein temporäres

	<p>Attribut → mIn.</p> <p>//Verarbeitung</p> <p>4. Umrechnung des → ergebnisses am Objekt der Fachklasse. Dazu werden die Werte für → mVon und → mIn als Parameter übermittelt.</p> <p>//Ausgabe</p> <p>5. Ermittlung des berechneten → ergebnisses aus dem Objekt der Fachklassen und Anzeige des Wertes auf der Benutzeroberfläche im Ergebnisfeld.</p> <p>Ansonsten soll die Aktivität geschlossen werden.</p>
<pre>31 //Assoziation 32 private Waehrungsrechner derRechner = new Waehrungsrechner();</pre>	<p><i>Assoziation. MainActivity → Währungsrechner</i></p> <p>Die Verarbeitung der Eigenschaftswerte soll am Objekt der Fachklasse (Modell) erfolgen. Damit brauchen wir in unserer Activity eine Waehrungsrechner-Objekt das wir nutzen können.</p> <p>Deklarieren und initialisieren Sie dieses Objekt unterhalb der bereits deklarierten Komponenten in der Klasse → MainActivity.java.</p>
<p>Lese-Methode ohne Rückgabewert für den Betrag (EditText) :</p> <pre>private void leseBetrag(){ double mBetrag= Double.valueOf(betrag.getText().toString()); derRechner.setBetrag(mBetrag); }</pre> <p>Erläuterung: Von Innen nach Außen</p> <p><code>betrag.getText().toString()</code> Der Wert für den → betrag wird ermittelt und in einen String umgewandelt.</p> <p><code>Double.valueOf(...)</code> Mit Hilfe der Wrapper-Klasse → Double wird sichergestellt, dass das Parameterattribut in einen Double umgewandelt wird.</p> <p><code>double mBetrag =</code> Der umgewandelte Wert wird einem lokalen Attribut</p>	<p><i>EVA-Prinzip. Die Eingabe.</i></p> <p>Wir implementieren dazu die Lese-Methoden am unteren Rand der MainActivity-Klasse. Erzeugen Sie einen Kommentar</p> <pre>180 181 //Hilfsmethoden</pre> <p>damit Sie die Methoden künftig schnell finden.</p> <p>Implementieren Sie die Lese-Methode für den betrag, wie nebenstehend angezeigt.</p> <p>Erzeugen Sie dann die Lese-Methode für die Spinnerwerte (→ von und → in) nach dem gleichen Muster.</p> <p>Rufen Sie dann an entsprechender Stelle der</p>

→ mBetrag zugewiesen.

```
derRechner.setBetrag(mBetrag);
```

Übermitteln des Wertes an das Objekt der Fachklasse

Lese-Methode mit Rückgabewert für den Spinnerwert Ausgangswährung (von):

```
private String leseVon(){
    String mVon =
        von.getSelectedItemAt().toString().trim();
    return mVon;
}
```

Erläuterung: *Von Innen nach Außen*

`von.getSelectedItemAt().toString().trim();`
Der Wert für den ausgewählten Wert → von wird ermittelt, in einen String umgewandelt. Mit `trim()` werden vor- bzw. nachgelagerte Leerzeichen entfernt.

`String mVon =`
Der Wert wird einem lokalen Attribut → mVon zugewiesen.

`return mVon;`
Der Wert des lokalen Attributs wird zurückgegeben.

onClick(View v)-Methode, die Lese-Methoden auf.

Methodenaufrufe in der `onClick(View v)`-Methode

```
69 //Listener für den Button
70 umrechnen.setOnClickListener(
71     new View.OnClickListener() {
72         @Override
73         public void onClick(View v) {
74             if(v == umrechnen){
75                 //Eingabe
76                 leseBetrag();
77                 String mVon = leseVon();
78                 String mIn = leseIn();
```

EVA-Prinzip. Die Verarbeitung.

Die Anweisung für die Berechnung entspricht genau einer Zeile. Am Objekt der Fachklasse wird dazu die Methode `umrechnen(von, in)` aufgerufen.

Implementieren Sie den Methodenaufwurf an entsprechender Stelle der `onClick(View v)`-Methode.

```
81 //Verarbeitung
82 derRechner.umrechnen(mVon, mIn);
83
```

Schreibe-Methode für das Ergebnis:

EVA-Prinzip. Die Ausgabe.

Wir schreiben die benötigten Anweisungen zur Ermittlung und Ausgabe des Ergebnisses in die Hilfsmethode → `schreibeErgebnis()`.

Implementieren Sie die Hilfsmethode → `schreibeErgebnis()`.

```

200 private void schreibeErgebnis(){
201     String newline = "\n";
202     ergebnis.setText(
203         String.valueOf(
204             f.format(
205                 derRechner.getErgebnis())
206             + " "
207             + derRechner.getIn()
208             + newline + " Zum Wechselkurs von:"
209             + derRechner.getWechselkurs());
210     }

```

Erläuterung: *Von Innen nach Außen*

`String newline = "\n";`

Der reguläre Ausdruck für einen Zeilenumbruch wird einem lokalen Attribut → `newline` als String zugewiesen.

`derRechner.getErgebnis()`

Ermitteln des berechneten Wertes aus dem Objekt der Fachklasse.

`f.format(...)`

Am Objekt `f` der Klasse `DecimalFormat` wird der ermittelte Wert anschließend formatiert.*

`String.valueOf(...)`

Dann wird der Wert in einen String (Zeichenkette) umgewandelt.

`ergebnis.setText(...);`

Der Wert wird in das Ergebnisfeld (TextView) der Benutzeroberfläche übermittelt.

Strings mit dem „+“-Zeichen verketteten:

```

+ " "
+ derRechner.getIn()
+ newline + " Zum Wechselkurs von:"
+ derRechner.getWechselkurs();

```

```

*private DecimalFormat f =
    new DecimalFormat("#0.00");

```

Rufen Sie dann an entsprechender Stelle der `onClick(View v)`-Methode, die `schreibe`-Methoden auf.

Methodenaufwurf in der `onClick(View v)`-Methode:

```

120 //Ausgabe
121 schreibeErgebnis();

```

Gewünschte Ausgabe:

72.00 Britisches Pfund (GBP)
Zum Wechselkurs von:0.72085



Prototyp testen.

So nun sollte unser kleiner, einfacher Währungsrechner funktionieren.

Klicken Sie auf den grünen Pfeil in der Symbolleiste oberhalb des Designers.

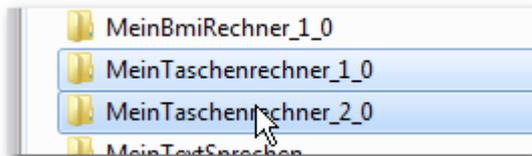
<p style="text-align: center;">Gut! Aber es geht immer besser!</p>	<p><i>Gestalten und umstrukturieren.</i></p> <p>Im letzten Kapitel werden wir u.a. einige Maßnahmen ergreifen, um die Ausgabe schöner zu gestalten. Außerdem wählen wir testweise einen Datencontainer nutzen, um die Daten zu verwalten.</p>
--	---

6 Projekte und Erweiterungen

6.1 Erweiterung des BMI-Rechners 2.0

<pre>24 private DecimalFormat f = new DecimalFormat("#0.00");</pre>	<p><i>Gerundete Ergebnisse erzeugen.</i></p> <p>Deklariieren Sie dazu in der MainActivity-Klasse unterhalb der deklarierten Komponenten ein Objekt „f“ vom Typ DecimalFormat.</p> <p>Übergeben Sie dabei im Konstruktoraufruf</p> <pre>new DecimalFormat("#0.00")</pre> <p>als Parameter den Format-String.</p>
<pre>141 private void schreibeErgebnis(){ 142 bmi.setText(String.valueOf(143 f.format(144 derRechner.getBmi())); 145 }</pre>	<p><i>Gerundete Ergebnisse anzeigen.</i></p> <p>Erweitern Sie dazu die Schreibe-Methode der MainActivity-Klasse.</p> <p>Rufen Sie wie nebenstehend angezeigt die Methode</p> <pre>f.format(double)</pre> <p>am Objekt „f“ der Klasse DecimalFormat auf.</p>

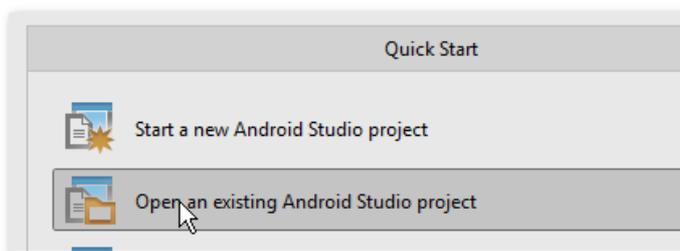
6.2 Erweiterung des Taschenrechners 2.0



Projektverzeichnis kopieren und einfügen.

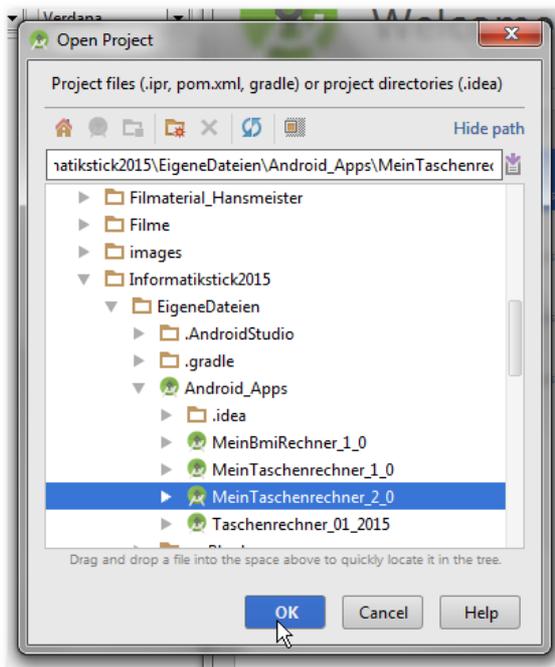
Wechseln Sie in den Android Workspace und kopieren Sie mit STRG+C das Projekt.

Fügen Sie die Kopie mit STRG+V ein und benennen Sie das Verzeichnis wie gewünscht um.



Quick-Start-Menü nutzen.

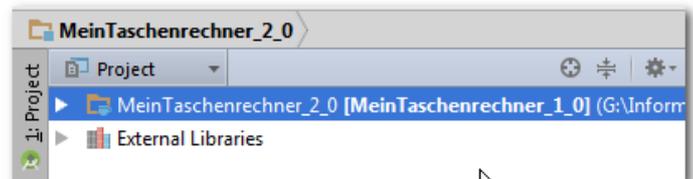
Klicken Sie im Quick Start-Menü die Option „Open an existing Android Studio project“.



Projekt öffnen.

Wählen Sie das Projekt der Version 2.0 aus und klicken Sie auf die Schaltfläche „OK“.

In der Anzeige sehen Sie dass eine Version 2.0 aufbauend auf der Version 1.0 entstehen soll:



Nun können Sie die folgenden Erweiterungen Stück für Stück implementieren.

6.2.1 DecimalFormat zu Anzeige gerundeter Ergebnisse

<pre>24 private DecimalFormat f = new DecimalFormat("#0.00");</pre>	<p><i>Gerundete Ergebnisse erzeugen.</i></p> <p>Deklarieren Sie dazu in der MainActivity-Klasse unterhalb der deklarierten Komponenten ein Objekt „f“ vom Typ DecimalFormat.</p> <p>Übergeben Sie dabei im Konstruktoraufruf</p> <p>new DecimalFormat("#0.00")</p> <p>als Parameter den Format-String.</p>
<pre>218 private void schreibeErgebnis(){ 219 ergebnis.setText(220 String.valueOf(221 f.format(222 derRechner.getErgebnis())); 223 }</pre>	<p><i>Gerundete Ergebnisse anzeigen.</i></p> <p>Erweitern Sie dazu die Schreibe-Methode der MainActivity-Klasse.</p> <p>Rufen Sie wie nebenstehend angezeigt die Methode</p> <p>f.format(double)</p> <p>am Objekt „f“ der Klasse DecimalFormat auf.</p>

6.2.2 Effizienter Quellcode: Hilfsmethoden

An dieser Stelle wird sich der Quellcode reduzieren. Bei allen anderen editierbaren Komponenten auch!

```
new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {
    }

    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
    }

    @Override
    public void afterTextChanged(Editable s) { zahl1.setEnabled(s.length() >= 0); }
}
```

An dieser Stelle wird sich u.a. der Quellcode reduzieren. Bei allen anderen Buttons auch!

```
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
addieren.setOnClickListener(
    new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (v == addieren){
                //Eingabe
                leseZahl1();
                leseZahl2();

                //Verarbeitung
                derRechner.addieren();

                //Ausgabe
                schreibeErgebnis();
            }else{
                finish();
            }
        }
    }
);
```

Wiederholungen im Quellcode vermeiden.

Bestenfalls verzichten wir gänzlich auf Wiederholungen im Quellcode. Wenn wir die MainActivity-Klasse betrachten, kommt es allerdings derzeit noch an zwei Stellen zu Dopplungen im Quellcode:

```
new TextWatcher() {...}
```

und für die Berechnung

```
button.setOnClickListener(...);
```

Entsprechende Hilfsmethoden, die wir am unteren Ende der MainActivity-Klasse platzieren, können das verhindern.

Implementieren Sie die folgenden Hilfsmethoden und nutzen Sie diese, um auf die Wiederholungen zu verzichten.

```

106 //Beobachter für alle TextWatcheraufrufe
107 private TextWatcher beobachte(){
108     return new TextWatcher(){
109
110         @Override
111         public void onTextChanged(CharSequence s, int start,int before, int count){
112
113         }
114
115         @Override
116         public void beforeTextChanged(CharSequence s, int start,int count, int after){
117
118         }
119
120         @Override
121         public void afterTextChanged(Editable s){
122             zahl1.setEnabled(s.length() >= 0);
123             zahl2.setEnabled(s.length() >= 0);
124         }
125     };
126 }
127

```

TextWatcher in Hilfsmethode auslagern.

Dazu wird eine neue private Hilfsmethode beobachte() liefert uns nun das gewünschte TextWatcher-Objekt zurück.

```

46 /*Die TextChangedListener nutzen nun die Hilfsmethode
47 beobachte um den TextWatcher zu verwenden*/
48 zahl1.addTextChangedListener(beobachte());
49 zahl2.addTextChangedListener(beobachte());

```

Den ausgelagerten TextWatcher über die private Hilfsmethode beobachte() nutzen.

Die TextChangedListener nutzen nun die Hilfsmethode beobachte um den TextWatcher zu verwenden.

Der Quellcode der MainActivity-Klasse in der onCreate-Methode reduziert sich dementsprechend an dieser Stelle, wo des Listener-Objekt erwartet wird, auf jeweils eine Zeile.

```

133 //Berechner für alle Listeneraufrufe
134 private View.OnClickListener berechne() {
135     return new View.OnClickListener() {
136         public void onClick(View v) {
137             //Eingabe
138             leseEingaben();
139
140             //Prueft welche Schaltflaeche angeklickt wurde
141             ermittleOperationUndBerechne(v);
142
143             //Ausgabe: Aktualisiere die Benutzeroberflaeche
144             updateGUI();
145         }
146     };
147 }

```

Den OnClickListener (die Berechnung) in eine private Hilfsmethode auslagern.

Wir erzeugen einen Listener der, egal welche Operation (addieren, subtrahieren, multiplizieren, dividieren) angeklickt wurde, die Eingaben liest, verarbeitet und das Ergebnis ausgibt.

Implementieren Sie die Auslagerung der Berechnung, wie nebenstehend angezeigt als private Hilfsmethode der ActivityMain-Klasse. Platzieren Sie dazu die Methode unterhalb der bereits erzeugten Hilfsmethoden.

Implementieren Sie danach die benötigten privaten Hilfsmethoden:

```

→ leseEingaben();
→ ermittleOperationUndBerechne(View v)
→ updateGUI();

```

wie im Anschluss erläutert.

Zeile 138

leseEingaben();

Ist eine private Hilfsmethode die alle Eingaben liest. Die Methode nutzt die bereits erzeugten Lese-Methoden:

- leseZahl1() und
- leseZahl2()

Zeile 141

ermittleOperationUndBerechne(View v)

Ist eine private Hilfsmethode die prüft welche Rechenoperation ausgeführt wurde und berechnet dann das Ergebnis.

Zeile 144

```
updateGUI();
```

Ist eine private Hilfsmethode die abschließend gewünschte Ausgabe erzeugt. Dazu wird das angezeigte Ergebnis auf der Benutzeroberfläche aktualisiert (überschrieben) und das CnClickListener-Objekt zurückgegeben.

```

98 private void leseEingaben() {
99     leseZahl1();
100    leseZahl2();
101 }
```

Private Hilfsmethode leseEingabe().

Diese Hilfsmethode fasst die bereits implementierten lese-Methoden für die zahl1 und zahl2 zusammen.

Implementieren Sie die Hilfsmethode „leseEingabe()“, wie nebenstehend angezeigt.

```

141 private void ermittleOperationUndBerechne(View v) {
142     if (v == addieren) {
143         //Verarbeitung
144         derRechner.addieren();
145     } else if (v == subtrahieren) {
146         //Verarbeitung
147         derRechner.subtrahieren();
148     } else if (v == multiplizieren) {
149         //Verarbeitung
150         derRechner.multiplizieren();
151     } else if (v == dividieren) {
152         if (derRechner.getZahl2() > 0) {
153             derRechner.dividieren();
154         } else {
155             showToast("Division durch 0 nicht möglich!");
156         }
157     } else {
158         finish();
159     }
160 }
```

Private Hilfsmethode ermittleOperationUndBerechne(View v).

Mit Hilfe der Kontrollstruktur ELSE-IF und dem View-Objekt → v wird geprüft welche Schaltfläche (Button) ausgewählt wurde.

Im Rahmen der Verarbeitung wird dann die entsprechende Operation:

- addieren(),
- subtrahieren(),
- multiplizieren() oder
- dividieren()

am Objekt der Fachklasse aufgerufen.

Zeile 155

```
showToast (
    "Division durch 0 nicht möglich!");
```

Ist ebenfalls eine private Hilfsmethode die zusätzlich implementiert werden muss:

```

180 private void showToast(CharSequence msg) {
181     Toast.makeText(this, msg, Toast.LENGTH_LONG).show();
182 }
```

Sonderfall Dividieren:

Bevor die Division erfolgt, wird geprüft ob der Wert von → zahl2 größer null ist. Ist das der Fall wird die Operation → dividieren() aufgerufen. Ist der Wert null, wird eine Meldung in einer Infobox (Toast) ausgegeben:

Division durch 0 nicht möglich!

```
162 private void schreibeZahl1(){
163     zahl1.setText(String.valueOf(derRechner.getZahl1()));
164 }
165
166 private void schreibeZahl2(){
167     zahl2.setText(String.valueOf(derRechner.getZahl2()));
168 }
169
170 private void schreibeErgebnis(){
171     ergebnis.setText(String.valueOf(f.format(derRechner.getErgebnis())));
172 }
173
174 private void updateGUI(){
175     schreibeZahl1();
176     schreibeZahl2();
177     schreibeErgebnis();
178 }
```

Private Hilfsmethode updateGUI().

Die Ausgabe soll nach der Berechnung aktualisiert werden. Dazu werden die aktuellen Werte aus dem Objekt der Fachklasse ermittelt, umgewandelt und in die jeweilige Komponente geschrieben.

Implementieren Sie die Methode → updateGUI() und fügen Sie alle noch fehlenden privaten Hilfsmethoden hinzu, wie nebenstehend angezeigt.

```
52 /*Die OnClickListener nutzen nun die Hilfsmethode
53 berechne um den TextWatcher zu verwenden*/
54 addieren.setOnClickListener(berechne());
55 subtrahieren.setOnClickListener(berechne());
56 multiplizieren.setOnClickListener(berechne());
57 dividieren.setOnClickListener(berechne());
```

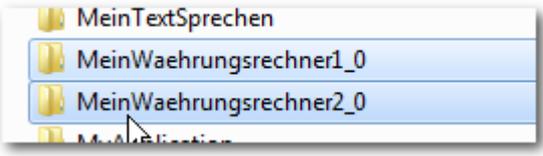
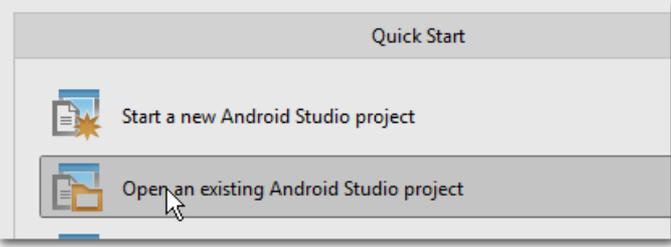
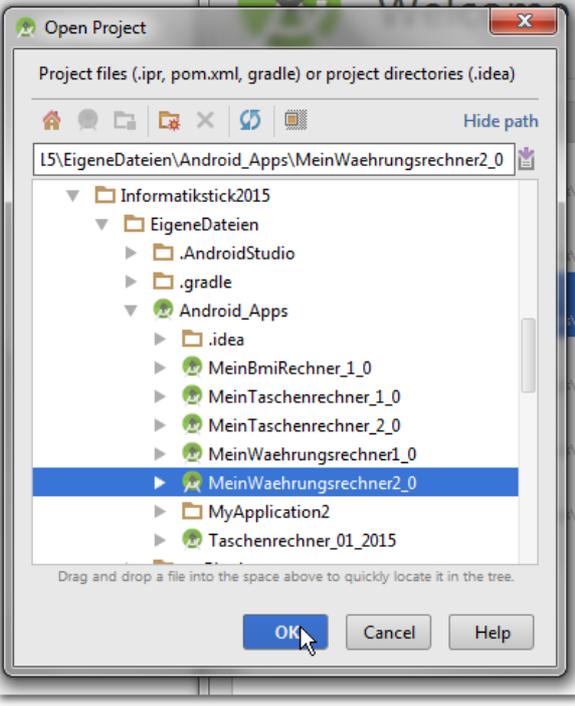
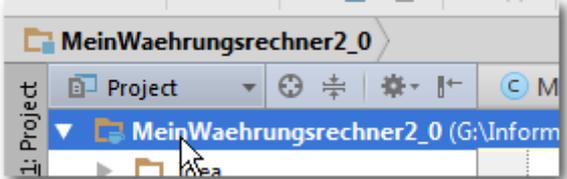
Den ausgelagerten View.OnClickListener über die private Hilfsmethode nutzen.

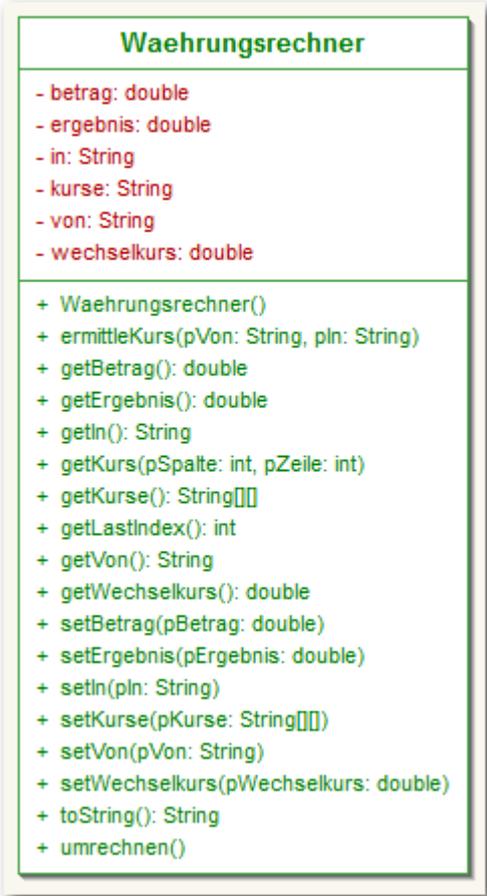
Die Methode setOnClickListener(...) nutzt nun die Hilfsmethode → berechne() um das OnClickListener-Objekt zu verwenden.

Der Quellcode der MainActivity-Klasse in der onCreate-Methode reduziert sich dementsprechend an der Stelle, wo zuvor die Listener-Objekte erwartet wurden, auf jeweils eine Zeile.

Testen Sie die Anwendung erneut und prüfen Sie die implementierten Optimierungen und Erweiterungen!

6.3 Variante des Währungsrechners 2.0

	<p><i>Projektverzeichnis kopieren und einfügen.</i></p> <p>Wechseln Sie in den Android Workspace und kopieren Sie mit STRG+C das Projekt.</p> <p>Fügen Sie die Kopie mit STRG+V ein und benennen Sie das Verzeichnis wie gewünscht um.</p>
	<p><i>Quick-Start-Menü nutzen.</i></p> <p>Klicken Sie im Quick Start-Menü die Option „Open an existing Android Studio project“.</p>
	<p><i>Projekt öffnen.</i></p> <p>Wählen Sie das Projekt der Version 2.0 aus und klicken Sie auf die Schaltfläche „OK“.</p> <p>In der Anzeige sehen Sie dass eine Version 2.0 aufbauend auf der Version 1.0 entstehen soll:</p>  <p>Nun können Sie die folgenden Erweiterungen Stück für Stück implementieren.</p>

 <p>The UML class diagram for Waehrungsrechner shows the following structure:</p> <ul style="list-style-type: none"> Attributes: <ul style="list-style-type: none"> - betrag: double - ergebnis: double - in: String - kurse: String - von: String - wechselkurs: double Constructor & Methods: <ul style="list-style-type: none"> + Waehrungsrechner() + ermittleKurs(pVon: String, pln: String) + getBetrag(): double + getErgebnis(): double + getIn(): String + getKurs(pSpalte: int, pZeile: int) + getKurse(): String[][] + getLastIndex(): int + getVon(): String + getWechselkurs(): double + setBetrag(pBetrag: double) + setErgebnis(pErgebnis: double) + setIn(pln: String) + setKurse(pKurse: String[][]) + setVon(pVon: String) + setWechselkurs(pWechselkurs: double) + toString(): String + umrechnen() <p>Labels on the right side of the diagram indicate: <ul style="list-style-type: none"> Klasse (Class) for the class name. Attribute for the list of attributes. Konstruktor & Methoden (Constructor & Methods) for the list of methods. </p> <p>UML-Klasse: <i>Waehrungsrechner</i></p>	<p>Wir implementieren die Fachklasse „Waehrungsrechner“ entsprechend den Vorgaben der angezeigten UML-Klasse.</p> <p>Besonderheiten</p> <pre>private String[][] kurse = new String[3][16];</pre> <p>Das Attribut <code>kurse</code> ist ein 2-dimensionales Array aus Strings mit 3 Spalten und 16 Zeilen.</p>
<pre>class Waehrungsrechner { // Deklaration der Eigenschaften (Attribute) private double betrag; private String von; private String in; private double ergebnis; private double wechselkurs; }</pre>	<p><i>Deklaration der Attribute.</i></p> <pre>private double betrag;</pre> <p>Der Zugriffsmodifikator → <code>private</code> stellt sicher, dass nur die Objekte der Klasse selbst auf die Eigenschaften direkt zugreifen kann.</p> <p>Der primitive Datentyp → <code>double</code> bestimmt den Wertebereich und das Zahlenformat für eine Gleitkommazahl mit doppelter Genauigkeit. Sobald in Java eine Gleitkommazahl verarbeitet werden soll, greifen die meisten Programmierer zum Datentyp → <code>double</code>. Mit der Bestimmung des geeigneten Datentyps für ein Attribut wird gleichzeitig der maximal benötigte Speicherplatz vorab reserviert.</p>

	→ betrag ist der Attributname. Attribute werden in Java kleingeschrieben und enthalten keine Umlaute und/oder Sonderzeichen.
<pre> 15 // Standard (Default) Konstruktor 16 public Waehrungsrechner() { 17 18 } </pre>	<p><i>Deklaration des Konstruktors.</i></p> <p>Der Konstruktor einer Klasse sorgt dafür, dass beliebig viele Objekte der Klasse erzeugt „konstruiert“ werden können.</p> <p>Jeder Benutzer erzeugt damit sein eigenes Bmirechner-Objekt.</p> <p>Wir nutzen den Standard Konstruktor, ohne Parameter und ohne Initialisierung von Anfangswerten. Neu erzeugte Waehrungsrechner-Objekte sind also am Anfang ihrer Entstehung „wertelos“.</p>
<p><i>Beispiel Attribut „ergebnis“:</i></p> <p>Get-Methode (Ermittlung)</p> <pre> public double getErgebnis() { return ergebnis; } </pre> <p>Set-Methode (Übermittlung)</p> <pre> public void setErgebnis(double ergebnis) { this.ergebnis = ergebnis; } </pre>	<p><i>Deklaration und Implementierung der Get- und Set-Methoden.</i></p> <p>Berücksichtigen Sie, dass wir auf die Eigenschaftswerte der Waehrungsrechner-Objekte von außerhalb der Klasse zugreifen müssen. Jedes Attribut benötigt deshalb eine Get- und Set-Methode.</p> <p>Implementieren Sie außerdem nach dem gleichen Muster die Get- und Set-Methoden für die übrigen Attribute.</p>
<p>Deklaration des zweidimensionalen Arrays:</p> <pre>private String[][] kurse = new String[3][16];</pre> <p>Deklarieren Sie auch die zugehörige Get- und Set-Methode für das Kursarray:</p> <p>Getter:</p> <pre>public String[][] getKurse() { return kurse; }</pre>	<p><i>Abhängig von der Wahl des Benutzers (von, in) soll der Wechselkurs bestimmt werden.</i></p> <p>Danach soll dann der Betrag in die Zielwährung umgerechnet werden.</p> <p>Hier sind viele Lösungsansätze möglich!</p> <p>Wir werden hier einen Ansatz wählen der einen zweidimensionalen Datencontainer verwendet.</p>

}

Setter:

```
public void setKurse(String[][] pKurse) {
    this.kurse = pKurse;
}
```

Initialisierung des zweidimensionalen Arrays im Konstruktor der Fachklasse:

```
// Erweiterter Konstruktor
public Waehrungsrechner(){
    kurse[0][0]="1.0000";
    kurse[1][0]="Euro (EUR)";
    kurse[2][0]="Euro (EUR)";

    kurse[0][1]="0.72085";
    kurse[1][1]="Euro (EUR)";
    kurse[2][1]="Britische Pfund (GBP)";

    kurse[0][2]="1.00000";
    kurse[1][2]="Britische Pfund (GBP)";
    kurse[2][2]="Britische Pfund (GBP)";

    kurse[0][3]="1.05987";
    kurse[1][3]="Euro (EUR)";
    kurse[2][3]="US Dollar (USD)";

    kurse[0][4]="1.0000";
    kurse[1][4]="US Dollar (USD)";
    kurse[2][4]="US Dollar (USD)";

    kurse[0][5]="126.86000";
    kurse[1][5]="Euro (EUR)";
    kurse[2][5]="Japanischer Yen (JPY)";

    kurse[0][6]="1.0000";
    kurse[1][6]="Japanischer Yen (JPY)";
    kurse[2][6]="Japanischer Yen (JPY)";

    kurse[0][7]="1.38690";
    kurse[1][7]="Britische Pfund (GBP)";
    kurse[2][7]="Euro (EUR)";

    kurse[0][8]="1.47011";
    kurse[1][8]="Britische Pfund (GBP)";
    kurse[2][8]="US Dollar (USD)";

    kurse[0][9]="175.96000";
    kurse[1][9]="Britische Pfund (GBP)";
    kurse[2][9]="Japanischer Yen (JPY)";

    kurse[0][10]="0.94337";
    kurse[1][10]="US Dollar (USD)";
    kurse[2][10]="Euro (EUR)";

    kurse[0][11]="0.68012";
    kurse[1][11]="US Dollar (USD)";
    kurse[2][11]="Britische Pfund (GBP)";
```

Dieser Ansatz wurde u.a. gewählt, um wichtige die Themen

1. Containerklassen (Array)
2. Kontrollstrukturen

im Unterricht zu behandeln.

Dazu deklarieren und initialisieren wir für alle Fälle die Werte für den Wechselkurs, die Ausgangs (von) - und Zielwährung (in).

Da wir uns für den Anfang auf 4 Währungen konzentrieren, ergeben sich aber trotzdem schon mal 16 Kombinationsmöglichkeiten. Es gibt also, egal welche Lösungsmöglichkeit wir wählen, jede Menge zu initialisieren.

Ein zweidimensionales Array (Container) deklarieren:

```
private String[][] arrayname =
    new String[spalten][zeilen];
```

Deklarieren Sie das Array unterhalb der bereits deklarierten Attribute in der Fachklasse, wie nebenstehend angezeigt.

Die Initialisierung erfolgt im Konstruktor der Fachklasse. Dabei erhält jeder Wert eine genaue Adresse.

```
arrayname[0][0]="Wert1";
arrayname[1][0]="Wert2";
arrayname[2][0]="Wert3";
```

```

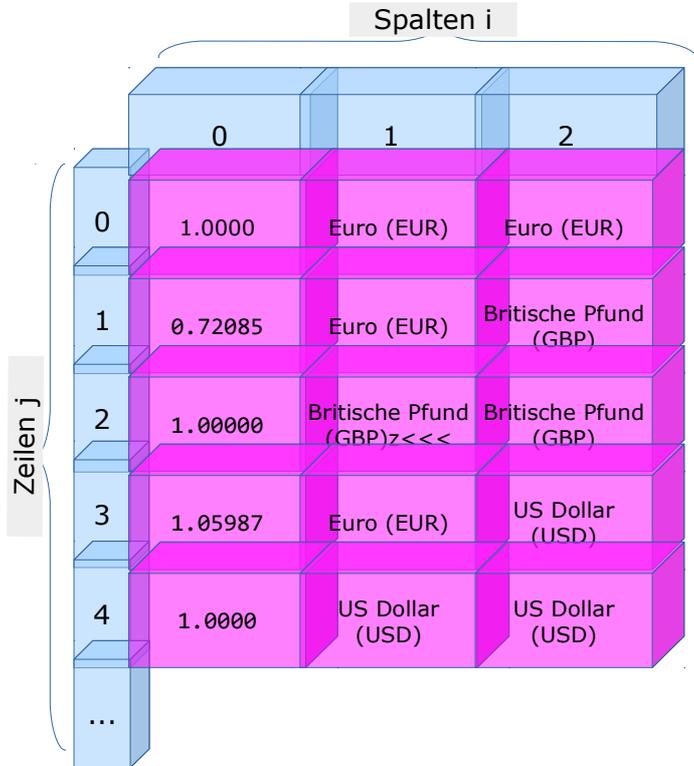
kurse[0][12]="119.69000";
kurse[1][12]="US Dollar (USD)";
kurse[2][12]="Japanischer Yen (JPY)";

kurse[0][13]="0.00788";
kurse[1][13]="Japanischer Yen (JPY)";
kurse[2][13]="Euro (EUR)";

kurse[0][14]="0.00568";
kurse[1][14]="Japanischer Yen (JPY)";
kurse[2][14]="Britische Pfund (GBP)";

kurse[0][15]="0.00835";
kurse[1][15]="Japanischer Yen (JPY)";
kurse[2][15]="US Dollar (USD)";
    }
    
```

Initialisieren Sie das Array im Konstruktor der Fachklasse, wie nebenstehend angezeigt.



```

180 public void ermittleKurs(String pVon, String pIn){
181     ① int spalten=0;
182     int zeilen=0;
183     hier:
184     ② while(spalten< kurse.length){
185         int anzahl = this.getLastIndex()+1; ③
186
187         while(zeilen<= anzahl){
188             this.getKurs(spalten, zeilen);
189             ④ if(this.getVon().equals(pVon) &&
190                 this.getIn().equals(pIn) ){
191                 ⑤ this.getKurs(spalten, zeilen);
192
193                 break hier; ⑥
194             }else {
195                 zeilen++;
196             }
197         }
198         spalten++;
199     }
200 }
201
    
```

Deklaration und Implementierung Höherer Methoden. Ermittlung eines Kurses.

Erläuterung zu ermittleKurs(von, in):
 Um aus dem Kurs-Array einen Kurs zu ermitteln muss mittels zweier geschachtelter Wiederholstrukturen (Schleifen) das Array durchlaufen werden. Es werden zwei Schleifen benötigt, da sowie die Zeilen als auch die Spalten im Array durchlaufen werden müssen.

Die Werte werden dann jeweils mit den Eingaben (Von, In) des Nutzers verglichen. Für den Fall, dass die Werte übereinstimmen, erfolgt die Übernahme der Wechselkurswerte (wechselkurs, von, in) in das aktuelle Objekt der Fachklasse. Ansonsten wird hochgezählt (inkrementiert). Die Methode → ermittleKurs(von, in)“ ist eine Hilfsmethode, die den Wechselkurs für die Umrechnung benötigt wird → umrechnen().

Zeilen und Spalten im Array durchlaufen: ①
 Zur Schalchtelung einer While-Schleife benötigen wir zwei Zählervariablen. Da wir diese benötigen

um die Zeilen und Spalten unseres Arrays zu durchlaufen, nennen wir sie auch so und initialisieren die Zähler jeweils mit 0:

```
int spalten = 0;
int zeilen = 0;
```

Zur Schachtelung: **2**

```
while(spalten < kurse.length){
    //hier soll etwas passieren
    while(zeilen < kurse[spalten].length){
        //hier soll etwas passieren
        zeilen++;
    }
    spalten++;
}
```

Das Attribut → length liefert uns für das Array die Länge der Liste (Anzahl der Werte).

Für den Fall (erste Schleife) von

```
kurse.length
```

= 3 Spalten → wobei der Index mit 0 startet und mit 2 am Ende ist, was bedeutet wir müssen mit dem Vergleichsoperator kleiner als „<“ sicherstellen, dass er niemals Werte 3 und größer prüft.

Außerdem stellen wir sicher, dass am Ende der Schleife die Spaltenanzahl hochgezählt (inkrementiert) wird, sonst würden wir in einer Endlosschleife landen (Abbruchbedingung):

```
spalten++;
```

Gleiches gilt für den Fall (zweite Schleife) von

```
kurse[spalten].length
```

= 16 Zeilen → wobei der Index mit 0 startet und mit 15 am Ende ist, was bedeutet wir müssen mit dem Vergleichsoperator kleiner als „<“ sicherstellen, dass er niemals 16 und größer prüft.

Auch hier stellen wir sicher, dass am Ende der Schleife die Zeilenanzahl hochgezählt (inkre-

4

Vergleich mit IF-ELSE:

Innerhalb der zweiten Schleife werden die Eingabewerte (von, in) mit den Werten aus dem Array verglichen. Für den Vergleich wird eine einfache IF-Else-Kontrollstruktur verwendet:

```
if(this.getVon().equals(pVon)&&
    this.getIn().equals(pIn) ){
    this.getKurs(spalten, zeilen);

    break hier;
}else {
    zeilen++;
}
```

5

Hilfsmethode „getKurs(int i, int j)“:

Die Methode ermittelt im Array den Wert des Wechselkurses an der Stelle i,j, wandelt den Wert um in einen Double-Wert und setzt den Wert im aktuellen Objekt der Fachklasse.

```
public void getKurs(int pSpalte, int pZeile){
    int i = pSpalte;
    int j = pZeile;

    this.setWechselkurs(
        Double.valueOf(this.getKurse()[i][j]));
    this.setVon(this.getKurse()[i+1][j]);
    this.setIn(this.getKurse()[i+2][j]);
}
```

In zwei weiteren Anweisungen wird anschließend der zugehörige Wert für die Ausgangswährung (von) und die Zielwährung an der Stelle i+1 und i+2.

mentiert) wird (Abbruchbedingung):

```
zeilen++;
```

Die Erläuterung zum Vergleich erfolgt nebenstehend.

Implementieren Sie die Methode `ermittleKurs(von, in)` in der Fachklasse „`Waehrungsrechner.java`“ als Höhere Methode, wie nebenstehend angezeigt.

Hilfsmethode „`getLastIndex()`“:

3

Ermittelt den index des letzten Elements der Kursliste. Die Methode wird genutzt, um zu ermitteln wieviele Kursdatensätze im Kursarray enthalten sind, nämlich „`lastIndex + 1`“. Ist die Abbruchbedingung der zweiten Schleife.

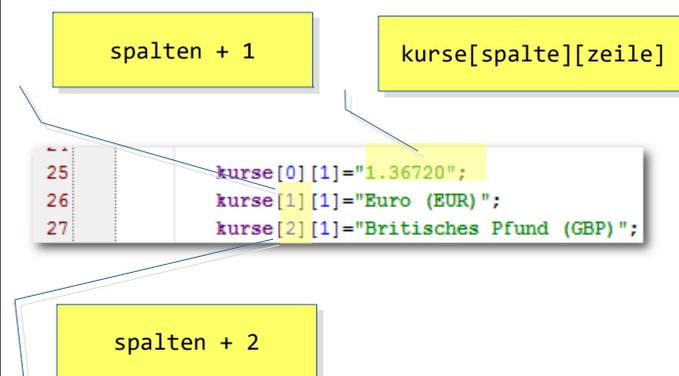
```
public int getLastIndex(){
    int mIndex = 0;
    int i = 0;
    int j = 0;
    while(i < this.kurse.length){
        while(j < this.kurse[i].length){
            try{
                if(!this.kurse[i][j].equals(null)){
                    mIndex = j;
                }
            }catch(NullPointerException e){
                break;
            }
            j++;
        }
        i++;
    }
    return mIndex;
}
```

```
public void umrechnen(){
    ermittleKurs(this.von,this.in);
    this.ergebnis =
```

Erläuterung am Beispiel:

Der Nutzer hat für „von“ den Wert „Euro (EUR)“ und für „in“ den Wert „Britisches Pfund (GBR)“ ausgewählt.

Diese Werte finden wir auch im Array an einer bestimmten Stelle:



Mit der Methode

```
.equals(Object)
```

vergleichen wir den Wert aus dem Array mit den Eingabewerten.

Mit dem Logischen Operator

```
&&
```

stellen wir sicher, dass beide Bedingungen zutreffen.

Falls dies der Fall ist wird der Wechselkurswert aus dem Array ermittelt, umgewandelt in einen double-Wert und im aktuellen Objekt der Fachklasse gesetzt:

```
this.wechselkurs =
    Double.valueOf(this.kurse[spalten][zeilen]);
```

6 Sprungmarke (flag) im „break“:

Sobald der richtige Wechselkurs gefunden ist wird die Suche abgebrochen. Die Sprungmarke „hier“ gibt an, dass mit dem `break` alle drei Kontrollstrukturen verlassen werden.

Wir nutzen die Methode `ermittleKurs(von,in)` für die Umrechnung in das Ergebnis.

```
Math.round((this.betrag *  
this.wechselkurs)*100)/ 100;  
}
```

Ein Betrag X in Ausgangswährung soll in ein →
ergebnis Y in Zielwährung umgerechnet wer-
den.

Dazu sind zwei Schritte notwendig:

1. ermittleKurs(von, in) → wechselkurs
2. ermittle this.ergebnis

Formel für Umrechnung in das Ergebnis:

$$\text{Ergebnis} = ((\text{Betrag} * \text{Wechselkurs}) * 100) / 100:$$

In dieser Variante wird nur eine Funktionalität
implementiert, nämlich die Möglichkeit zu
umzurechnen.

Da das Ergebnis gerundet dargestellt werden
soll, nutzen wir das statische Objekt Math und
rufen dazu die Methode → round() auf.

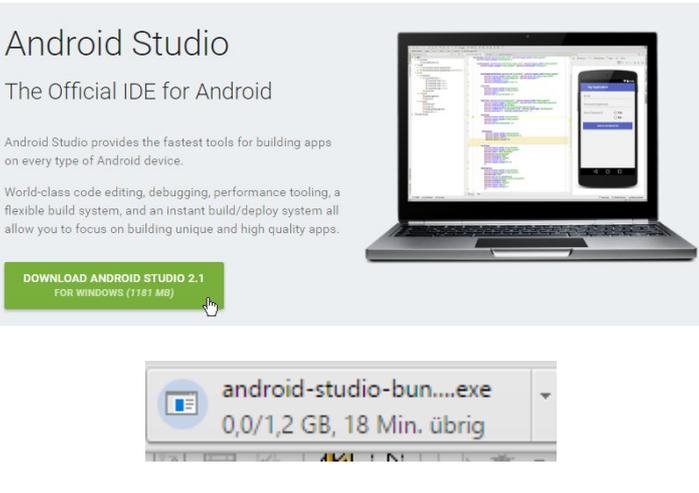
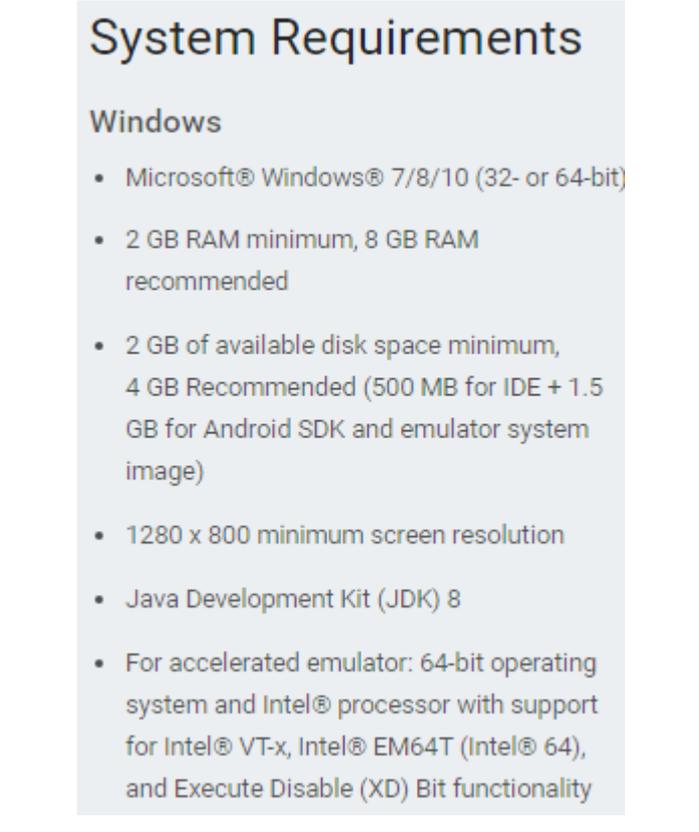
`Math.round(...)`

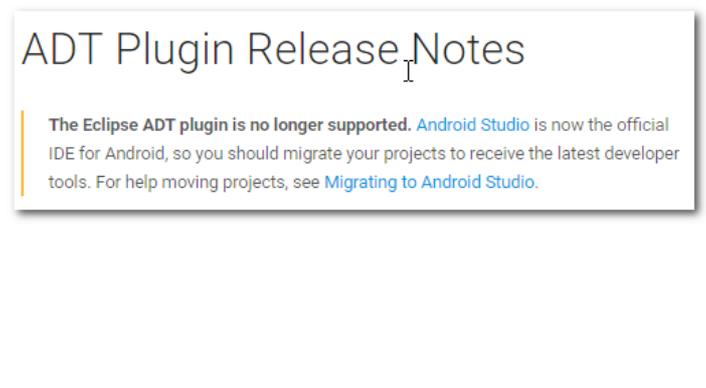
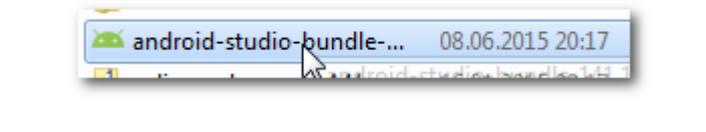
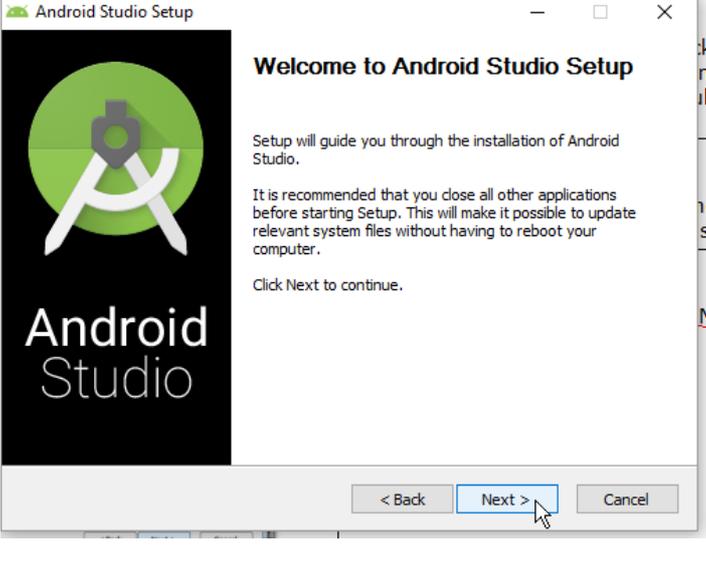
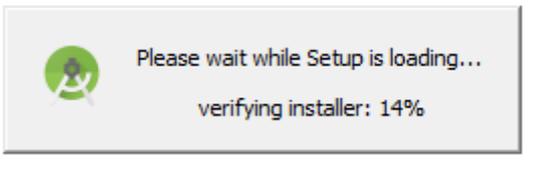
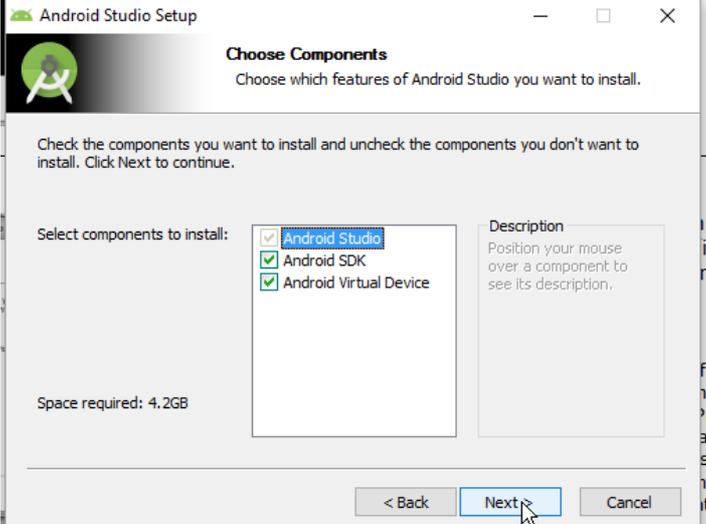
Die Wechselkurse sind von Tag zu Tag unter-
schiedlich und sind vorgegebene statische
Werte (siehe Array).

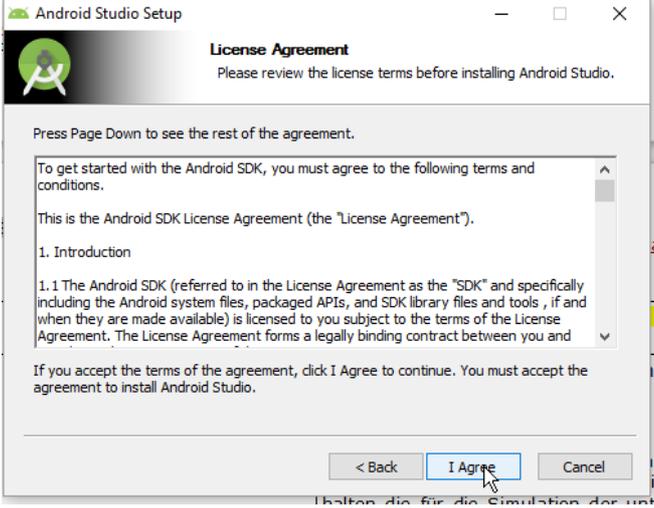
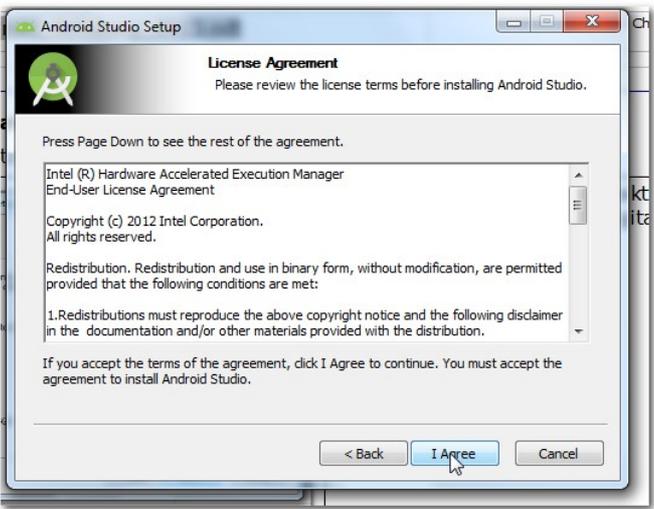
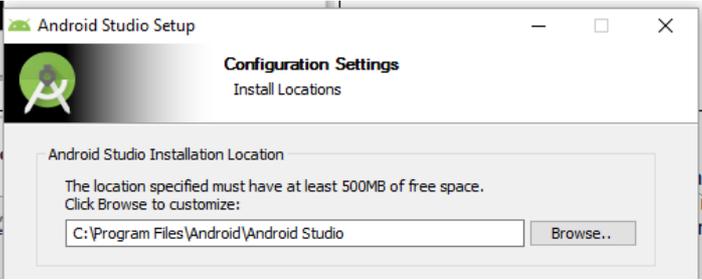
Implementieren Sie auch die Methode umrech-
nen() in der Fachklasse
„Waehrungsrechner.java“ als Höhere Methode,
wie nebenstehend angezeigt.

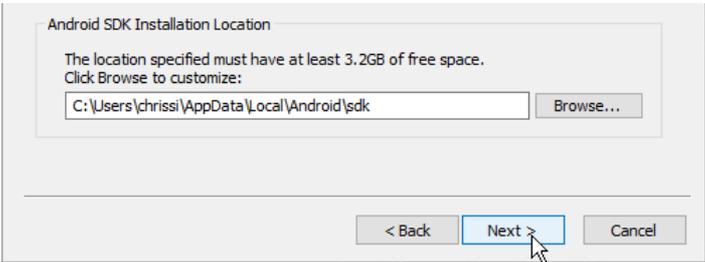
7 Installation und Konfiguration der Entwicklungsumgebung

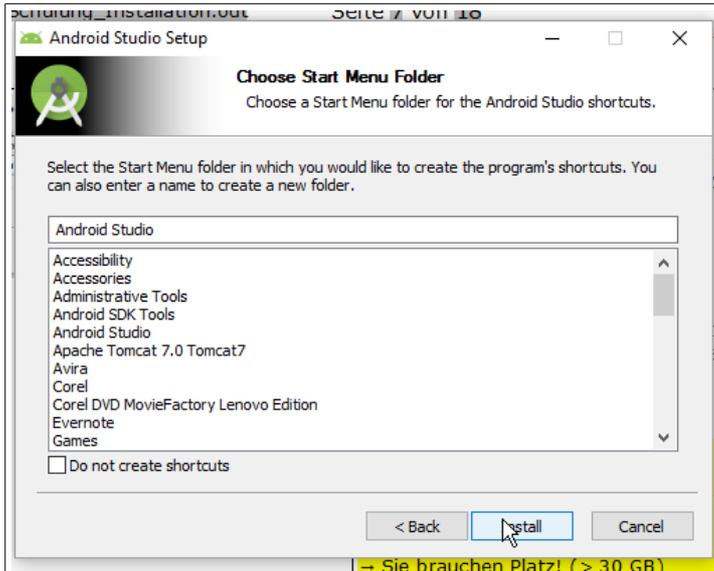
7.1 Installation

	<p><i>Android Studio download.</i></p> <p>Das Android Studio ist aktuell die offizielle Entwicklungsumgebung für die Entwicklung von Anwendungen für mobile Endgeräte mit Android Betriebssystem.</p> <p>Die aktuellste Version (für Windows) finden Sie zum Download auf den Entwicklerseiten:</p> <p>https://developer.android.com/studio/index.html</p> <p>Es sind auch Versionen für MAC OSX und Linux (Ubuntu) zur Verfügung.</p>
	<p><i>Systemvoraussetzungen.</i></p> <p>Die Voraussetzungen an das System (für Windows) sind nebenstehend aufgeführt.</p>

 <p>ADT Plugin Release Notes</p> <p>The Eclipse ADT plugin is no longer supported. Android Studio is now the official IDE for Android, so you should migrate your projects to receive the latest developer tools. For help moving projects, see Migrating to Android Studio.</p>	<p><i>Warum zu Android Studio wechseln?</i></p> <p>Für den Fall, dass Sie noch mit Eclipse und den entsprechenden Erweiterungen arbeiten, findet man zwischenzeitlich auf den Entwicklerseiten einen Hinweis:</p> <p>Darin wird empfohlen, die Entwicklungsumgebung mittelfristig zu wechseln, um die Versorgung mit Updates für die Zukunft sicherzustellen.</p>
	<p><i>Installation starten.</i></p> <p>Klicken Sie die heruntergeladene exe-Datei doppelt an, um die Installation zu starten.</p>
 <p>Welcome to Android Studio Setup</p> <p>Setup will guide you through the installation of Android Studio.</p> <p>It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.</p> <p>Click Next to continue.</p> <p>< Back Next > Cancel</p>	<p><i>Setup fortfahren.</i></p>  <p>Klicken Sie auf die Schaltfläche → Next.</p>
 <p>Choose Components</p> <p>Choose which features of Android Studio you want to install.</p> <p>Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.</p> <p>Select components to install:</p> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Android Studio <input checked="" type="checkbox"/> Android SDK <input checked="" type="checkbox"/> Android Virtual Device <p>Space required: 4.2GB</p> <p>Description: Position your mouse over a component to see its description.</p> <p>< Back Next > Cancel</p>	<p><i>Die Option Android SDK .</i></p> <p>Der Android SDK enthält je nach Umfang die Bibliotheken mit den gerätespezifischen Angaben für die Emulation der Geräte (Handy-Typen).</p> <p>Empfehlung: Den SDK kann bei zu wenig Speicherplatz auch auf andere Partitionen/Datenträger ausgelagert werden. Achten Sie darauf die System- und Umgebungsvariablen anzupassen (siehe Einstellungen).</p> <p>Hinweis: Damit der SDK und damit auch der Emulator in</p>

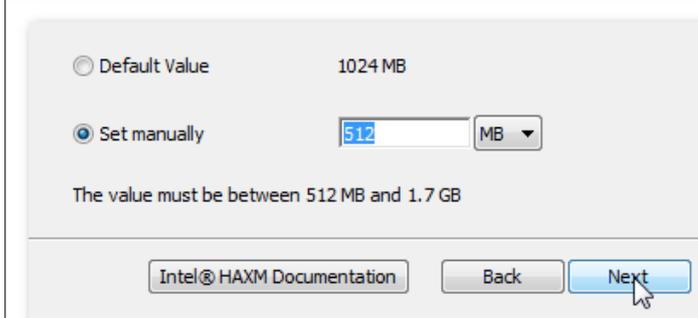
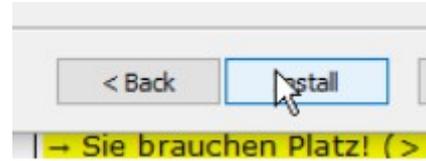
	<p>ganzem Umfang funktionstüchtig sind, benötigt der Entwickler auf dem SDK-Verzeichnis die vollständigen Zugriffsrechte. Gleiches ist für das Programmverzeichnis „Android Studio“ empfohlen, soweit Updates durch den Entwickler eingespielt werden sollen.</p>
	<p>Akzeptieren Sie die Nutzungsbedingungen für den SDK.</p> <p>Software-Development-Kit (SDK): Je nach Ausstattung sind darin u.a. die für den Emulator notwendigen Softwareerweiterungen enthalten die für die Simulation der unterschiedlichen mobilen Endgeräte und deren unterschiedlichen Betriebssystemversionen notwendig sind. Android Studio ist ohne die SDK nicht funktionsfähig!</p>
	<p>Akzeptieren Sie die Nutzungsbedingungen für die HAXM.</p> <p>Hardware Accelerated Execution Manager Ist eine Software, die im speziellen Intel Prozessoren bei der Emulation von mobilen Endgeräten mit Android Betriebssystem beschleunigen soll.</p>
	<p><i>Android Studio Installationsort wählen.</i></p> <p>Geben Sie hier den Pfad für die Installation des Android Studios an. Hier im Beispiel wurde dazu im Programmverzeichnis das vorgeschlagene Verzeichnis → Android Studio verwendet:</p> <p>Variante 1: Lokal C:\Program Files\Android\Android Studio</p>

	<p>Alternative: Falls gewünscht...</p> <p>Variante 2: In der Digitalen Tasche → G:\Informatikstick2016\Programme\Android\Android Studio</p> <p>Laufwerksbuchstaben können variieren! Die Entwicklungsumgebung ist nur bedingt portable da Sie von den Hard- und Systemvoraussetzungen des Rechners abhängig ist.</p> <p>Bitte entscheiden Sie sich für eine Variante!</p>
	<p>SDK-Pfad angeben.</p> <p>Der Installationsassistent schlägt als Installationsort ein lokales benutzerspezifisches sdk-Verzeichnis vor.</p> <p>Variante 1.0: Belassen Sie die Einstellungen. C:\Users\<<Benutzer>\AppData\Local\sdk</p> <p>Alternative: Falls gewünscht...</p> <p>Variante 1.1: Lokales SDK-Verzeichnis: C:\Program Files\Android\sdk</p> <p>Variante 2: SDK-Verzeichnis der Digitalen Tasche: → G:\Informatikstick2016\Programme\Android\sdk</p> <p>Klicken Sie dann auf → Next.</p> <p>Hinweis: Der Ort für das SDK-Verzeichnis kann hier über die Schaltfläche → Browse individuell gewählt werden.</p> <p>→ Merken Sie sich den Ort unbedingt! → Sie brauchen Platz! (> 30 GB) → Laufwerksbuchstaben können variieren</p> <p>Bitte entscheiden Sie sich für eine Variante!</p>



Startmenü konfigurieren

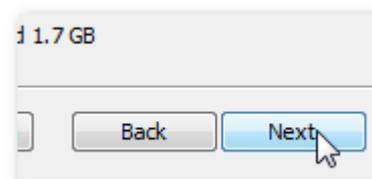
Erzeugen Sie , wie vorgeschlagen, ein Programmstartverzeichnis im Startmenü.

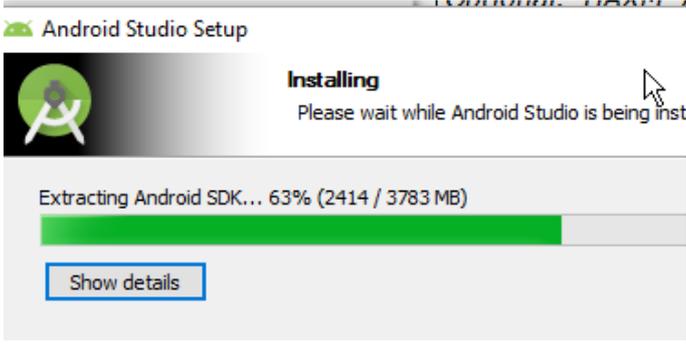
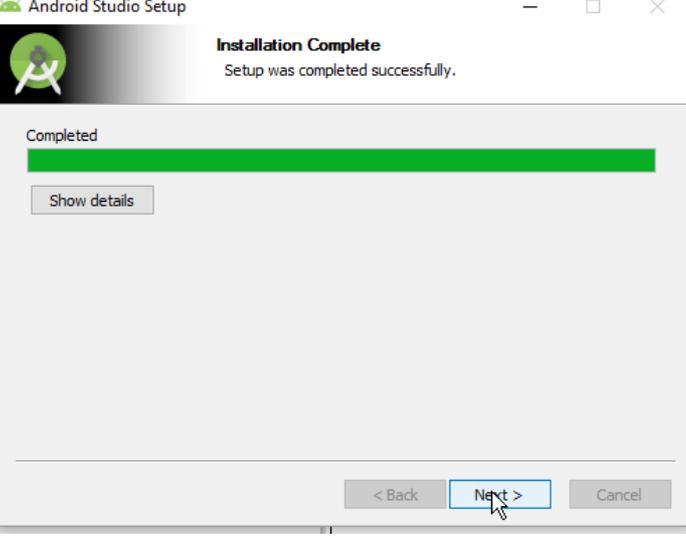
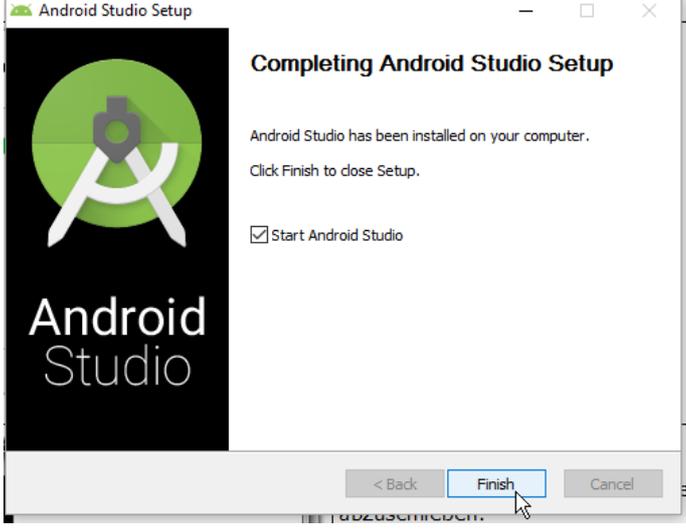


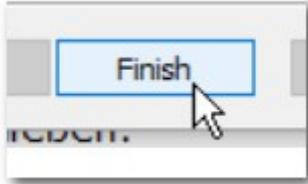
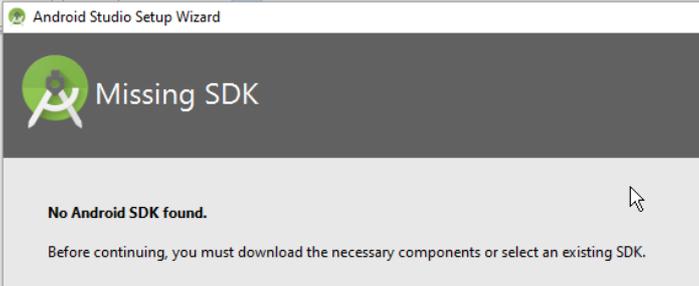
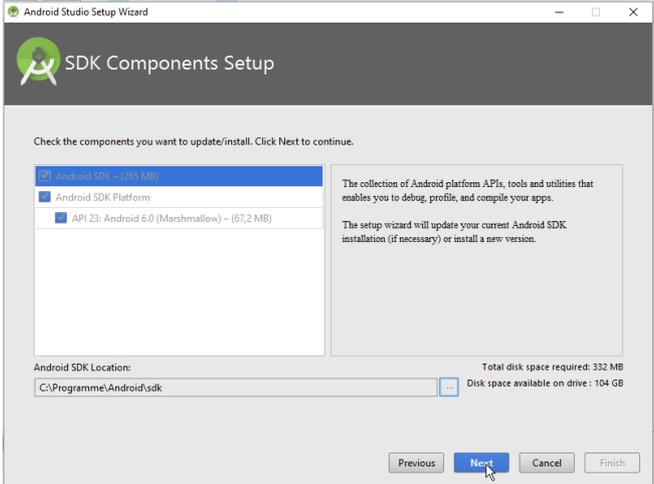
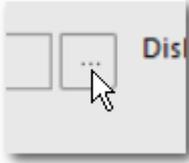
Optional. HAXM Arbeitsspeicher manuell verkleinern.

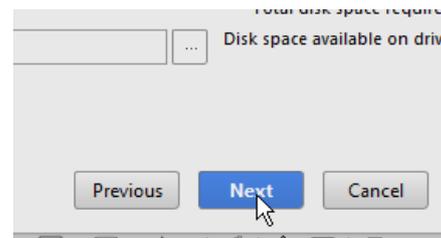
Setzen Sie den Arbeitsspeicher ggf. manuell auf 512 MB. Das ist nur dann sinnvoll, wenn Ihr Rechner gerade einmal die Mindestanforderung von 2 GB Arbeitsspeicher erfüllt. Für Geräte mit hoher Auflösung benötigt der Emulator viel virtuellen Arbeitsspeicher.

Klicken Sie dann auch die Schaltfläche → Next



	<p><i>Installation durchführen.</i></p> <p>Starten Sie nun die Installation mit einem Klick auf die Schaltfläche → Install:</p> 
	<p><i>Installation abschließen.</i></p> <p>Klicken Sie auf → Next.</p>
	<p><i>Installation abschließen.</i></p> <p>Klicken Sie auf → Finish, um die Installation abzuschließen.</p> <p>Nur für den Fall, dass Sie als Administrator keinen Zugang zum Internet haben:</p> <p>Entfernen Sie das Häkchen → Start Android Studio.</p> <p>Klicken Sie auf die Schaltfläche → Finish</p>

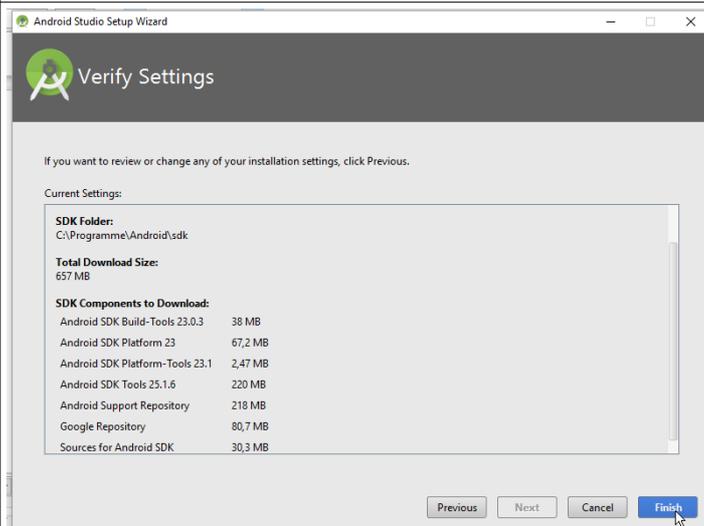
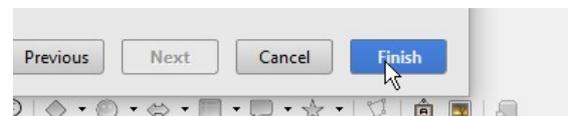
	
	<p>SDK Missing.</p> <p>Klicken Sie auf die Schaltfläche → Next</p> 
	<p>SDK installieren.</p> <p>Je nachdem für welchen Ort Sie sich eingangs entschieden haben, sollten Sie den Pfad über die Schaltfläche → ... anpassen.</p>  <p>Variante 1.0: Standard-Einstellung C:\Users\<<Benutzer>\AppData\Local\sdk</p> <p>Alternative: Falls anders gewählt...</p> <p>Variante 1.1: Lokales SDK-Verzeichnis: C:\Program Files\Android\sdk</p> <p>Variante 2: SDK-Verzeichnis der Digitalen Tasche: → G:\Informatikstick2016\Programme\Android\sdk</p> <p>Klicken Sie auf die Schaltfläche → Next</p>



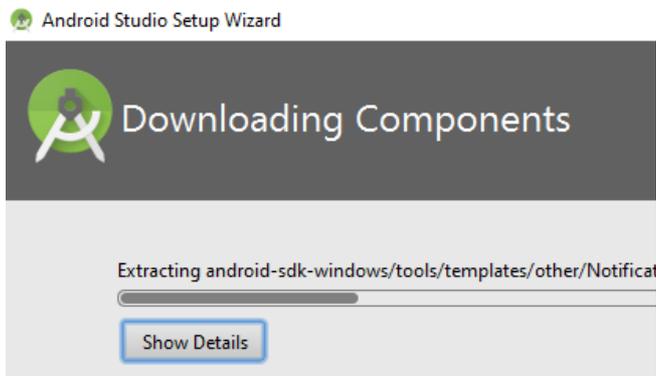
Bitte entscheiden Sie sich für eine Variante!

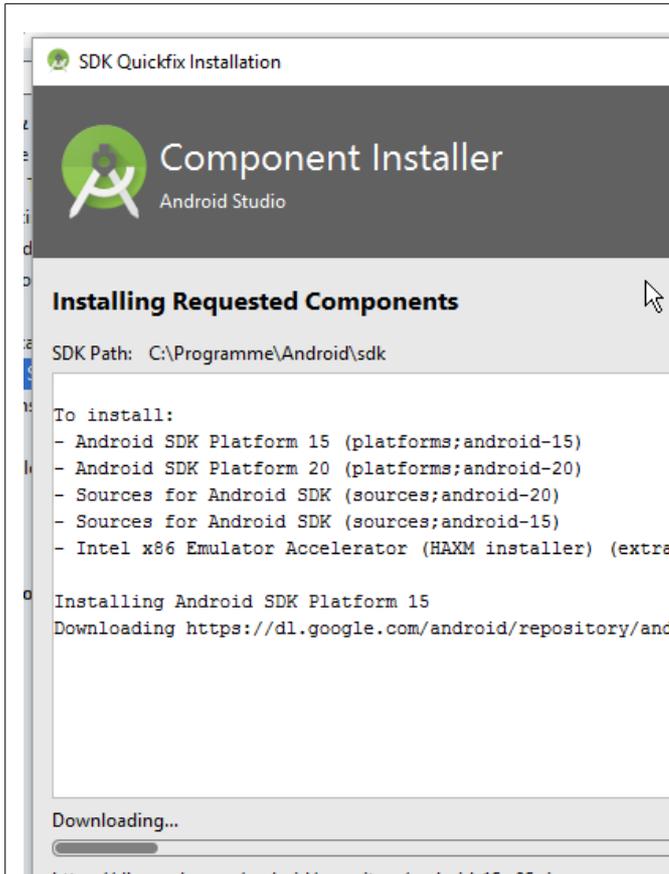
Einstellungen übernehmen.

Klicken Sie auf die Schaltfläche → Finish!



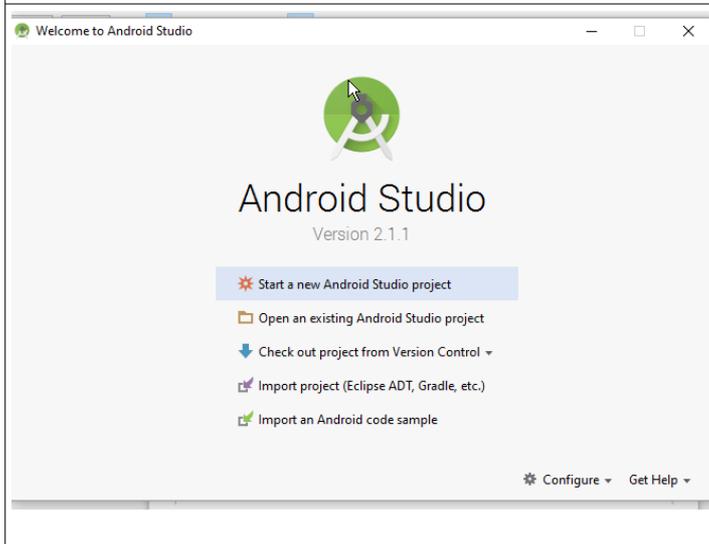
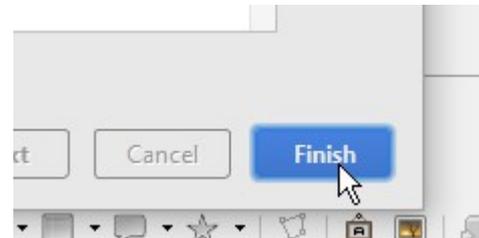
Installation durchführen.





Installation abschließen.

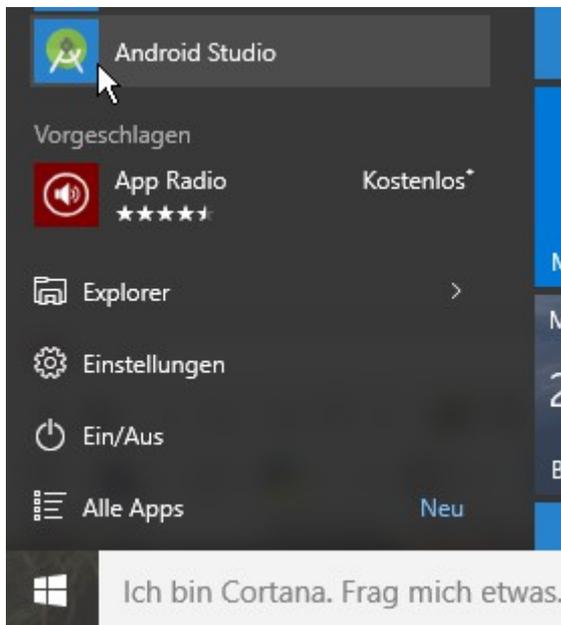
Klicken Sie nach Abschluss der Download-Phase auf die Schaltfläche → Finish



Android Studio öffnen

Sie können Android Studio nun verwenden.

7.2 Einstellungen

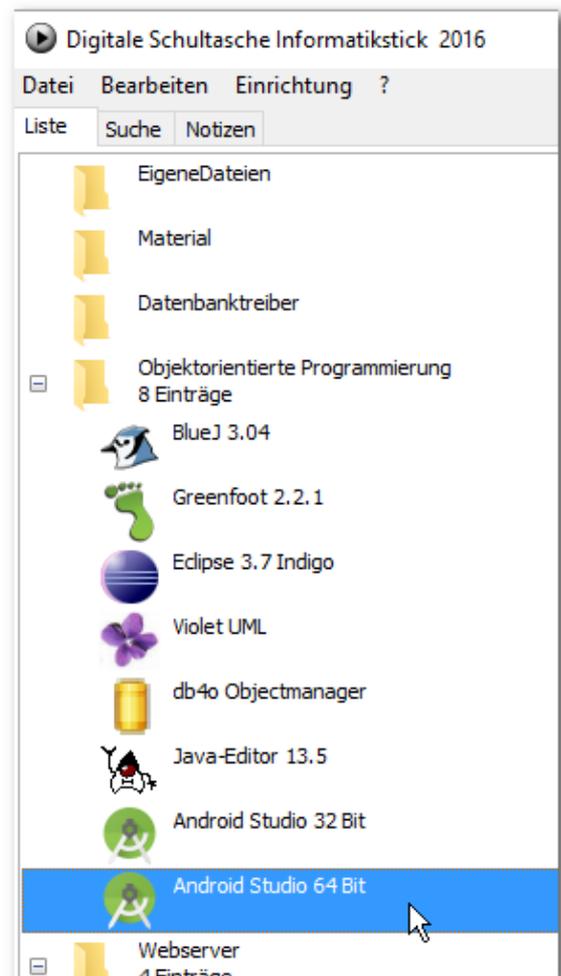


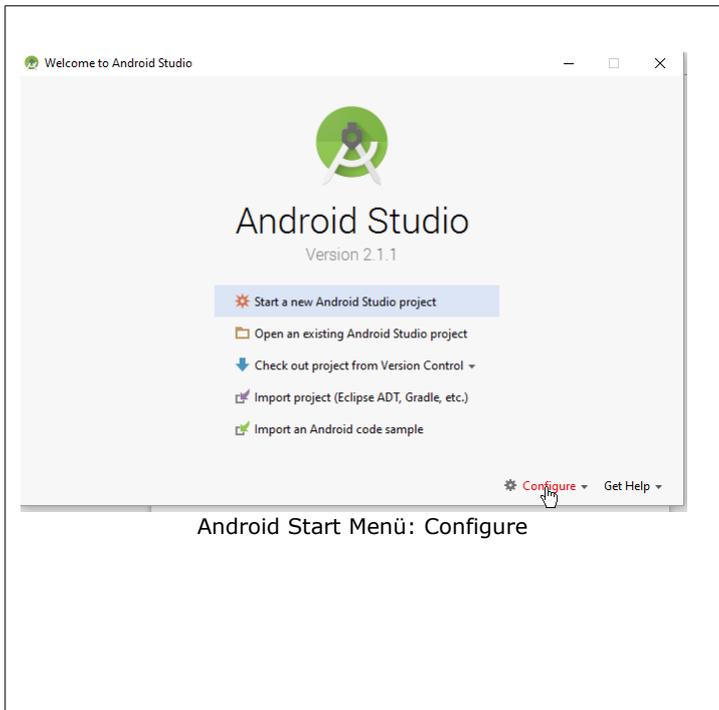
Android Studio starten.

Variante 1:

Klicken Sie dazu im Menü → Start → Android Studio.

Variante 2:

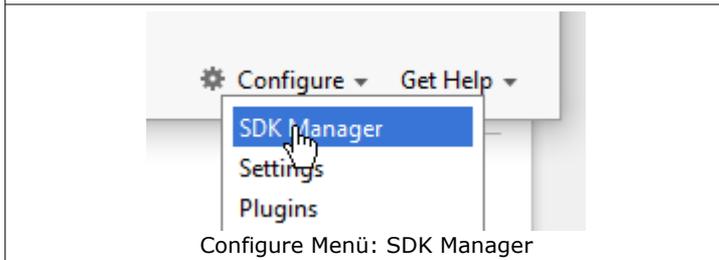
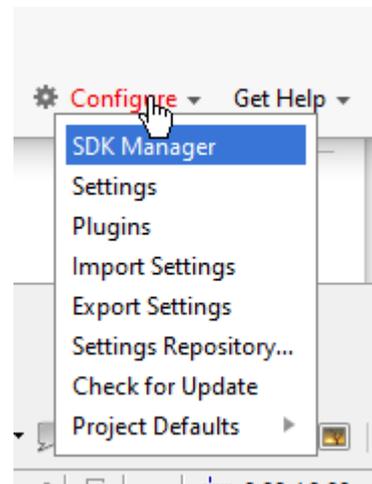




Android Start Menü: Configure

Konfigurieren.

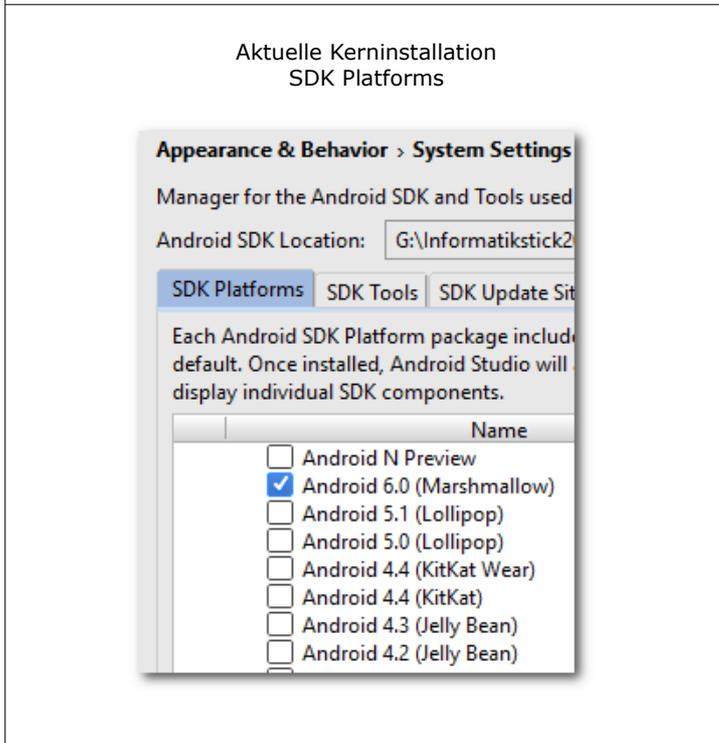
Auswahl im Konfigurationsmenü:



Configure Menü: SDK Manager

SDK Manager.

Wählen Sie die Option → SDK Manager



Aktuelle Kerninstallation
 SDK Plattformen

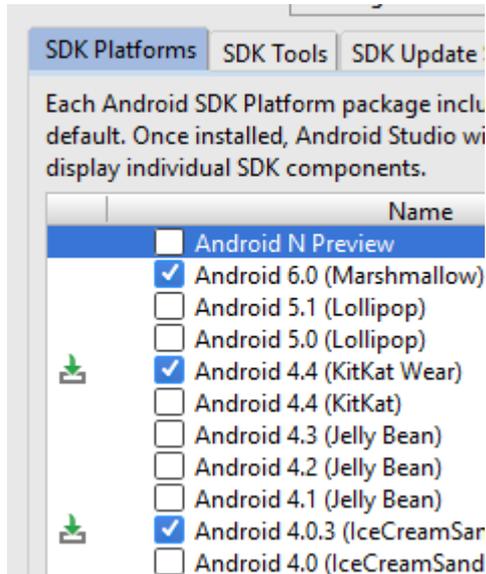
SDK Plattformen.

Ergänzen Sie ggf. in Ihrer Auswahl:

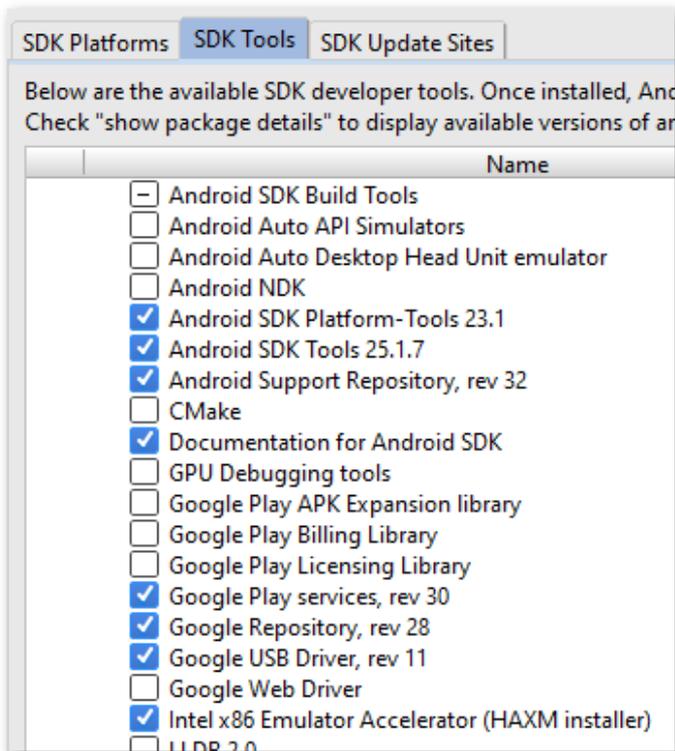
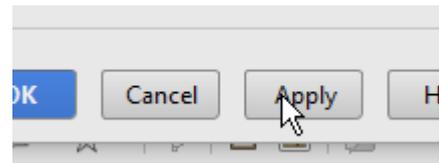
<input type="checkbox"/>	Android 6.0 (Marshmallow)	20
<input checked="" type="checkbox"/>	Android 4.4 (KitKat Wear)	19
<input type="checkbox"/>	Android 4.4 (KitKat)	18
<input type="checkbox"/>	Android 4.3 (Jelly Bean)	17
<input type="checkbox"/>	Android 4.2 (Jelly Bean)	16
<input type="checkbox"/>	Android 4.1 (Jelly Bean)	15
<input checked="" type="checkbox"/>	Android 4.0.3 (IceCreamSandwich)	14
<input type="checkbox"/>	Android 4.0 (IceCreamSandwich)	

Klicken Sie dann die Schaltfläche → Apply.

Zeitintensiv!!



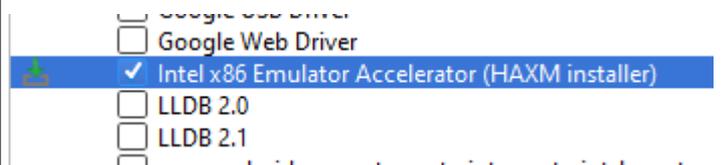
SDK Platforms



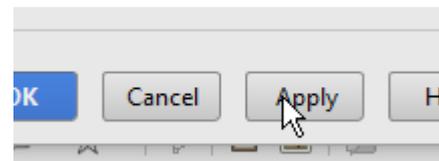
SDK Tools

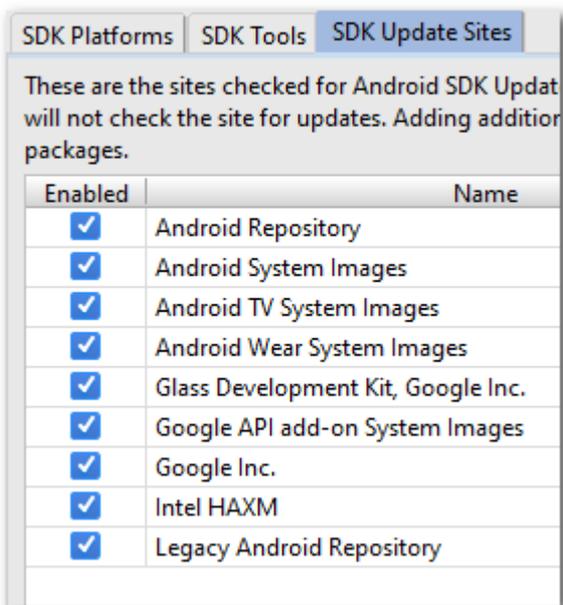
SDK Tools.

Wählen Sie ggf. HAXM als zusätzliches Tool aus und installieren Sie diese nachträglich.



Schaltfläche → Apply klicken.



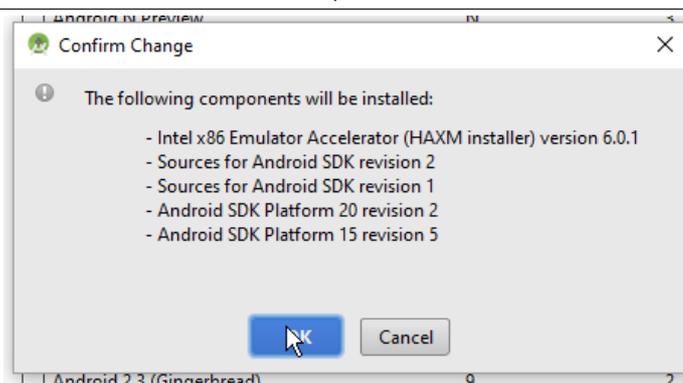
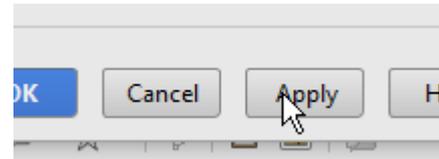


SDK Update Sites

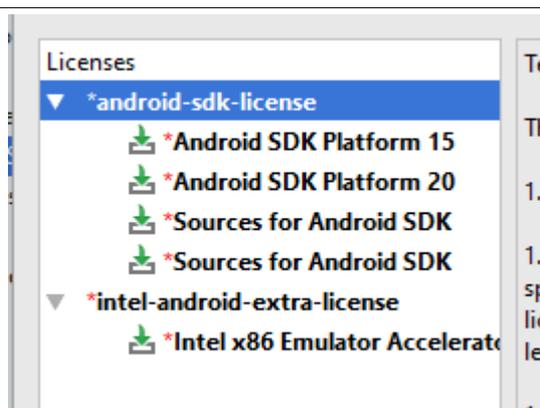
SDK Update Sites.

Einstellungen belassen.

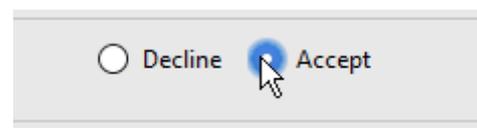
Schaltfläche → Apply klicken.

*Änderungen bestätigen.*

Abschließend mit der Schaltfläche → OK den die Änderungen durchführen.

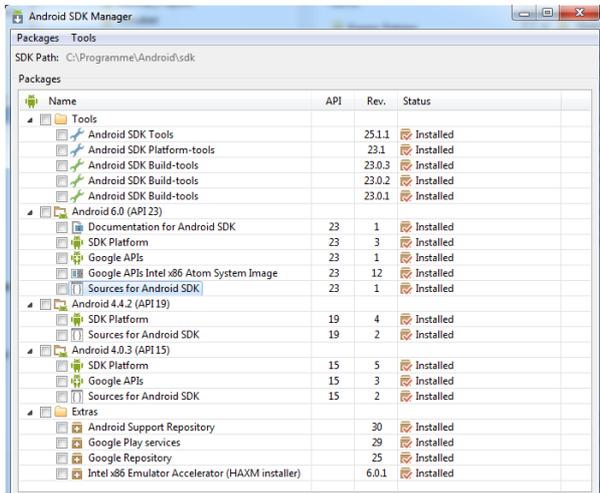
*SDK Lizenzen.*

Wählen Sie das Element → android sdk license im Fenster Lizenz aus und markieren Sie die die Option → Accept.





Standalone SDK Manager



Überblick: empfohlener Umfang

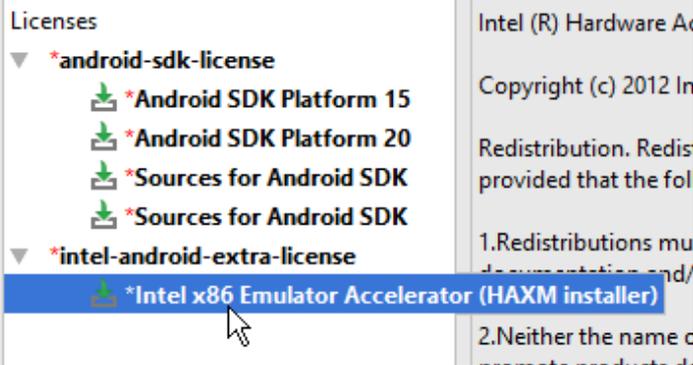
Standalone SDK Manager

Klicken Sie dazu unterhalb des Fensters Setting → Appearance & Behavior → System Settings → Android SDK auf den Link → Launch Standalone SDK Manager.

Prüfen Sie die Liste. Die Bereiche mit dem Vermerk → Installed sind bereits verfügbar. Falls Sie weitere Bereiche nachinstallieren möchten müssen Sie das Häkchen links setzen und dann auf die Schaltfläche → Install packages klicken.

Hinweis:

Die Installation kann je nach Umfang sehr zeitintensiv sein.

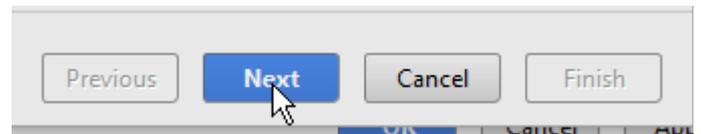
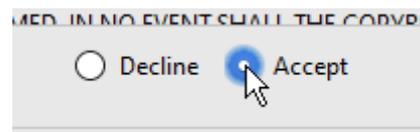


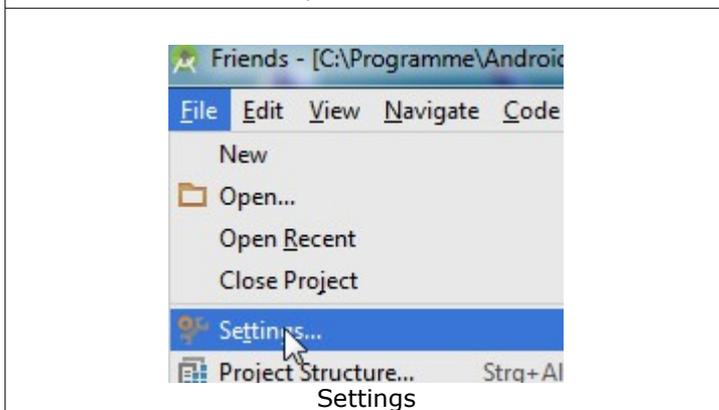
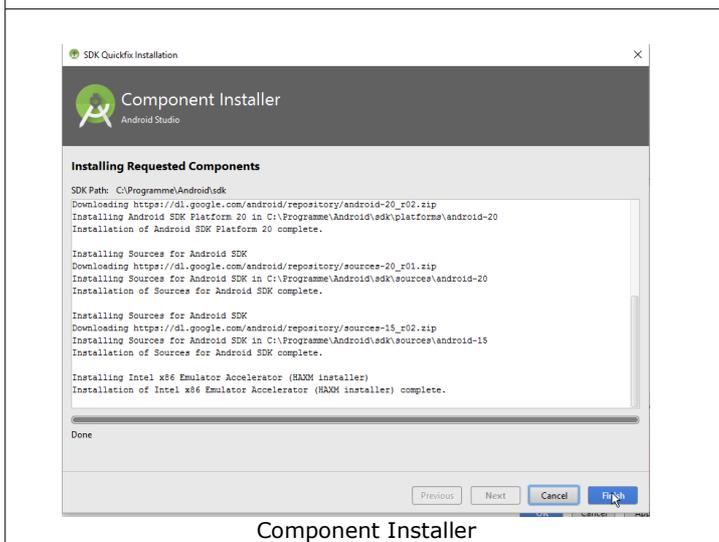
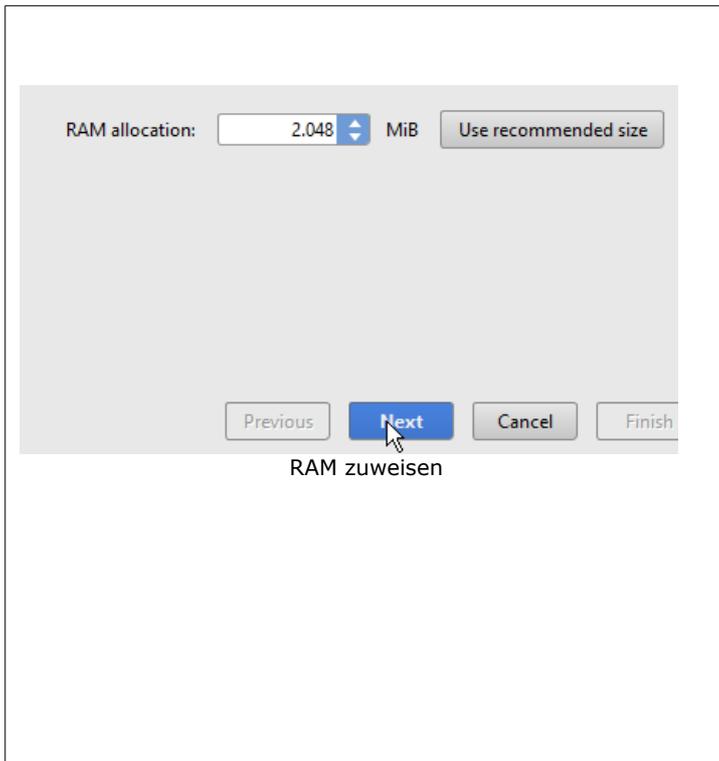
HAXM Accelerator

HAXM.

Für den Fall dass Sie den Accelerator nachinstallieren möchten.

Wählen Sie das Element → Intel x86 Emulator Accelerator, im Fenster Lizenz aus und markieren Sie die die Option → Accept.



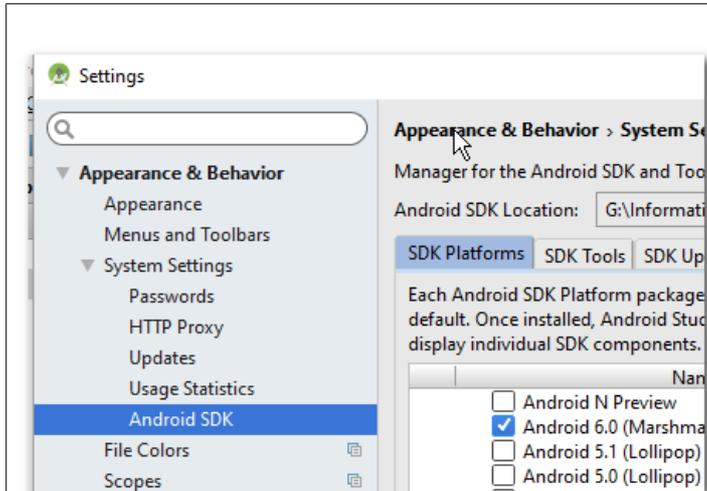


Settings.

Die Settings finden Sie auch in Android Studio in der Menü-Leiste File →

In der Baumstruktur Links finden Sie alle Einstellungsmöglichkeit.

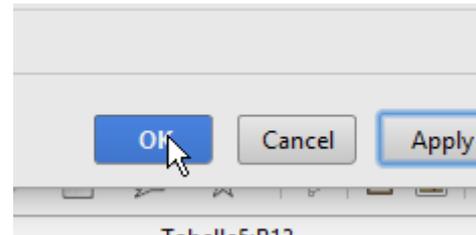
Beispiel:
 Unter anderem finden Sie dazu die Settings für die Android SDK in der Baumstruktur unter →



Android SDK

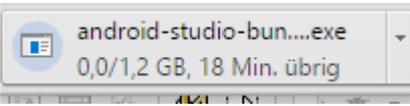
Appearance & Behaviour → System Settings → Android SDK.

Falls Sie hier Änderungen in der Auswahl tätigen bestätigen Änderungen mit → Apply und schließen das Konfigurationsfenster mit einem Klick auf die Schaltfläche → OK.



Fertig!

7.3 Hinweise

<p>Android Studio The Official IDE for Android</p> <p>Android Studio provides the fastest tools for building apps on every type of Android device.</p> <p>World-class code editing, debugging, performance tooling, a flexible build system, and an instant build/deploy system all allow you to focus on building unique and high quality apps.</p> <p>DOWNLOAD ANDROID STUDIO 2.1 FOR WINDOWS (1181 MB)</p> 	<p><i>Android Studio download.</i></p> <p>Das Android Studio ist aktuell die offizielle Entwicklungsumgebung für die Entwicklung von Anwendungen für mobile Endgeräte mit Android Betriebssystem.</p> <p>Die aktuellste Version (für Windows) finden Sie zum Download auf den Entwicklerseiten:</p> <p>https://developer.android.com/studio/index.html</p> <p>Es sind auch Versionen für MAC OSX und Linux (Ubuntu) zur Verfügung.</p>
<p>System Requirements</p> <p>Windows</p> <ul style="list-style-type: none"> • Microsoft® Windows® 7/8/10 (32- or 64-bit) • 2 GB RAM minimum, 8 GB RAM recommended • 2 GB of available disk space minimum, 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image) • 1280 x 800 minimum screen resolution • Java Development Kit (JDK) 8 • For accelerated emulator: 64-bit operating system and Intel® processor with support for Intel® VT-x, Intel® EM64T (Intel® 64), and Execute Disable (XD) Bit functionality 	<p><i>Systemvoraussetzungen.</i></p> <p>Die Voraussetzungen an das System (für Windows) sind nebenstehend aufgeführt.</p>
	<p><i>Warum zu Android Studio wechseln?</i></p> <p>Für den Fall, dass Sie noch mit Eclipse und den entsprechenden Erweiterungen arbeiten, findet man zwischenzeitlich auf den Entwicklerseiten</p>

ADT Plugin Release Notes

The Eclipse ADT plugin is no longer supported. **Android Studio** is now the official IDE for Android, so you should migrate your projects to receive the latest developer tools. For help moving projects, see [Migrating to Android Studio](#).

einen Hinweis:

Darin wird empfohlen, die Entwicklungsumgebung mittelfristig zu wechseln, um die Versorgung mit Updates für die Zukunft sicherzustellen.



Umgebungsvariablen setzen.

Die Benutzervariablen und Systemvariablen für die JDK und SDK sollten gesetzt sein.

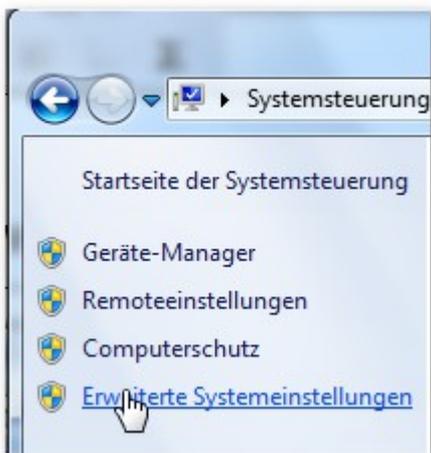
Die Namen der Umgebungsvariablen sind:
→ ANDROID_SDK_HOME
→ JAVA_HOME

Getestet auf Computern mit:

1. 64 Bit Intel Core i3 2120 Prozessor
2. Windows 7, 32 Bit Version

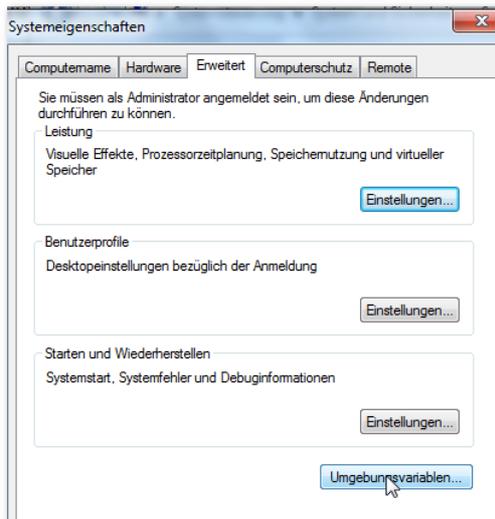
Außerdem müssen diese Ressourcen verfügbar, also vorhanden sein.

Wählen Sie dazu die Option Start → Systemsteuerung → System und Sicherheit → System.



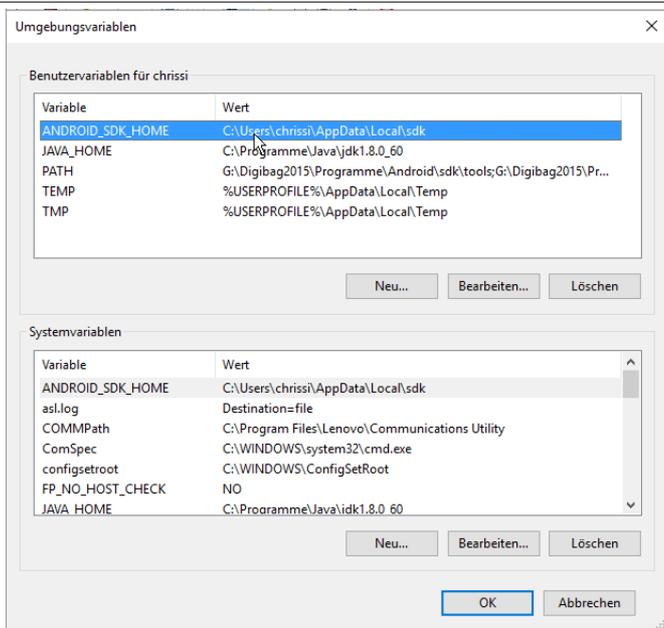
Erweiterte Systemeinstellungen.

Klicken Sie im linken Frame auf die Option „Erweiterte Einstellungen“.



Umgebungsvariablen.

Klicken Sie auf die Schaltfläche „Umgebungsvariablen“.



Information über JDK und SDK.

Um die Pfadangaben für die Umgebungsvariablen (Benutzer- und Systemvariablen) erfolgreich anzugeben müssen die Verzeichnisse für die „sdk“ und „jdk“ bekannt sein!

Hinweis:

Sie haben im Installationsprozess die Möglichkeit die vorgeschlagenen Pfade individuell anzupassen. Merken Sie sich bitte die Pfade, um die Umgebungsvariablen setzen zu können.

Java Development Kit (jdk):

Im Programm-Verzeichnis befindet sich i.d.R. im Unterverzeichnis „Java“ die hier verwendete „jdk“. Sie können dieses Verzeichnis problemlos an die gewünschte Stelle kopieren.

Hinweis:

Falls Sie eine neuere Version der „jdk“ verwenden oder das Installationsverzeichnis variiert müssen Sie den Pfad entsprechend der Realität anpassen.

Der aktuelle Java Development Kit (Java SE, Oracle, Download) können Sie sich auf den Oracle-Seiten je nach Betriebssystem auswählen, herunter-

Annahme die Pfade sind:

1. Variante 1.0: Standardverzeichnis liegt lokal
→ C:\Users\<<Benutzer>\AppData\Local\sdk

2. Oder ein individuelles sdk-Verzeichnis:

Variante 1.1: Lokal

→ C:\Program Files\Android\sdk

Variante 2: Digitale Tasche

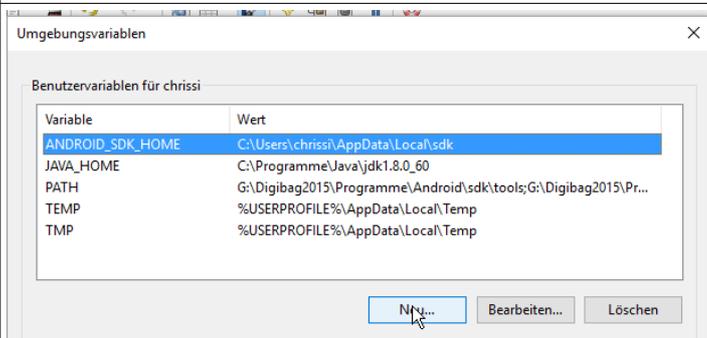
→ G:\Informatikstick2016\Programme\Android\sdk

3. Das Verzeichnis jdk liegt i. d. R. lokal im Programmverzeichnis auf dem Laufwerk C:
→ C:\Programme\Java\jdk1.8.0_60

Bitte entscheiden Sie sich für eine Variante!

terladen und installieren:

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>



Neue Benutzervariablen für die SDK setzen.

Variante 1.0: Standard-Verzeichnis

Klicken Sie im Bereich → Benutzervariablen auf die Schaltfläche → Neu und machen Sie folgende Angaben:

Name der Variablen:

Wert der Variablen:

Angabe:

→ ANDROID_SDK_HOME
→ C:\Users\<Benutzer>\AppData\Local\sdk

Kontrollieren Sie zuvor ob das sdk-Verzeichnis einen Unterordner → .android enthält.

Variante 1.1: Lokal, individuell

Für den Fall, dass Sie ein individuelles Verzeichnis gewählt haben:

Name der Variablen:

Wert der Variablen:

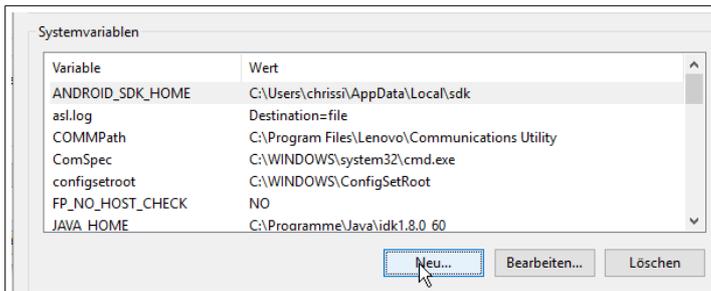
Angabe:

→ ANDROID_SDK_HOME
→ C:\Program Files\Android\sdk

Variante 2: Digitale Tasche

Angabe:

→ ANDROID_SDK_HOME
→ C:\Program Files\Android\sdk



Neue Systemvariablen für die SDK setzen.

Variante 1.0: Standard-Verzeichnis

Name der Variablen:

Wert der Variablen:

Klicken Sie im Bereich → Systemvariablen auf die Schaltfläche → Neu und machen Sie erneut die folgenden Angaben:

Angabe:

→ ANDROID_SDK_HOME

→ C:\Users\<Benutzer>\AppData\Local\sdk

Variante 1.1: Lokal, individuell

Für den Fall, dass Sie ein individuelles Verzeichnis gewählt haben:

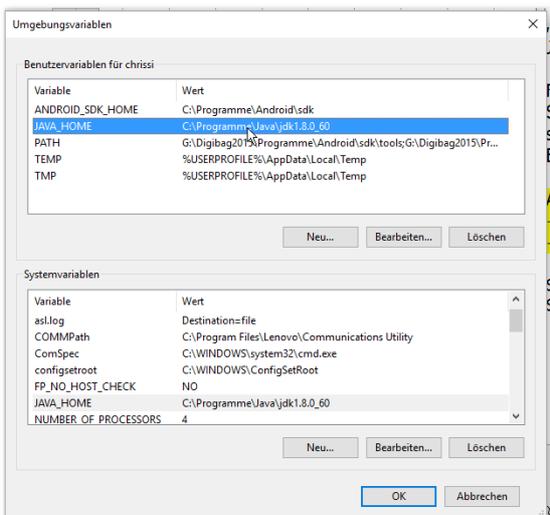
Name der Variablen:

Wert der Variablen:

Angabe:

→ ANDROID_SDK_HOME

→ C:\Program Files\Android\sdk



Neue Benutzer- und Systemvariablen für die JDK.

Falls für die JDK bisher keine Benutzer- und Systemvariablen (JAVA_HOME) vorhanden ist, sollten Sie den Vorgang wiederholen und die Benutzer- und Systemvariablen setzen:

Angabe:

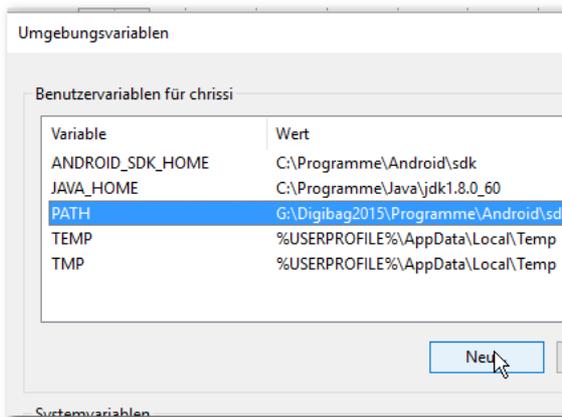
→ JAVA_HOME

→ C:\Programme\Java\jdk1.8.0_60

Schließen Sie den Vorgang mit einem Klick auf die Schaltfläche → OK ab.

Beachten Sie:

Je nach Betriebssystem müssen Sie die JDK in der 32Bit bzw. 64Bit Version installieren. Googeln Sie dazu einfach nach den Begriffen → Java SE download.

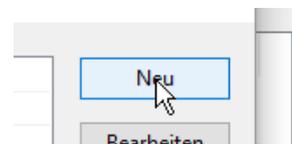


PATH-Variablen setzen.

Umgebungsvariable bearbeiten

G:\Digibag2015\Programme\Android\sdk\tools
G:\Digibag2015\Programme\Android\sdk\platform-tools
C:\Program Files (x86)\Sophos\Sophos SSL VPN Client\bin
G:\Programme\Android\sdk\tools
G:\Programme\Android\sdk\tools\platform-tools
C:\Programme\Android\sdk\tools
C:\Programme\Android\sdk\tools\platform-tools

Über die Schaltfläche → Neu

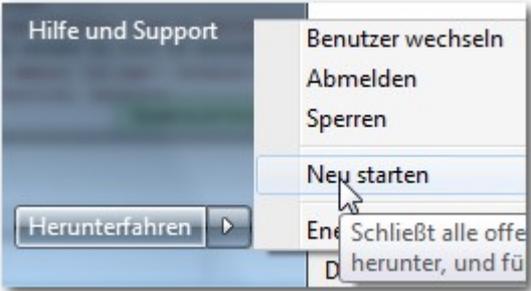


hinzufügen.

Variante 1.1.: Lokal, individuell

→ C:\Programme\Android\sdk\tools

→ C:\Programme\Android\sdk\tools\platform-tools

	<p>Variante 2: Digitale Tasche Falls vorhanden...</p> <p><code>G:\Informatikstick2016\Programme\Android\jdk\platform-tools</code> <code>G:\Informatikstick2016\Programme\Android\jdk\tools</code></p>
 <p>The screenshot shows a Windows Start menu with the text 'Hilfe und Support' at the top. Below it is a search bar with the text 'Herunterfahren' and a right-pointing arrow. A context menu is open over the search bar, listing the following options: 'Benutzer wechseln', 'Abmelden', 'Sperren', 'Neu starten' (highlighted with a mouse cursor), 'Ene Schließt alle offe', and 'D herunter, und fü'.</p>	<p>Start Sie Ihren Rechner neu, damit die neu konfigurierten Einstellungen erkannt werden.</p> <p>Melden Sie sich als normaler Benutzer an und versuchen Sie die Anwendung zu starten.</p>

7.4 Fehler

„HAX is not working and emulator runs in emulation mode“

```

Inode size: 256
Journal blocks: 1024
Label:
Blocks: 16896
Block groups: 1
Reserved block group size: 7
Created filesystem with 11/4224 inodes and 1302/16896 blocks
emulator: WARNING: Requested RAM size of 1536MB is too large for your environment, and is reduced to 1152MB.
emulator: device fd:604
HAX is not working and emulator runs in emulation mode
emulator: The memory needed by this VM exceeds the driver limit.
Cannot set up guest memory 'pc.ram': Invalid argument
Error accepting connection, aborting
  
```

Gradle build finished in 6 min 18 sec

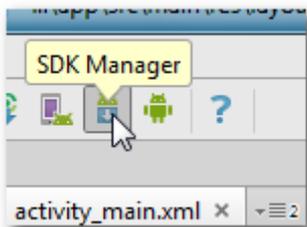
HAXM Fehler.

Ein Fehler kann bereits bei der Installation auftreten oder aber er tritt auf, wenn erstmals ein virtuelles, mobiles Endgerät emuliert wird.

Hinweis:

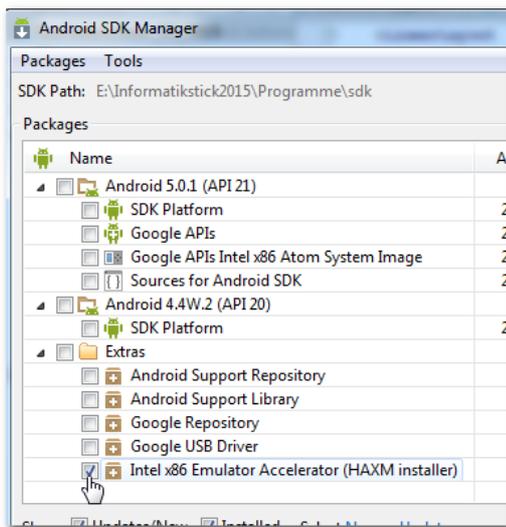
Dazu muss die → Virtualization Technology bei manchen neueren Rechner eventuell im Bios aktiviert (→ enabled) werden.

Starten Sie dazu den Rechner neu und wechseln mit den Tasten F2, Delete oder ESC ins Bios. Im Bereich der CPU/Prozessor-Angaben suchen Sie nach Bezeichnungen, wie VT-x, Virtualization Technology oder VT-d. Diese Funktionalitäten müssen ggf. aktiviert, also → enabled werden.



HAXM nachinstallieren.

Öffnen Sie Android Studio und klicken Sie auf die Schaltfläche für den → SDK Manager.



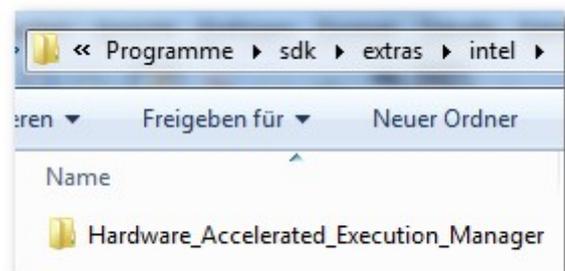
HAXM installer herunterladen.

Normalerweise erfolgt die Installation von HAXM bei der Erstinstallation des Android Studio zwischenzeitlich automatisch.

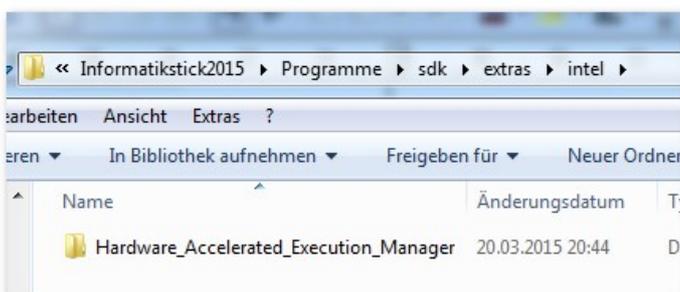
Für den Fall, dass das unter Extras aufgeführte Paket noch nicht heruntergeladen wurde, können Sie dies nun über den SDK-Manager nachholen.

Für diesen Fall laden Sie den Sie den fehlenden HAXM installer herunter.

→ Das Verzeichnis wird im Ordner „sdk“ abgelegt:



In jedem Fall muss die Installation dieses Pakets als Administrator erfolgen. Folgen Sie weiter den Angaben.

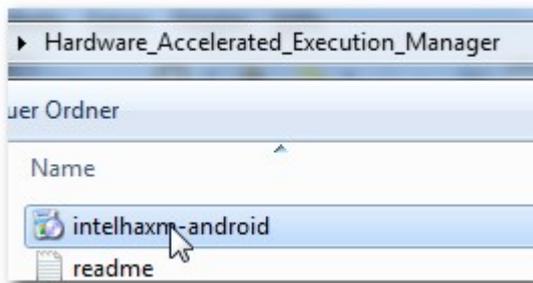


HAXM-Verzeichnis finden.

Sie benötigen dazu Administratorenrechte. Melden Sie sich deshalb als Administrator an.

Wechseln Sie ins Verzeichnis:

→ Programme → sdk → extras → intel



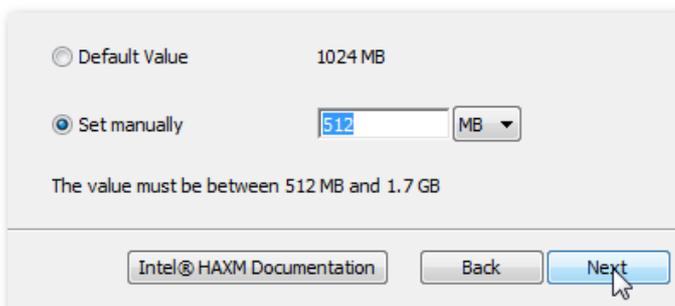
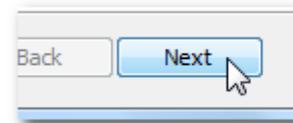
HAXM installieren.

Öffnen Sie dazu das Verzeichnis:
→ Hardware Accelerated Execution Manager
und klicken Sie die im Verzeichnis enthaltene Datei „intelhaxm-android.exe“ (Anwendung) doppelt an.



HAXM Assistent.

Klicken Sie im ersten Schritt auf die Schaltfläche → Next



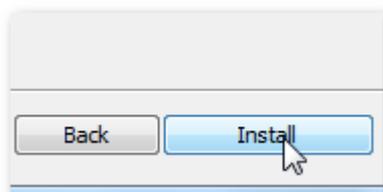
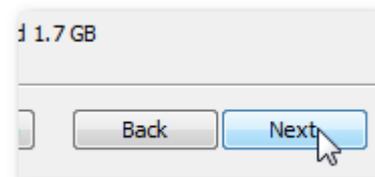
HAXM Arbeitsspeicher manuell verkleinern.

Setzen Sie den Arbeitsspeicher manuell auf 512 MB. Damit stellen Sie sicher, dass Sie relativ viele Geräte zum testen verwenden können.

Hinweis:

Für Anwendungen die viel Arbeitsspeicher benötigen, kann diese Einstellung eventuell an Grenzen stoßen. → Andere Anwendungen werden langsam.

Klicken Sie dann auch die Schaltfläche → Next

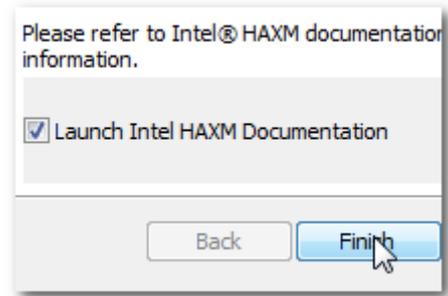


Klicken Sie auf die Schaltfläche → Install, um die Installation durchzuführen.

Warten Sie einen Moment bis die Installation durchgeführt wurde.

Klicken Sie Abschließend auf die Schaltfläche →

Finish



Schließen Sie die Intel-Seite und Starten Sie Ihren Rechner erneut.

7.5 Top 10 der Hilfestellungen

Maßnahmen bei Anzeigefehlern.

1. Menü-Leiste → Built → Clean Project
2. Menü-Leiste → Built → Rebuilt Project
3. Preview Editor → Refresh Rendering → Sync-Button
4. Menü-Leiste → File → Open File → styles.xml → Add the word „Base.“ to the beginning of the theme name so that it reads "Base.Theme.AppCompat.Light.DarkActionBar"
5. Menü-Leiste → Tools → Android → Sync Project with Gradle Files
6. Menü-Leiste → File → Invalidate Caches / Rebuilt

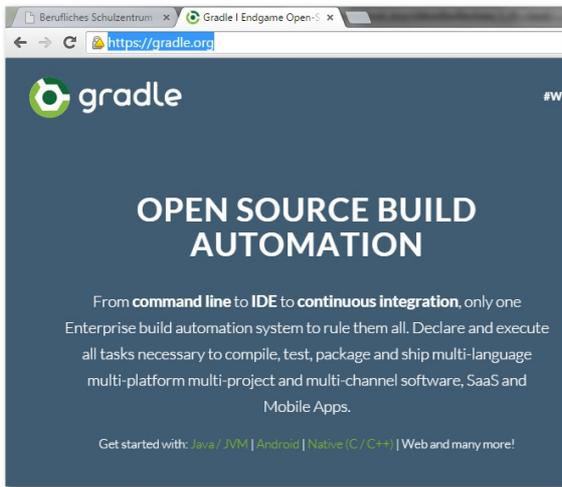
Maßnahmen bei fehlendem Verständnis:

1. Methode markieren → Menü-Leiste → View → Quick Documentation

Sonstiges:

1. Quellcode einrücken → STRG + A → STRG + I
2. Getter und Setter erzeugen → Kontext-Menü → Generate → Getters and Setters
3. Methoden Überschreiben → Kontext-Menü → Generate → Override Methods → Methode auswählen → Implementierung modifizieren.
4. Methoden generieren (Override/Implement) → ALT + Einfg
5. Import einer Klasse → ALT + ENTER

7.6 Gradle

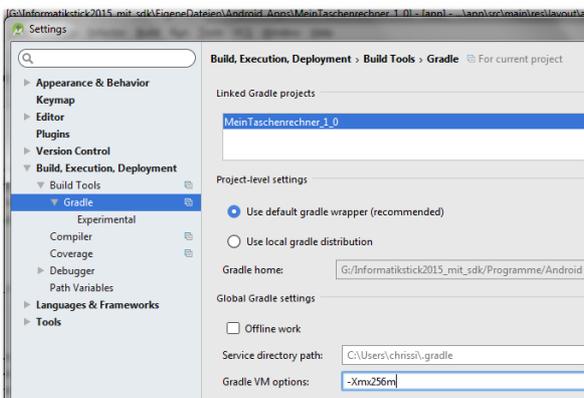


<https://gradle.org/>

```
Terminal
+ Microsoft Windows [Version 6.1.7601]
+ Copyright (c) 2009 Microsoft Corporation. Alle Rechte vorbehalten.
G:\Informatikstick2015_mit_sdk\EigeneDateien\Android_Apps\MeinTaschenrechner_1_0
>gradlew clean build
```

Im Terminal:

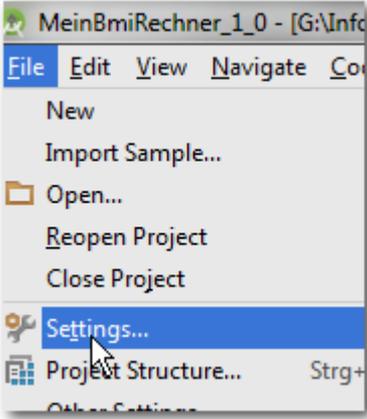
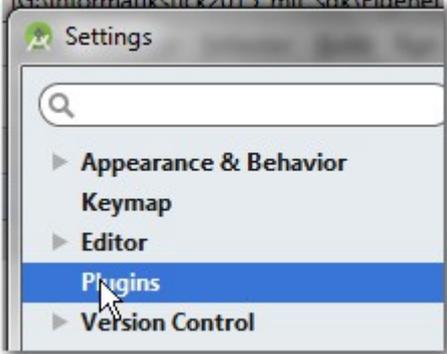
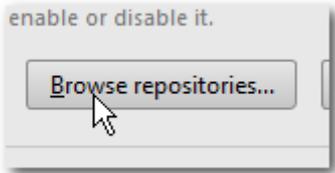
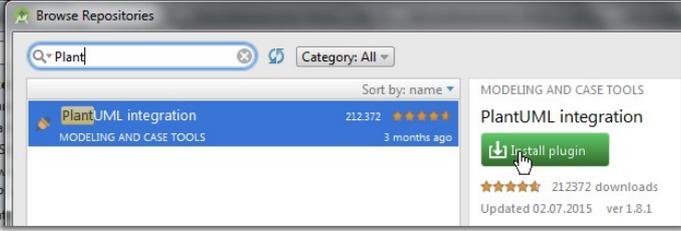
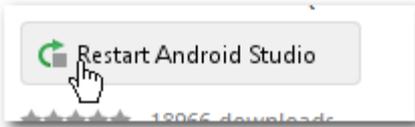
→ gradlew clean build

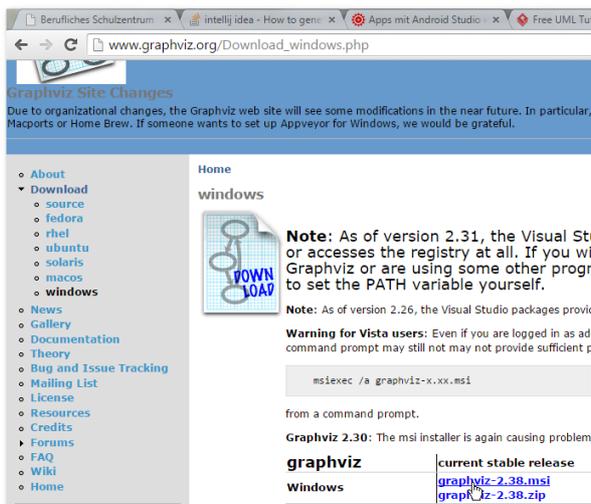


Gradle VM options:

-Xmx256m

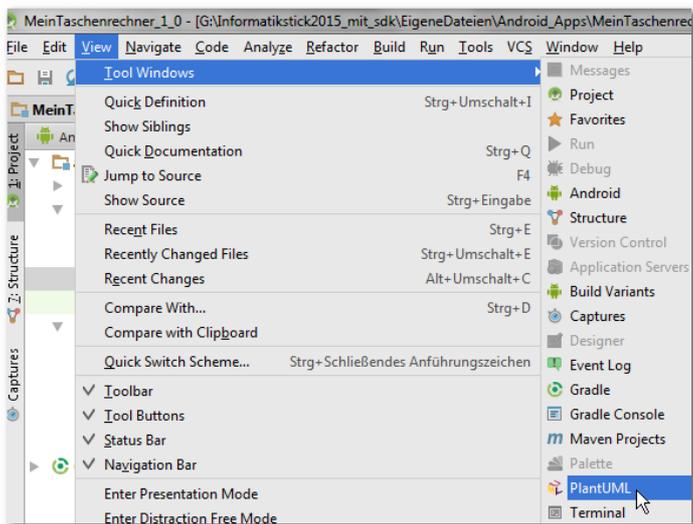
7.7 UML-Modelling PlugIn

	<p><i>File</i> → <i>Settings</i>.</p>
	<p><i>Fenster Settings</i> → <i>Plugins</i>. Schaltfläche Browse repositories klicken</p> 
	

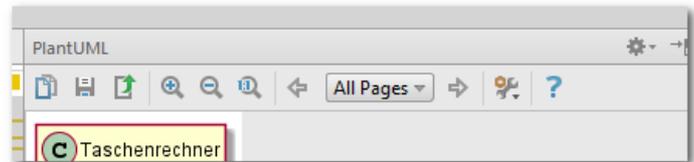


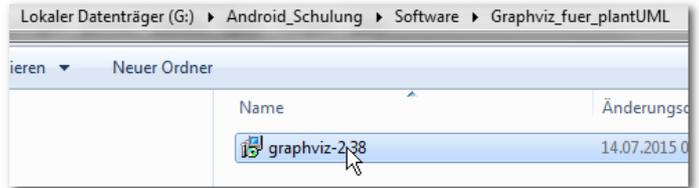
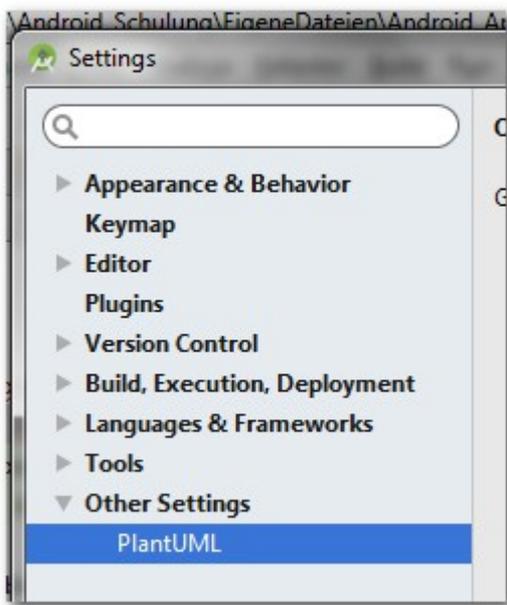
Erweiterung graphviz.

http://www.graphviz.org/Download_windows.php



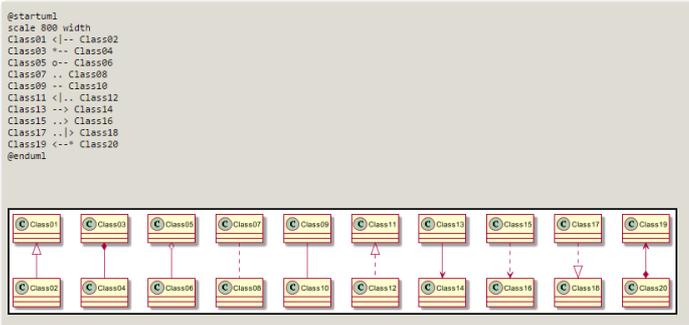
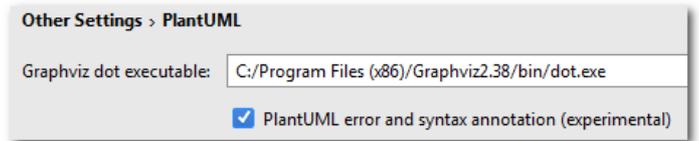
PlantUML View.



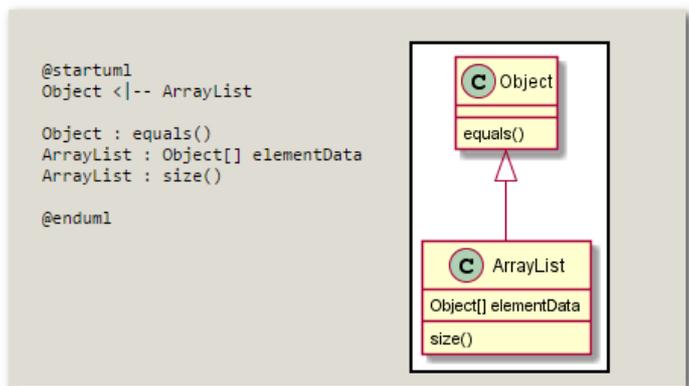


Installieren

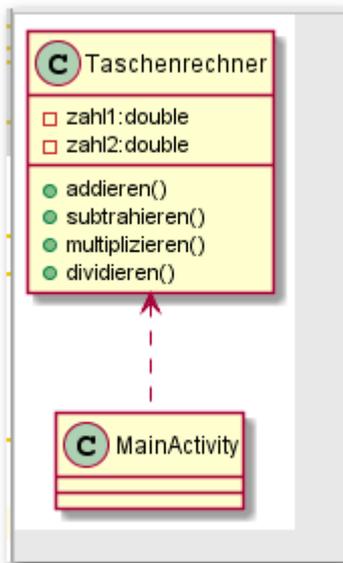
Dann den Pfad setzen



UML Notation Einführung → Pfeile.



UML Notation Einführung → Klassen.

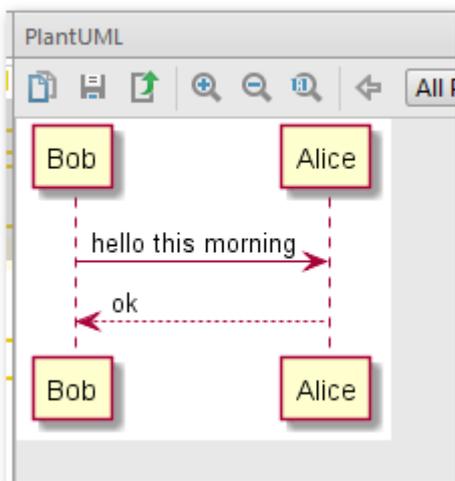


Übung → Taschenrechner

```

*
* @startuml
* Taschenrechner <.. MainActivity
* Taschenrechner : - zahl1:double
* Taschenrechner : - zahl2:double
* Taschenrechner : + addieren()
* Taschenrechner : + subtrahieren()
* Taschenrechner : + multiplizieren()
* Taschenrechner : + dividieren()
* @enduml
* * */
    
```

UML-Klasse



```

/**
 * Created by Chrissi on 07.04.2015.
 * @startuml
 * Bob -> Alice : hello this morning
 * Alice --> Bob : ok
 * @enduml
 *
    
```

Sequenzdiagramm