

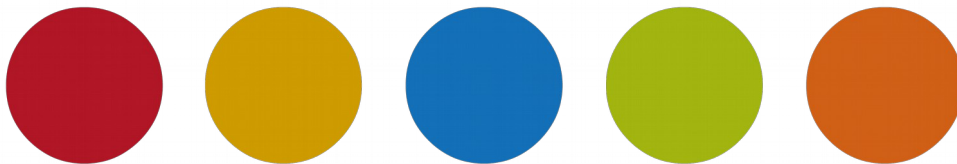
Friends-App

Skript 2016

Konfigurations- und Schulungsunterlagen

Schulung:	Didaktische Ansätze zur Android-Programmierung
Referent:	Christine Janischek

Stand: 7. Jun 2016



© [Christine Janischek](#)

Inhaltsverzeichnis

1 Allgemeines.....	3
2 Das Projekt Friends.....	5
2.1 Überblick.....	5
2.2 Anwendungsfälle.....	6
2.3 Grundlagen: Projekt erzeugen.....	9
2.4 View: Layouts, Komponenten & XML für die Benutzeroberfläche.....	12
2.5 Modell: Implementierung der Fachklasse Friend.....	29
2.6 Controller: Steuerung und Zugriff auf die Datenbank.....	34
2.6.1 Daten anzeigen.....	35
2.6.2 Daten einfügen.....	55
2.6.3 Daten entfernen.....	64
2.6.4 Daten aktualisieren.....	76

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 für die Projekte (für Fortgeschrittene):

→ [Alle Arbeitsmaterialien 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

Weitere Skripte und Sources online:

[Einführung in die Programmierung von Android Apps anhand klassischer Unterrichtsbeispiele](#)

[Fortgeschrittene Apps mit Android Studio erstellen](#)

[Android Apps erstellen](#)

[Java Programmieren im Unterricht](#)

[Java-E-Learning zum Unterricht](#)

[Objektorientierte Sytementwicklung in Java](#)

[Dynamische Webseiten mit PHP \(objektorientiert\) programmieren](#)

[Webprogrammierung im Unterricht](#)

[Entwickeln mit Javascript Framework \(jQuery, JQuery mobile\)](#)

[Einführung in PHP und die WordPress-Theme-Entwicklung](#)

[Relationale Datenbanken](#)

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

[BUC01]

Buchalka, Tim, "Master Android 6.0 Marshmallow Apps Development Using Java", timbuchalka.com, 2016, Udemy Course

[KUE01]

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

[WAC00]

Wagner, Chris, "Das Android SQLite Datenbank Tutorial", <http://www.programmierenlernenhq.de/android-sqlite-datenbank-tutorial/>, 2016, programmierenlernenhq.de, zuletzt getestet am 09.04.2016

[FLE00]

Flowers, Eric, "WeatherIcons", <https://github.com/erikflowers/weather-icons/tree/master/font>, 2016, <http://www.helloerik.com>, zuletzt getestet am 26.04.2016

[HAA00]

Hathibelagal, Ashraff „Create a Weather App on Android“, <http://code.tutsplus.com/tutorials/create-a-weather-app-on-android--cms-21587>, zuletzt getestet am 26.04.2016

[AZF00]


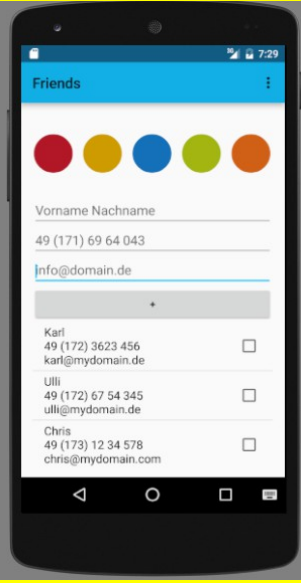
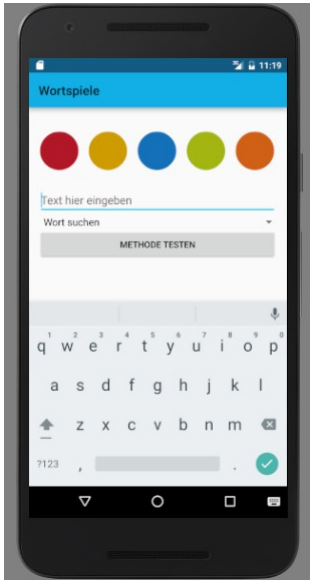
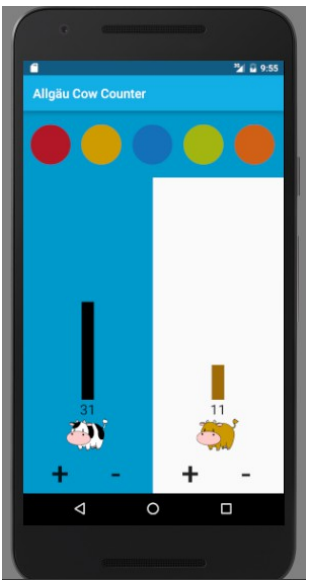
Azzola, Francesco „Android: Build real weather app: JSON, HTTP and Openweathermap“, <https://www.javacodegeeks.com/2013/06/android-build-real-weather-app-json-http-and-openweathermap.html>, 2013, zuletzt getestet am 30.04.2016

2 Das Projekt Friends

2.1 Überblick

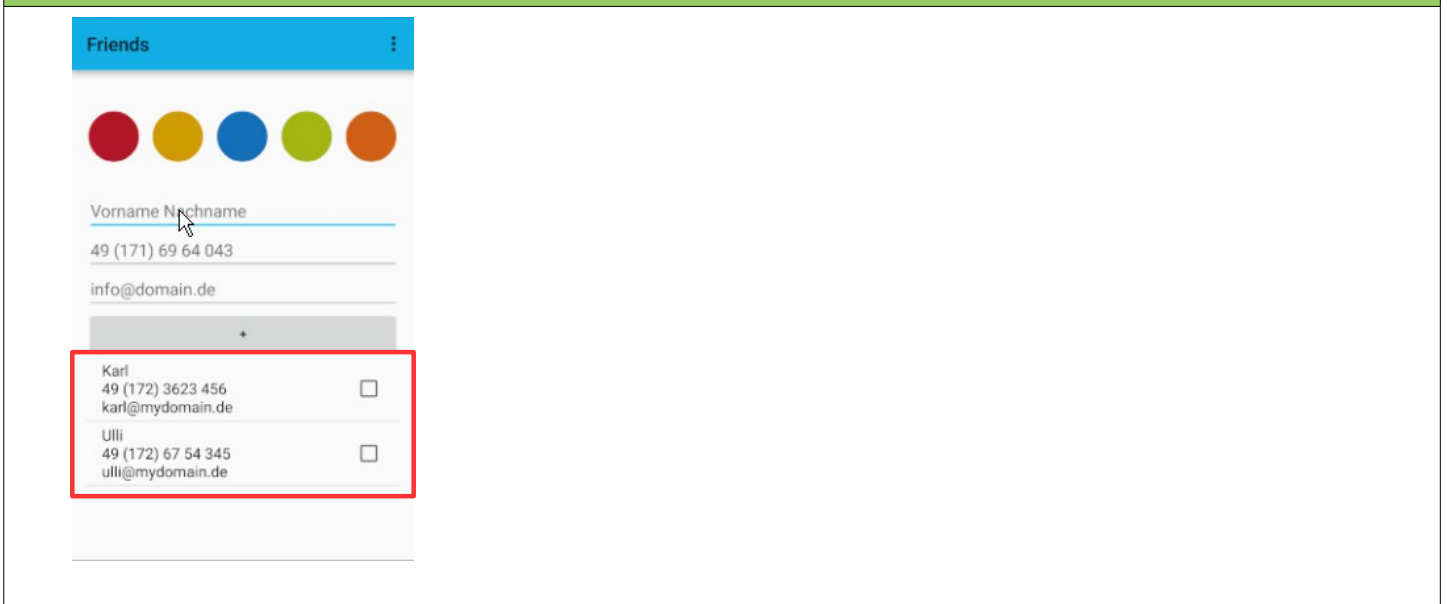
Friends App:

Das Projekt soll an einer einfachen Datenbank zeigen auf welche Weise die Verwaltung von Daten realisiert werden kann. Die klassischen Datenbankoperationen (Anzeigen, Einfügen, Aktualisieren, Löschen) werden dazu in die Anwendung integriert. Folgen Sie für die Umsetzung der Friends-App den Schritt-für-Schritt-Anleitungen im Anschluss an den Überblick und bedenken Sie, dass Sie auf keinen der erläuterten Schritte verzichten können.

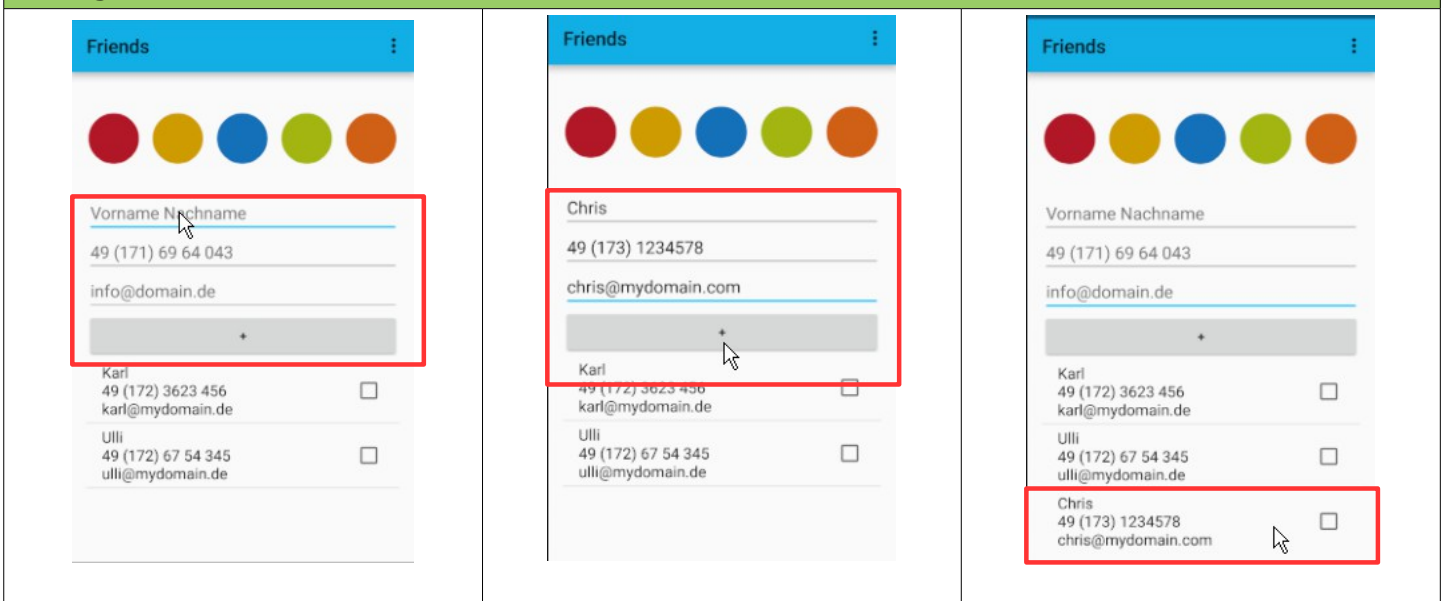
Weather App	Friends App	Wortspiele App	Cow-Counter App
			
			
<p>Tags: OpenWeatherMap, http, Netzwerk, JSONObject, Fragment, Schrift, Exceptions, Fehlerbehandlung, Thread, Dialog</p>	<p>Tags: Datenbankzugriff, SQLite, ListView, Menüs, Dialog</p>	<p>Tags: Stringverarbeitung, Kontrollstrukturen, Spinner, Dialoge, Fallunterscheidungen, Schleifen, Algorithmen</p>	<p>Tags: Zähler, Inkrementieren, Dekrementieren, Layouts, Balkendiagramm</p>

2.2 Anwendungsfälle

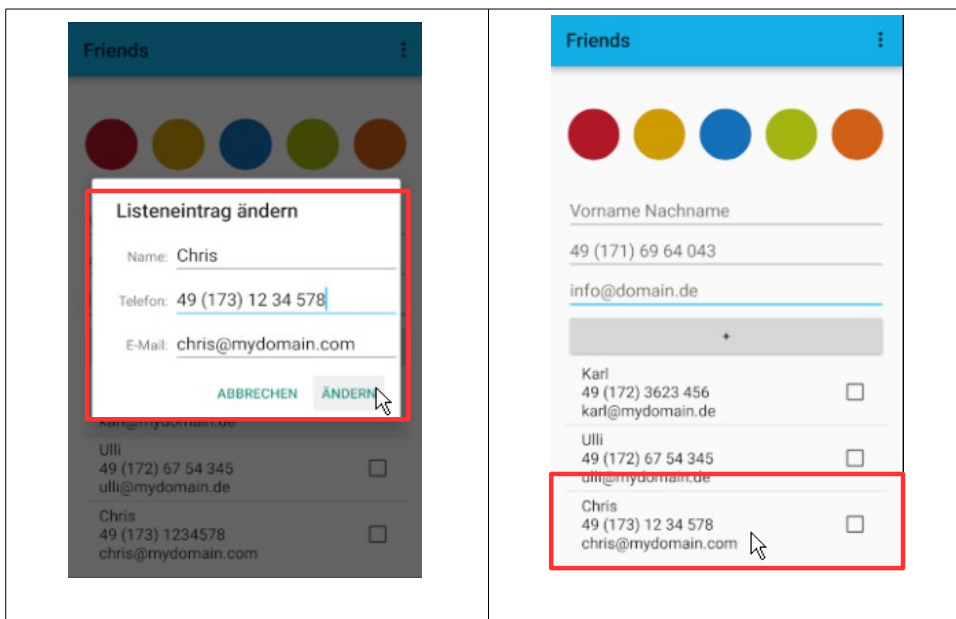
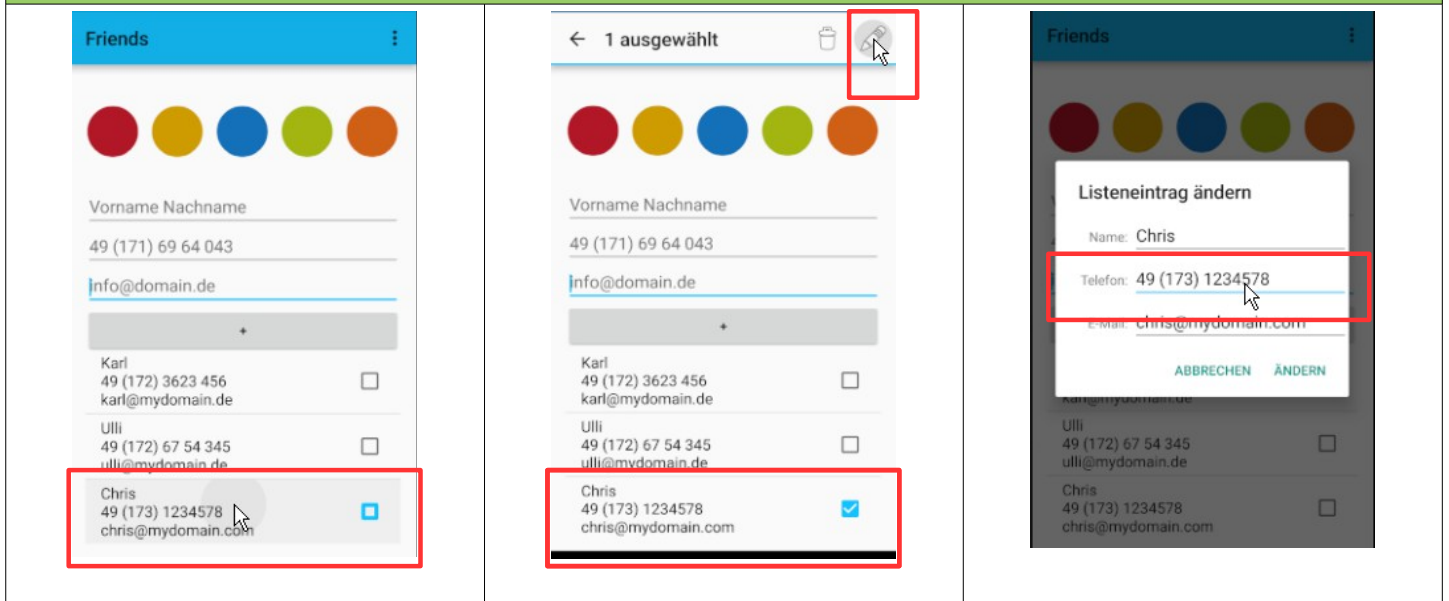
Anzeigen



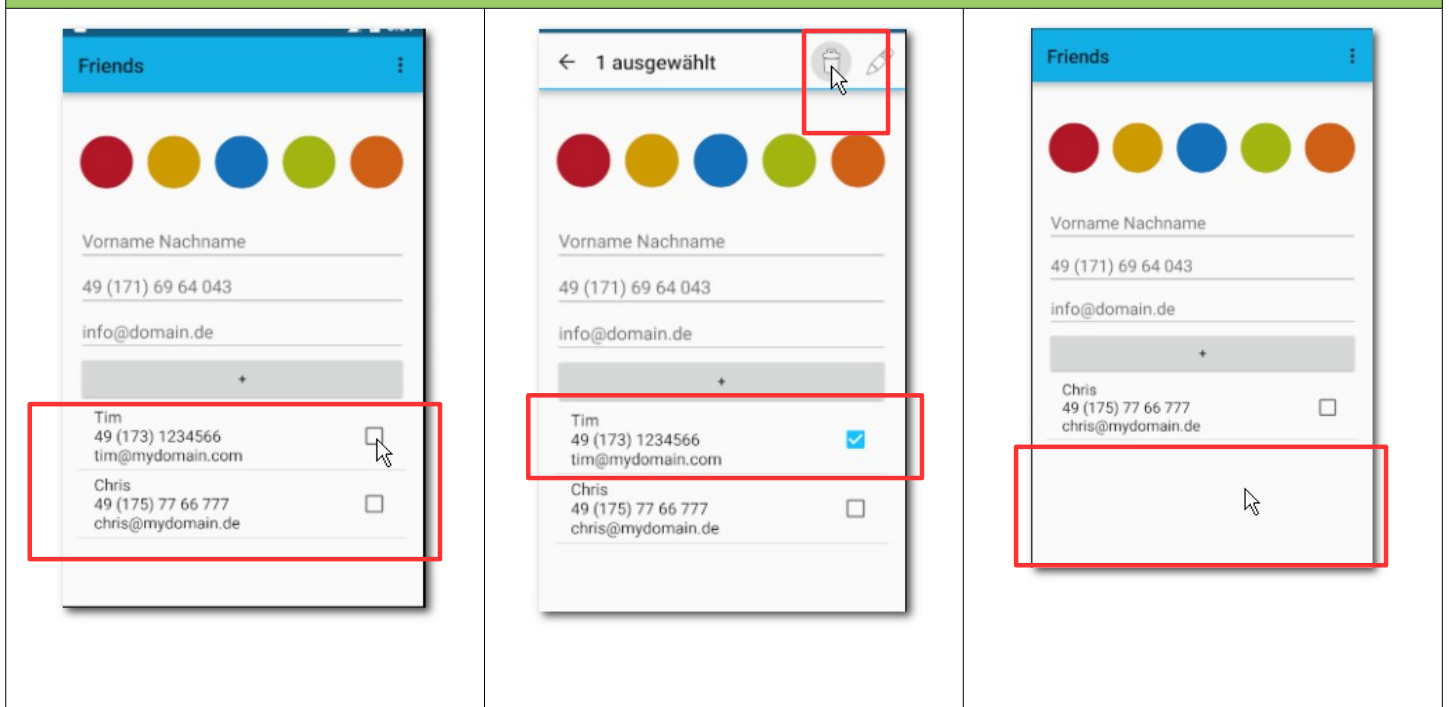
Einfügen



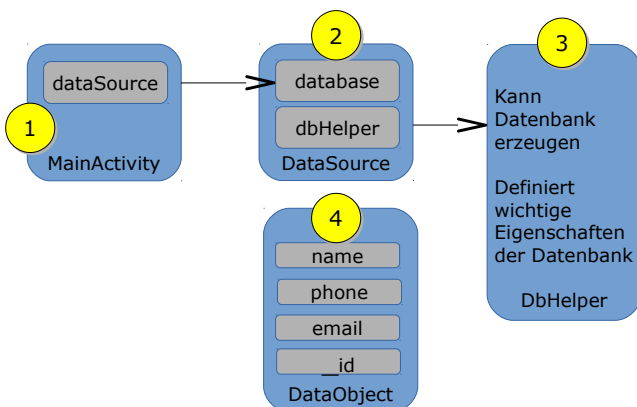
Aktualisierung



Löschen



Realisierung des Datenbankzugriffs



Überblick zu den notwendigen Klassen

MainActivity.java (1)

In dieser Klasse werden wir unsere Datenquelle öffnen und wieder schließen. Später werden wir von hier aus alle Datenbankzugriffe mit Hilfe der Datenquelle steuern.

FriendsDataSource.java (2)

Arbeiterklasse. Objekte dieser Klasse übernehmen die Steuerung aller Datenbankoperationen. Sie ist unsere Datenquelle und besitzt daher eine dauerhafte Verbindung zur SQLite Datenbank.

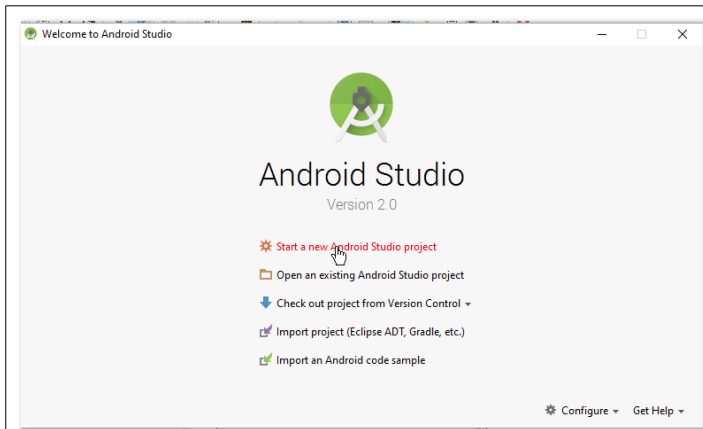
FriendsDbHelper.java (3)

Diese Klasse ist unsere Hilfsklasse und von der SQLiteOpenHelper-Klasse abgeleitet. Mit ihrer Hilfe werden wir u.a. die Tabelle in die Datenbank einfügen. Vorher legen wir aber die wichtigsten Datenbank- und Tabellen-Eigenschaften in String-Konstanten in dieser Klasse fest.

Friend.java (4)

Fachklasse.

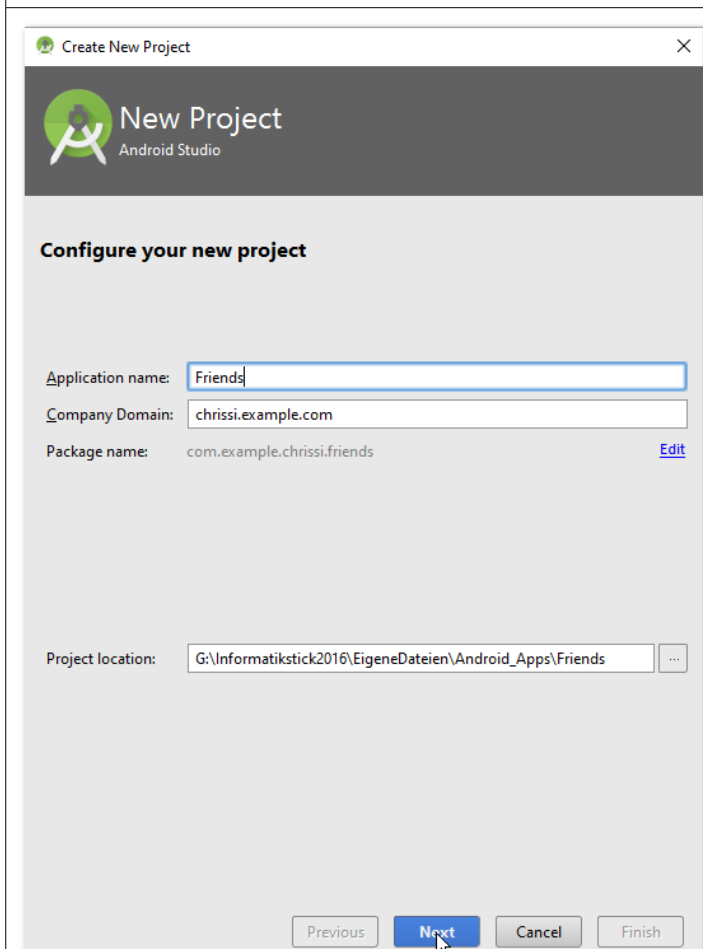
2.3 Grundlagen: Projekt erzeugen



Ein Neues Projekt erzeugen.

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

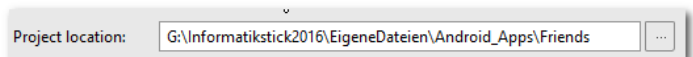
Um ein neues Projekt zu erzeugen, wählen Sie im Quick Start-Menü die Option → Start a new Android Studio project.



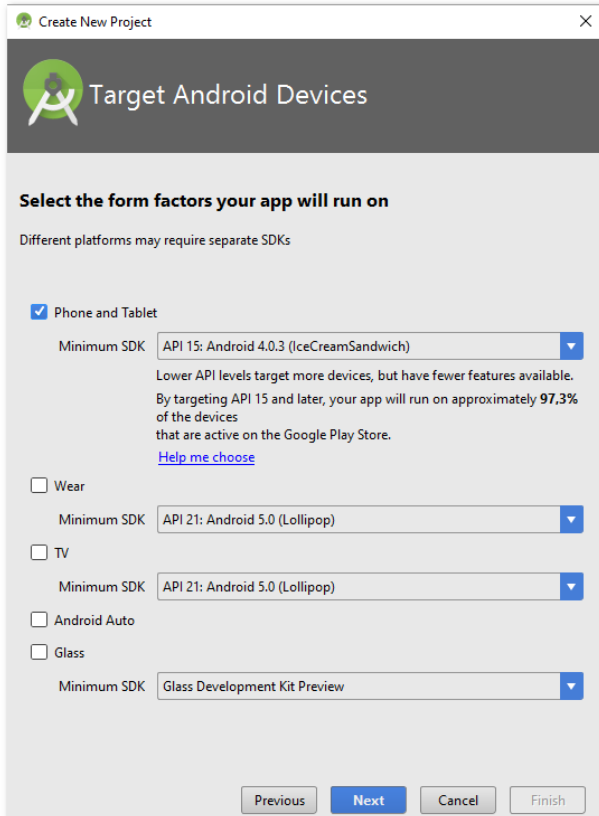
Legen Sie nun schrittweise die Eigenschaften für Ihr neues Android-Projekt fest.

Geben Sie dazu die nebenstehend angezeigten Angaben für

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



Je nach Konfiguration können diese Angaben variieren

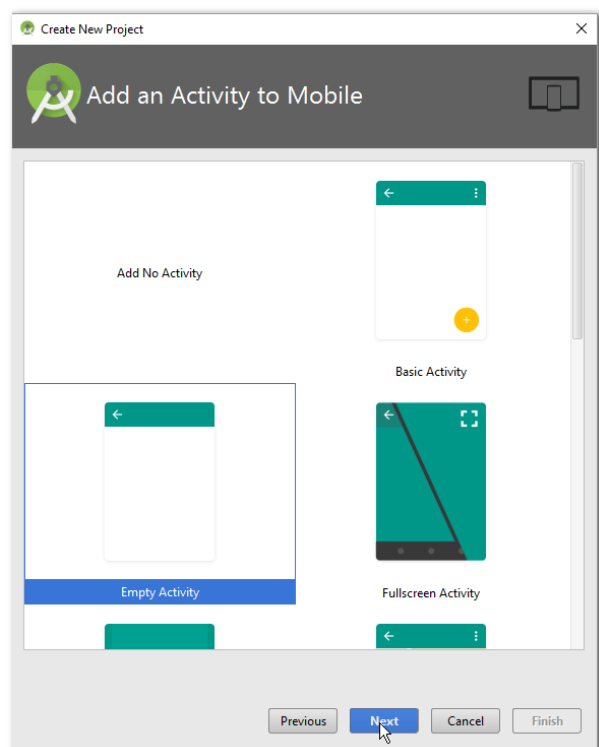


Laufzeitumgebung unserer Anwendung.

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.



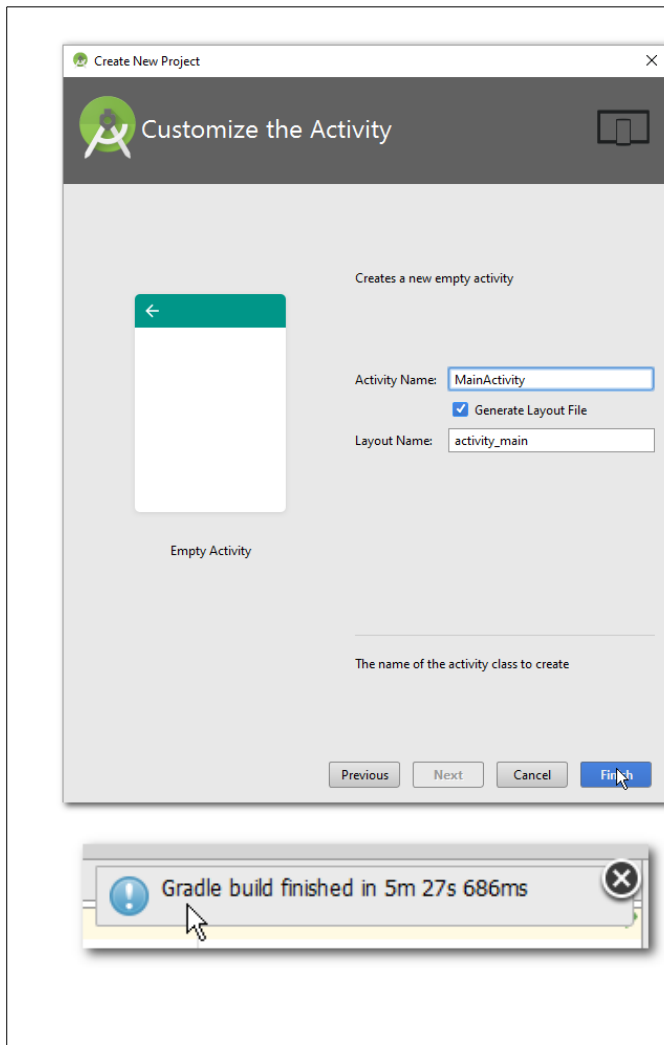
Aktivität wählen.

Im ersten Schritt nutzen wir die einfachste Form zur Steuerung von Ereignissen. Die → Empty Activity. Wählen wir diese Aktivität bekommen wir einige Standards mitgeliefert.

Wir wählen die → Empty Activity und klicken Sie auf die Schaltfläche → Next.

Hinweis:

Alternativ können wir auch die Option → Add No Activity wählen und können dann nachträglich alle Maßnahmen für die Implementierung der Activity selber treffen.



Aktivität anpassen.

Activities enthalten die Ereignissteuerung für einen bzw. eine ganze Reihe von zusammengehörenden Vorgänge (Interaktionen, Verhaltensweisen) einer App.

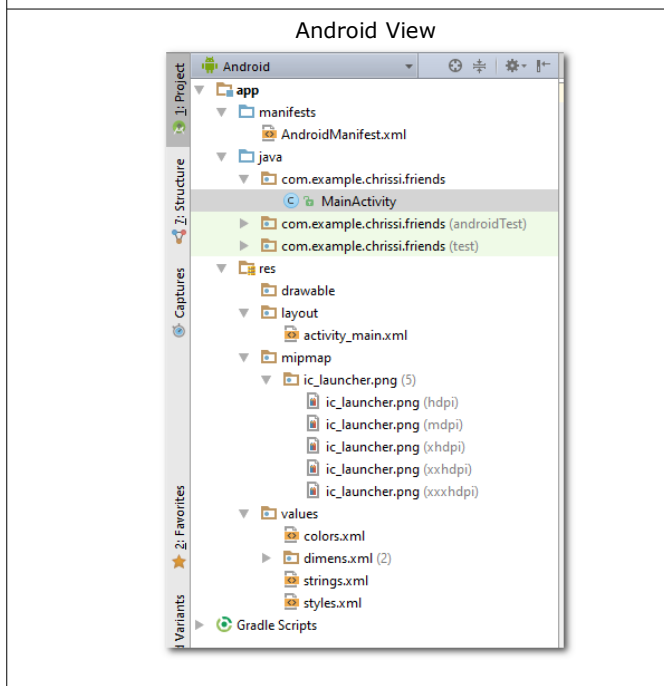
Wir belassen die Standard-Einstellungen. Klicken Sie dazu auf die Schaltfläche → Finish.

Mit dem Klick auf → Finish wird die Projektstruktur (Architektur) erzeugt.

Hinweis:

Je nach Rechnerausstattung kann die Erzeugung einen Moment dauern.

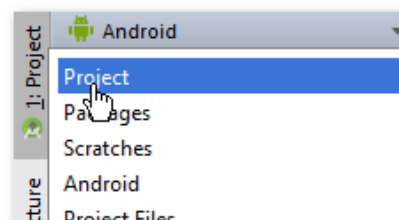
Android Studio nutzt u.a. das Gradle-PlugIn als Buildsystem. Gradle ist dabei ein Werkzeug das komplett in Android Studio integriert ist und zur Build-Automatisierung und - Management genutzt wird. Jede Anwendung muss nach jeder Änderungen im Quellcode neu erzeugt werden, dabei werden außer der Kompilierung viele weitere Bindungsprozesse (z.B. mit den Ressourcen) durchgeführt.



Projektstruktur am Anfang.

Im Anschluss an den abgeschlossenen Build-Prozess finden Sie im linken Frame die folgende Projektstruktur vor.

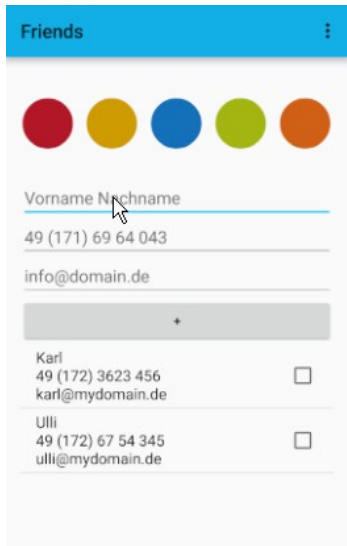
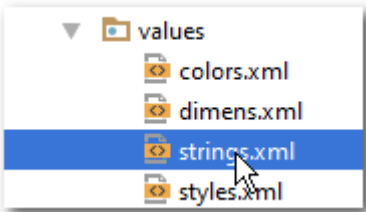

Klicken Sie oberhalb auf den Androiden um die Projektansicht (→ Project View) zu wählen:



Folgen Sie den nächsten Schritten, um ein ersten Entwurf der Benutzeroberfläche zu erzeugen.

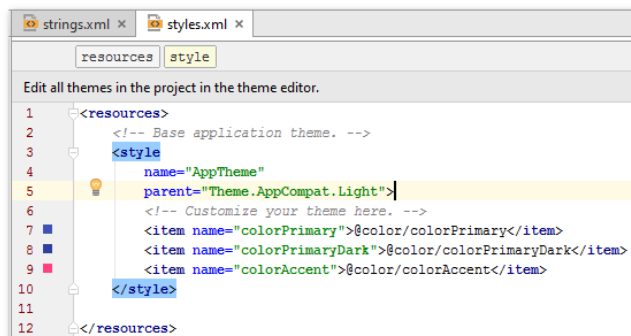
2.4 View: Layouts, Komponenten & XML für die Benutzeroberfläche

Wir erzeugen ein erstes Layout und stellen die Verbindung zur Benutzeroberfläche her.

	<p><i>Erstes Layout.</i></p> <p>Wir werden nun die Benutzeroberfläche für unsere Friends-App erzeugen.</p> <p>Benutzeroberflächen werden in Android-Apps in der Seitenbeschreibungssprache XML beschrieben.</p>
 <p>Hinweis: Aus gutem Grund werden alle Bezeichner der Benutzeroberfläche in eine String-Ressource → (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.</p>	<p><i>XML-Deklarationen.</i></p> <p>Dazu definieren wir in einem ersten Schritt alle verwendeten Bezeichnungen für die Komponenten die wir auf unserer Benutzeroberfläche verwenden möchten. Sie sollten in der Datei strings.xml definiert werden.</p> <p>Öffnen Sie dazu die Datei strings.xml.</p> <p>Sie finden diese Datei im Unterverzeichnis → app → res → values → strings.xml.</p>
	<p><i>Bezeichner definieren.</i></p> <p>Öffnen Sie die Datei → strings.xml mit einem Doppelklick auf den Dateinamen und ändern Sie die darin enthaltenen Angaben wie nebenstehend angezeigt.</p> <p>Hinweis: Vergleichen Sie die definierten Strings mit der Benutzeroberfläche und identifizieren Sie die Bezeichner.</p>

Eingabehilfe:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">Friends</string>
  <string name="hello_world">Hello world!</string>
  <string name="action_settings">
    Settings</string>
  <string name="logo_description">
    Logo-Banner</string>
  <string name="etName_hint_name">
    Vorname Nachname</string>
  <string name="etPhone_hint_phone">
    49 (171) 69 64 043</string>
  <string name="etEmail_hint_email">
    info@domain.de</string>
  <string name="btAdd_zeichen">+</string>
  <string name="etErrorMessage">
    Das Feld darf nicht leer sein.</string>
</resources>
```



styles.xml

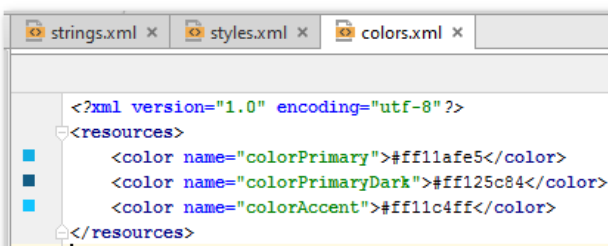
Styles der App ändern.

Öffnen Sie dazu die Datei → styles.xml mit einem Doppelklick auf den Dateinamen.

Ändern Sie die Angaben ggf. wie nebenstehend angezeigt.

```
<style
  name="AppTheme"
  parent="Theme.AppCompat.Light">
```

Sie finden diese Datei im Unterverzeichnis
→ app → res → values → styles.xml.



colors.xml

Farben der App ändern.

Öffnen Sie die Datei → colors.xml mit einem Doppelklick auf den Dateinamen und ändern Sie die Farbcodes, wie nebenstehend angezeigt.

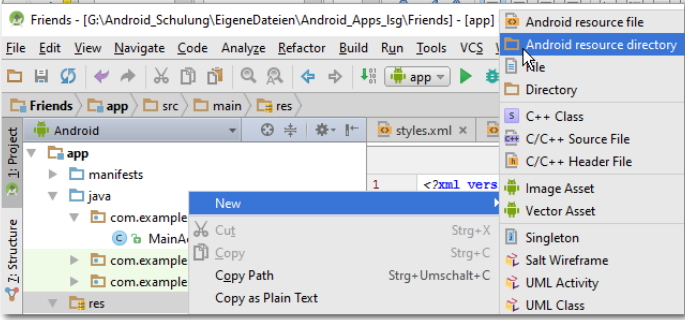
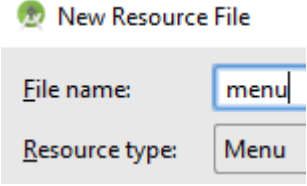
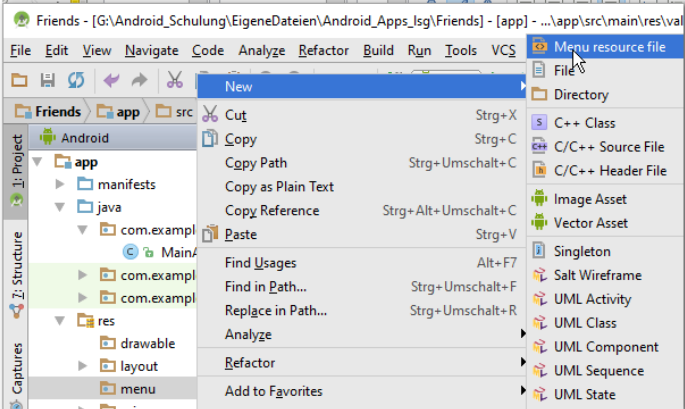
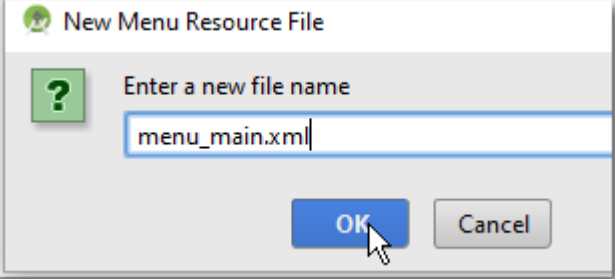
```
<color name="colorPrimary">#ff11afe5</color>
<color name="colorPrimaryDark">#ff125c84</color>
<color name="colorAccent">#ff11c4ff</color>
```

Sie finden diese Datei im Unterverzeichnis
→ app → res → values → colors.xml.

Die styles-Datei holt sich dabei an entsprechender Stelle die Farbangaben aus der colors-Datei.

Hinweis:

Themes enthalten Layoutvorgaben. Diese können wir nach belieben anpassen. Auch wenn unsere Anwendung nur eine Beispielanwendung ist soll sie trotzdem schick sein. Alle generellen Angaben zum Format und Aussehen unserer App werden in der Datei → colors.xml und → sty-

	<p>les.xml definiert:</p> <ol style="list-style-type: none"> 1. Aussehen: res/values/styles.xml 2. Farben: res/values/colors.xml
 <p>Klicken Sie dazu mit der rechten Maustaste im Linken Frame auf das Verzeichnis → res.</p> <p>Wählen Sie dazu im Kontext-Menü (rechte Maustaste) die Optionen → New → Android resource directory</p>	<p><i>Menü-Ressourcen-Verzeichnis erstellen.</i></p> <p>Wir werden zu einem späteren Zeitpunkt ein Menü für die Lösch- und Änderungsoperationen erweitern. Deshalb erzeugen wir nun im Vorfeld ein Menü-Verzeichnis und integrieren schon mal die zugehörige Menü-Datei.</p>  <p>Übernehmen Sie die Angaben, wie angezeigt und klicken Sie auf die Schaltfläche → OK.</p>
 <p>Klicken Sie dazu mit der rechten Maustaste im Linken Frame auf das Verzeichnis → res.</p> <p>Wählen Sie dazu im Kontext-Menü (rechte Maustaste) die Optionen → New → Android resource file</p>	<p><i>Menü-Ressourcen-Datei erstellen.</i></p> <p>Folgen Sie dazu der nebenstehenden Pfadangabe.</p>  <p>Übernehmen Sie die Angaben, wie angezeigt und klicken Sie auf die Schaltfläche → OK.</p>

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <menu
3      xmlns:android=
4          "http://schemas.android.com/apk/res/android"
5      xmlns:app=
6          "http://schemas.android.com/apk/res-auto"
7      xmlns:tools="http://schemas.android.com/tools"
8      tools:context=".MainActivity">
9      <item
10         android:id="@+id/action_settings"
11         android:title="@string/action_settings"
12         android:orderInCategory="100"
13         app:showAsAction="never" />
14  </menu>

```

Inhalt der Menu-Datei ändern.

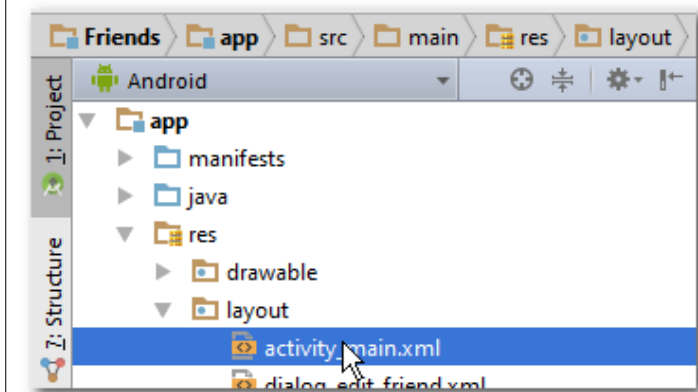
Eingabehilfe:

```

<?xml version="1.0" encoding="utf-8"?>
<menu
    xmlns:android=
        "http://schemas.android.com/apk/res/android"
    xmlns:app=
        "http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".MainActivity">
    <item
        android:id="@+id/action_settings"
        android:title="@string/action_settings"
        android:orderInCategory="100"
        app:showAsAction="never" />
</menu>

```

Übernehmen Sie dazu die Angaben, wie angezeigt.



Layout erstellen.

Wir werden nun das Layout unserer eigentlichen Benutzeroberfläche erstellen.

Öffnen Sie dazu das Verzeichnis
app → res → layout

Öffnen Sie die Datei → activity_main.xml mit einem Doppelklick auf den Dateinamen.

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context="com.example.chrissi.friends.MainActivity">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />
</LinearLayout>

```

Layout definieren.

Wir ersetzen das → RelativeLayout durch ein → LinearLayout.

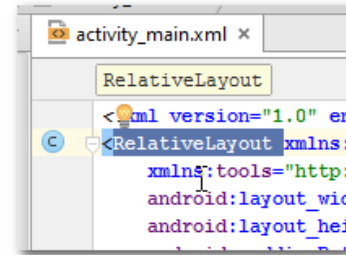
Übernehmen Sie die Angaben, wie nebenstehend angezeigt.

Das Lineare Layout (vertikal):

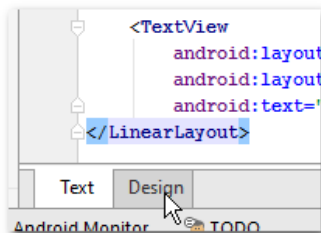
Die in einem vertikalen Linearen Layout platzierten Komponenten werden untereinander angeordnet.

Das Lineare Layout (horizontal):

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

**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.



Designer

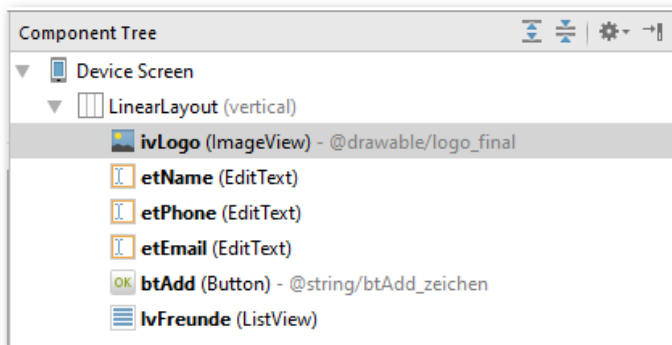
In den Design-Modus wechseln.

Um das Design zu erstellen nutzen wir den Oberflächendesigner.

Klicken Sie dazu auf den Reiter → Design unterhalb des angezeigten XML-Quellcodes.

Hinweis:

Die Anwendung besitzt ähnlich, wie in Eclipse der Swing-Designer einen Quellcode-Generator. Im Gegensatz zu Eclipse erzeugt der Quellcode-Generator in Android Studio XML-Quellcode. Wir können jederzeit zwischen den Ansichten → Text und → Design wechseln.

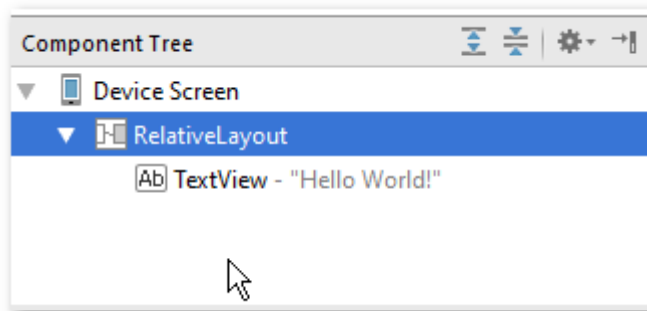


Gewünschtes Ergebnis

Vorgehensweise: Component Tree.

1. Layouts (wenn nötig) schachteln
2. Komponenten im Layout platzieren
3. Komponenteneigenschaften definieren

Nun folgen die Änderungen im aktuellen Komponenten-Baum um das nebenstehende gewünschte Ergebnis zu erzeugen.



Aktueller Komponenten-Baum

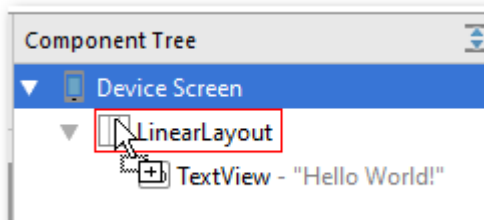
Der Komponenten-Baum.

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.

Das Standard-Layout ist das → Relative Layout.

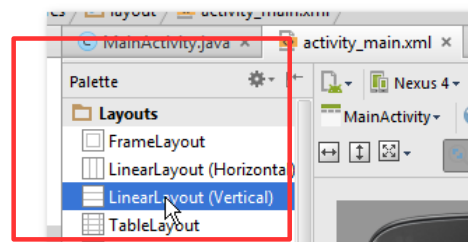


Fenster Component Tree

LinearesLayout (Vertical) verwenden.

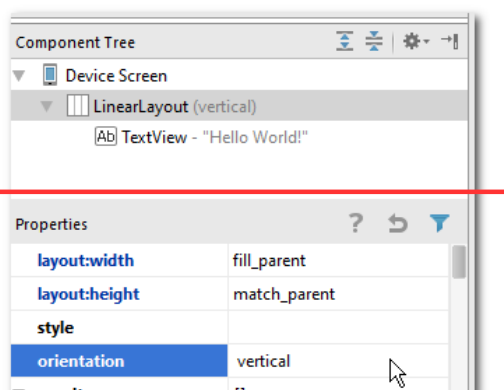
Erster Schritt:

Fenster Palette



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

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



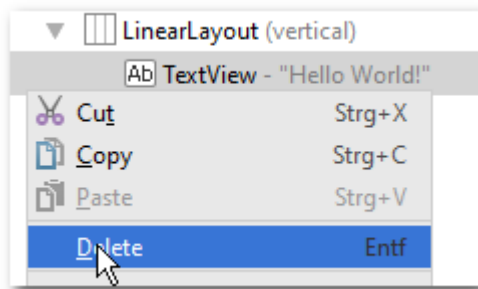
Fenster Properties

Eigenschaften des Layouts ändern.

Klicken Sie dazu im Fenster → Component Tree auf das → LinearesLayout (vertical).

Ändern Sie dann die nebenstehend angezeigten Eigenschaften der Komponente im darunterliegenden Fenster → Properties ab.

layout:width: fill_parent
 layout:height: match_parent
 orientation: vertical

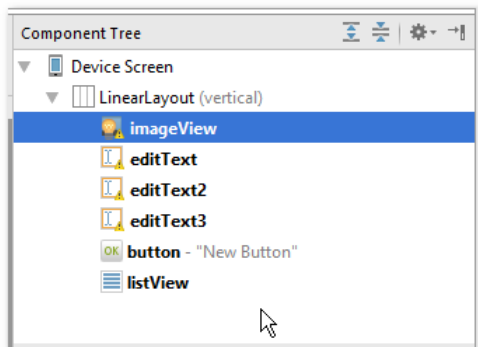


Fenster Component Tree

Komponente löschen.

Löschen Sie die nicht benötigte TextView-Komponente → Hello World!.

Klicken Sie dazu die Komponente im Fenster → Component Tree mit der rechten Maustaste an und wählen Sie im Kontext-Menü die Option → Delete.

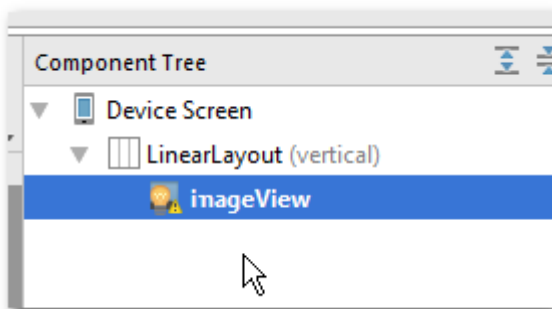


Fenster Component Tree

Komponenten platzieren.

Alle Komponenten werden wir untereinander in das LineareLayout integrieren. Anschließend werden wir für jede Komponente die Eigenschaften festlegen.

Gehen Sie auf gleiche Weise vor. Suchen Sie in der Palette die Komponente und ziehen Sie dazu diese Komponente mit gedrückter linker Maustaste in das rechte, obere Frame-Fenster → Component Tree.



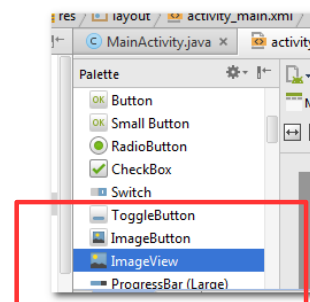
Fenster Component Tree

Platz für das Logo schaffen.

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

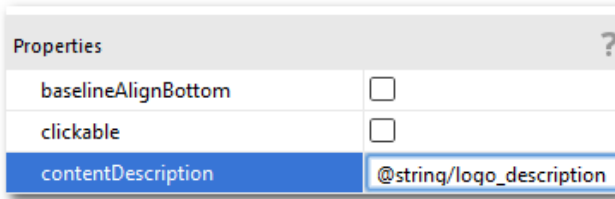
Hier ist eine Komponente vom Typ *ImageView* nötig. Eine → *ImageView*-Komponente (Platzhalter für ein Bild oder eine Grafik).

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

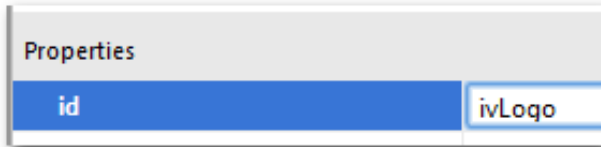


Fenster Palette

Wählen Sie dazu erst die *ImageView*-Komponente im linken Frame-Fenster → Palette aus.



Fenster Properties



Fenster Properties

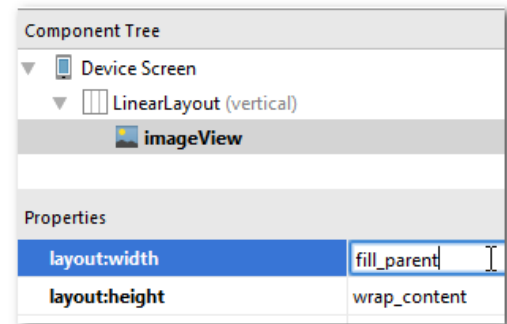
Nutzen Sie dann die vertikale Bildlaufleiste im Fenster → Properties, um die Eigenschaft für die layout:width, → contentDescription und → id, wie nebenstehend angezeigt, ändern zu können.

```
layout:width:      fill_parent
contentDescription: @string/logo_description
id:                ivLogo
```

Die Anpassung der restlichen Eigenschaften werden wir am Ende des Kapitels durchführen.

Eigenschaften für die ImageView-Komponente festlegen.

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.



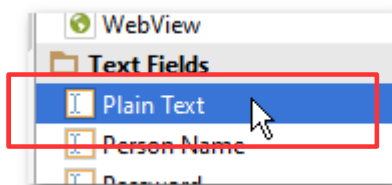
Fenster Component Tree

Eingabe-Komponenten einfügen.

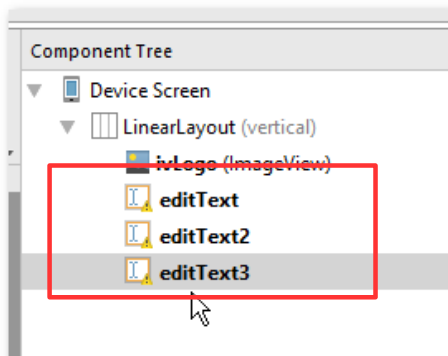
Hier sind drei Komponenten vom Typ EditText nötig. Eine → EditText-Komponente (Plain Text, Texteingabefeld):

Wählen Sie dazu die EditText-Komponente im linken Frame-Fenster → Palette aus.

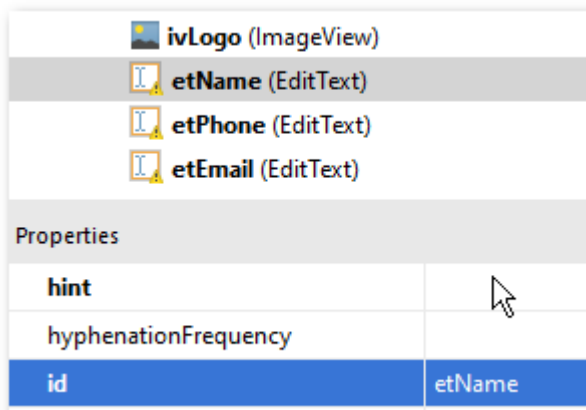
Ziehen Sie dazu diese Komponente drei mal nacheinander mit gedrückter linker Maustaste in das rechte, obere Frame-Fenster → Component Tree.



Fenster Palette



Fenster Component Tree



Fenster Properties

EditText: für die Eingabe des Namens

```
layout:width:    fill_parent
hint:            @string/etName_hint_name
id:             etName
inputType:      [text]
```

Legen Sie auch die Eigenschaften für die Komponenten etPhone und etEmail fest.

Eigenschaften für die Eingabe-Komponenten festlegen.

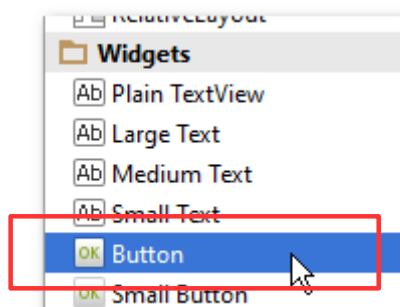
Klicken Sie die Komponente im Fenster → Component Tree und nutzen Sie dann die vertikale Bildlaufleiste im Fenster → Properties, um die Eigenschaft für die layout:width, → hint, → id, und → inputType, wie nebenstehend angezeigt, ändern zu können.

EditText: für die Eingabe der Telefonnummer

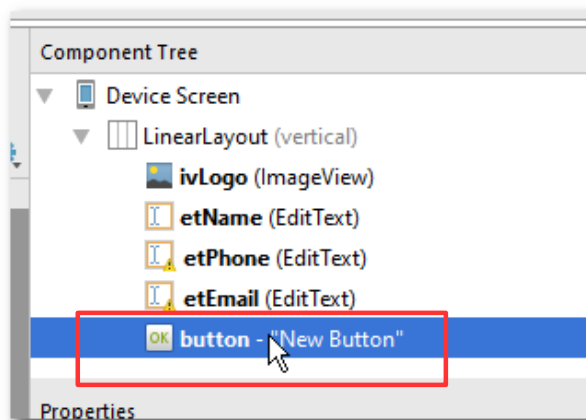
```
layout:width:    fill_parent
hint:            @string/etPhone_hint_phone
id:             etPhone
inputType:      [phone]
```

EditText: für die Eingabe der E-Mail-Adresse

```
layout:width:    fill_parent
hint:            @string/etEmail_hint_email
id:             etEmail
inputType:      [textEmailAddress]
```



Fenster Palette



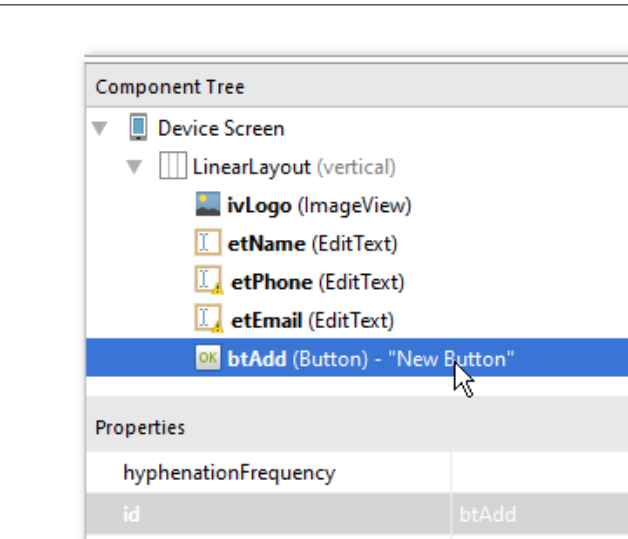
Fenster Component Tree

Schaltfläche einfügen.

Hier ist eine Komponente vom Typ Button nötig. Eine → Button-Komponente (Button, Schaltfläche).

Wählen Sie dazu die Button-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.



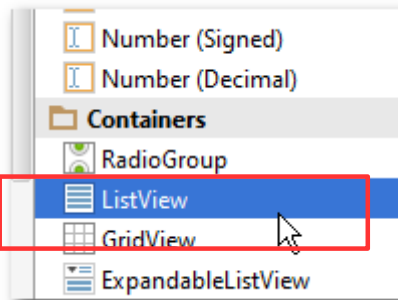
Fenster Component Tree

Eigenschaften für die Schaltfläche festlegen.

Klicken Sie die Komponente im Fenster → Component Tree und nutzen Sie dann die vertikale Bildlaufleiste im Fenster → Properties, um die Eigenschaft für die layout:width, → id und → text, wie nebenstehend angezeigt, ändern zu können.

Button: zum Einfügen von Daten

```
layout:width:    fill_parent
id:              btAdd
text:            @string/btAdd_zeichen
```



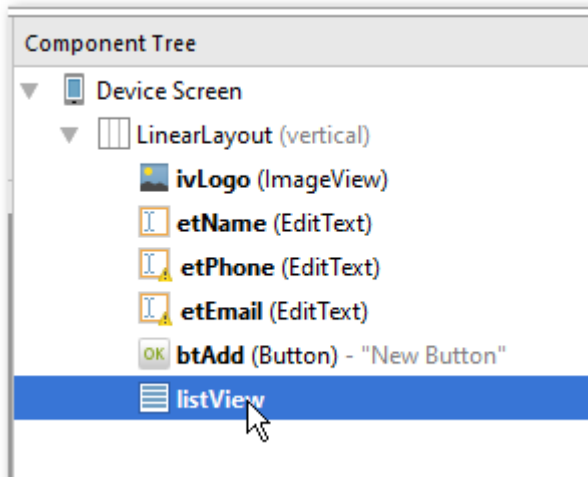
Fenster Palette

Liste-Komponente einfügen.

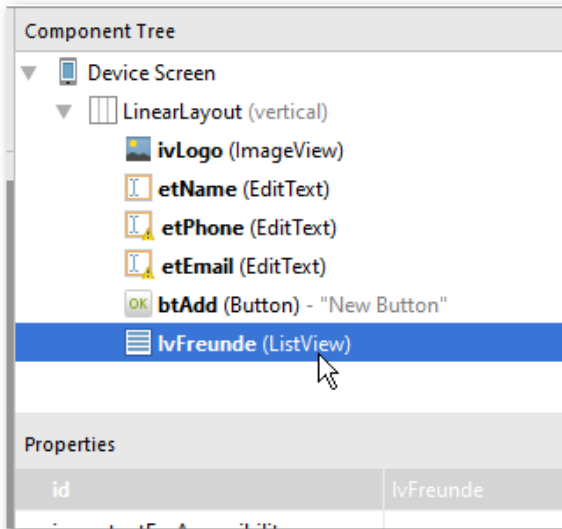
Hier ist eine Komponente vom Typ ListView nötig. Eine → ListView-Komponente (ListView, Listenansicht):

Wählen Sie dazu die ListView-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.



Fenster Component Tree



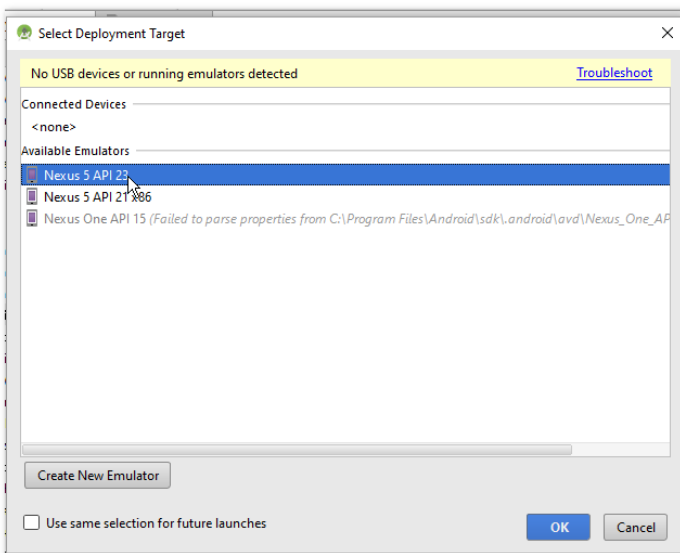
Fenster Component Tree

Eigenschaften für die Listen-Komponente festlegen.

Klicken Sie die Komponente im Fenster → Component Tree und nutzen Sie dann die vertikale Bildlaufleiste im Fenster → Properties, um die Eigenschaft für die layout:width, → layout-gravity und → id, wie nebenstehend angezeigt, ändern zu können.

Liste: zum Anzeigen von Daten

```
layout:width:      fill_parent
layout-gravity:   [center]
id:               lvFreunde
```

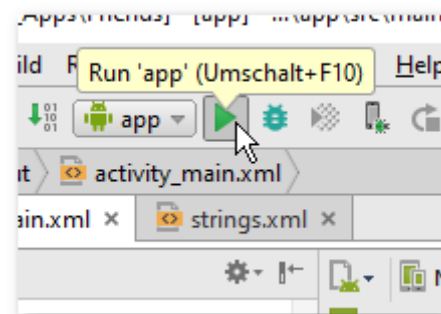


Alternativ → Create New Emulator:

Für wenig leistungsfähige Rechner empfiehlt sich ein neues Gerät → Nexus One Device mit API 15 (SanwichIceCream) zu erzeugen:

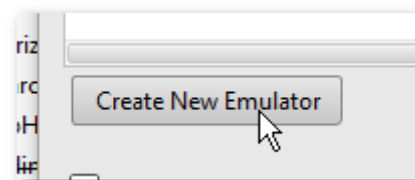
Testen der View.

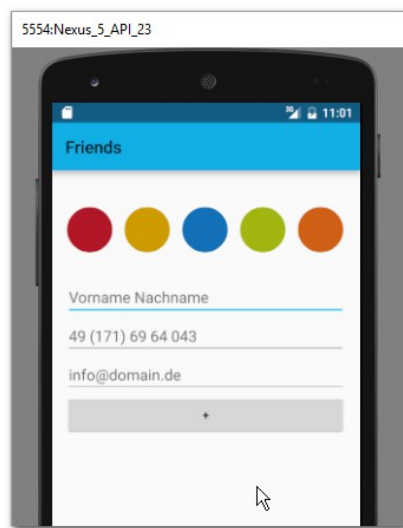
Wir starten nun den Emulator.



Emulator:

Der Emulator simuliert vorliegendem Fall ein virtuelles Mobiltelefon vom Typ → Nexus 5 API 23.





Der Emulator öffnet sich.

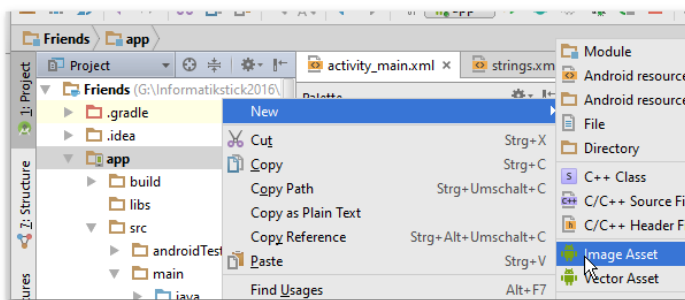
Beim ersten Öffnen kann das einen Moment dauern.

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

Wenn Sie nicht ungeduldig werden, startet der Emulator die App nach Abschluss des Built-Prozesses von selbst.

Im Ergebnis sollte die Benutzeroberfläche erscheinen.

Die Listenansicht ist nicht zu sehen, da nicht keine Anzeigedaten vorhanden sind und um das Logo werden wir uns gleich kümmern.



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`.

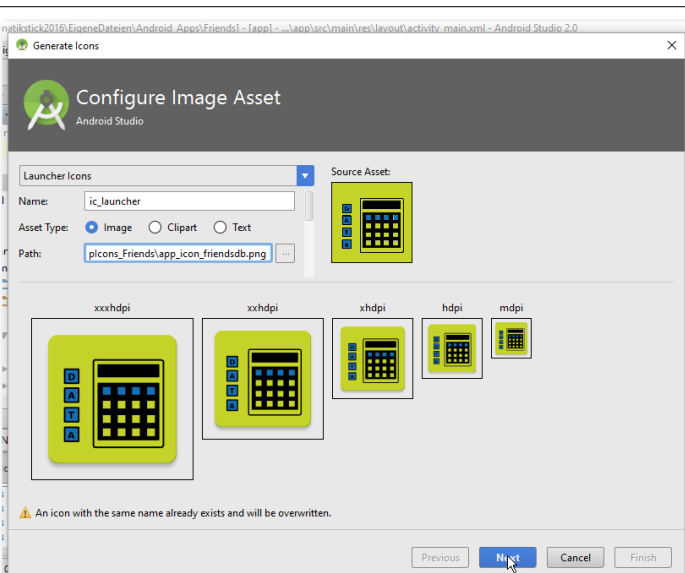
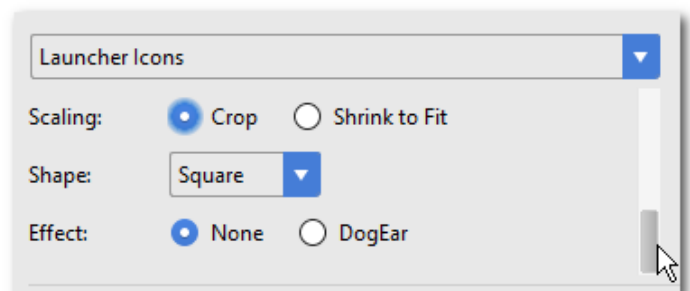


Image Icon definieren.

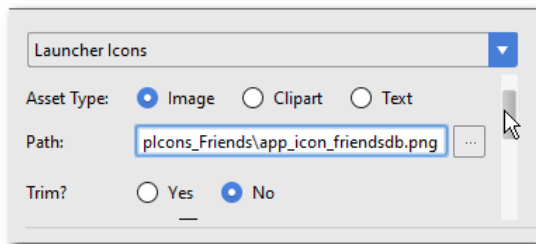
AppIcons_Friends\app_icon_friendsdb.png
Aktivieren Sie für die Eigenschaft → `Scaling` die Option → `Crop` und für die Eigenschaft → `Shape` die Option → `Square` aus:



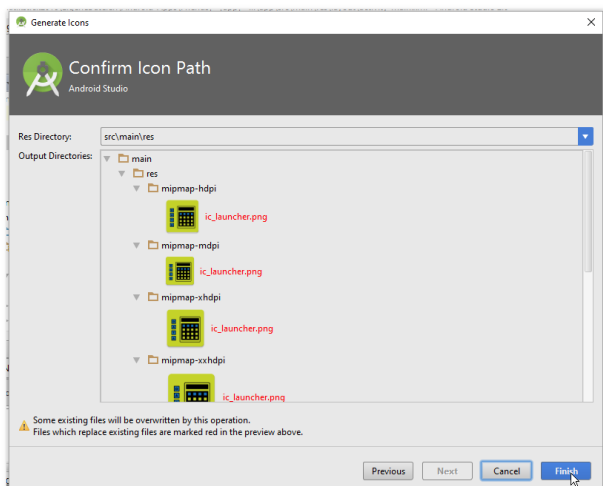
G:\Android_Schulung\Material\Applcons_Friends

Bildquelle

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



Klicken Sie auf die Schaltfläche → Next.



Icon Konfiguration abschließen.

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

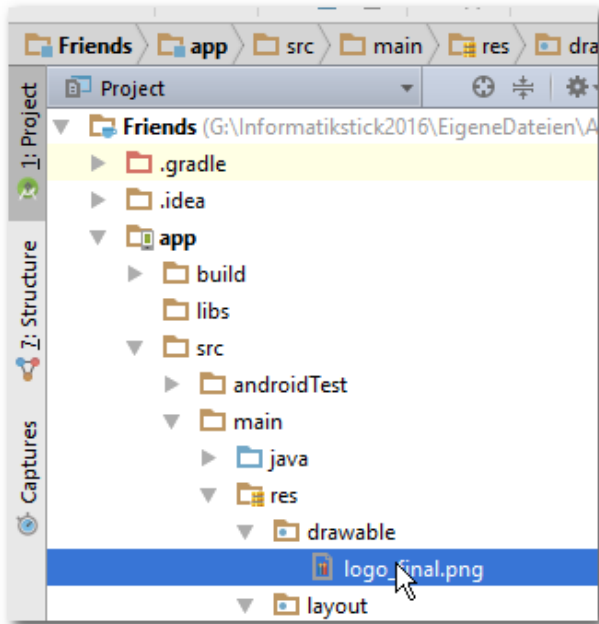
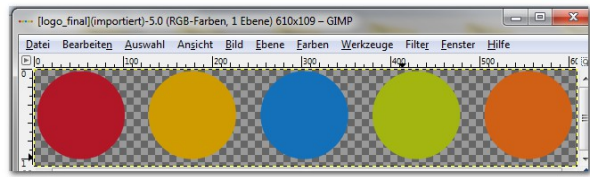
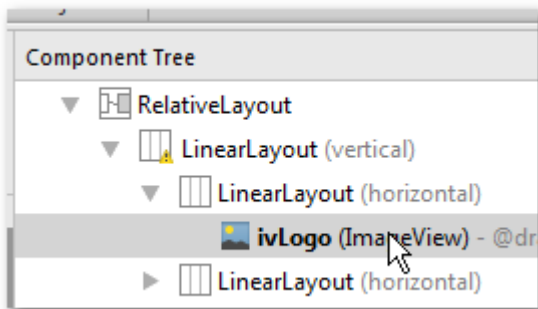
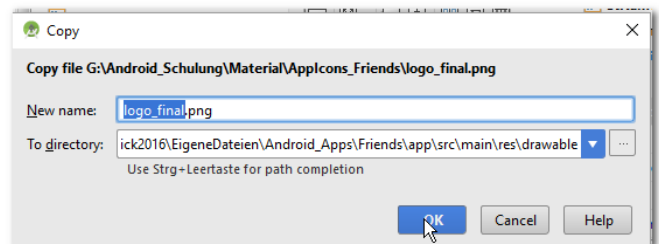


Bild (Banner) einfügen.



AppIcons_Fiends\logo_final.png

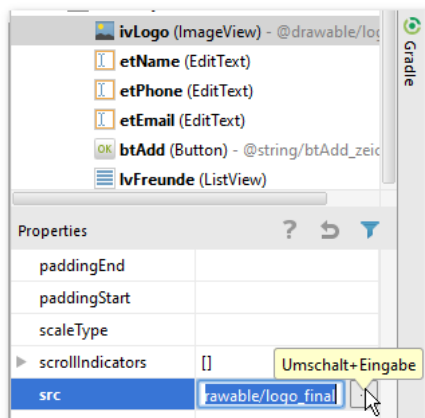
Bilddatei „logo_final.png“ in das Verzeichnis → res → drawable kopieren:



Fenster Component Tree

Bildquelle definieren.

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



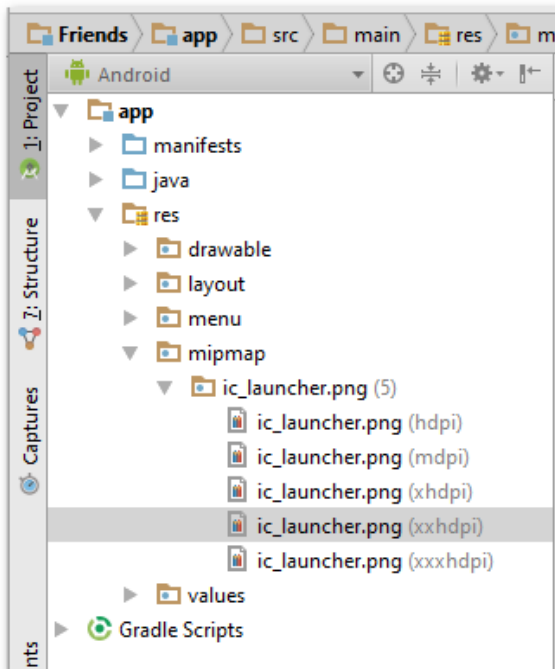
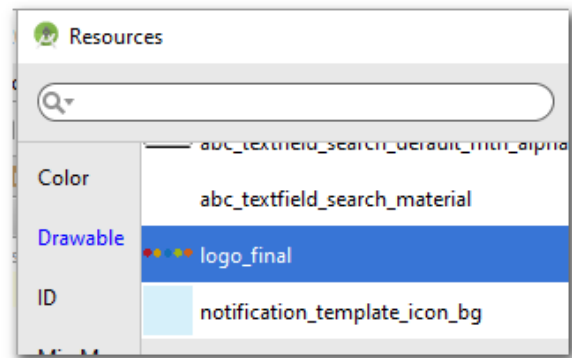
Fenster Properties

Im Fenster Properties die Bildquelle eingeben.

Dazu für die Eigenschaft → src die Quelle:

@drawable/logo_final

definieren.

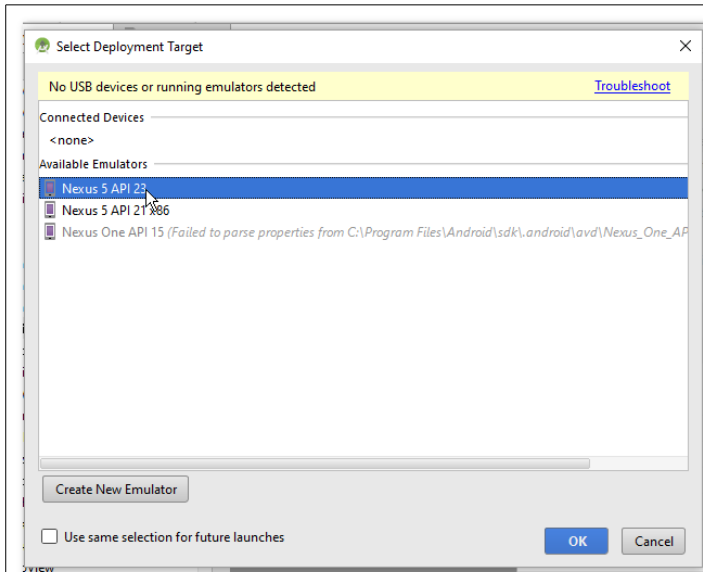


Kontrolle.

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.



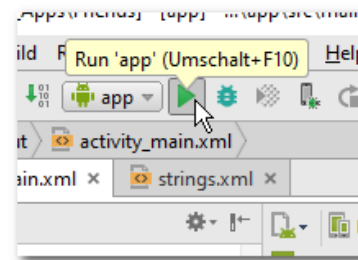


Alternativ → Create New Emulator:

Für wenig leistungsfähige Rechner empfiehlt sich ein neues Gerät → Nexus One Device mit API 15 (SanwichIceCream) zu erzeugen:

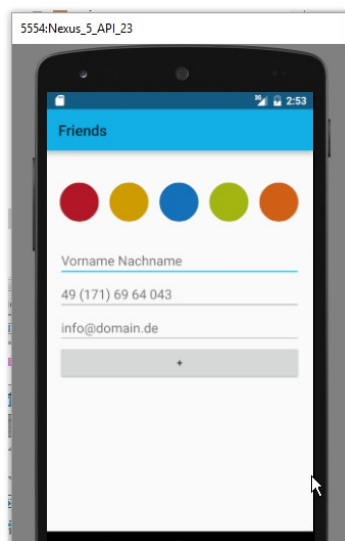
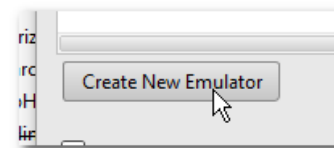
Testen der View.

Wir starten nun den Emulator.



Emulator:

Der Emulator simuliert vorliegenden Fall ein virtuelles Mobiltelefon vom Typ → Nexus 5 API 23.



Der Emulator öffnet sich.

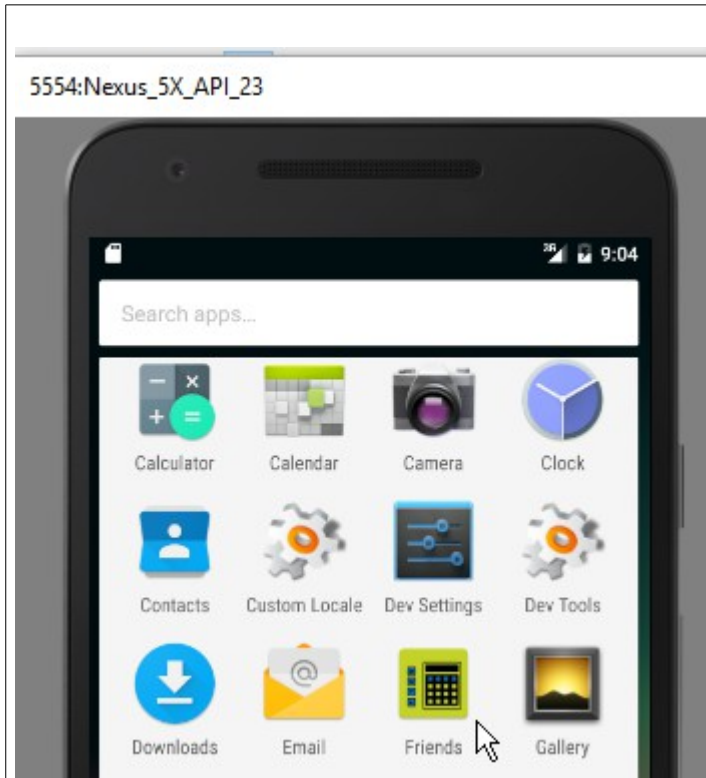
Beim ersten öffnen kann das einen Moment dauern.

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

Wenn Sie nicht ungeduldig werden, startet der Emulator die App nach Abschluss des Built-Prozesses von selbst.

Im Ergebnis sollte die Benutzeroberfläche erscheinen:

Die Listenansicht ist nicht zu sehen, da nicht keine Anzeigedaten vorhanden sind.

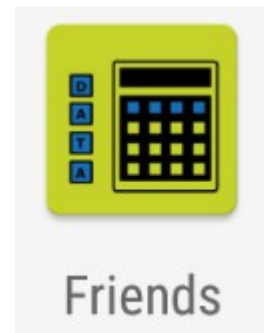


Icon anzeigen.

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



Auf dem Display ist das App Icon nun aufgeführt.

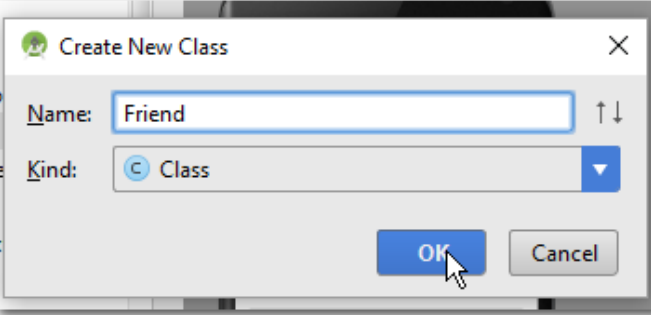


App Icon

Gratulation das Layout ist erstellt!

2.5 Modell: Implementierung der Fachklasse Friend

Wir möchten unserer App die Daten vieler Freunde verwalten. Wir schaffen im nächsten Schritt eine Mustervorlage die wir für alle unsere Freunde verwenden können.

	<p><i>Vorgehensweise erläutern.</i></p> <p>Es folgen nun die Erläuterungen zur Erstellung unseres Modells. Das Modell enthält die Deklaration und Implementierung aller systemrelevanten Objekteigenschaften und -verhaltensweisen die der zeitweisen Datenhaltung dienen.</p>
	<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 → Friend ein und klicken Sie auf die Schaltfläche → OK.</p>

```

1 package com.example.chrissi.friends;
2
3 /**
4  * Created by chrissi on 14.04.2016.
5  */
6 public class Friend {
7
8
9

```

```

1 package com.example.chrissi.friends;
2
3 /**
4  * Created by chrissi on 14.04.2016.
5  */
6 public class Friend {
7
8     //Attribute: Deklaration der Eigenschaften einer Klasse
9
10
11     //Konstruktor: mit Parameter
12
13
14     //Getter: Ermittelt Eigenschaftswert eines eines Objektes
15     * Setter: Übermittelt Eigenschaftswert an das Attribut eines Objektes*/
16
17
18     //Sonstige Methoden: können mehr als nur er- und übermitteln
19     * Hier: Die von Object vererbte toString-Methode wird überschrieben*/
20
21 }
22

```

Eingabehilfe:

```
//Attribute: Deklaration der Eigenschaften einer Klasse
```

```
//Konstruktor: mit Parameter
```

```
/*Getter: Ermittelt Eigenschaftswert eines eines Objektes
Setter: Übermittelt Eigenschaftswert an das Attribut eines Objektes*/
```

```
/*Sonstige Methoden: können mehr als nur er- und übermitteln.
Hier: Die von Object vererbte toString-Methode wird überschrieben*/
```

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.

Grundgerüst einer Klasse festlegen.

Übernehmen Sie die nebenstehend angezeigten Kommentare.

Deklariieren:

In der objektorientierten Programmierung ist mit der Deklaration die

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),

gemeint.

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.

<div data-bbox="98 241 513 824" style="border: 1px solid black; padding: 5px;"> <p style="text-align: center;">Friend</p> <pre> - _id:long - name:String - phone:String - email:String + Friend(String pName, String pPhone, String pEmail, long pId) + setId(long pId) + setName(String pName) + setPhone(String pPhone) + setEmail(String pEmail) + getId():long + getName():String + getPhone():String + getEmail():String + toString():String </pre> </div> <p style="text-align: center;">UML-Klasse: Friend</p> <div data-bbox="603 264 766 302" style="background-color: yellow; padding: 2px;">Klasse</div> <div data-bbox="603 385 766 423" style="background-color: yellow; padding: 2px;">Attribute</div> <div data-bbox="603 618 766 685" style="background-color: yellow; padding: 2px;">Konstruktor & Methoden</div>	<p><i>Fachklasse implementieren.</i></p> <p>Wir erzeugen eine Mustervorlage für all unsere Freunde.</p> <p>Dazu implementieren die Fachklasse → Friend, 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>
<div data-bbox="130 929 742 1169" style="border: 1px solid gray; padding: 5px;"> <pre> 8 //Attribute: Deklaration der 9 //Eigenschaften einer Klasse 10 private String name; 11 private String phone; 12 private String email; 13 private long _id; </pre> </div> <p style="text-align: center;">Attribute</p> <p>Eingabehilfe:</p> <pre> private String name; private String phone; private String email; private long _id; </pre> <p>Der Datentyp String: Der <i>komplexe Datentyp</i> → String bestimmt den Wertebereich einer Zeichenkette (Implementierung: Array aus Characters, siehe auch Oracles API → String).</p> <p>Der Datentyp long: Der <i>primitive Datentyp</i> long legt den Wertebereich für große ganze Zahlen fest. Sobald wir in Java mit der Verarbeitung einer großen Menge von identifizierbaren Objekten aus einer Datenbank rechnen müssen, nutzen die meisten Programmierer den Datentyp → long für die identifizierende Eigenschaft, die → _id.</p>	<p><i>Deklaration der Attribute.</i></p> <div data-bbox="801 987 1513 1025" style="background-color: yellow; padding: 2px;"><code>private String name;</code></div> <p>Der Zugriffsmodifikator → private stellt sicher, dass nur die Objekte der Klasse selbst auf die Eigenschaftswerte direkt zugreifen können.</p> <p>Mit der Bestimmung des geeigneten Datentyps für ein Attribut wird gleichzeitig der maximal benötigte Speicherplatz vorab reserviert.</p> <p>→ name ist der Attributname. Attribute werden in Java kleingeschrieben und enthalten keine Umlaute und/oder Sonderzeichen.</p> <p>Hinweis: Leerzeichen sind auch Sonderzeichen!</p> <div data-bbox="801 1585 1513 1624" style="background-color: yellow; padding: 2px;">Deklarieren Sie auch die übrigen Attribute.</div>

```

14      //Konstruktor: mit Parameter
15      public Friend(String pName,
16                    String pPhone,
17                    String pEmail,
18                    long pId){
19          this.name = pName;
20          this.phone = pPhone;
21          this.email = pEmail;
22          this._id = pId;
23      }

```

Konstruktor

Beispiel Konstruktoraufruf:

```

Friend einFreund =
    new Friend(
        "Tim Müller",
        "49 (172) 67 54 678",
        "tim@mydomein.de", 1);

```

Der Konstruktor einer Klasse sorgt dafür, dass beliebig viele Objekte der Klasse erzeugt, „konstruiert“ werden können.

```

24      /*Getter:
25      * Ermittelt Eigenschaftswert
26      * eines eines Objektes
27      *
28      * Setter:
29      * Übermittelt Eigenschaftswert an das
30      * Attribut eines Objektes*/
31      public String getName() {
32          return name;
33      }
34
35      public void setName(String pName) {
36          this.name = pName;
37      }

```

Getter und Setter:

```

public String getName() {
    return name;
}
public void setName(String pName) {
    this.name = pName;
}

```

Deklaration eines Konstruktors mit Parameter.

Jeder Benutzer erzeugt damit seine eigenen Freunde.

Der Standard Konstruktor enthält keine Parameter und initialisiert keine Anfangswerte. Neu erzeugte Objekte sind also am Anfang ihrer Entstehung „wertelos“.

Wir wollen verhindern, dass es „wertelose“ Freunde gibt und nutzen im vorliegenden Fall einen parameterbehafteten Konstruktor. Es wird also in unserem System so sein, dass wir nur in Situationen, in denen alle Eigenschaften bekannt sind, ein neues Objekt der Klasse → Friend erzeugen werden.

Deklaration und Implementierung der Get- und Set-Methoden.

Berücksichtigen Sie, dass wir auf die Eigenschaftswerte der Friend-Objekte von außerhalb der Klasse (z.B. von der Benutzeroberfläche 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 Fachklassen Friend und Kontakt verteilen. Damit könnten wir für jeden Friend Container für eine ganze Menge von E-Mail-Adressen und Telefonnummern schaffen (Multiplizität → 1:N).

Wir haben es uns in unserem Beispiel also sehr einfach gemacht indem wir jedem Freund genau ein Name, eine E-Mail-Adresse und eine Telefonnummer zuordnen werden (Multiplizität → 1:1).


```
69  /*Sonstige Methoden: können mehr
70  * als nur er- und übermitteln von Werten
71  * Hier: Die von Object vererbte
72  * toString-Methode wird überschrieben*/
73  public String toString(){
74      String newline = "\n";
75      String output = this.name + newline
76                  + this.phone + newline
77                  + this.email;
78      return output;
79  }
```

Die Methode `toString():String` ist eine Methode die von der Klasse → Object vererbt wird.

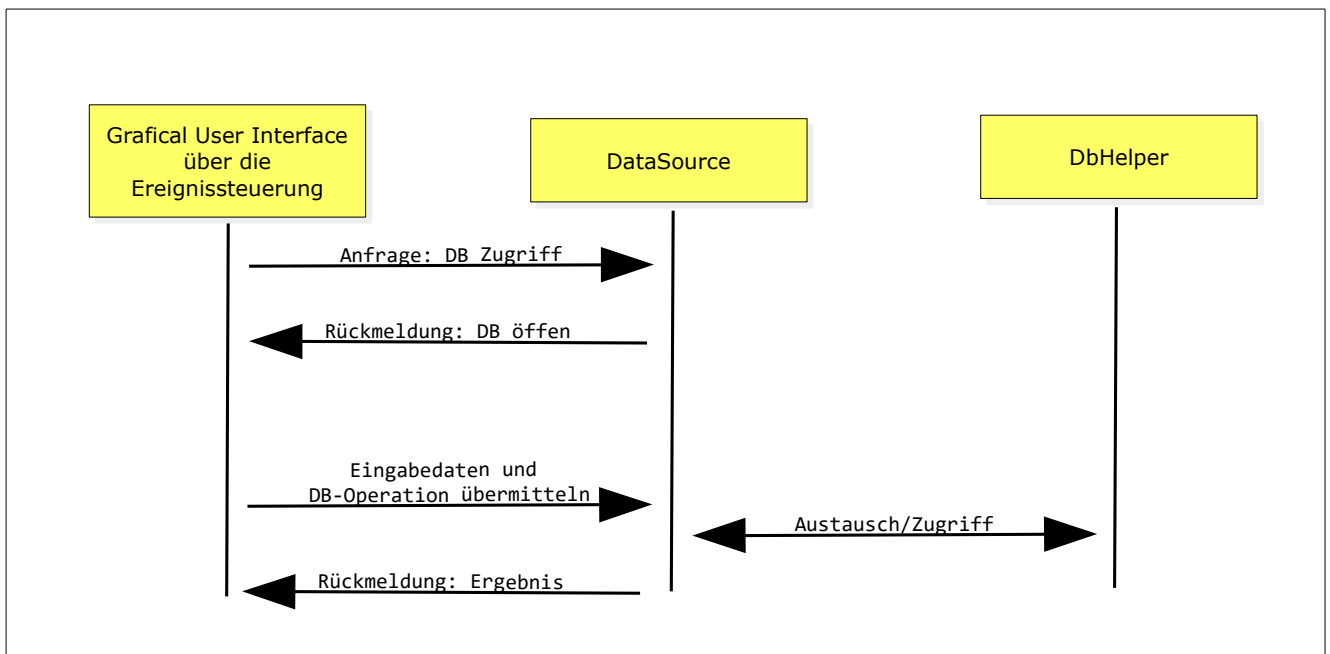
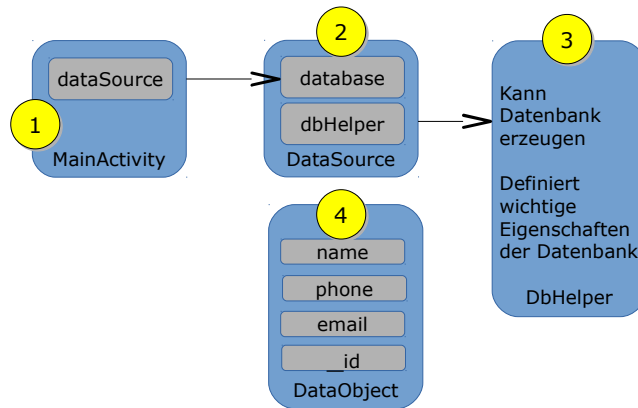
Deklaration und Implementierung sonstiger Methoden.

Eine Methode mit einem Rückgabewert vom Typ String wird dazu genutzt die Art und Weise der Ausgabe der Objekteigenschaften zu definieren. Wir nutzen darin den regulären Ausdruck → \n um nach jedem Eigenschaftswert einen Zeilenumbruch zu forcieren. Die Methode liefert im Ergebnis einen Datensatz der Form zurück:

```
Karl
49 (173) 1234566
karl@mydomain.com
```

2.6 Controller: Steuerung und Zugriff auf die Datenbank

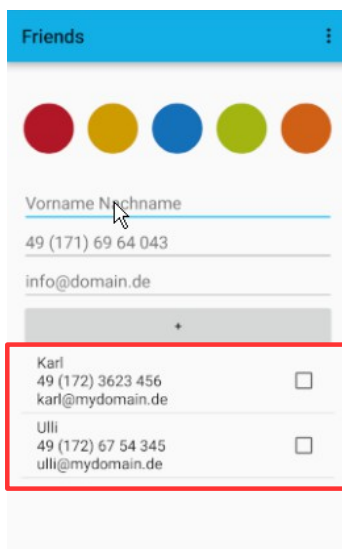
Datenbankzugriff



2.6.1 Daten anzeigen

Für die Anzeige von Daten benötigen wir eine *Auswahloperation* und müssen wir zuvor einige Strukturen schaffen die den Zugriff auf die Datenbank sicherstellt. Wir benötigen die Datenbanktabellen und müssen sicherstellen, dass wir vorab Testdaten einfügen und auslesen können. Das Anzeigen von Daten setzt also den Zugriff auf die Datenbank voraus.

Anzeigen



FriendsDbHelper

```
- LOG_TAG:String
    = FriendsDbHelper.class.getSimpleName()
- DB_NAME:String = "friends.db"
- DB_VERSION:String = "contact"
+ COLUMN_ID:Long = "_id"
+ COLUMN_NAME:String = "name"
+ COLUMN_PHONE:String = "phone"
+ COLUMN_EMAIL:String = "email"
+ SQL_CREATE:String =
    "CREATE TABLE " + TABLE_CONTACT_LIST
    +"(" + COLUMN_ID
    + " INTEGER PRIMARY KEY AUTOINCREMENT, "
    + COLUMN_NAME
    + " TEXT NOT NULL, "
    + COLUMN_PHONE
    + " TEXT NOT NULL, "
    + COLUMN_EMAIL + " TEXT NOT NULL);";
+ Friend(Context context)
+ onCreate(SQLiteDatabase db)
+ onUpgrade(SQLiteDatabase db,
            int oldVersion, int newVersion)
```

UML-Klasse: FriendsDbHelper.java

Die Klasse FriendsDbHelper.java.

Alle Eigenschaften dieser Klasse sind *statisch* und *final*. Jede Eigenschaft erhält einen fixen Werte von uns.

static:

Ist ein Schlüsselwort (keyword) für Attribute und Methoden. Wenn in Java eine Eigenschaft als static deklariert wird bedeutet das, dass alle Objekte dieser Klasse den selben Eigenschaftswert nutzen. Die Attributnamen statischer Eigenschaften werden kursiv geschrieben.

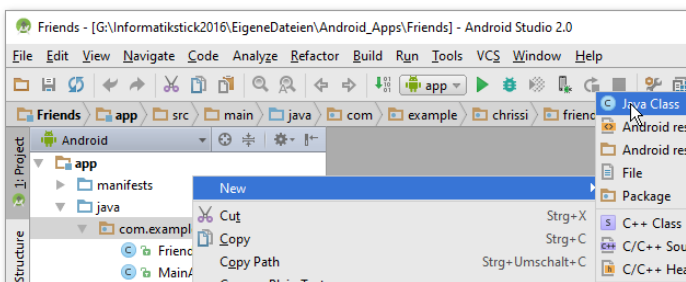
final:

Ist ein Schlüsselwort (keyword) für Attribute in Java. Wenn in Java eine Eigenschaft als final deklariert wird ist eine Änderung des Eigenschaftswertes unerwünscht. Auch deshalb haben finale Eigenschaften keine implementierten Getter und Setter. Die Attributnamen finaler Eigenschaften werden in Großbuchstaben geschrieben.

In den folgenden Schritten werden wir die Klasse

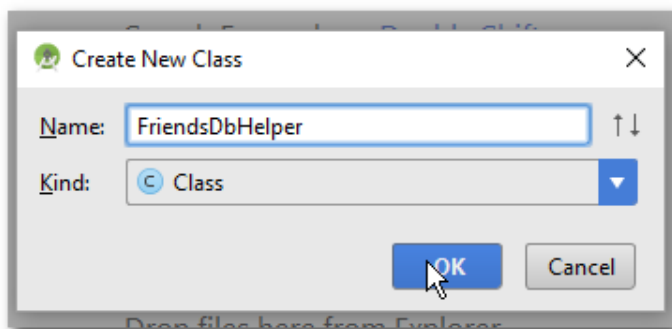
Diese Klasse ist unsere Helferklasse und wird von der SQLiteOpenHelper-Klasse abgeleitet. Mit ihrer Hilfe werden wir u.a. die Tabelle in die Datenbank einfügen. Vorher legen wir aber die wichtigsten Datenbank- und Tabellen-Eigenschaften in String-Konstanten in dieser Klasse fest.

erzeugen und mit den vorerst wichtigsten Eigenschaften und Methoden ausstatten.



Die Klasse FriendsDbHelper.java erzeugen.

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



Klassenname festlegen.



Geben Sie als Klassennamen → FriendsDbHelper ein und klicken Sie auf die Schaltfläche → OK.

Führen Sie nun Schrittweise die Implementierung dieser Klassen durch.



vorher

Die vererbende Klasse angeben.

Die Klasse soll von der Übergeordneten Klasse SQLiteOpenHelper erben.

Geben Sie dazu nach dem Schlüsselwort (keyword) → extends den Klassennamen → SQLiteOpenHelper ein wählen Sie den Klassennamen im Kontext-Menü aus. Im Ergebnis sollte oberhalb der Klassendeklaration der Importbefehl für die Superklasse eingefügt werden.

SQLiteOpenHelper:

Ist eine Arbeiterklasse und unterstützt die Kommu-

 <pre> 1 package com.example.chrissi.friends; 2 3 import android.database.sqlite.SQLiteOpenHelper; 4 5 /** 6 * Created by chrissi on 15.04.2016. 7 */ 8 public class FriendsDbHelper extends SQLiteOpenHelper{ 9 10 11 } </pre> <p>nachher</p>	<p>nikation mit der Datenbank des mobilen Endgeräts.</p> <p>In den folgenden Schritten implementieren wir die erbende Klasse → Friendshelper mit den Methoden → onCreate() und → onUpgrade(). Um das Anlegen der Datenbank müssen wir uns vorerst nicht kümmern.</p>
 <pre> 12 /*Ein String-Objekt LOG-TAG um die Abarbeitung der Programmlogik 13 sichtbar zu machen. Die Meldungen im Logcat-Fenster helfen 14 während der Entwicklung beim Debugging.*/ 15 private static final String LOG_TAG 16 = FriendsDbHelper.class.getSimpleName(); 17 </pre> <p>Eingabehilfe:</p> <pre> /*Ein String-Objekt LOG-TAG um die Abarbeitung * der Programmlogik * sichtbar zu machen. Die Meldungen im * Logcat-Fenster helfen während der * Entwicklung beim Debugging.*/ private static final String LOG_TAG = FriendsDbHelper.class.getSimpleName(); </pre>	<p>Die Eigenschaft LOG_TAG.</p> <p>Im Ersten Schritt deklarieren wir ein String-Objekt → LOG-TAG um die Abarbeitung der Programmlogik protokollieren zu können. Wir lassen uns später die Meldung im Logcat-Fenster ausgeben. Das hilft während der Entwicklung, vor allem dann, wenn wir Fehler suchen.</p> <p>Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.</p>
 <pre> 17 //Attribute für die Datenbankeigenschaften 18 private static final String DB_NAME ="friends.db"; 19 private static final int DB_VERSION =1; </pre> <p>Eingabehilfe:</p> <pre> //Attribute für die Datenbankeigenschaften private static final String DB_NAME ="friends.db"; private static final int DB_VERSION =1; </pre>	<p>Eigenschaften der Datenbank.</p> <p>Legen Sie den Namen und die Versionsnummer für die Datenbank fest.</p> <p>Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.</p>
 <pre> 21 //Attribute für die Tabelleneigenschaften einer Datenbank 22 public static final String TABLE_CONTACT_LIST = "contact"; 23 public static final String COLUMN_ID = "_id"; 24 public static final String COLUMN_NAME = "name"; 25 public static final String COLUMN_PHONE= "phone"; 26 public static final String COLUMN_EMAIL= "email"; </pre> <p>Eingabehilfe:</p> <pre> //Attribute für die Tabelleneigenschaften //einer Datenbank public static final String TABLE_CONTACT_LIST = "contact"; </pre>	<p>Die Datenbanktabelle.</p> <p>Legen Sie für unsere Datenbank den Tabellennamen und die einzelnen Spaltennamen der Tabelle fest.</p> <p>Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.</p>

```
public static final String COLUMN_ID = "_id";
public static final String COLUMN_NAME = "name";
public static final String COLUMN_PHONE = "phone";
public static final String COLUMN_EMAIL = "email";
```

SQL-String für den CREATE-TABLE-Befehl.

Für die Datenbank nutzen wir das auf allen mobilen Endgeräten vorhandene Datenbanksystem SQLite.

Die SQL-Syntax unterscheidet sich nur geringfügig von der uns bekannten MySQL-Syntax. Die Unterschiede lassen sich gegebenenfalls mit Hilfe einer Internetrecherche ermitteln.

Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.

```
28 //String zum erzeugen des SQL-Create-Table-Befehls
29 public static final String SQL_CREATE =
30     "CREATE TABLE "
31     + TABLE_CONTACT_LIST
32     + "(" + COLUMN_ID
33     + " INTEGER PRIMARY KEY AUTOINCREMENT, "
34     + COLUMN_NAME + " TEXT NOT NULL, "
35     + COLUMN_PHONE + " TEXT NOT NULL, "
36     + COLUMN_EMAIL + " TEXT NOT NULL);";
37
```

Eingabehilfe:

```
//String zum erzeugen des SQL-Create-Table-Befehls
public static final String SQL_CREATE
    ="CREATE TABLE "
    + TABLE_CONTACT_LIST
    + "(" + COLUMN_ID
    + " INTEGER PRIMARY KEY AUTOINCREMENT, "
    + COLUMN_NAME + " TEXT NOT NULL, "
    + COLUMN_PHONE + " TEXT NOT NULL, "
    + COLUMN_EMAIL + " TEXT NOT NULL);";
```

Der Konstruktor der FriendsHelper-Klasse.

Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.

Fehlende Importanweisungen einfügen: Klicken auf die rot angezeigten Klassennamen und wählen Sie die Tastenkombination ALT + ENTER, um die fehlende Importanweisung einzufügen.

Für die Klasse Context:

```
? android.content.Context? Alt+Eingabe
onstruktor: mit Parameter vom Typ Context
lic FriendsDbHelper(Context context){
//Übermittelt die bekannten Eigenschaftswerte
```

```
40 //Konstruktor: mit Parameter vom Typ Context
41 public FriendsDbHelper(Context context){
42
43     //Übermittelt die bekannten Eigenschaftswerte
44     super(context,DB_NAME,null,DB_VERSION);
45
46     //Log-Meldung erzeugen und im Logcat ausgeben.
47     Log.d(LOG_TAG,"DBHelper hat die Datenbank "+ getDatabaseName() + " erzeugt.");
48 }
```

Eingabehilfe:

```
//Konstruktor: mit Parameter vom Typ Context
public FriendsDbHelper(Context context){
    //Übermittelt die bekannten Eigenschaftswerte
    super(context,DB_NAME,null,DB_VERSION);

    //Log-Meldung erzeugen und im Logcat ausgeben.
    Log.d(LOG_TAG,"DBHelper hat die Datenbank "
        + getDatabaseName() + " erzeugt.");
}
```

Für die Klasse Log:

```
super(context,DB
? android.util.Log? Alt+Eingabe
//Log-Meldung erz
Log.d(LOG_TAG,"DB
}
```

```

50 //Die onCreate-Methode wird nur aufgerufen, falls die Datenbank noch nicht existiert
51 @Override
52 public void onCreate(SQLiteDatabase db) {
53     try {
54         Log.d(LOG_TAG, "Die Tabelle wird mit SQL-Befehl: " + SQL_CREATE + " angelegt.");
55         db.execSQL(SQL_CREATE);
56     }
57     catch (Exception ex) {
58         Log.e(LOG_TAG, "Fehler beim Anlegen der Tabelle: " + ex.getMessage());
59     }
60 }

```

Eingabehilfe:

```

//Die onCreate-Methode wird nur aufgerufen,
//falls die Datenbank noch nicht existiert

```

```

@Override
public void onCreate(SQLiteDatabase db) {
    try {
        Log.d(LOG_TAG,
            "Die Tabelle wird mit SQL-Befehl: "
            + SQL_CREATE + " angelegt.");
        db.execSQL(SQL_CREATE);
    }
    catch (Exception ex) {
        Log.e(LOG_TAG,
            "Fehler beim Anlegen der Tabelle: "
            + ex.getMessage());
    }
}

```

Die onCreate-Methode wird nur aufgerufen, falls die Datenbank noch nicht existiert. Wir überschreiben die Implementierung und übermitteln den SQL CREATE-String an die Datenbank. Diese Aktion wird aber nur in einem Falle ausgeführt, nämlich ausschließlich dann, wenn die App zuvor nicht installiert wurde.

```

62 //Im Fall einer Ausführung die DB_VERSION zuvor auf 3 setzen
63 @Override
64 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
65     Log.d(LOG_TAG, String.format("FriendsSQLiteDatabase.onUpgrade(%d -> %d)",
66         oldVersion, newVersion));
67     switch(newVersion) {
68         case 2:
69             Log.d(LOG_TAG, "Tabelle contact erzeugen");
70             db.execSQL(SQL_CREATE);
71             break;
72         default:
73             throw new IllegalStateException(
74                 "onUpgrade() mit unbekannter newVersion" + newVersion);
75     }
76 }

```

Eingabehilfe:

```

//Im Fall einer Ausführung die DB_VERSION
//zuvor auf 3 setzen

```

```

@Override
public void onUpgrade(
    SQLiteDatabase db,
    int oldVersion,
    int newVersion) {
    Log.d(LOG_TAG,

```

Die onCreate()-Methode.

Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.

Fehlende Importanweisungen einfügen: Klicken auf die rot angezeigten Klassennamen und wählen Sie die Tastenkombination ALT + ENTER, um die fehlende Importanweisung einzufügen.

Für die Klasse SQLiteDatabase:

```

@Override
public void onCreate(SQLiteDatabase db) {
    try {
        Log.d(LOG_TAG, "Die Tabelle wird mit SQL-Befehl: "

```

Die onUpgrade()-Methode.

Die Methode nutzen wir wenn die Version der Datenbank ändern möchten. Dies ist dann der Fall, wenn wir die Tabelle nachträglich ändern möchten. Wir müssen gegebenenfalls weitere Methoden implementieren. Mittels eines SQL-String ALTER TABLE zur Änderung einer Datenbanktabelle. Für den Fall, dass wir die Datenbankstruktur erhöhen wir die Versionsnummer um 1 damit werden dann die Änderungen auf der Datenbank ausgeführt.

Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.

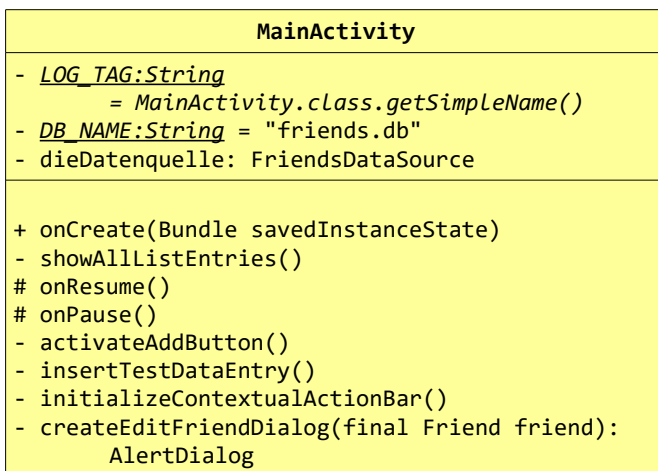
Damit sind die vorerst benötigten Verhaltensweisen implementiert, sodass wir uns um die Erweiterung der MainActivity kümmern können.

```
String.format(
    "FriendsSQLiteDatabase.onUpgrade(%d -> %d)",
    oldVersion, newVersion));

switch(newVersion) {
case 2:
    Log.d(LOG_TAG, "Tabelle contact erzeugen");
    db.execSQL(SQL_CREATE);
    break;

default:
    throw new IllegalStateException(
        "onUpgrade() mit unbekannter newVersion"
        + newVersion);
}
}
```

nen, um erste Tests auf unserer Datenbank ausführen zu können.



UML-Klasse: MainActivity.java

Deklaration der Klasse MainActivity:

```
//Ereignissteuerung für die Benutzeroberfläche.
public class MainActivity extends AppCompatActivity {
    /*Ein String-Objekt LOG-TAG um die Abarbeitung der Pro
```

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.

Unsere Klasse → MainActivity wird alle Ereignisse unserer Kontaktdatenbank → Friends handeln.

Im aktuellen Zustand ist die Klasse nur mit der → onCreate()-Methode ausgestattet. Wir werden Sie nach und nach mit den nötigen ereignissteuernden Methoden ausstatten.

Hinweis:

Die MainActivity erbt zwischenzeitlich standardmäßig von der Klasse AppCompatActivity:

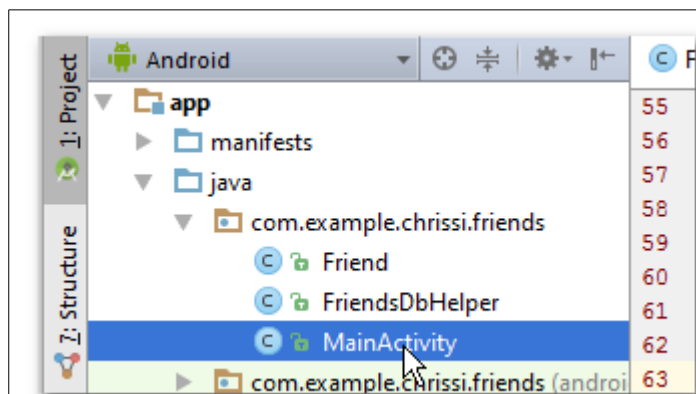
MainActivity extends AppCompatActivity

Bei den meisten älteren Projekten erbt die MainActivity von der ActionBarActivity

MainActivity extends ActionBarActivity

Die Verwendung der Klasse ActionBarActivity ist hinfällig (depreceated).

Das macht aber bisher keine Probleme, d.h. alte Projekte müssen nicht zwingend abgeändert werden.



Öffnen Sie die Klassen mit einem Klick auf die Datei → MainActivity im Verzeichnis → app → java → packagename und erweitern Sie die Klasse in den folgenden Schritten.

MainActivity.java zum testen erweitern.

```

1 package com.example.chrissi.friends;
2
3 import android.support.v7.app.AppCompatActivity;
4 import android.os.Bundle;
5
6 public class MainActivity extends AppCompatActivity {
7
8     @Override
9     protected void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.activity_main);
12     }
13 }

```

In den nächsten Schritten erweitern wir die Klasse MainActivity, um erste Tests ausführen zu können.

```

1 package com.example.chrissi.friends;
2
3 //Importanweisungen: Datenbanktabelle erzeugen und ändern
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.util.Log;
7 import android.view.View;
8 import android.view.ViewGroup;
9 import android.widget.ArrayAdapter;
10 import android.widget.ListView;
11 import java.util.List;
12

```

Ergänzungen der Importanweisungen

Alternativ können Sie jede fehlende Importanweisung situativ dann erzeugen, wenn Sie benötigt wird.

Mit der Tastenkombination ALT + ENTER werden fehlende Importe an entsprechender Stelle angezeigt und können generiert werden.

Eingabehilfe:

```

//Importanweisungen: Datenbanktabelle
//erzeugen und ändern
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import java.util.List;

```

```

1 package com.example.chrissi.friends;
2
3 import android.support.v7.app.AppCompatActivity;
4 import android.os.Bundle;
5
6 public class MainActivity extends AppCompatActivity {
7     //Ein String-Objekt LOG-TAG um die Abarbeitung der Programmlogik
8     //sichtbar zu machen. Die Meldungen im Logcat-Fenster helfen
9     //während der Entwicklung beim Debugging.*/
10    public static final String LOG_TAG = MainActivity.class.getSimpleName();

```

Die Eigenschaft LOG_TAG.

Im Ersten Schritt deklarieren wir ein String-Objekt → LOG-TAG um die Abarbeitung der Programmlogik protokollieren zu können. Wir lassen uns später die Meldung im Logcat-Fenster ausgeben. Das hilft während der Entwicklung, vor allem dann, wenn wir Fehlersuchen.

Eingabehilfe:

```
/*Ein String-Objekt LOG-TAG um die Abarbeitung der
Programmlogik sichtbar zu machen. Die Meldungen im
Logcat-Fenster helfen während der Entwicklung beim
Debugging.*/
public static final String LOG_TAG
    = MainActivity.class.getSimpleName();
```

Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.

```
12 //Deklaration des Objektes für die Steuerung der Datenbank.
13 private FriendsDataSource dieDatenquelle;
14
```

Deklaration unserer Datenquelle.

Eingabehilfe:

```
//Deklaration des Objektes für die Steuerung
//der Datenbank.
private FriendsDataSource dieDatenquelle;
```

Der Klasse → FriendsDataSource. Die Klasse dient der Datenbanksteuerung. Diese Klasse werden wir gleich im Anschluss an die Ergänzungen dieser Klasse durchführen.

Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.

```
15 @Override
16 protected void onCreate(Bundle savedInstanceState) {
17     super.onCreate(savedInstanceState);
18     setContentView(R.layout.activity_main);
19     //Log-Meldung erzeugen und im Logcat ausgeben.
20     Log.d(LOG_TAG, "Das Datenquellen-Objekt wird angelegt.");
21     dieDatenquelle = new FriendsDataSource(this);
22 }
23
```

Ergänzung der Testausgabe.

In der onCreate()-Methode ergänzen wir im Konstruktor eine erste Testausgabe des Log-Objektes, erzeugen ein Objekt der Klasse → FriendsDataSource und übermitteln dabei den aktuellen Objektzustand

Eingabehilfe:

```
//Log-Meldung erzeugen und im Logcat ausgeben.
Log.d(LOG_TAG,
    "Das Datenquellen-Objekt wird angelegt.");
dieDatenquelle = new FriendsDataSource(this);
```

Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.

Der Klasse → FriendsDataSource. Die Klasse dient der Datenbanksteuerung. Diese Klasse werden wir gleich im Anschluss an die Ergänzungen dieser Klasse durchführen.

```
32 //Einfügen eines Testdatensatzes
33 //(nur nutzen falls keiner enthalten ist).
34 insertTestDataEntry();
```

Einfügen eines Testdatensatzes.

Diese Methode nutzen später nur noch optional, nämlich dann, wenn noch keine Testdaten enthalten sind.

Eingabehilfe:

```
//Einfügen eines Testdatensatzes
//(nur nutzen falls keiner enthalten ist).
insertTestDataEntry();
```

Die Implementierung erfolgt nun im Anschluss. Wir beginnen mit

- → showAlldataEntry() und
- → insertTestDataEntry()

Fügen Sie den Methodenaufruf zum Einfügen eines Testdatensatzes im Anschluss an die bestehenden Anweisungen in die onCreate(Bundle savedInstanceState)-Methode, ein.

```

36
37 //HILFSMETHODEN
38
39 //Anzeigen aller Freunde in einer ListView.
40 private void showAllListEntries () {
41     /*Deklaration einer Liste vom Typ Freund.
42     * Initialisierung mit allen bisher in der
43     * Datenbank gespeicherten Freunden.*/
44     List<Friend> friendList
45         = dieDatenquelle.getAllFriends();
46
47     /*Erzeugen einer ListView-Komponente um die
48     * Freundesliste auf der Benutzeroberfläche
49     * anzuzeigen.*/
50     ListView lvFreunde_anzeigen
51         = (ListView) findViewById(R.id.lvFreunde);
52
53     /*Erzeugen eines Adapters um die Anzeige
54     * der Freundesliste um eine Auswahl-Komponente
55     * (Multiple Choice) zu erweitern*/
56     ArrayAdapter<Friend> friendArrayAdapter
57         = new ArrayAdapter<Friend>(
58             this,
59             android.R.layout.simple_list_item_multiple_choice,
60             friendList){

```

```

61
62     /* Da die Anzeige standardmäßig einzeilig ist,
63     * wir aber mit name,phone und email drei Zeilen
64     * anzeigen möchten, ergänzen wir Quellcode
65     * um die Höhe der Items in der Itemlist zu
66     * regulieren/anzupassen.*/
67     @Override
68     public View getView(int position,
69                         View convertView,
70                         ViewGroup parent){
71         /*Hole das aktuelle Listenelement aus der
72         * ListView und initialisiere damit das
73         * lokale view-Objekt.*/
74         View view = super.getView(position,convertView,parent);
75
76         /*Ermittle daran die aktuellen Layout-Parameter.*/
77         ViewGroup.LayoutParams params = view.getLayoutParams();
78
79         /*Die Höhe des Listenelements manuell
80         * regulieren und setzen.*/
81         params.height = 200;
82         view.setLayoutParams(params);
83
84         //Rückgabe der überarbeiteten Listenansicht
85         return view;
86     }
87 };
88
89 /*Übermittlung des konfigurierten Adapters an die
90 ListView-Komponente.*/
91 lvFreunde_anzeigen.setAdapter(friendArrayAdapter);
92 }

```

Hinweis:

Die Methode → `getAllFriends()` werden wir gleich noch in die noch fehlende Klasse → `FriendsDataSource` implementieren.

Hilfsmethode `showAllListEntries()` einfügen.

Die Methode ermittelt alle Freunde und zeigt sie mit ihren Eigenschaftswerten in der ListView auf der Benutzeroberfläche an.

Eingabehilfe:

```

//Anzeigen aller Freunde in einer ListView.
private void showAllListEntries () {
    /*Deklaration einer Liste vom Typ Freund.
    * Initialisierung mit allen bisher in der
    * Datenbank gespeicherten Freunden.*/
    List<Friend> friendList
        = dieDatenquelle.getAllFriends();

    /*Erzeugen einer ListView-Komponente um die
    * Freundesliste auf der Benutzeroberfläche
    * anzuzeigen.*/
    ListView lvFreunde_anzeigen
        = (ListView) findViewById(R.id.lvFreunde);

    /*Erzeugen eines Adapters um die Anzeige
    * der Freundesliste um eine Auswahl-Komponente
    * (Multiple Choice) zu erweitern*/
    ArrayAdapter<Friend> friendArrayAdapter
        = new ArrayAdapter<Friend>(
            this,
            android.R.layout.simple_list_item_multiple_choice,
            friendList){

        /* Da die Anzeige standardmäßig einzeilig ist,
        * wir aber mit name,phone und email drei Zeilen
        * anzeigen möchten, ergänzen wir Quellcode
        * um die Höhe der Items in der Itemlist zu
        * regulieren/anzupassen.*/
        @Override
        public View getView(int position,
                            View convertView,
                            ViewGroup parent){

            /*Hole das aktuelle Listenelement aus der
            * ListView und initialisiere damit das
            * lokale view-Objekt.*/
            View view = super
                .getView(position,convertView,parent);

            /*Ermittle daran die aktuellen Layout-Parameter.*/
            LayoutParams params = view.getLayoutParams();

            /*Die Höhe des Listenelements manuell
            * regulieren und setzen.*/
            params.height = 200;
            view.setLayoutParams(params);

            //Rückgabe der überarbeiteten Listenansicht
            return view;
        }
    };

    /*Übermittlung des konfigurierten Adapters an die

```

ListView-Komponente./**

```
lvFreunde_anzeigen.setAdapter(friendArrayAdapter);
}
```

```

94 //Einfügen eines Testdatensatzes
95 private void insertTestDataEntry(){
96 //Test 2: Erweiterung
97 Log.d(LOG_TAG, "Öffnen der Datenquelle.");
98 dieDatenquelle.open();
99
100 //Test 3: Erweiterung
101 Friend einFreund = dieDatenquelle.createFriend(
102     "Tim", "49 (173) 1234566", "tim@mydomain.com");
103
104 //Log-Meldung erzeugen und im Logcat ausgeben.
105 Log.d(LOG_TAG, "Es wurde der folgende Datensatz eingefügt.");
106 Log.d(LOG_TAG, "ID: " + einFreund.getId() + "\n, Inhalt:\n"
107     + einFreund.toString());
108 Log.d(LOG_TAG, "Folgende Einträge sind in der Datenbank vorhanden.");
109
110 //Anzeigen aller Freunde
111 showAllListEntries();
112
113
114 //zu Test 2: Erweiterung
115 //Log-Meldung erzeugen und im Logcat ausgeben.
116 Log.d(LOG_TAG, "Schließen der Datenquelle.");
117
118 //Schließen der Datenbank
119 dieDatenquelle.close();
120 }
121

```

Hilfsmethode insertTestDataEntries() einfügen.

Die Methode öffnet die Datenquelle, fügt einen Testdatensatz ein und schließt die Datenquelle anschließend wieder.

Hinweis:

Die Methode → open(), → createFriend() und → close() werden wir gleich noch in die noch fehlende Klasse → FriendsDataSource implementieren.

Eingabehilfe:

```

//Einfügen eines Testdatensatzes
private void insertTestDataEntry(){

    //Test: Öffnen
    dieDatenquelle.open();
    Log.d(LOG_TAG, "Öffnen der Datenquelle.");

    //Test 3: Einfügen
    Friend einFreund
    = dieDatenquelle.createFriend("Tim",
    "49 (173) 1234566", "tim@mydomain.com");

    //Log-Meldung erzeugen und im Logcat ausgeben.
    Log.d(LOG_TAG,
    "Es wurde der folgende Datensatz eingefügt.");

    Log.d(LOG_TAG, "ID: "
    + einFreund.getId()
    + "\n, Inhalt:\n"
    + einFreund.toString());

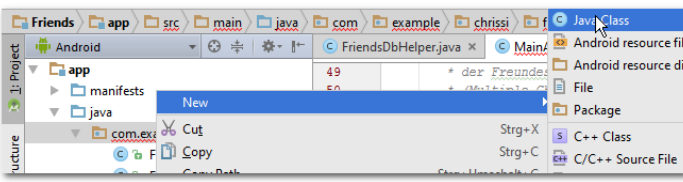
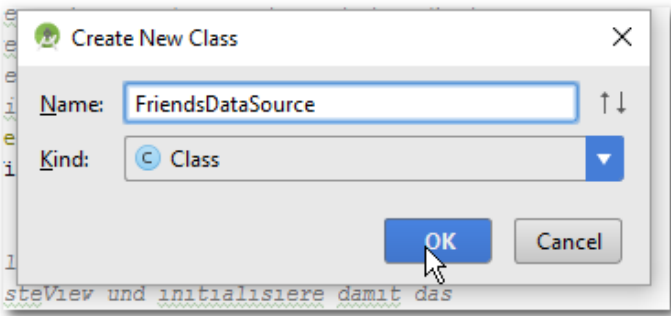
    Log.d(LOG_TAG,
    "Folgende Einträge sind in der Datenbank
    vorhanden.");


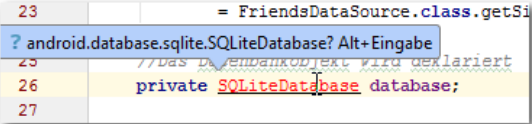
    //Anzeigen aller Freunde
    showAllListEntries();

    //zu Test 2: Erweiterung
    //Log-Meldung erzeugen und im Logcat ausgeben.
    Log.d(LOG_TAG, "Schließen der Datenquelle.");

    //Schließen der Datenbank
    dieDatenquelle.close();
}

```

}	
<div data-bbox="98 327 764 1043" style="border: 1px solid black; padding: 5px;"> <p style="text-align: center;">FriendsDataSource</p> <pre> - LOG_TAG:String = FriendsDataSource.class.getSimpleName() - database:SQLiteDatabase - dbHelper: FriendsDbHelper - columns: String[]= { FriendsDbHelper.COLUMN_ID, FriendsDbHelper.COLUMN_NAME, FriendsDbHelper.COLUMN_PHONE, FriendsDbHelper.COLUMN_EMAIL }; + FriendsDataSource(Context context) + open() + close() + createFriend(String name, String phone,String email): Friend - cursorToFriend(Cursor cursor): Friend + getAllFriends():Liste<Friend> + deleteFriend(Friend friend) + updateFriend(long id, String newName, String newPhone, String newEmail): Friend </pre> <p style="text-align: center;">UML-Klasse: FriendsDataSource.java</p> </div>	<p><i>Die Klasse FriendsDataSource.java.</i></p> <p>Arbeiterklasse. Objekte dieser Klasse übernehmen die Steuerung aller Datenbankoperationen. Sie ist unsere Datenquelle und besitzt daher eine dauerhafte Verbindung zur SQLite Datenbank.</p> <p>In den folgenden Schritten werden wir die Klasse erzeugen und mit den vorerst wichtigsten Eigenschaften und Methoden ausstatten.</p> <p>Die Lösch- und Änderungsoperationen werden wir zu einem späteren Zeitpunkt implementieren.</p>
	<p><i>Die Klasse FriendsDataSource.java erzeugen.</i></p> <p>Klicken Sie dazu im → app-Verzeichnis mit der rechten Maustaste auf das Package und wählen Sie die Option New → Java Class.</p> <p>Führen Sie nun Schrittweise die Implementierung dieser Klassen durch.</p>
	<p><i>Klassenname festlegen.</i></p> <p>Geben Sie als Klassennamen → FriendsDataSource ein und klicken Sie auf die Schaltfläche → OK.</p>

	
<pre> 19 public class FriendsDataSource { 20 21 //ATTRIBUTE: <u>Eigenschaften</u> der Klasse 22 23 /*Ein String-Objekt LOG-TAG um die <u>Abarbeitung</u> 24 der <u>Programmlogik sichtbar zu machen</u>. Die <u>Meldungen</u> 25 im <u>Logcat-Fenster</u> helfen während der <u>Entwicklung</u> 26 beim <u>Debugging</u>.*/ 27 private static final String LOG_TAG 28 = FriendsDataSource.class.getSimpleName(); 29 </pre> <p>Eingabehilfe:</p> <pre> /*Ein String-Objekt LOG-TAG um die Abarbeitung der Programmlogik sichtbar zu machen. Die Meldungen im Logcat-Fenster helfen während der Entwicklung beim Debugging.*/ private static final String LOG_TAG = FriendsDataSource.class.getSimpleName(); </pre>	<p><i>Die Eigenschaft LOG_TAG.</i></p> <p>Im Ersten Schritt deklarieren wir ein String-Objekt → LOG-TAG um die Abarbeitung der Programmlogik protokollieren zu können. Wir lassen uns später die Meldung im Logcat-Fenster ausgeben. Das hilft während der Entwicklung, vor allem dann, wenn wir Fehler suchen.</p> <p>Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.</p>
<pre> 27 //Das Datenbankobjekt wird deklariert 28 private SQLiteDatabase database; 29 </pre> <p>Eingabehilfe:</p> <pre> //Das Datenbankobjekt wird deklariert private SQLiteDatabase database; </pre> 	<p><i>Datenbankobjekt deklarieren.</i></p> <p>Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.</p> <p>Fehlende Importanweisungen einfügen: Klicken auf die rot angezeigten Klassennamen und wählen Sie die Tastenkombination ALT + ENTER, um die fehlende Importanweisung einzufügen.</p>
<pre> 30 //Ein Steuerungsobjekt für die Datenbank wird deklariert 31 private FriendsDbHelper dbHelper; 32 </pre> <p>Eingabehilfe:</p> <pre> //Ein Steuerungsobjekt für die //Datenbank wird deklariert private FriendsDbHelper dbHelper; </pre>	<p><i>FriendsDbHelperobjekt deklarieren.</i></p> <p>Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.</p>

```

33      /*Ein Array mit Strings wird deklariert und mittels
34      der Datenbanksteuerung mit den Feldern aus der
35      Datenbank initialisiert*/
36      private String[] columns = {
37          FriendsDbHelper.COLUMN_ID,
38          FriendsDbHelper.COLUMN_NAME,
39          FriendsDbHelper.COLUMN_PHONE,
40          FriendsDbHelper.COLUMN_EMAIL
41      };

```

String-Array deklarieren und initialisieren.

Der Container dient später zur Verwaltung der ermittelten Eigenschaftswerte aus der Datenbanktabelle.

Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.

Eingabehilfe:

```

/*Ein Array mit Strings wird deklariert und mittels
der Datenbanksteuerung mit den Feldern aus der
Datenbank initialisiert*/
private String[] columns = {
    FriendsDbHelper.COLUMN_ID,
    FriendsDbHelper.COLUMN_NAME,
    FriendsDbHelper.COLUMN_PHONE,
    FriendsDbHelper.COLUMN_EMAIL
};

```

```

43      //Konstruktor
44      public FriendsDataSource(Context context) {
45          //Log-Meldung erzeugen und im Logcat ausgeben.
46          Log.d(LOG_TAG, "Unsere Datenquelle erzeugt jetzt unere Helferklasse.");
47          dbHelper = new FriendsDbHelper(context);
48      }

```

Konstruktor implementieren.

Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.

Eingabehilfe:

```

//Konstruktor
public FriendsDataSource(Context context) {
    //Log-Meldung erzeugen und im Logcat ausgeben.
    Log.d(LOG_TAG,
        "Unsere Datenquelle erzeugt jetzt unere Hel-
ferklasse.");
    dbHelper = new FriendsDbHelper(context);
}

```

Fehlende Importanweisungen einfügen.

Klicken dazu auf die rot angezeigten Klassen-
namen und wählen Sie die Tastenkombination
ALT + ENTER, um die fehlende Importanwei-
sung einzufügen.

Für die Klasse Context:

```

41      ..
42      //Konstruktor
43      public FriendsDataSource(Context context) {
44          //Log-Meldung erzeugen und im Logcat ausgeben.

```

Für die Klasse Log:

```

43      //Konstruktor
44      public FriendsDataSource(Context context) {
45          //Log-Meldung erzeugen und im Logcat ausgeben.
46          Log.d(LOG_TAG, "Unsere Datenquelle erzeugt jetzt unere Hel-
47          ferklasse.");

```

```

50 //SONSTIGE METHODEN
51
52 //Öffnen der Datenbank
53 public void open() {
54     //Log-Meldung erzeugen und im Logcat ausgeben.
55     Log.d(LOG_TAG,
56         "Der Ort unserer Datenbank wird jetzt angefragt.");
57
58     //Erzeuge ein beschreibbares Datenbankobjekt (Rechte setzen)
59     database = dbHelper.getWritableDatabase();
60
61     //Log-Meldung erzeugen und im Logcat ausgeben.
62     Log.d(LOG_TAG, "Pfad erhalten. Pfad zur Datenbank: "
63         + database.getPath());
64 }

```

Öffnen einer Datenquelle.

Mittels des Helferobjektes → dbHelper öffnen wir die Datenquelle. Der Methodenaufwurf → getWritableDatabase(), stellt sicher, dass wir rechtlich gesehen auch Schreibrechte haben.

Die Logs stellen sicher, dass die Abarbeitung der einzelnen Vorgänge protokolliert wird.

Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.

Eingabehilfe:

```

//Öffnen der Datenbank
public void open() {
//Log-Meldung erzeugen und im Logcat ausgeben.
Log.d(LOG_TAG,
"Der Ort unserer Datenbank wird jetzt angefragt.");

//Erzeuge ein beschreibbares Datenbankobjekt (Rechte
setzen)
    database = dbHelper.getWritableDatabase();

//Log-Meldung erzeugen und im Logcat ausgeben.
Log.d(LOG_TAG,
    "Pfad erhalten. Pfad zur Datenbank: "
    + database.getPath());
}

```

```

66 //Schließen der Datenbank
67 public void close() {
68     dbHelper.close();
69
70     //Log-Meldung erzeugen und im Logcat ausgeben.
71     Log.d(LOG_TAG,
72         "Datenbank mit Hilfe der Helferklasse geschlossen.");
73 }

```

Schließen einer Datenquelle.

Mittels des Helferobjektes → dbHelper schließen wir die Datenquelle.

Die Logs stellen sicher, dass die Abarbeitung der einzelnen Vorgänge protokolliert wird.

Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.

Eingabehilfe:

```

//Schließen der Datenbank
public void close() {
    //Schließen der Datenquelle
    dbHelper.close();

//Log-Meldung erzeugen und im Logcat ausgeben.
Log.d(LOG_TAG,
"Datenbank mit Hilfe der Helferklasse geschlossen.");
}

```



```

C FriendsDbHelper.java x C MainActivity.java x C FriendsDataSource.java x
75 //Datensatz (Freund) einfügen
76 public Friend createFriend(String name, String phone,String email) {
77     /*ContentValues eignet sich zur Verwaltung von Daten aus der
78     Datenbank. Dazu wird ein solches Listenobjekt erzeugt.
79     Dann werden mit dem Methodenaufruf put die Werte in
80     die Liste aufgenommen.*/
81     ContentValues values = new ContentValues();
82     values.put(FriendsDbHelper.COLUMN_NAME, name);
83     values.put(FriendsDbHelper.COLUMN_PHONE, phone);
84     values.put(FriendsDbHelper.COLUMN_EMAIL, email);
85
86     /*Einfügen der Daten (values) in die Datenbank,
87     *dabei wird die verwendete id
88     zurückgegeben.*/
89     long insertId = database.insert(
90         FriendsDbHelper.TABLE_CONTACT_LIST, null, values);
91
92     /*Das Cursorobjekt enthält das Resultset
93     (Ergebnisliste) der Abfrage,
94     legt es quasi frei, sodass systematisch auf d
95     ie Ergebniselemente
96     zugegriffen werden kann.*/
97     Cursor cursor = database.query(
98         FriendsDbHelper.TABLE_CONTACT_LIST,
99         columns, FriendsDbHelper.COLUMN_ID + "="
100         + insertId,null, null, null, null);
101
102     /*Ermittelt das erste Element der Ergebnisliste*/
103     cursor.moveToFirst();

```

```

105 /*Ermittelt die Eigenschaftswerte des Datensatzes,
106 erzeugt ein neues Freund-Objekt und gibt das Objekt zurück.*/
107 Friend einFreund = cursorToFriend(cursor);
108
109 //Schließt die Ergebnisliste
110 cursor.close();
111
112 //Gibt das ermittelte Objekt zurück
113 return einFreund;
114 }

```

Eingabehilfe:

```

//Datensatz (Freund) einfügen
public Friend createFriend(String name,
    String phone,String email) {

/*ContentValues eignet sich zur Verwaltung von Daten
aus der Datenbank. Dazu wird ein solches Listenobjekt
erzeugt. Dann werden mit dem Methodenaufruf put die
Werte in die Liste aufgenommen.*/
    ContentValues values = new ContentValues();
    values.put(FriendsDbHelper.COLUMN_NAME, name);
    values.put(FriendsDbHelper.COLUMN_PHONE, phone);
    values.put(FriendsDbHelper.COLUMN_EMAIL, email);

/*Einfügen der Daten (values) in die Datenbank,
*dabei wird die verwendete id
zurückgegeben.*/
    long insertId =
        database.insert(
            FriendsDbHelper.TABLE_CONTACT_LIST, null, values);

/*Das Cursorobjekt enthält das Resultset
(Ergebnisliste) der Abfrage,
legt es quasi frei, sodass systematisch auf d
ie Ergebniselemente
zugegriffen werden kann.*/
    Cursor cursor

```

Einen Freund erzeugen und einfügen.

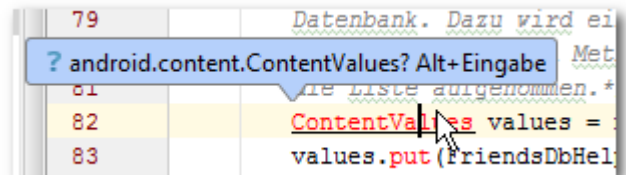
Fügt die Eigenschaftswerte eines Freundes in ein Inhaltsobjekt ein, übermittelt den SQL-String mit dem Einfüge-Befehl an die Datenbank, ermittelt dabei die dazu genutzte id und führt den String auf der Datenbank aus.

In der Ergebnismenge wird das Erste Element ermittelt und zurückgegeben.

Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.

Fehlende Importanweisungen einfügen.

Klicken Sie dazu auf die rot angezeigten Klassennamen und wählen Sie die Tastenkombination ALT + ENTER, um die fehlende Importanweisung einzufügen.



```

    = database.query(
        FriendsDbHelper.TABLE_CONTACT_LIST,
        columns,
        FriendsDbHelper.COLUMN_ID + "="
        + insertId,null, null, null, null);

    /*Ermittelt das erste Element der Ergebnisliste*/
    cursor.moveToFirst();

    /*Ermittelt die Eigenschaftswerte des Datensatzes,
    erzeugt ein neues Freund-Objekt und gibt das Ob-
    jekt zurück.*/
    Friend einFreund = cursorToFriend(cursor);

    //Schließt die Ergebnisliste
    cursor.close();

    //Gibt das ermittelte Objekt zurück
    return einFreund;
}

```

```

120 //Hilfsmethode
121 private Friend cursorToFriend(Cursor cursor) {
122     //Position der Eigenschaftswerte ermitteln
123     int idIndex = cursor.getColumnIndex(FriendsDbHelper.COLUMN_ID);
124     int idName = cursor.getColumnIndex(FriendsDbHelper.COLUMN_NAME);
125     int idPhone = cursor.getColumnIndex(FriendsDbHelper.COLUMN_PHONE);
126     int idEmail = cursor.getColumnIndex(FriendsDbHelper.COLUMN_EMAIL);
127
128     //Eigenschaftswerte anhand der Position ermitteln
129     String name = cursor.getString(idName);
130     String phone = cursor.getString(idPhone);
131     String email = cursor.getString(idEmail);
132     long id = cursor.getLong(idIndex);
133
134     /*Erzeugt ein neues Objekt vom Typ Freund und übermittelt die
135     * Eigenschaftswerte für name, phone, email und id*/
136     Friend einFreund = new Freund(name, phone, email, id);
137
138     //Schließt die Ergebnisliste
139     return einFreund;
140 }
141

```

Hilfsmethode `cursorToFriend(Cursor cursor)` einfügen.

Übermittelt die Eigenschaftswerte des Datensatzes an ein neues Objekt der Klasse Freund und gibt das Objekt zurück.

Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.

Eingabehilfe:

```

//Hilfsmethode
private Friend cursorToFriend(Cursor cursor) {
    //Position der Eigenschaftswerte ermitteln
    int idIndex
= cursor.getColumnIndex(
    FriendsDbHelper.COLUMN_ID);

    int idName
= cursor.getColumnIndex(
    FriendsDbHelper.COLUMN_NAME);

    int idPhone
= cursor.getColumnIndex(
    FriendsDbHelper.COLUMN_PHONE);

    int idEmail
= cursor.getColumnIndex(
    FriendsDbHelper.COLUMN_EMAIL);

    //Eigenschaftswerte anhand der Position ermitteln
    String name = cursor.getString(idName);

```

```

String phone = cursor.getString(idPhone);
String email = cursor.getString(idEmail);
long id = cursor.getLong(idIndex);

    /*Erzeugt ein neues Objekt vom Typ Freund und
    übermittle die Eigenschaftswerte für name, phone,
    email und id*/
    Friend einFreund
        = new Friend(name, phone,email, id);

    //Gibt das Objekt zurück
    return einFreund;
}

```

```

FriendsDataSource.java x
143 public List<Friend> getAllFriends() {
144     //Erzeugen einer ArrayList (Container) vom Typ Friend
145     List<Friend> friendList = new ArrayList<>();
146
147     /*Das Cursorobjekt enthält das Resultset
148     (Ergebnisliste) der Abfrage, legt es quasi frei,
149     sodass systematisch auf die Ergebniselemente
150     zugegriffen werden kann.*/
151     Cursor cursor = database.query(
152         FriendsDbHelper.TABLE_CONTACT_LIST,
153         columns, null, null, null, null);
154
155     /*Ermittelt das erste Element der Ergebnisliste*/
156     cursor.moveToFirst();
157
158     //Deklaration eines Objektes vom Typ Friend
159     Friend friend;
160

```

```

161     //Solange das Ende der Ergebnisliste noch nicht
162     //erreicht ist
163     while(!cursor.isAfterLast()) {
164
165         /*Initialisiere das Freundobjekt mit dem
166         Freundobjekt aus der Ergebnisliste*/
167         friend = cursorToFriend(cursor);
168
169         //Füge den Freund in die ArrayList hinzu
170         friendList.add(friend);
171
172         //Log-Meldung erzeugen und im Logcat ausgeben.
173         Log.d(LOG_TAG, "ID: " + friend.getId() + ", Inhalt: "
174             + friend.toString());
175         cursor.moveToNext();
176     }
177
178     //Schließe die Ergebnisliste
179     cursor.close();
180
181     //Rückgabe der ArrayList mit allen Freunden
182     return friendList;
183 }
184

```

Hilfsmethode getAllFriends() einfügen.

Wählt alle Freunde aus der Datenbank aus und gibt die Liste mit dem Ergebnis (Allen Datensätzen) zurück.

Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.

Eingabehilfe:

```

/*Alle gespeicherten Freunde aus der Datenbank auslesen*/
public List<Friend> getAllFriends() {
    //Erzeugen einer ArrayList (Container)
    //vom Typ Friend
    List<Friend> friendList = new ArrayList<>();

    /*Das Cursorobjekt enthält das Resultset
    (Ergebnisliste) der Abfrage, legt es quasi frei,
    sodass systematisch auf die Ergebniselemente
    zugegriffen werden kann.*/
    Cursor cursor = database.query(
        FriendsDbHelper.TABLE_CONTACT_LIST,
        columns, null, null, null, null);

    /*Ermittelt das erste Element der Ergebnisliste*/
    cursor.moveToFirst();
    //Deklaration eines Objektes vom Typ Friend
    Friend friend;

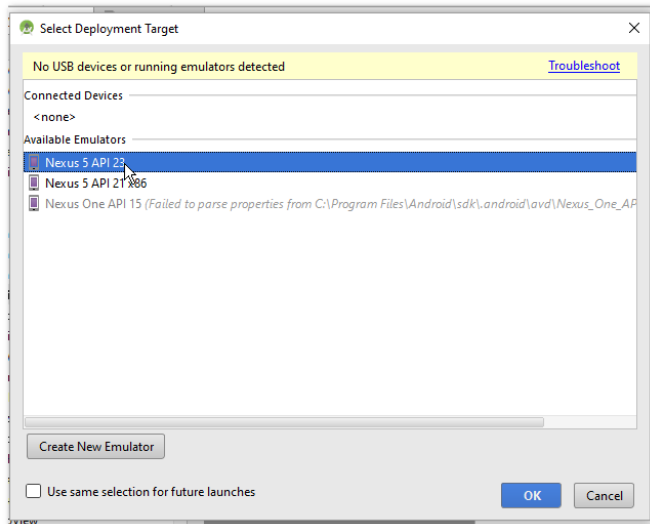
    //Solange das Ende der Ergebnisliste noch nicht
    //erreicht ist
    while(!cursor.isAfterLast()) {
        /*Initialisiere das Freundobjekt mit dem
        Freundobjekt aus der Ergebnisliste*/
        friend = cursorToFriend(cursor);
        //Füge den Freund in die ArrayList hinzu
        friendList.add(friend);
        //Log-Meldung erzeugen und im Logcat
        // ausgeben.
        Log.d(LOG_TAG, "ID: " + friend.getId()
            + ", Inhalt: " + friend.toString());
        cursor.moveToNext();
    }
}

```

```
//Schließe die Ergebnisliste
cursor.close();

//Rückgabe der ArrayList mit allen Freunden
return friendList;
}
```

Die Logs stellen sicher, dass die Abarbeitung der einzelnen Vorgänge protokolliert wird.

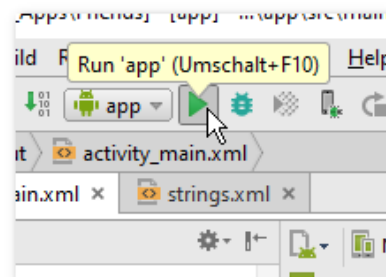


Create New Emulator:

Für wenig leistungsfähige Rechner empfiehlt sich ein neues Gerät → Nexus One Device mit API 15 (SanwichIceCream) zu erzeugen:

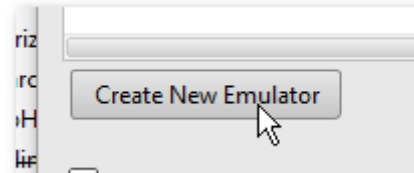
Testen der Anwendung.

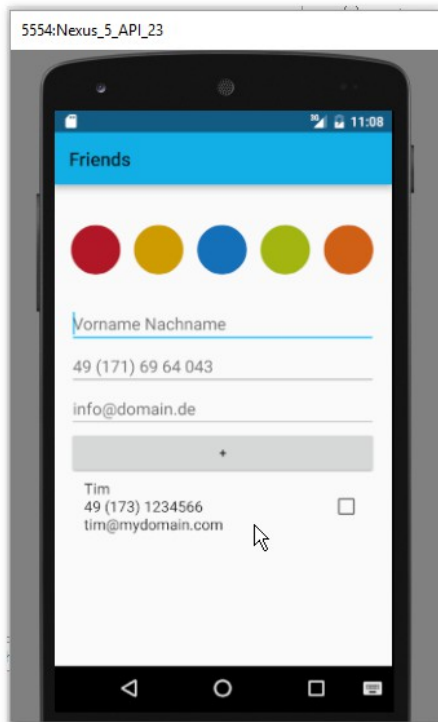
Wir starten nun den Emulator.



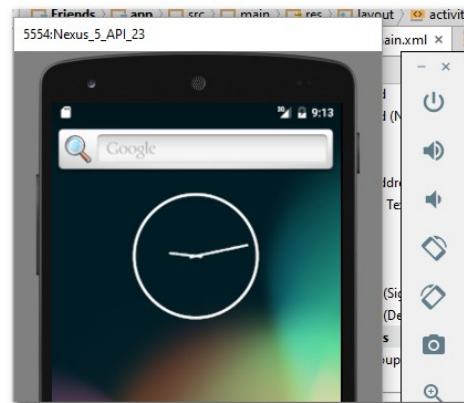
Emulator:

Der Emulator simuliert im vorliegenden Fall ein virtuelles Mobiltelefon vom Typ → Nexus 5 API 23.





Der Emulator öffnet sich.



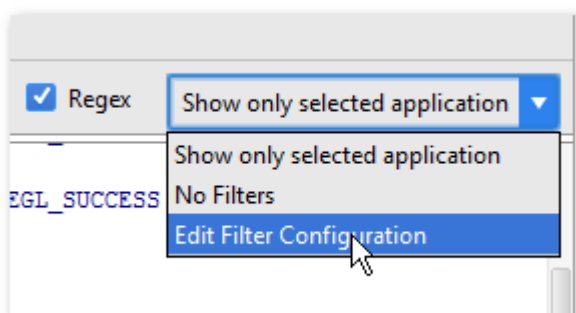
Beim ersten öffnen kann das einen Moment dauern.

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

Wenn Sie nicht ungeduldig werden, startet der Emulator die App nach Abschluss des Built-Prozesses von selbst.

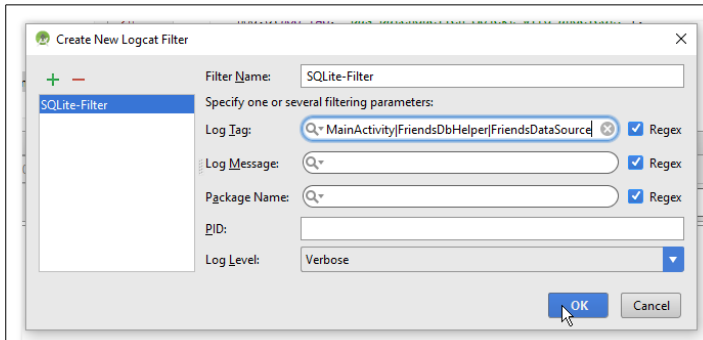
Im Ergebnis sollte die Benutzeroberfläche erscheinen:

Die Listenansicht ist nun erstmals zu sehen, da wir einen Testdatensatz eingefügt haben.



Locat-Filter nutzen.

Wählen Sie dazu im unteren Frame → Android Monitor am rechten Fensterrand im Drop-Down-Menü die Option → Edit Filter Configuration.



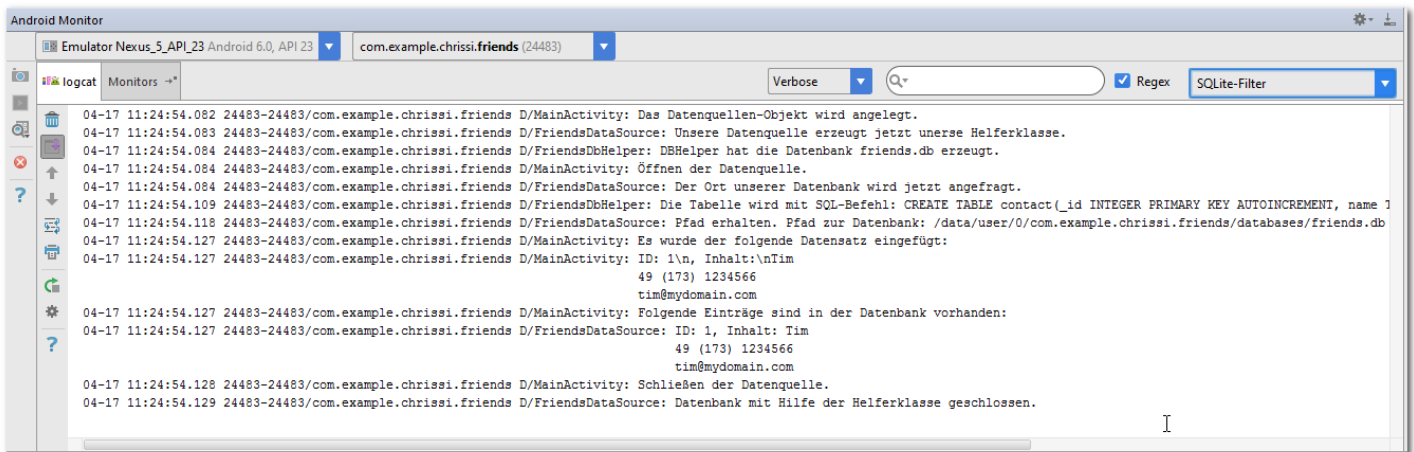
Filterkriterien festlegen.

Geben Sie dem Filter einen Namen und legen Sie die regulären Ausdrücke fest. Es sollen nur die Log Tags aus diesen Klassen im Logcat-Fenster angezeigt werden.

Filter Name:
SQLite-Filter

Log Tag:
MainActivity|FriendsDbHelper|FriendsDataSource

Anzeige der Logs.



Nachdem der Zugriff auf die Datenbank funktioniert, widmen wir uns der Interaktion mit der Benutzeroberfläche. Wir erweitern damit die Ereignissteuerung. Das Einfügen, Ändern und Löschen von Datensätzen in der Datenbank! Alle Operationen sollen bequem über die Benutzeroberfläche gesteuert werden können. Viel Spaß bei der weiteren Umsetzung!

2.6.2 Daten einfügen

Zum Einfügen eines neuen Datensatzes benötigen wir u.a. eine *Einfügeoperation*. Im konkreten Fall sollen die Daten eines neuen Freundes in der Datenbank gespeichert und anschließend in der Freundesliste auf der Benutzeroberfläche angezeigt werden. Der Benutzer gibt dazu die Werte für den Namen, die Telefonnummer und E-Mail-Adresse in die Texteingabefelder ein. Mit einem Klick auf die Schaltfläche „+“ wird der Datensatz eingefügt. Außerdem sollen im gleichen Schritt die Freundesliste auf der Benutzeroberfläche aktualisiert und angezeigt werden.

Einfügen

MainActivity erweitern.

Um die Daten über die Schaltfläche → + zum Einfügen von Datensätzen in die Datenbank nutzen zu können, implementieren wir die Methode → `activateAddButton()`.

Wir erweitern dazu die Klasse um die ereignissteuernde Methode:

→ `activateAddButton():void`

```

125 //Listener und Ereignissteuerung für die Schaltfläche Add
126 private void activateAddButton() {
127
128
129 }

```

Eingabehilfe:
//Listener und Ereignissteuerung

activateAddButton()-Methode deklarieren.

Eingabekomponenten benötigen dazu immer eine Art Fühler. Dieser Fühler ist ein Listenerobjekt. Dieses Objekt nimmt Zustandsveränderungen der Eingabekomponente war.

```
//für die Schaltfläche Add
private void activateAddButton() {
    //Hier fehlt Quellcode
}
```

Deklarieren Sie im Ersten Schritt die Methode → activateAddButton(). Ergänzen Sie dazu den Quellcode und die Kommentare, wie nebenstehend angezeigt.

```
128 private void activateAddButton() {
129
130     //Initialisierung der Komponenten
131     Button buttonAddFriend = (Button) findViewById(R.id.btAdd);
132     final EditText etName = (EditText) findViewById(R.id.etName);
133     final EditText etPhone = (EditText) findViewById(R.id.etPhone);
134     final EditText etEmail = (EditText) findViewById(R.id.etEmail);
135 }
```

Komponenten deklarieren und initialisieren.

Wir implementieren nun die Methode → activateAddButton() Schritt-Für-Schritt.

Wir müssen dazu sicherstellen, dass Komponenten, deren Inhalte gelesen bzw. in die geschrieben werden soll, zuvor initialisiert werden. Wir ergänzen dazu den Quellcode, wie nebenstehend angezeigt.

Eingabehilfe:

```
//Initialisierung der Komponenten
Button buttonAddFriend
    = (Button) findViewById(R.id.btAdd);
final EditText etName
    = (EditText) findViewById(R.id.etName);
final EditText etPhone
    = (EditText) findViewById(R.id.etPhone);
final EditText etEmail
    = (EditText) findViewById(R.id.etEmail);
```

Deklarieren und Initialisieren Sie die Eingabekomponenten. Ergänzen Sie dazu den Quellcode und die Kommentare, wie nebenstehend angezeigt.

Erklärung:

```
//Initialisierung der Komponenten
Button buttonAddFriend = (Button) findViewById(R.id.btAdd);
```

- **buttonAddFriend:**
Ist u.a. ein Klassenattribut der Activity-Klasse vom Typ EditText (siehe Deklaration).
- **(Button):**
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.btAdd**
R liefert zum String btAdd den entsprechenden int-Wert zurück. Den entsprechenden Schlüsselwert.

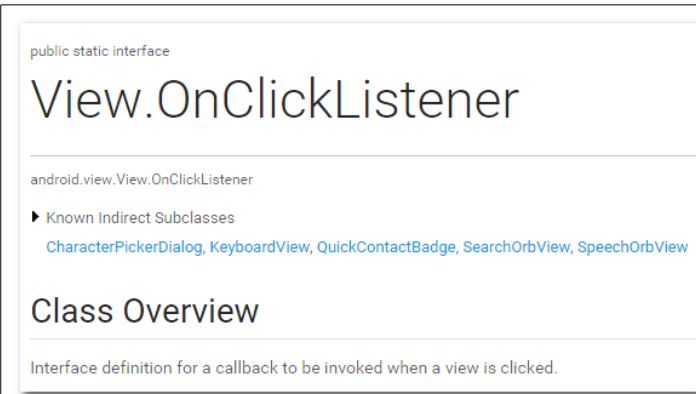
```
136 //Listener für die Button-Komponente add
137 buttonAddFriend.setOnClickListener(new View.OnClickListener() {
138     |
139     });
```

Listener für die Schaltfläche erstellen.

Auch der Button braucht einen Fühler der Aktivitäten registriert.

Ergänzen Sie dazu den Quellcode und die Kommentare, wie nebenstehend angezeigt.

Wählen Sie dabei im Dropdown-Menü die Option

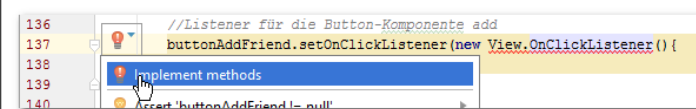


Auszug der API

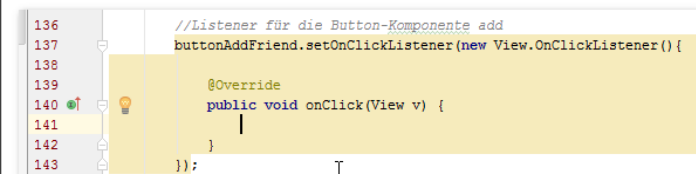
```
OnClickListener{
    //Hier fehlt Quellcode
}
```

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.



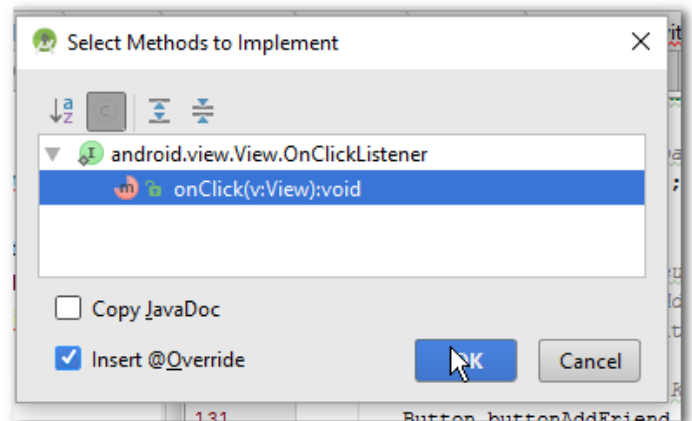
Vorher



Nachher

View.OnClickListener-Methoden deklarieren.

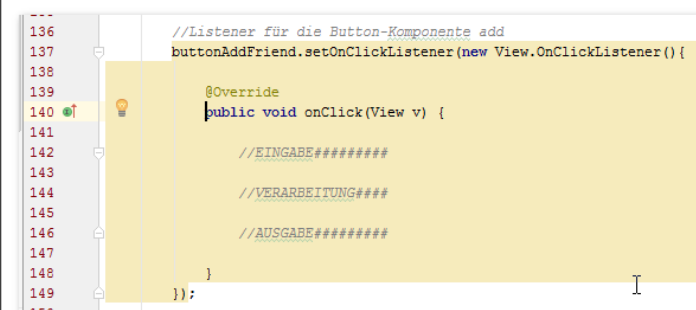
Klicken Sie dazu auf den Klassennamen → View. Mit einem Klick auf die kleine rote Glühbirne am linken Rand und der Tastenkombination ALT+ Eingabe (Enter) werden die fehlenden Methoden implementiert.



Summary

Public Methods	
abstract void	onClick(View v) Called when a view has been clicked.

Auszug der API



Die Implementierung der onCreate()-Methode:

Eingabehilfe:

```
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)

Ergänzen Sie die Kommentare und implementieren Sie dann anschließend den Quellcode schrittweise, wie es anschließend beschrieben wird.

```

142 //EINGABE*****
143 //Lesen der Inhalte aus den Texteingabefeldern für den Namen,
144 Telefonnummer und E-Mail-Adresse. Übermittlung der Werte
145 an lokale Attribute.*/
146 String nameString = etName.getText().toString();
147 String phoneString = etPhone.getText().toString();
148 String emailString = etEmail.getText().toString();
149

```

Inhalte aus den Texteingabefeldern lesen.

Der Wert für den → name wird ermittelt, in einen String umgewandelt und einem lokalen attribut → nameString zugewiesen.

Eingabehilfe:

```

String nameString
    = etName.getText().toString();
String phoneString
    = etPhone.getText().toString();
String emailString
    = etEmail.getText().toString();

```

Implementieren Sie die Lese-Anweisungen für die Texteingabefelder, wie nebenstehend angezeigt.

```

167 //*****VERARBEITUNG
168 //Prüfung ob eine der Texteingabefelder Leer ist.
169 * Wenn das der Fall ist wird eine entsprechende Meldung
170 * ausgegeben.*/
171 if (TextUtils.isEmpty(nameString)) {
172     etName.setError("Das Feld darf nicht leer sein.");
173     return;
174 }
175 if (TextUtils.isEmpty(phoneString)) {
176     etPhone.setError("Das Feld darf nicht leer sein.");
177     return;
178 }
179 if (TextUtils.isEmpty(emailString)) {
180     etEmail.setError("Das Feld darf nicht leer sein.");
181     return;
182 }

```

Prüfung der Texteingabefeldern.

Für den Fall, dass eine der Texteingabefelder leer ist soll der Nutzer den Hinweis → „das Feld darf nicht leer sein“ gegeben werden.

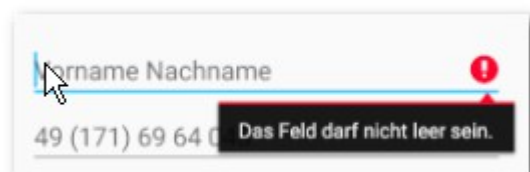
Ergänzen Sie dazu den Quellcode und die Kommentare, wie nebenstehend angezeigt.

Eingabehilfe:

```

/*Prüfung ob eine der Texteingabefelder Leer ist.
* Wenn das der Fall ist wird eine entsprechende Meldung ausgegeben.*/
if (TextUtils.isEmpty(nameString)) {
    etName.setError(getString(
        R.string.etErrorMessage));
    return;
}
if (TextUtils.isEmpty(phoneString)) {
    etPhone.setError(getString(
        R.string.etErrorMessage));
    return;
}
if (TextUtils.isEmpty(emailString)) {
    etEmail.setError(getString(
        R.string.etErrorMessage));
    return;
}

```



```

167      /*Texteingabefelder leeren: Überschreiben der
168      Texteingabefelder mit nichts*/
169      etName.setText("");
170      etPhone.setText("");
171      etEmail.setText("");
172

```

Eingabehilfe:

```

/*Texteingabefelder leeren: Überschreiben der
Texteingabefelder mit nichts*/
etName.setText("");
etPhone.setText("");
etEmail.setText("");

```

Anschließendes Leeren der Texteingabefelder.

Dazu wird der Inhalt der Texteingabefelder mittels der Methode → `.setText("")` mit nichts überschrieben.

Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.

```

173      /*Erzeuge ein neues Objekt vom Typ Freund, übermittle
174      die Eingabewerte für den Namen, die Telefonnummer und die
175      E-Mail-Adresse. Füge das Objekt in die Datenbank ein.*/
176      dieDatenquelle.createFriend(nameString,
177      phoneString, emailString);
178

```

Eingabehilfe:

```

/*Erzeuge ein neues Objekt vom Typ Freund, übermittle
die Eingabewerte für den Namen, die Telefonnummer und
die E-Mail-Adresse. Füge das Objekt in die Datenbank
ein.*/
dieDatenquelle.createFriend(
    nameString, phoneString, emailString);

```

Ein Objekt vom Typ Friend erzeugen.

Wir nutzen das Datenbankobjekt → dieDatenquelle um mittels der entwickelten `.createFriend(String name, String phone, String email)` ein neues Objekt von Typ Friend zu erzeugen und in die Datenbank einzufügen.

Klicken Sie mit gedrückter STRG-Taste auf die `.createFriend(String name, String phone, String email)`-Methode. Damit springen Sie in die Implementierung Methode. In diesem Falle springen Sie in die Klasse: → `FriendsDataSource`

```

FriendsDataSource
public Friend createFriend (String name, String phone, String email)
und die
E-Mail-Adresse. Füge das Objekt in die Datenbank ein.*/
dieDatenquelle.createFriend(nameString,
    phoneString, emailString);

```

Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.

```

178      /*Manager-Objekt für die Eingaben erzeugen. Nutzen des Services
179      zur Überprüfung der Eingaben. (variiert je nach
180      gewähltem inputType)*/
181      InputMethodManager inputMethodManager;
182      inputMethodManager
183      = (InputMethodManager) getSystemService(
184      INPUT_METHOD_SERVICE);
185      if (getCurrentFocus() != null) {
186          inputMethodManager
187          .hideSoftInputFromWindow(
188          getCurrentFocus().getWindowToken(), 0);
189

```

Eingabehilfe:

```

/*Manager-Objekt für die Eingaben erzeugen. Nutzen
des Services zur Überprüfung der Eingaben. (variiert
je nach gewähltem inputType)*/
InputMethodManager inputMethodManager;
inputMethodManager
    = (InputMethodManager) getSystemService(
INPUT_METHOD_SERVICE);
if (getCurrentFocus() != null) {
    inputMethodManager
        .hideSoftInputFromWindow(
getCurrentFocus()

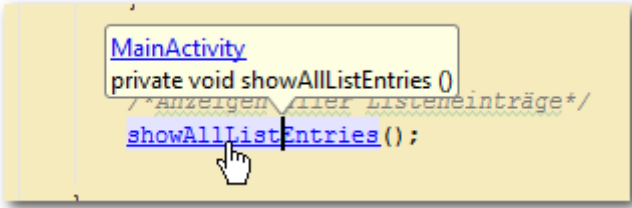
```

Den InputMethodManager nutzen.

Das Framework ermöglicht auch Geräten ohne Hardwaretastatur die Eingabe und Prüfung der Eingabe abhängig von dem gewählten → `inputType`.

Wir erzeugen ein neues Manager-Objekt für die Eingaben und aktivieren den Service. Wir nutzen diesen Service (Framework) zur Überprüfung der Eingaben. Je nach gewähltem `inputType` variieren die Prüfungen und Meldungen für den Benutzer.

Ergänzen Sie dazu den Quellcode und die Kommentare, wie nebenstehend angezeigt.

<pre> } .getWindowToken(), 0); } </pre> <p>public final class InputMethodManager <small>Summary: Constants Methods Inherited Methods [Expand All] Added in API level 3</small></p> <p>extends <code>Object</code></p> <p><small>java.lang.Object</small> <small>↳ android.view.inputmethod.InputMethodManager</small></p> <h3>~ Class Overview</h3> <p>Central system API to the overall input method framework (IMF) architecture, which arbitrates interaction between applications and the current input method. You can retrieve an instance of this interface with <code>Context.getSystemService()</code>.</p> <p>API Klasse InputMethodManager</p>	
<pre> 192 //AUSGABE##### 193 /*Anzeigen aller Listeneinträge*/ 194 showAllListEntries(); </pre> <p>Eingabehilfe:</p> <pre> /*Anzeigen aller Listeneinträge*/ showAllListEntries(); </pre> 	<p><i>Anzeige der Listeneinträge.</i></p> <p>Fügen Sie den Methodenaufruf zur Anzeige aller Listeneinträge hinzu.</p> <p>Klicken Sie mit gedrückter STRG-Taste auf die → <code>.showAllListEntries()</code>-Methode. Damit springen Sie zur Implementierung in die bereits implementierte Hilfsmethode.</p> <p>Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.</p>
<pre> 39 40 //Listener und Ereignissteuerung für die Schaltfläche Add 41 activateAddButton(); </pre> <p>Eingabehilfe:</p> <pre> //Listener und Ereignissteuerung //für die Schaltfläche Add activateAddButton(); </pre>	<p><i>Methodenaufruf für activateAddButton().</i></p> <p>Wechseln Sie dazu in die → <code>onCreate()</code>-Methode der Klasse → MainActivity.</p> <p>Fügen Sie den Methodenaufruf nun abschließend in der <code>onCreate()</code>-Methode unterhalb der Methode → <code>insertTestDataEntry()</code> ein.</p> <p>Ergänzen Sie dazu den Quellcode und die Kommentare, wie nebenstehend angezeigt.</p>

```

MainActivity.java x FriendsDbHelper.java x FriendsDataSource.java x
202
203 //ERGÄNZUNG: Für alle Änderungen innerhalb
204 // des Activity-Life-Circles
205 @Override
206 protected void onResume() {
207     super.onResume();
208
209     Log.d(LOG_TAG, "Die Datenquelle wird geöffnet.");
210     dieDatenquelle.open();
211
212     Log.d(LOG_TAG, "Folgende Einträge sind in der Datenbank vorhanden.");
213     showAllListEntries();
214 }
215
216 @Override
217 protected void onPause() {
218     super.onPause();
219
220     //Log-Meldung erzeugen und im Logcat ausgeben.
221     Log.d(LOG_TAG, "Die Datenquelle wird geschlossen.");
222     dieDatenquelle.close();
223 }

```

Eingabehilfe:

```

//ERGÄNZUNG: Für alle Änderungen innerhalb
// des Activity-Life-Circles
@Override
protected void onResume() {
    super.onResume();

    Log.d(LOG_TAG,
        "Die Datenquelle wird geöffnet.");

    dieDatenquelle.open();

    Log.d(LOG_TAG,
        "Folgende Einträge sind in der Datenbank
        vorhanden.");

    showAllListEntries();
}

@Override
protected void onPause() {
    super.onPause();

    //Log-Meldung erzeugen und im Logcat ausgeben.
    Log.d(LOG_TAG,
        "Die Datenquelle wird geschlossen.");

    dieDatenquelle.close();
}

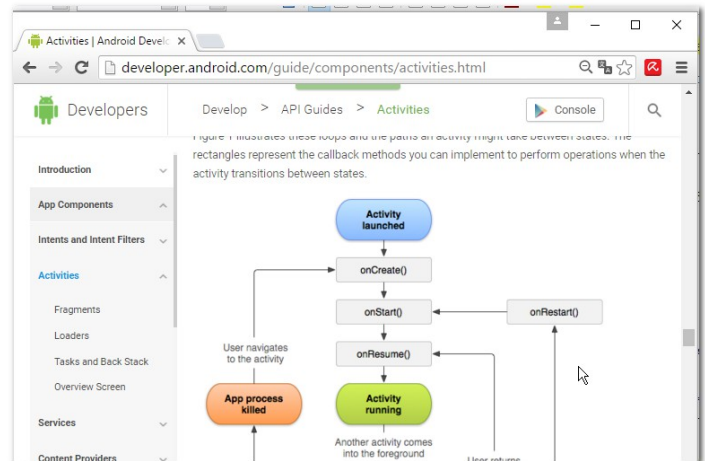
```

Überschreiben von Methoden.

Die beide Methoden → onResume() und → onPause() sind fester Bestandteil des Lebenszyklus einer Aktivität.

Hinweis:

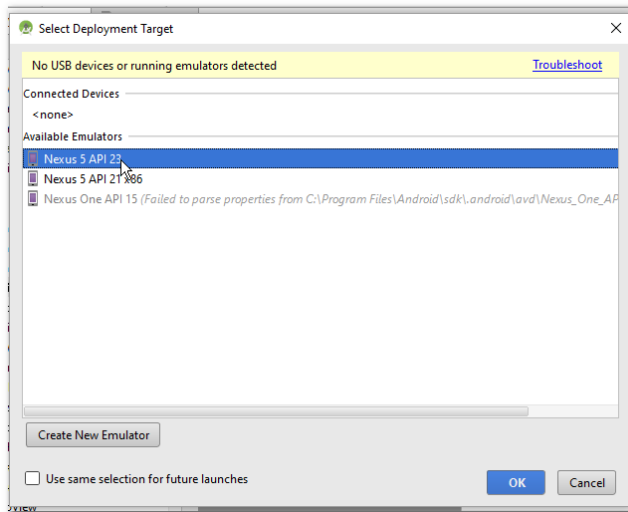
Auf den Developerseiten können Sie weitere Fakten zum Lebenszyklus erfahren.



<http://developer.android.com/guide/components/activities.html>

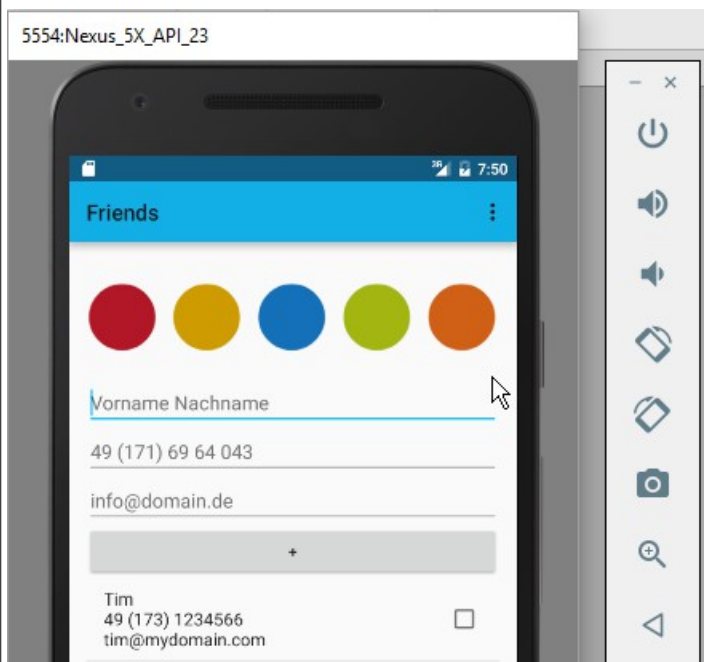
Ergänzen Sie dazu den Quellcode und die Kommentare, wie nebenstehend angezeigt.

Fertig! Wir sind nun soweit und können das Einfügen eines Datensatzes über die Benutzeroberfläche testen.



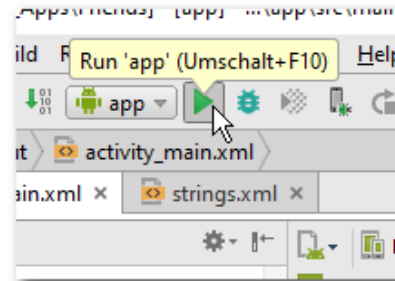
Create New Emulator:

Für wenig leistungsfähige Rechner empfiehlt sich ein neues Gerät → Nexus One Device mit API 15 (SanwichIceCream) zu erzeugen:

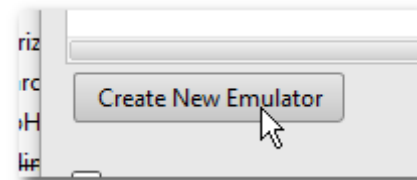


Testen der Anwendung.

Wir starten nun den Emulator.
Emulator:



Der Emulator simuliert im vorliegenden Fall ein virtuelles Mobiltelefon vom Typ → Nexus 5 API 23.



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.

Wenn Sie nicht ungeduldig werden, startet der Emulator die App nach Abschluss des Built-Prozesses von selbst.

Fügen Sie über die Benutzeroberfläche einen Datensatz ein.

Einfügen

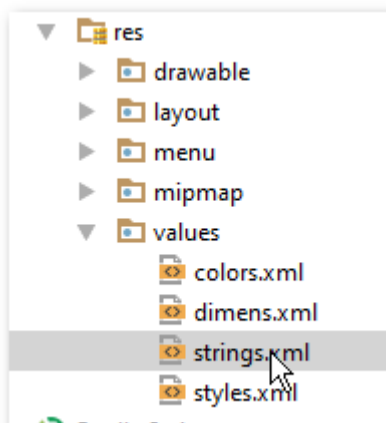
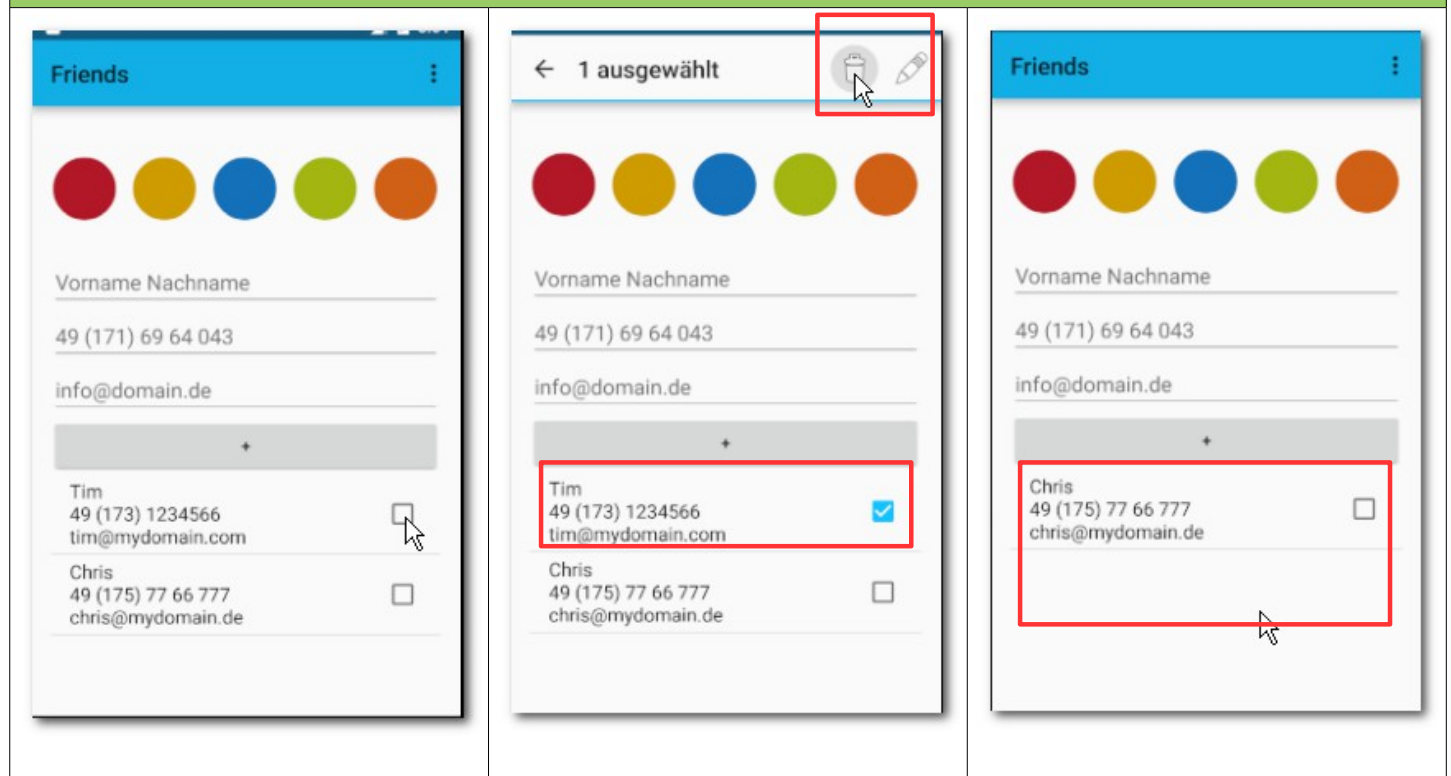
The image shows three sequential screenshots of an Android contact card interface. Each card has a header with five colored circles (red, yellow, blue, green, orange).
1. The first screenshot shows a contact card for 'Tim' with fields for 'Vorname Nachname', '49 (171) 69 64 043', and 'info@domain.de'. A red box highlights the 'Vorname Nachname' field, and a mouse cursor is pointing at it.
2. The second screenshot shows the same card, but the 'Vorname Nachname' field now contains 'Chris'. A mouse cursor is pointing at a circular icon with a plus sign below the email field.
3. The third screenshot shows the card with 'Chris' in the name field and 'chris@mydomain.de' in the email field. A mouse cursor is pointing at the bottom right corner of the card.

Gratulation!

2.6.3 Daten entfernen

Um einen Datensatz aus unserer Datenbank zu entfernen benötigen wir u.a. eine *Löschoption*. In unserem konkreten Fall soll die Löschoption über ein neues Menü steuerbar sein. Dieses Menü soll erscheinen wenn ein oder mehrere angezeigte Datensätze auf der Benutzeroberfläche ausgewählt wurden. Im folgenden werden wir also das Menü und die zugehörige Ereignissteuerung für die Löschoption ergänzen und testen.

Löschen



Bezeichner ergänzen.

Öffnen Sie dazu die Datei → strings.xml im Verzeichnis → app → res → values.

Wir erweitern im Ersten Schritt die Datei → strings.xml um die Bezeichnung die wir für Löschoption benötigen werden.

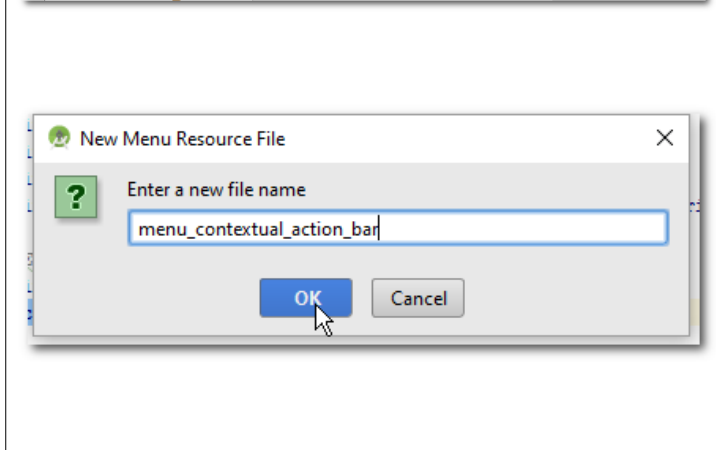
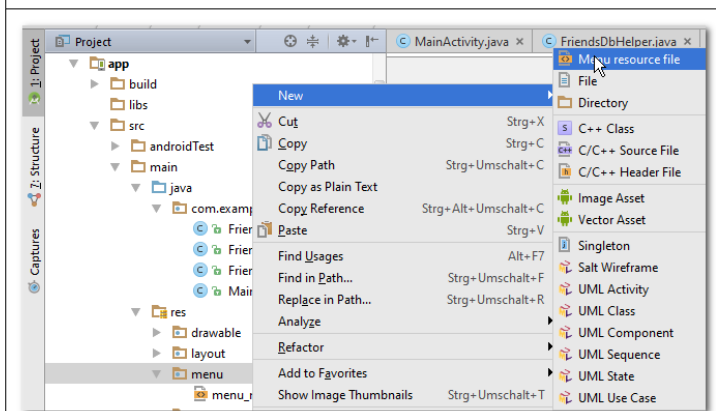
Eingabehilfe:

```
<!--Ergänzung Löschen von Datensätzen -->
<string name="cab_checked_string">ausgewählt</string>
<string name="cab_delete">Einträge löschen</string>
```

Ergänzen Sie dazu den Quellcode und die Kommentare, wie nebenstehend angezeigt.


```

1  <?xml version="1.0" encoding="utf-8" ?>
2  <resources>
3      <string name="app_name">Friends</string>
4      <string name="hello_world">Hello world!</string>
5
6      <string name="action_settings">Settings</string>
7      <string name="logo_description">Logo-Banner</string>
8      <string name="etName_hint_name">Vorname Nachname</string>
9      <string name="etPhone_hint_phone">49 (171) 69 64 043</string>
10     <string name="etEmail_hint_email">info@domain.de</string>
11     <string name="btAdd_zeichen">+</string>
12     <string name="etErrorMessage">Das Feld darf nicht leer sein.</string>
13
14     <!--Ergänzung löschen von Datensätzen -->
15     <string name="cab_checked_string">ausgewählt</string>
16     <string name="cab_delete">Einträge löschen</string>
17 </resources>
18
    
```



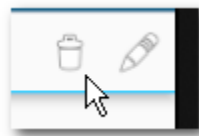
Vorher:

```

1  <?xml version="1.0" encoding="utf-8" ?>
2  <menu xmlns:android="http://schemas.android.com/apk/res/android">
3
4
5  </menu>
    
```

Datei für das Untermenü erstellen.


Eine extra Menü-Datei soll die Anzeige der Operationen (Löschen und Ändern) in der → Contextual Action Bar sicherstellen.



Klicken Sie dazu auf das Verzeichnis → app → res → menu. Wählen Sie im Kontext-Menü (rechte Maustaste) → New → Menu resource file.

Legen Sie für das neue Menü den Namen → menu_contextual_action_bar ein. Klicken Sie auf die Schaltfläche → OK.

Menüpunkt im Kontext-Menü ergänzen.



Wir nutzen dazu ein bereits vorhandenes Icon aus der Android-Bibliothek.
Eingabehilfe:

Nachher:

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <menu xmlns:android="http://schemas.android.com/apk/res/android">
3     <item
4         android:id="@+id/cab_delete"
5         android:icon="@android:drawable/ic_menu_delete"
6         android:title="@string/cab_delete" />
7
8 </menu>

```

```

<item
    android:id="@+id/cab_delete"
    android:icon="@android:drawable/ic_menu_delete"
    android:title="@string/cab_delete" />

```

Ergänzen Sie dazu den Quellcode und die Kommentare, wie nebenstehend angezeigt.

```

223 //Ergänzung löschen eines Datensatzes
224 public void deleteFriend(Friend friend) {
225     //Ermittelt die id des aktuellen Freundes
226     long id = friend.getId();
227
228     //Löscht den Freund aus der Datenbank
229     database.delete(FriendsDbHelper.TABLE_CONTACT_LIST,
230         FriendsDbHelper.COLUMN_ID + "=" + id,
231         null);
232
233     //Log-Meldung erzeugen und im Logcat ausgeben.
234     Log.d(LOG_TAG, "Eintrag gelöscht! ID: " + id
235         + " Inhalt: " + friend.toString());
236 }

```

Löschoperation implementieren.

Diese Methode soll einen ausgewählten Datensatz aus der Datenbank löschen.

Identifiziert wird der ausgewählte Datensatz an seinem identifizierendem Attribut, der → `_id`

Wechseln Sie in die Klasse → `FriendsDataSource` und implementieren Sie darin die Methode → `deleteFriend(Friend friend)`.

Eingabehilfe:

```

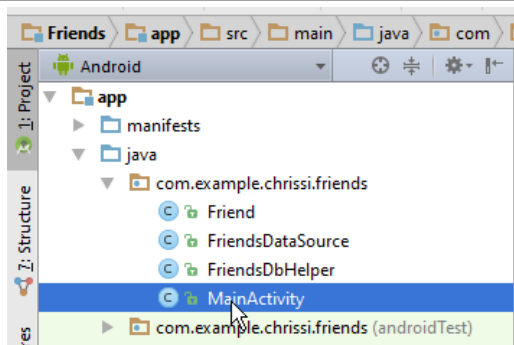
//Ergänzung löschen eines Datensatzes
public void deleteFriend(Friend friend) {
    //Ermittelt die id des aktuellen Freundes
    long id = friend.getId();

    //Löscht den Freund aus der Datenbank
    database.delete(
        FriendsDbHelper.TABLE_CONTACT_LIST,
        FriendsDbHelper.COLUMN_ID + "="
            + id,null);

    //Log-Meldung erzeugen und im Logcat ausgeben.
    Log.d(LOG_TAG, "Eintrag gelöscht! ID: " + id
        + " Inhalt: " + friend.toString());
}

```

Ergänzen Sie dazu den Quellcode und die Kommentare, wie nebenstehend angezeigt.

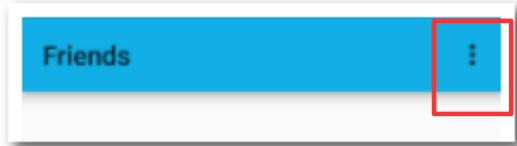


Android View

Ereignissteuerung für die Löschoperation implementieren.

Wechseln Sie dazu in die Klasse → `MainActivity`.

Da wir die Lösch- und Änderungsoperationen in in das Kontext-Menü integrieren möchten überschreiben zunächst zwei Methoden die u.a. dazu dienen die verfügbaren Menü-Einträge zu händeln.

<p>Context menu and contextual action mode</p> <p>A context menu is a floating menu that appears when the user performs a long-click on an element. It provides actions that affect the selected content or context frame.</p> <p>The contextual action mode displays action items that affect the selected content in a bar at the top of the screen and allows the user to select multiple items.</p> <p>See the section about Creating Contextual Menus.</p>	<p>Das Kontext-Menü.</p> <p>http://developer.android.com/guide/topics/ui/menus.html</p>
<pre> 243 //Überschriebene Methoden 244 @Override 245 public boolean onCreateOptionsMenu(Menu menu) { 246 //Befüllt das Menü und ergänzt die verfügbaren 247 // Menüeinträge im Action Bar Menü. 248 getMenuInflater().inflate(R.menu.menu_main, menu); 249 return true; 250 } </pre> <pre> //Überschriebene Methoden @Override public boolean onCreateOptionsMenu(Menu menu) { //Befüllt das Menü und ergänzt die verfügbaren // Menüeinträge im Action Bar Menü. getMenuInflater().inflate(R.menu.menu_main, menu); return true; } </pre>	<p>Callback-Methode onCreateOptionsMenu() überschreiben.</p> 
<pre> 249 @Override 250 public boolean onOptionsItemSelected(MenuItem item) { 251 //Abarbeitung der angeklickten Menüeinträge. Das 252 // Action Bar Menü wird die Abarbeitung der Kicks auf die Schaltflächen 253 // Home und Up automatisch durchführen sofern diese in der übergeordneten 254 // Aktivität im Android Manifest festgelegt wurden*/ 255 int id = item.getItemId(); 256 257 //noinspection SimplifiableIfStatement 258 if (id == R.id.action_settings) { 259 return true; 260 } 261 262 return super.onOptionsItemSelected(item); 263 } </pre> <pre> @Override public boolean onOptionsItemSelected(MenuItem item) { int id = item.getItemId(); //Für den Fall dass kein Eintrag gewählt wurde if (id == R.id.action_settings) { return true; } return super.onOptionsItemSelected(item); } </pre>	<p>Callback-Methode onOptionsItemSelected() überschreiben.</p> <p>Organisiert die Abarbeitung der angeklickten Menüeinträge. Das Action Bar Menü wird die Abarbeitung der Kicks auf die Schaltflächen Home und Up automatisch durchführen, sofern diese in der übergeordneten Aktivität im Android Manifest festgelegt wurden.</p> 
<pre> 230 //ERGÄNZUNG: löschen und ändern eines Datensatzes 231 232 private void initializeContextualActionBar() { 233 234 } </pre>	<p>Das Kontext-Menü.</p> <p>Implementieren Sie darin die Methode → initializeContextualActionBar().</p> <p>Deklariieren Sie die Methode, wie nebenstehend angezeigt und ergänzen Sie innerhalb</p>

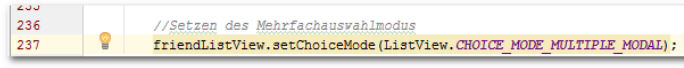
Eingabehilfe:
 //ERGÄNZUNG: löschen und ändern eines Datensatzes
private void initializeContextualActionBar() {
 //Hier fehlt Quellcode
}

dieser Methode den fehlenden Quellcode und die Kommentare, nun Schritt für Schritt.



Eingabehilfe:
 //Initialisierung der ListView-Komponente
final ListView friendListView
 = (ListView) findViewById(R.id.LvFreunde);

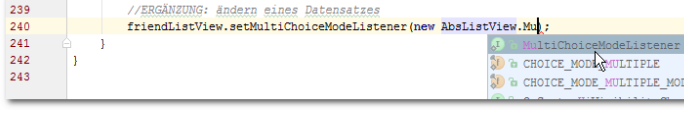
ListView-Objekt initialisieren.
 Ergänzen Sie dazu den Quellcode und die Kommentare, wie nebenstehend angezeigt.



Eingabehilfe:
 //Setzen des Mehrfachauswahlmodus
 friendListView.setChoiceMode(
 ListView.CHOICE_MODE_MULTIPLE_MODAL);

Wir nutzen die statische und finale Variable der ListView-Komponente:
 → CHOICE_MODE_MULTIPLE_MODAL

Die Mehrfachauswahl.
 Im Rahmen der Löschoperation soll der Benutzer der App ein oder mehrere Elemente der angezeigten Datensätze auswählen und löschen können.
 Ergänzen Sie dazu den Quellcode und die Kommentare, wie nebenstehend angezeigt.



public static interface
 Summary: Methods | Inherited Methods | [Expand All]
 Added in API level 11

AbsListView.MultiChoiceModeListener

implements ActionMode.Callback

android.widget.AbsListView.MultiChoiceModeListener

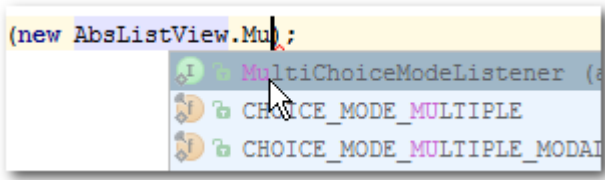
^ Class Overview

A MultiChoiceModeListener receives events for CHOICE_MODE_MULTIPLE_MODAL. It acts as the ActionMode.Callback for the selection mode and also receives onItemCheckedStateChanged(ActionMode, int, long, boolean) events when the user selects and deselects list items.

Auszug der API

Listener für die ListView erzeugen.
 Auch die ListView-Komponente braucht einen Fühler der Aktivitäten registriert.
 Klicken Sie zwischen die runden Klammern, erzeugen Sie ein Objekt der Klasse → AbsListView und wählen Sie im Dropdown-Menü die Option

MultiChoiceModeListener(//Hier fehlt Quellcode)
 {}



Eingabehilfe:
 //ERGÄNZUNG: ändern eines Datensatzes

[AbstractListView.MultiChoiceModeListener](#)
 Ist eine Interface-Klasse. Ein Interface ist so etwas



```
friendListView.setMultiChoiceModelListener();
```

wie eine Vorlage. Eigenschaften und Verhaltensweisen die im Interface deklariert sind, müssen implementiert werden, da sie eine zwingende Verhaltensweise eines Objektes darstellen.

Ergänzen Sie dazu den Quellcode und die Kommentare, wie nebenstehend angezeigt.

```
239 //ERGÄNZUNG: ändern eines Datensatzes
240 friendListView.setMultiChoiceModelListener(
241
242     new AbsListView.MultiChoiceModelListener() {
243         //Zählervariable
244         int selCount = 0;
```

Zählervariable einfügen.

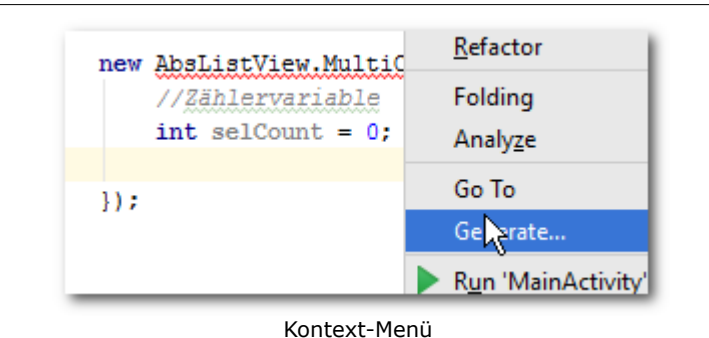
Fügen Sie den folgenden Quellcode in den gerade erstellten Listener ein.

Eingabehilfe:

```
//Zählervariable
int selCount = 0;
```

Deklariert und initialisieren Sie dazu im ersten Schritt die Zählervariable → selCount.

Ergänzen Sie dazu den Quellcode und die Kommentare, wie nebenstehend angezeigt.



Kontext-Menü

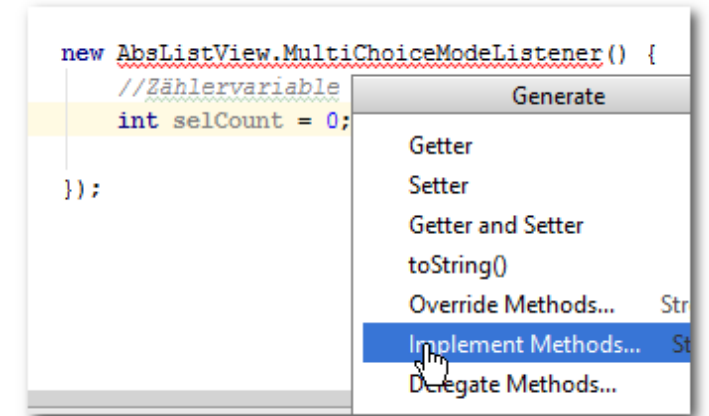
Methoden überschreiben.

Klicken Sie zum überschreiben der Methoden in den Listener und wählen Sie im Kontext-Menü die Option → Generate → implement Methods.

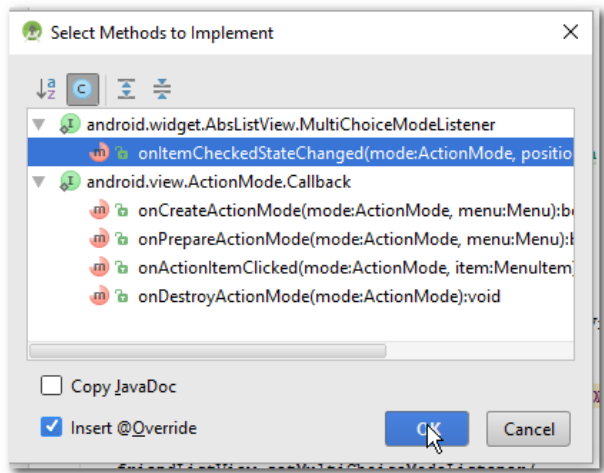
Wählen Sie dann die Methode:

Wir implementieren nun die fünf zwingend erforderlichen Listener Methoden.

```
onItemCheckedStateChanged(
    ActionMode mode,
    int position,
    long id,
    boolean checked)
```



Kontext-Menü



und Klicken Sie auf die Schaltfläche → OK.



```

245  /* Wir zählen mit der Callback-Methode die ausgewählten Listeneinträge mit
246  und fordern mit der Methode invalidate() die Aktualisierung
247  der Contextual Action Bar an.*/
248  @Override
249  public void onItemCheckedStateChanged(ActionMode mode,
250                                       int position,
251                                       long id,
252                                       boolean checked) {
253      if (checked) {
254          selCount++;
255      } else {
256          selCount--;
257      }
258      String cabTitle = selCount + " " + getString(R.string.cab_checked_string);
259      mode.setTitle(cabTitle);
260      mode.invalidate();
261  }

```

Callback-Methode `onItemCheckedStateChanged()` überschreiben.

Wir zählen mit der Callback-Methode die ausgewählten Listeneinträge mit und fordern mit der Methode `invalidate()` die Aktualisierung der Contextual Action Bar (unser neues Menü) an.

Eingabehilfe:

```

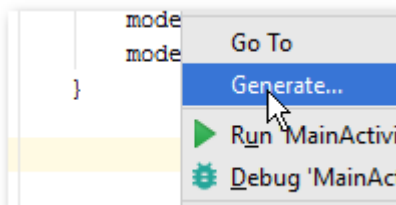
/* Wir zählen mit der Callback-Methode die ausgewählten
Listeneinträge mit und fordern mit der Methode
invalidate() die Aktualisierung der Contextual Action
Bar an.*/
@Override
public void onItemCheckedStateChanged(
    ActionMode mode,
    int position,
    long id,
    boolean checked) {

    if (checked) {
        selCount++;
    } else {
        selCount--;
    }

    String cabTitle
    = selCount + " "
    + getString(R.string.cab_checked_string);

    mode.setTitle(cabTitle);
    mode.invalidate();
}

```

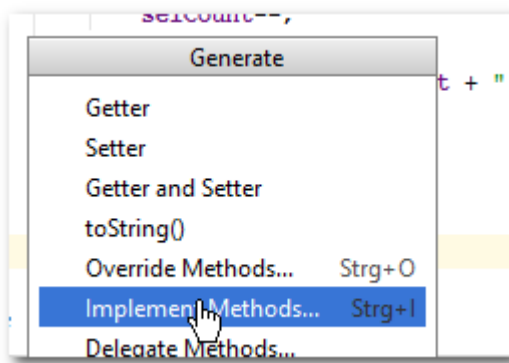


Kontext-Menü

Callback-Methode `onCreateActionMode()` überschreiben.

Klicken Sie zum überschreiben von Methoden in den Listener und wählen Sie im Kontext-Menü die Option → Generate → Implement Methods.

Wählen Sie dann die Methode → **boolean onCreateActionMode(**



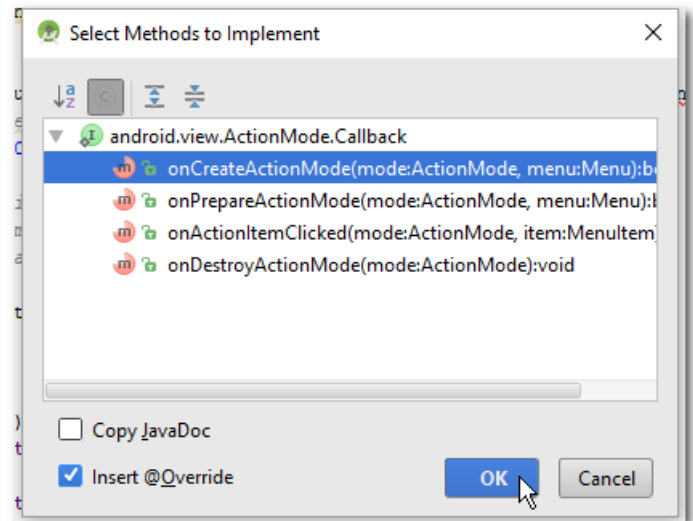
Kontext-Menü

Eingabehilfe:

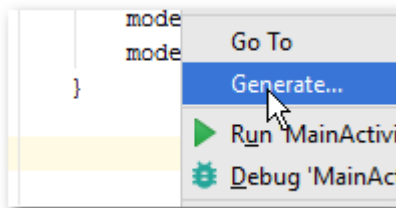
```
/* Mit dieser Callback-Methode Legen wir die Cab-
Menüeinträge in der Contextual Action Bar an.*/
@Override
public boolean onCreateActionMode(
    ActionMode mode, Menu menu) {

    getMenuInflater().inflate(
        R.menu.menu_contextual_action_bar, menu);
    return true;
}
```

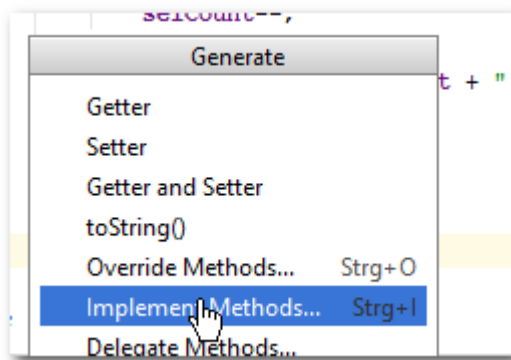
```
ActionMode mode, Menu menu){ }
```



Ergänzen Sie dazu den Quellcode und die Kommentare, wie nebenstehend angezeigt.



Kontext-Menü



Kontext-Menü

Eingabehilfe:

```
/*Mit dieser Callback-Methode reagieren wir auf Er-
eignisse (Item-Klicks). Je nachdem ob das Löschen-
oder Ändern-Symbol angeklickt wurde
soll das System reagieren.*/
@Override
```

Callback-Methode `onOptionsItemSelected()` überschreiben.

Implementieren Sie dazu den Quellcode auf gleiche Weise und ergänzen Sie die Kommentare, wie nebenstehend angezeigt.

Hinweis:

Die Kommentare erklären die erforderliche Algorithmen für die Ereignissteuerung der Löschoption über die Benutzeroberfläche.

Das Containerobjekt → `SparseBooleanArray` nutzen wir, um uns die Positionen der ausgewählten Datensätze aufzunehmen. Das Objekt ähnelt im Umgang einer `HashMap`.

```

public boolean onActionItemClicked(ActionMode mode,
MenuItem item) {

    boolean returnValue = true;
    /*Wir erzeugen ein Listenobjekt und übernehmen
    die Listenposition der ausgewählten Freunde
    aus der Liste*/
    SparseBooleanArray touchedFriendsPositions
    = friendListView
        .getCheckedItemPositions();
    switch (item.getItemId()) {
        /*ERGÄNZUNG: Löschen eines Datensatzes. Für den
        Fall dass die Löschoption gewählt wurde soll das
        gewählte Freund-Objekt anhand seiner aktuellen Posi-
        tion ermittelt und aus der Datenbank gelöscht
        werden.*/

        case R.id.cab_delete:
            for (int i = 0;
                i < touchedFriendsPositions.size(); i++) {

                boolean isChecked
                = touchedFriendsPositions.valueAt(i);
                if (isChecked) {
                    //Ermittelt die Position
                    // des gewählten, aktuellen Freundes
                    int positionInListView
                    = touchedFriendsPositions.keyAt(i);

                    //Deklaration und Initialisierung
                    // des Freundes an dieser Position
                    Friend friend
                    = (Friend) friendListView
                        .getItemAtPosition(positionInListView);

                    //Log-Meldung erzeugen und im
                    //Logcat ausgeben.
                    Log.d(LOG_TAG, "Position im ListView: "
                        + positionInListView + " Inhalt: "
                        + friend.toString());

                    //Löschen aus der Datenbank
                    dieDatenquelle.deleteFriend(friend);
                } //IF-Schließen
            } //SCHLEIFE-Schließen

            //Anzeigen der restlichen Freunde
            showAllListEntries();
            //Beenden der Aktion
            mode.finish();
            break;

        default:
            //Ansonsten soll nichts passieren
            returnValue = false;
            break;
    } //SWITCH-CASE schließen

    return returnValue;
}

```

public class **SparseBooleanArray** Summary: Ctors | Methods | Inherited Methods | [Expand All] Added in API level 1

extends [Object](#)
implements [Cloneable](#)

[java.lang.Object](#)
[↳ android.util.SparseBooleanArray](#)

^ Class Overview

SparseBooleanArrays map integers to booleans. Unlike a normal array of booleans there can be gaps in the indices. It is intended to be more memory efficient than using a [HashMap](#) to map [Integers](#) to [Booleans](#), both because it avoids auto-boxing keys and values and its data structure doesn't rely on an extra entry object for each mapping.

Note that this container keeps its mappings in an array data structure, using a binary search to find keys. The implementation is not intended to be appropriate for data structures that may contain large numbers of items. It is generally slower than a traditional [HashMap](#), since lookups require a binary search and adds and removes require inserting and deleting entries in the array. For containers holding up to hundreds of items, the performance difference is not significant, less than 50%.

It is possible to iterate over the items in this container using [keyAt\(int\)](#) and [valueAt\(int\)](#). Iterating over the keys using [keyAt\(int\)](#) with ascending values of the index will return the keys in ascending order, or the values corresponding to the keys in ascending order in the case of [valueAt\(int\)](#).

Wir können mit Hilfe der Methoden → [keyAt\(int i\)](#) und → [valueAt\(int i\)](#) die Elemente des Containers durchlaufen. Für den Fall, dass wir über die Schlüssel (keys) mit aufsteigenden Werten (int i) iterieren wird der Schlüsselwert in aufsteigender Reihenfolge ermittelt und zurückgegeben. Anderenfalls wird der zugehörigen Wert (values) selbst, ebenfalls in aufsteigender Reihenfolge ermittelt und zurückgegeben.


```

356
357 //ERGÄNZUNG: ändern eines Datensatzes
358 /* Mit dieser Callback-Methode reagieren wir auf das
359 Schließen der CAB und setzen den Zähler zurück auf den Wert 0*/
360 @Override
361 public void onDestroyActionMode(ActionMode mode) {
362
363     selCount = 0;
364 }
365

```

Callback-Methode onDestroyActionMode() überschreiben.

Implementieren Sie dazu den Quellcode auf gleiche Weise und ergänzen Sie die Kommentare, wie nebenstehend angezeigt.

Eingabehilfe:

```

//ERGÄNZUNG: ändern eines Datensatzes
/* Mit dieser Callback-Methode reagieren wir auf das
Schließen der CAB und setzen den Zähler zurück auf
den Wert 0*/
@Override
public void onDestroyActionMode(ActionMode mode) {
    selCount = 0;
}

```

```

334 @Override
335 public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
336     return false;
337 }

```

Callback-Methode onPrepareActionMode() überschreiben.

Diese Methode implementieren wir später. Wir benötigen Sie später zum Ändern von Datensätzen.

Ergänzen Sie vorerst den Quellcode und die Kommentare, wie nebenstehend angezeigt.

Eingabehilfe:

```

@Override
public boolean onPrepareActionMode(ActionMode mode,
Menu menu) {
    return false;
}

```

Damit haben wir die Löschoption fertig implementiert.

```

33 @Override
34 protected void onCreate(Bundle savedInstanceState) {
35     super.onCreate(savedInstanceState);
36     setContentView(R.layout.activity_main);
37
38     //Log-Meldung erzeugen und im Logcat ausgeben.
39     Log.d(LOG_TAG, "Das Datenquellen-Objekt wird angelegt.");
40     dieDatenquelle = new FriendsDataSource(this);
41
42     //Einfügen eines Testdatensatzes
43     // (nur nutzen falls keiner enthalten ist).
44     insertTestDataEntry();
45
46     //Listener und Ereignissteuerung für die Schaltfläche Add
47     activateAddButton();
48
49     //Löschen und ändern eines Datensatzes
50     initializeContextualActionBar();
51 }

```

Methodenaufruf des Kontext-Menüs.

Die gerade fertiggestellte Methode → initializeContextualActionBar()

müssen wir nur noch in der Methode → onCreate() aufrufen.

Wechseln Sie dazu in die Methode → onCreate() der Klasse MainActivity.

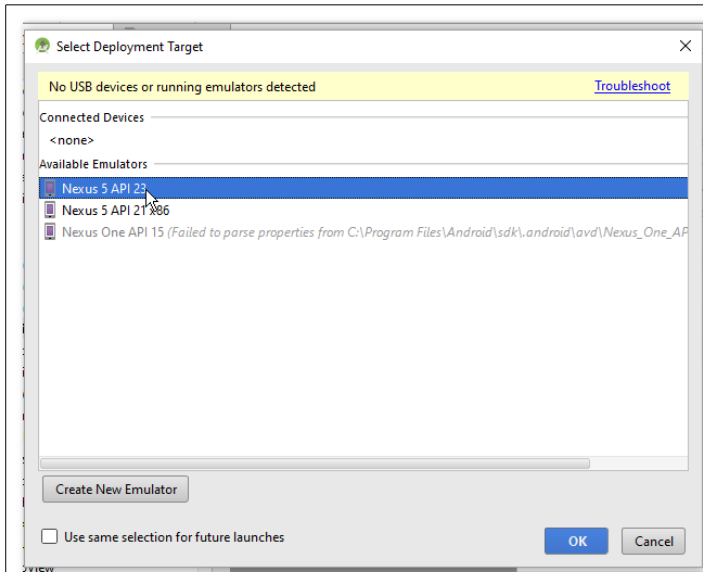
Ergänzen Sie abschließend den Quellcode und die Kommentare, wie nebenstehend angezeigt.

```

//Löschen und ändern eines Datensatzes
initializeContextualActionBar();

```

Speichern Sie alle Änderungen und testen Sie die Anwendung.

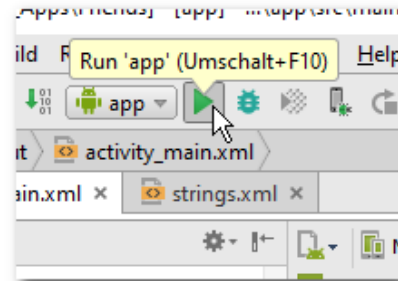


Create New Emulator:

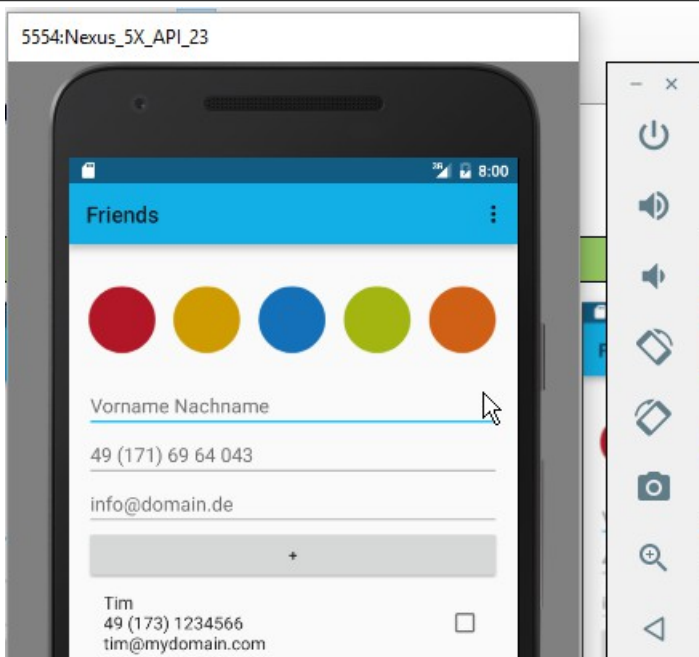
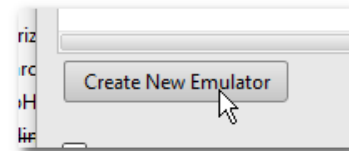
Für wenig leistungsfähige Rechner empfiehlt sich ein neues Gerät → Nexus One Device mit API 15 (SanwichIceCream) zu erzeugen:

Testen der Anwendung.

Wir starten nun den Emulator.
Emulator:



Der Emulator simuliert im vorliegenden Fall ein virtuelles Mobiltelefon vom Typ → Nexus 5 API 23.



Der Emulator öffnet sich.

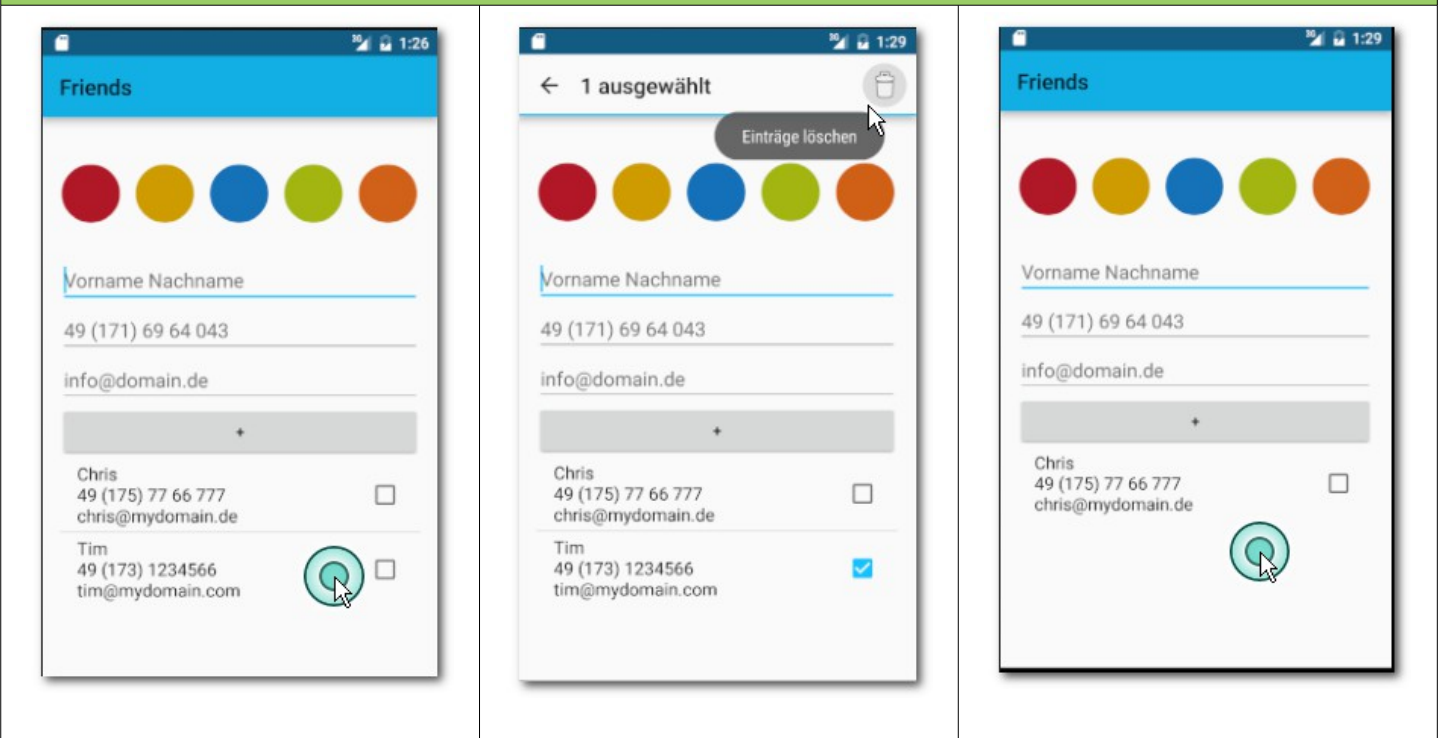
Beim ersten öffnen kann das einen Moment dauern.

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

Wenn Sie nicht ungeduldig werden, startet der Emulator die App nach Abschluss des Built-Prozesses von selbst.

Löschen Sie über die Benutzeroberfläche einen angezeigten Datensatz.

Löschen

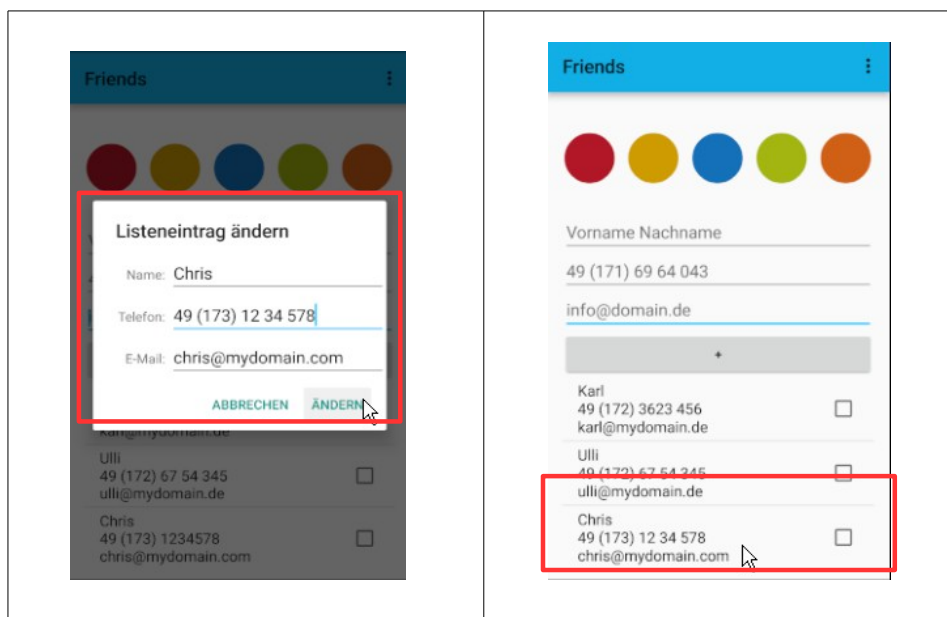
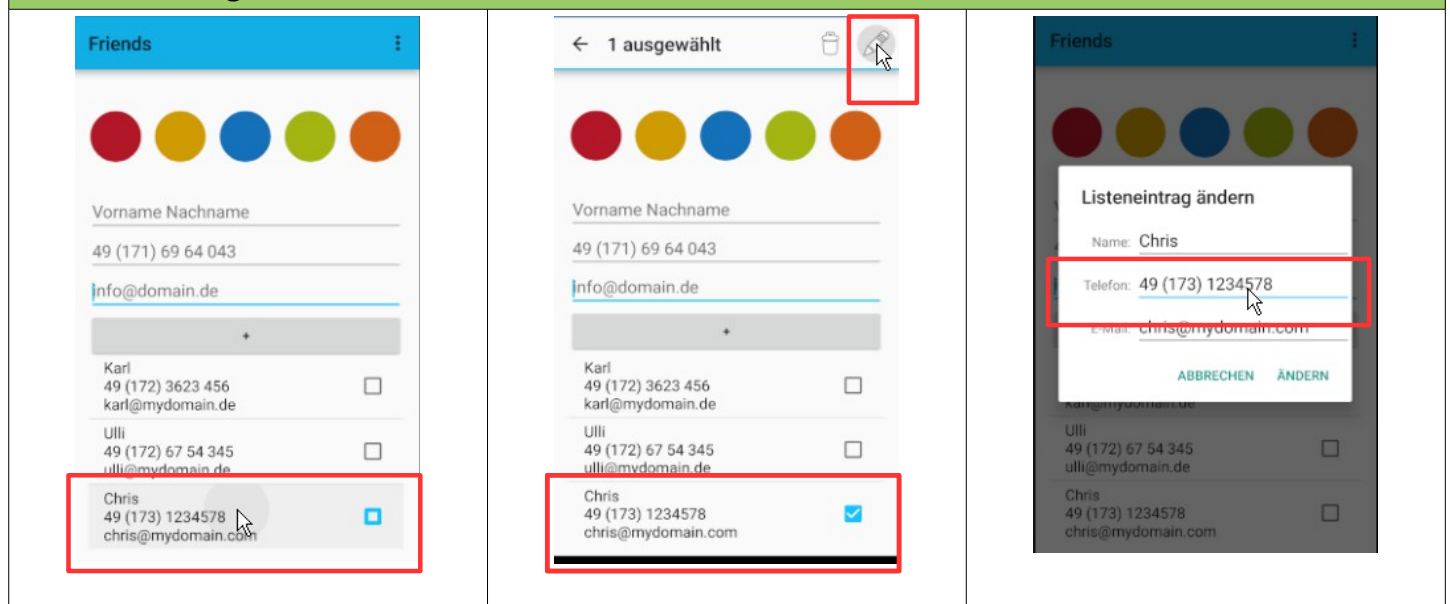


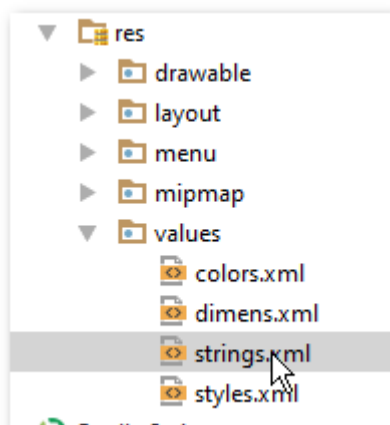
Gratulation!

2.6.4 Daten aktualisieren.

Für die Aktualisierung (Änderung) von Datensätzen in der Datenbank benötigen wir u.a. eine *Aktualisierungsoperation*. In unserem konkreten Fall ergänzen wir das gerade erstellte Menü um die Möglichkeit einen ausgewählten Datensatz zu bearbeiten und zu ändern. Der Menüpunkt zum ändern eines Datensatzes (Stift-Symbol) soll nur erscheinen wenn genau ein angezeigter Datensatz auf der Benutzeroberfläche ausgewählt wurde. Mit einem Klick auf das Stift-Symbol soll der ausgewählte Datensatz in einem Dialogfenster angezeigt werden. Der Benutzer kann die darin angezeigten Daten ändern und die Änderungen mit einem weiteren Klick auf die Schaltfläche → Ändern speichern. Die angezeigte Liste mit Daten wird daraufhin aktualisiert, sodass die Änderung gleich ersichtlich wird. Im folgenden werden wir hierfür das Menü, das Dialogfenster und die zugehörige Ereignissteuerung für die Änderungsoperation ergänzen und testen.

Aktualisierung





Android Project View

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <string name="app_name">Friends</string>
4      <string name="hello_world">Hello world!</string>
5
6      <string name="action_settings">Settings</string>
7      <string name="logo_description">Logo-Banner</string>
8      <string name="etName_hint_name">Vorname Nachname</string>
9      <string name="etPhone_hint_phone">49 (171) 69 64 043</string>
10     <string name="etEmail_hint_email">info@domain.de</string>
11     <string name="btAdd_zeichen">+</string>
12     <string name="etErrorMessage">Das Feld darf nicht leer sein.</string>
13
14     <!--Ergänzung löschen von Datensätzen -->
15     <string name="cab_checked_string">ausgewählt</string>
16     <string name="cab_delete">Einträge löschen</string>
17
18     <!--Ergänzung ändern von Datensätzen -->
19     <string name="cab_change">Eintrag ändern</string>
20
21     <!--Ergänzung Dialog ändern von Datensätzen -->
22     <string name="dialog_title">Listeneintrag ändern</string>
23     <string name="dialog_title_name">Name:</string>
24     <string name="dialog_title_phone">Telefon:</string>
25     <string name="dialog_title_email">E-Mail:</string>
26     <string name="dialog_button_positive">Ändern</string>
27     <string name="dialog_button_negative">Abbrechen</string>
28 </resources>

```

strings.xml

Bezeichner ergänzen.

Öffnen Sie dazu die Datei → strings.xml im Verzeichnis → app → res → values.

Wir erweitern die Datei → strings.xml um die Bezeichnungen, die wir für Änderungsoperationoperation benötigen werden.

Eingabehilfe:

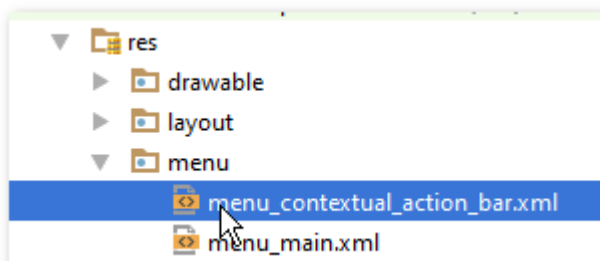
```

<!--Ergänzung ändern von Datensätzen -->
<string name="cab_change">Eintrag ändern</string>

<!--Ergänzung Dialog ändern von Datensätzen -->
<string name="dialog_title">
    Listeneintrag ändern</string>
<string name="dialog_title_name">Name:</string>
<string name="dialog_title_phone">Telefon:</string>
<string name="dialog_title_email">E-Mail:</string>
<string name="dialog_button_positive">Ändern</string>
<string name="dialog_button_negative">
    Abbrechen</string>

```

Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.



menu_contextual_action_bar.xml

Menüpunkt im Kontext-Menü ergänzen.

Öffnen Sie dazu die Datei → menu_contextual_action_bar.xml im Verzeichnis → app → res → menu.

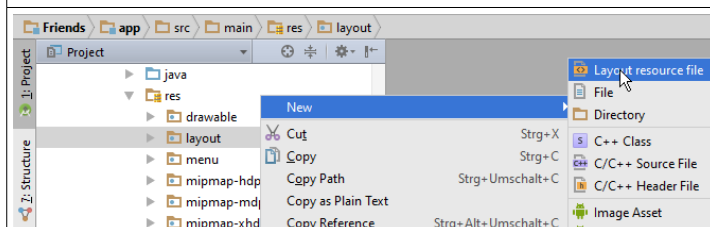
```

<!--Ergänzung ändern von Datensätzen -->
<item
    android:id="@+id/cab_change"
    android:icon="@android:drawable/ic_menu_edit"
    android:title="@string/cab_change" />

```

menu_contextual_action_bar.xml

Ergänzen Sie den Quellcode und die Kommentare, wie nebenstehend angezeigt.



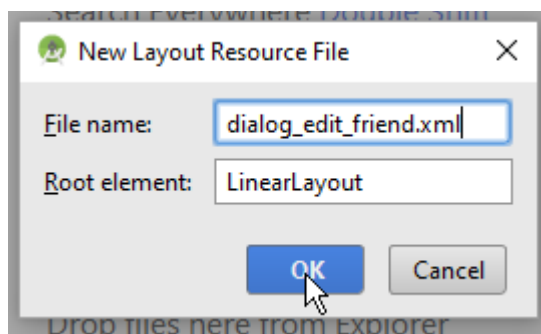
Kontext-Menü

Layout-Ressourcen-Datei erstellen.

Wir erstellen nun das Dialogfenster Layout. Dieses Fenster werden wir nutzen um Änderungen an einem zuvor ausgewählten Datensatz durchführen zu können.

Klicken Sie dazu mit der rechten Maustaste im Linken Frame auf das Verzeichnis.

→ res.



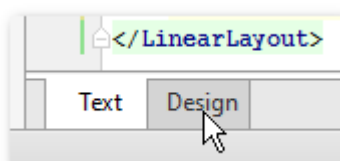
dialog_edit_friend.xml

Wählen Sie dazu im Kontext-Menü (rechte Maustaste) die Optionen.

→ New → Layout resource file

Übernehmen Sie die Angaben, wie nebenstehend angezeigt und klicken Sie auf die Schaltfläche.

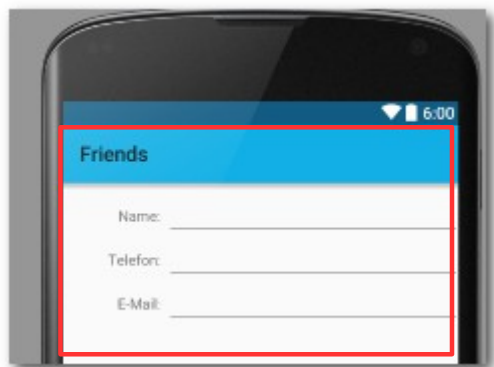
→ OK.



Dialogfenster mit dem Designer erstellen.

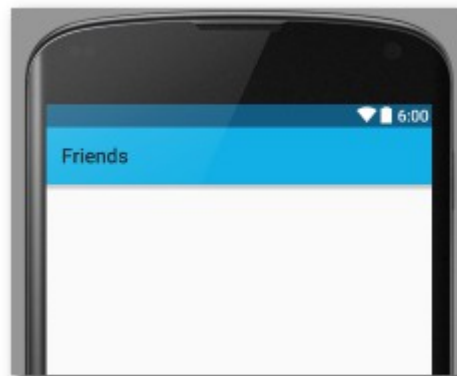
Klicken Sie dazu unterhalb der gerade erzeugten Datei auf den Reiter → Design. Wir werden den Oberflächendesigner verwenden, um das

Designer

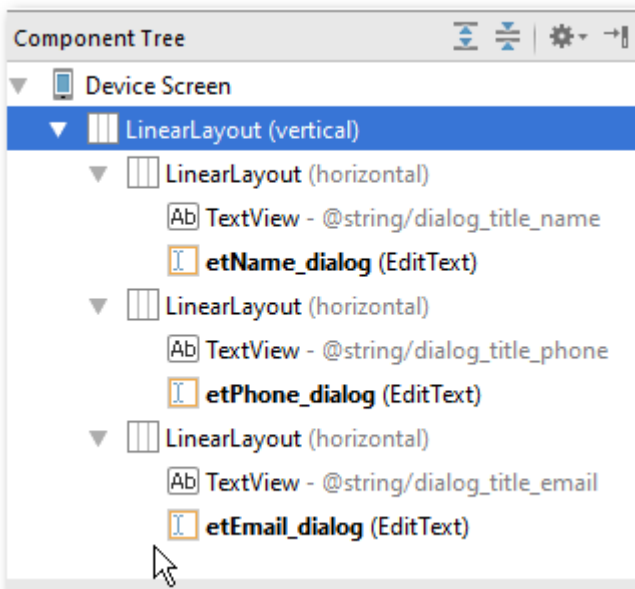


Dialogfenster-Layout (nachher)

Layout für das Dialogfenster zu erstellen.



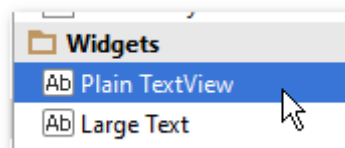
Aktueller Zustand des Dialogfensters



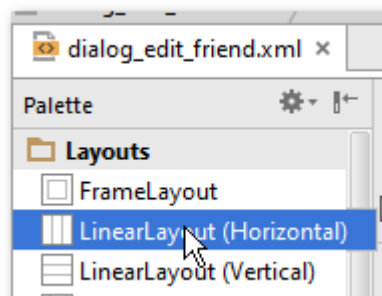
Gewünschtes Ergebnis im Fenster Component Tee

Dialogfenster-Layout erstellen.

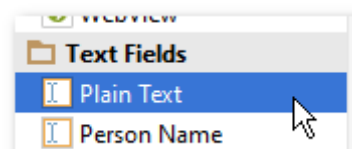
Ziehen Sie dazu, wie bereits eingangs für das Layout unserer Benutzeroberfläche (activity_main.xml) beschrieben die Komponenten aus dem Fenster → Palette in das Fenster → Component Tree.



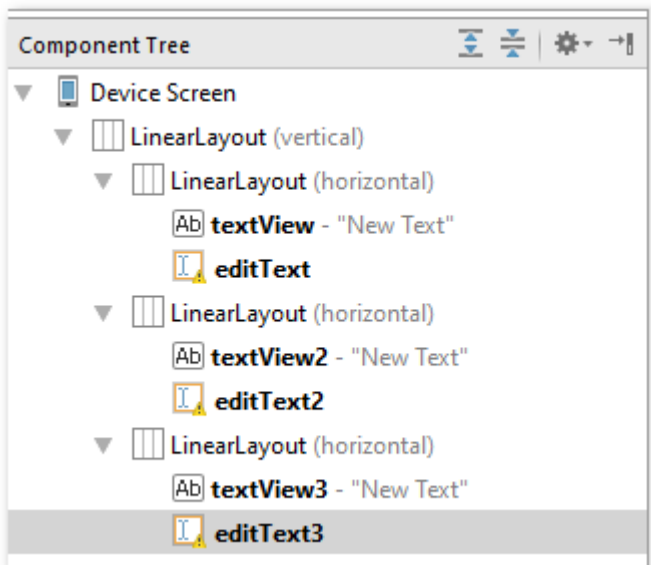
Fenster Palette: TextView-Komponente



Fenster Palette: Layout (Horizontal)-Komponente



Fenster Palette: EditText-Komponente



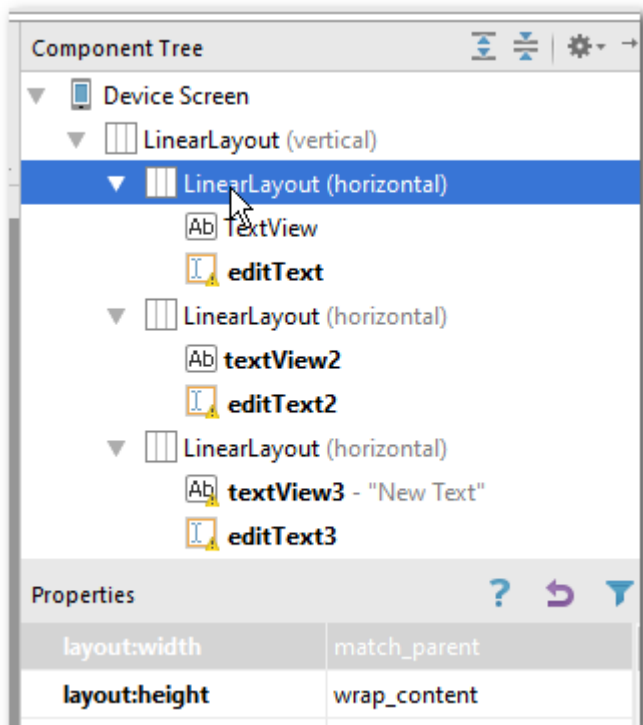
Platzieren der Komponenten

Vorgehensweise: Component Tree.

1. Layout (falls nötig) schachteln
2. Komponenten im Layout platzieren
3. Komponenteneigenschaften definieren

Nun folgen die Änderungen im aktuellen Komponenten-Baum, um das nebenstehende gewünschte Ergebnis zu erzeugen.

Ändern Sie anschließend die Eigenschaften für die einzelnen Komponenten im Fenster → Properties, wie in Folgendem Schritt für Schritt, ab.



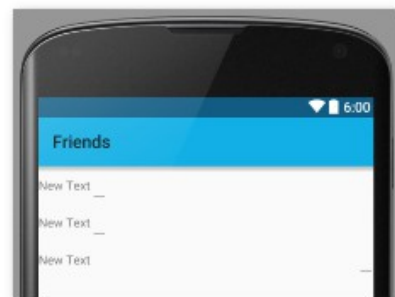
LinearLayouts (Horizontal)-Komponenten

Eigenschaft für die LinearLayout-Komponenten festlegen.

Klicken Sie jede Layout(Horizontal)-Komponente im Fenster → Component Tree nacheinander an und nutzen Sie dann die vertikale Bildlaufleiste im Fenster → Properties, um die Eigenschaft für die layout:height wie unten angezeigt, ändern zu können.

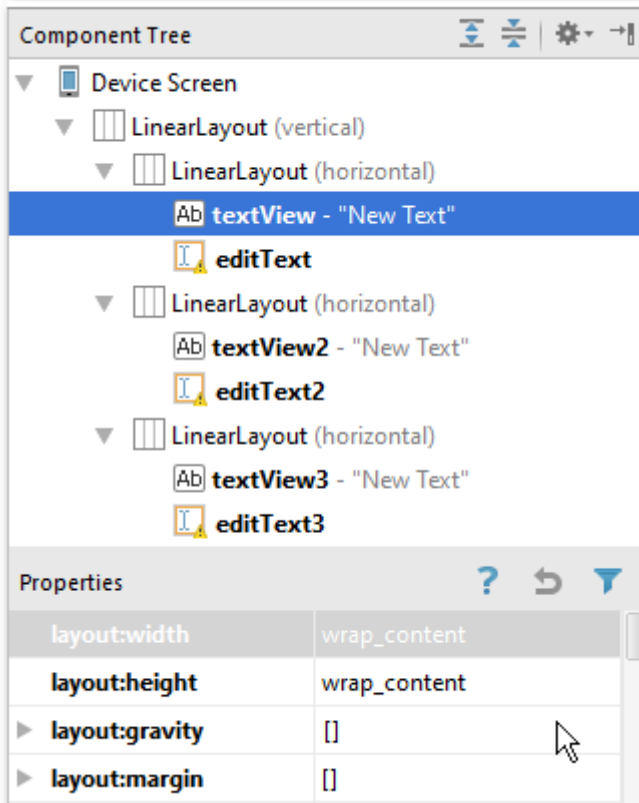
LinearLayout (Horizontal):

layout:height: wrap_content



Previewer

Bestätigen Sie jede Eingabe in den → Properties mit ENTER, damit der Quellcodegenerator den zugehörigen XML-Quellcode erzeugt.



TextViews-Komponenten

Zur Kontrolle:



Previewer

Eigenschaften für die Bezeichner-Komponenten festlegen.

Klicken Sie die TextView-Komponente im Fenster → Component Tree nacheinander an und nutzen Sie dann die vertikale Bildlaufleiste im Fenster → Properties, um die Eigenschaft für die layout:width, → gravity und → text, wie unten angezeigt, ändern zu können.

TextView (Plain TextView):
für den Bezeichner des Namens

```
layout-width: 0dp
layout-weight: 1
gravity: [end]
text: @string/dialog_title_name
```

TextView (Plain TextView):
für den Bezeichner des Telefonnummer

```
layout-width: 0dp
layout-weight: 1
gravity: [end]
text: @string/dialog_title_phone
```

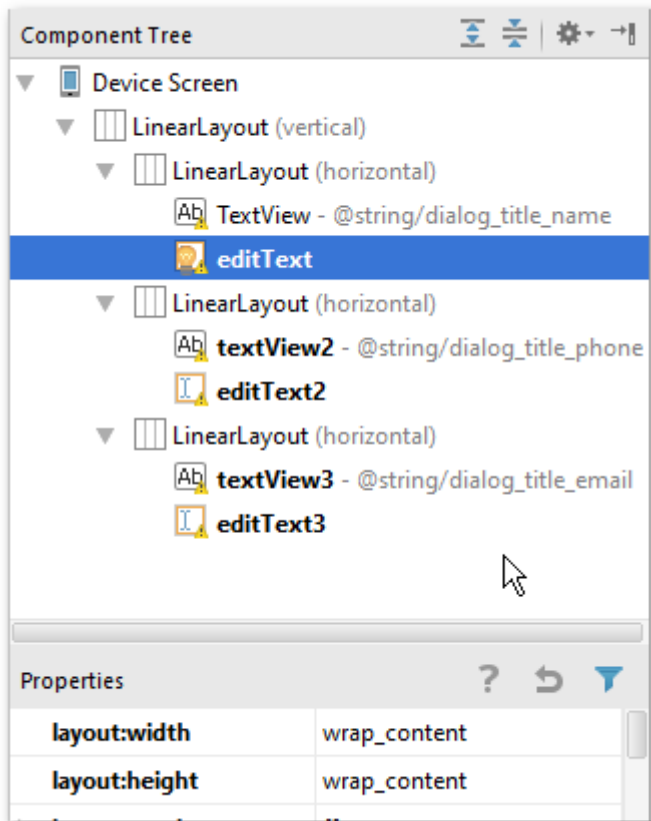
TextView (Plain TextView):
für den Bezeichner des E-Mail-Adresse

```
layout-width: 0dp
layout-weight: 1
gravity: [end]
text: @string/dialog_title_email
```

Bestätigen Sie jede Eingabe in den → Properties mit ENTER, damit der Quellcodegenerator den zugehörigen XML-Quellcode erzeugt.

Eigenschaften für die Eingabe-Komponenten festlegen.

Klicken Sie die EditText-Komponente im Fenster → Component Tree nacheinander an und nutzen Sie dann die vertikale Bildlaufleiste im Fenster → Properties, um die Eigenschaft für



EditTexts-Komponenten



Previewer

die layout:height, layout:weight, → id und → inputType, wie unten angezeigt, ändern zu können.

EditText (Plain Text):

für das Texteingabefeld des Name

```
layout-width: 0dp
layout-weight: 3
id: etName_dialog
inputType: [text]
```

EditText (Plain Text):

für das Texteingabefeld des Telefonnummer

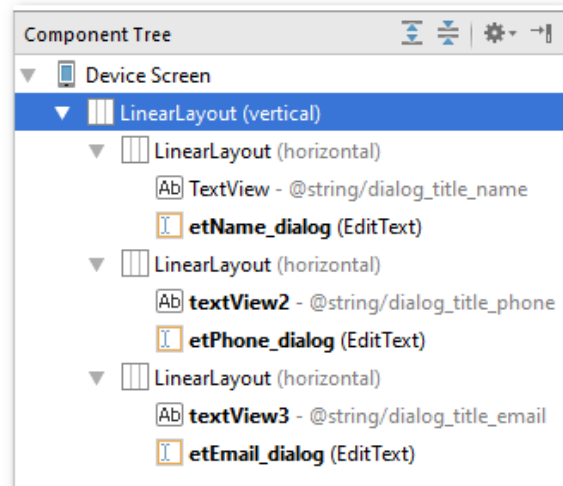
```
layout-width: 0dp
layout-weight: 3
id: etPhone_dialog
inputType: [phone]
```

EditText (Plain Text):

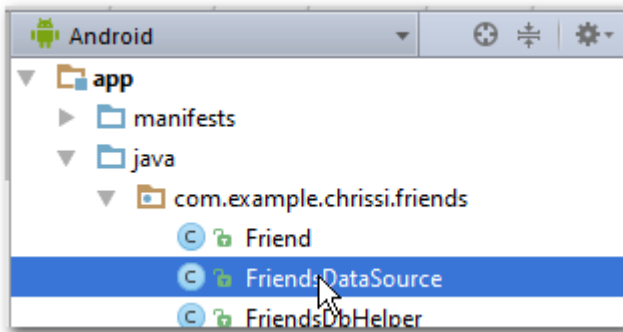
für das Texteingabefeld des E-Mail

```
layout-width: 0dp
layout-weight: 3
id: etEmail_dialog
inputType: [textEmailAddress]
```

Bestätigen Sie jede Eingabe in den → Properties mit ENTER, damit der Quellcodegenerator den zugehörigen XML-Quellcode erzeugt.



Fenster Component Tree



Android Project View

```

201  /*ERGÄNZUNG: ändern eines Datensatzes.*/
202  public Friend updateFriend(long id, String newName,
203                          String newPhone,
204                          String newEmail) {
205      /*ContentValues eignet sich zur Verwaltung von
206      Daten aus der Datenbank. Dazu wird ein solches
207      Listenobjekt erzeugt. Dann werden mit dem
208      Methodenaufruf put die Werte in die Liste aufgenommen.*/
209      ContentValues values = new ContentValues();
210
211      values.put(FriendsDbHelper.COLUMN_NAME, newName);
212      values.put(FriendsDbHelper.COLUMN_PHONE, newPhone);
213      values.put(FriendsDbHelper.COLUMN_EMAIL, newEmail);
214
215      /*Ausführen der Aktualisierungsabfrage für die Daten (values)
216      * durchführen.*/
217      database.update(FriendsDbHelper.TABLE_CONTACT_LIST,
218                    values,
219                    FriendsDbHelper.COLUMN_ID + "=" + id,
220                    null);
221
222      /*Das Cursorobjekt enthält das Resultset (Ergebnisliste) der Abfrage,
223      legt es quasi frei, sodass systematisch auf die Ergebniselemente
224      zugegriffen werden kann.*/
225      Cursor cursor = database.query(FriendsDbHelper.TABLE_CONTACT_LIST,
226                                  columns, FriendsDbHelper.COLUMN_ID + "=" + id,
227                                  null, null, null, null);
228
229      /*Ermittelt das erste Element der Ergebnisliste*/
230      cursor.moveToFirst();
231
232      /*Ermittelt die Eigenschaftswerte des Datensatzes,
233      erzeugt ein neues Freund-Objekt und gibt das Objekt zurück.*/
234      Friend derFreund = cursorToFriend(cursor);
235      cursor.close();
236
237      //Schließt die Ergebnisliste
238      return derFreund;
239  }

```

Änderungsoperation implementieren.

Diese Methode soll einen geänderten Datensatz in der Datenbank aktualisieren.

Identifiziert wird der geänderte Datensatz an seinem identifizierendem Attribut, der → `_id`

Wechseln Sie in die Klasse → `FriendsDataSource` und implementieren Sie darin die Methode → `updateFriend(Friend friend)`.

Eingabehilfe:

```

/*ERGÄNZUNG: ändern eines Datensatzes.*/
public Friend updateFriend(long id,
                          String newName,
                          String newPhone,
                          String newEmail) {

/*ContentValues eignet sich zur Verwaltung von Daten
aus der Datenbank. Dazu wird ein solches Listenobjekt
erzeugt. Mittels dem Methodenaufruf put die Werte in
die Liste aufgenommen.*/
    ContentValues values = new ContentValues();
    values.put(FriendsDbHelper.COLUMN_NAME, newName);
    values.put(FriendsDbHelper.COLUMN_PHONE,
              newPhone);
    values.put(FriendsDbHelper.COLUMN_EMAIL,
              newEmail);

/*Ausführen der Aktualisierungsabfrage für die Daten
(values) durchführen.*/
    database.update(
        FriendsDbHelper.TABLE_CONTACT_LIST,
        values,
        FriendsDbHelper.COLUMN_ID
        + "=" + id, null);

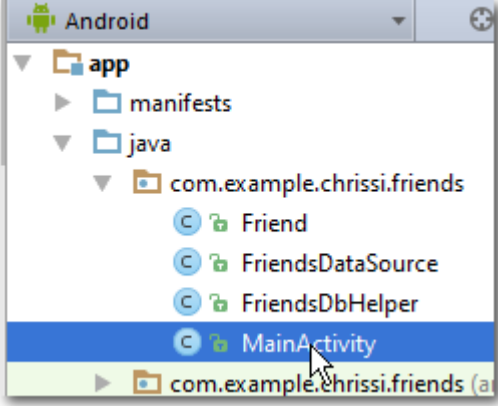
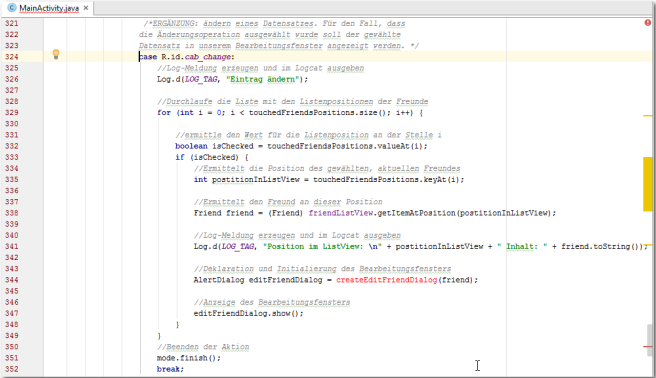
/*Das Cursorobjekt enthält das Resultset (Ergebnis-
liste) der Abfrage,legt es quasi frei, sodass
systematisch auf die Ergebniselemente zugegriffen
werden kann.*/
    Cursor cursor = database.query(
        FriendsDbHelper.TABLE_CONTACT_LIST,
        columns, FriendsDbHelper.COLUMN_ID
        + "=" + id, null, null, null, null);

/*Ermittelt das erste Element der Ergebnisliste*/
    cursor.moveToFirst();

/*Ermittelt die Eigenschaftswerte des Datensatzes,
erzeugt ein neues Freund-Objekt und gibt das Objekt
zurück.*/
    Friend derFreund = cursorToFriend(cursor);
    cursor.close();

//Schließt die Ergebnisliste
    return derFreund;
}

```

	<pre>} </pre> <p>Ergänzen Sie dazu den Quellcode und die Kommentare, wie nebenstehend angezeigt.</p>
 <p>Android Project View</p>	<p>Die Ereignissteuerung für die Änderungsoperation implementieren.</p> <p>Wechseln Sie dazu in die Klasse → MainActivity und erweitern Sie die Implementierung der Methode → initializeContextualActionBar().</p>
	<p><i>InitializeContextualActionBar()</i> erweitern.</p> <p>Für den Fall dass wir genau einen Datensatz ausgewählt haben, soll es möglich sein das Ereignis für die Änderung auszulösen. Deshalb müssen wir in der enthaltenen Listener Methode → onActionItemClicked() genau diesen Fall erweitern.</p> <p>Ermitteln Sie die beschriebene Stelle im Quellcode.</p> <p>Hinweis: Unterhalb der → break-Anweisung (Fall → cap_delete) → Methode onActionItemClicked().</p> <p>Die Methode → createEditFriendDialog() werden wir gleich noch als Hilfsmethode im unteren Teil der Klasse → MainActivity implementieren.</p>
<p>Eingabehilfe:</p> <pre>case R.id.cab_change: //Log-Meldung erzeugen und im Logcat ausgeben Log.d(LOG_TAG, "Eintrag ändern"); //Durchlaufe die Liste mit den Listenpositionen der Freunde for (int i = 0; i < touchedFriendsPositions.size(); i++) { //ermittle den Wert für die Listenposition an der</pre>	<p><i>Callback-Methode onActionItemClicked()</i> erweitern.</p> <p>Ergänzen Sie dazu den Quellcode und die Kommentare, wie nebenstehend angezeigt.</p> <p>Ergänzende Erläuterung: Das Containerobjekt → SparseBooleanArray nutzen wir, um uns die Positionen der ausgewählten Daten-</p>

```

Stelle i
boolean isChecked =
    touchedFriendsPositions.valueAt(i);
    if (isChecked) {
        //Ermittelt die Position des gewählten,
        //aktuellen Freundes

        int positionInListView
            = touchedFriendsPositions.keyAt(i);

        //Ermittelt den Freund an dieser Position
        Friend friend
            = (Friend)
            friendListView
            .getItemAtPosition(
                positionInListView);

        //Log-Meldung erzeugen und im Logcat ausgeben
        Log.d(LOG_TAG,
            "Position im ListView: \n"
            + positionInListView + " Inhalt: "
            + friend.toString());

        //Deklaration und Initialisierung
        //des Bearbeitungsfensters
        AlertDialog editFriendDialog
            = createEditFriendDialog(friend);

        //Anzeige des Bearbeitungsfensters
        editFriendDialog.show();
    }
}
//Beenden der Aktion
mode.finish();
break;

```

sätze aufzunehmen. Das Objekt ähnelt im Umgang einer HashMap.

Wir können mit Hilfe der Methoden → `keyAt(int i)` und → `valueAt(int i)` die Elemente des Containers durchlaufen. Für den Fall, dass wir über die Schlüssel (keys) mit aufsteigenden Werten (int i) iterieren wird der Schlüsselwert in aufsteigender Reihenfolge ermittelt und zurückgegeben. Anderenfalls wird der zugehörigen Wert (values) selbst, ebenfalls in aufsteigender Reihenfolge ermittelt und zurückgegeben.

public class **SparseBooleanArray** Summary: Ctors | Methods | Inherited Methods | [Expand All] Added in API level 1

extends `Object`
implements `Cloneable`

java.lang.Object
↳ android.util.SparseBooleanArray

^ Class Overview

SparseBooleanArrays map integers to booleans. Unlike a normal array of booleans there can be gaps in the indices. It is intended to be more memory efficient than using a HashMap to map Integers to Booleans, both because it avoids auto-boxing keys and values and its data structure doesn't rely on an extra entry object for each mapping.

Note that this container keeps its mappings in an array data structure, using a binary search to find keys. The implementation is not intended to be appropriate for data structures that may contain large numbers of items. It is generally slower than a traditional HashMap, since lookups require a binary search and adds and removes require inserting and deleting entries in the array. For containers holding up to hundreds of items, the performance difference is not significant, less than 50%.

It is possible to iterate over the items in this container using `keyAt(int)` and `valueAt(int)`. Iterating over the keys using `keyAt(int)` with ascending values of the index will return the keys in ascending order, or the values corresponding to the keys in ascending order in the case of `valueAt(int)`.

API Klasse SparseBooleanArray

```

369 //ÄNDERN VON DATEN
370 //Mit dieser Callback-Methode reagieren wir auf den
371 Methoden-Aufruf invalidate().Für den Fall, dass mehr als nur
372 1 Eintrag ausgewählt wurde lassen wir das Edit-Symbol verschwinden.*/
373 @Override
374 public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
375     MenuItem item = menu.findItem(R.id.cab_change);
376     if (selCount == 1) {
377         item.setVisible(true);
378     } else {
379         item.setVisible(false);
380     }
381     return true;
382 }
383

```

Callback-Methode `onPrepareActionMode()` implementieren.

Zum jetzigen Zeitpunkt haben wir die Methode zwar deklariert, aber noch nicht implementiert. Diese Callback-Methode soll aber künftig sicherstellen, dass die Bearbeitung nur dann möglich ist, wenn genau ein Datensatz ausgewählt wurde.

Eingabehilfe:

```

/*Mit dieser Callback-Methode reagieren wir auf den
Methoden-Aufruf invalidate().Für den Fall, dass mehr
als nur 1 Eintrag ausgewählt wurde lassen wir das
Edit-Symbol verschwinden.*/
@Override
public boolean onPrepareActionMode(ActionMode mode,
Menu menu) {
    MenuItem item = menu.findItem(R.id.cab_change);
    if (selCount == 1) {
        item.setVisible(true);
    }
}

```

Suchen Sie die Methodendeklaration! Ändern Sie dazu den Quellcode ab und ergänzen Sie die Kommentare, wie nebenstehend angezeigt.

```

} else {
    item.setVisible(false);
}
return true;
}

```

Ereignissteuerung des Dialogfensters.

Ergänzen Sie dazu den Quellcode und die Kommentare, wie nebenstehend angezeigt.

Fortsetzung:

```

//Titel im Bearbeitungsfenster anzeigen
builder.setView(dialogsView);
.setTitle(R.string.dialog_title);

//Ereignissteuerung für die Schaltfläche Eintrag ändern im Bearbeitungsfenster

.setPositiveButton(R.string.dialog_button_positive,
new DialogInterface.OnClickListener() {
    @Override
    public void onClick(
        DialogInterface dialog, int id) {
        //Lesen der geänderten Daten
        String nameString
            = etName_dialog.getText().toString();
        String phoneString
            = etPhone_dialog.getText().toString();
        String emailString
            = etEmail_dialog.getText().toString();

        //Prüfung ob Eingaben in den
        //Texteingabefeldern fehlen
        if ((TextUtils.isEmpty(nameString)
            || TextUtils.isEmpty(phoneString)
            || TextUtils.isEmpty(emailString))) {
            Log.d(LOG_TAG, "Ein Eintrag enthielt keinen Text. Daher Abbruch der Änderung.");
            return;
        }

        // An dieser Stelle schreiben wir die geänderten Daten
        // in die SQLite Datenbank
        Friend updatedFriend
            = dieDatenquelle.updateFriend(friend.getId(),
            nameString, phoneString, emailString);

        //Log-Meldung erzeugen und im Logcat ausgeben
        Log.d(LOG_TAG, "Alter Eintrag - ID: "
            + friend.getId() + " Inhalt: "
            + friend.toString());
        Log.d(LOG_TAG,
            "Neuer Eintrag - ID: "
            + updatedFriend.getId());
    }
});

```

```

389 //ERGÄNZUNG: ändern eines Datensatzes. Wir erzeugen ein Bearbeitungsfenster,
390 über nehmen die die Werte für name,phone und email des aktuellen Freundes
391 in die Texteingabefelder. Für den Fall, dass der Benutzer die Daten ändert
392 und auf die Schaltfläche Eintrag ändern klickt sollen die geänderten Daten
393 in der Datenbank gespeichert werden. Klickt der Benutzer anderenfalls
394 auf die Schaltfläche Abbrechen wird der Vorgang abgebrochen und das
395 Bearbeitungsfenster geschlossen */
396 private AlertDialog createEditFriendDialog(final Friend friend) {
397
398     //Erzeugung eines Dialogfensters(Bearbeitungsfenster)
399     AlertDialog.Builder builder = new AlertDialog.Builder(this);
400
401     //Den LayoutInflater (Befüller) initialisieren
402     LayoutInflater inflater = getLayoutInflater();
403
404     //Initialisiert (Befüllt) die View des Dialogfensters mit dem XML-Layout
405     View dialogsView = inflater.inflate(R.layout.dialog_edit_friend, null);
406
407     /*Schreibt die Werte für den ausgewählten Freund (name, phone und email)
408     in die Texteingabefelder im Bearbeitungsfenster*/
409     final EditText etName_dialog = (EditText) dialogsView.findViewById(R.id.etName_dialog);
410     etName_dialog.setText(friend.getName());
411
412     final EditText etPhone_dialog = (EditText) dialogsView.findViewById(R.id.etPhone_dialog);
413     etPhone_dialog.setText(friend.getPhone());
414
415     final EditText etEmail_dialog = (EditText) dialogsView.findViewById(R.id.etEmail_dialog);
416     etEmail_dialog.setText(friend.getEmail());
417

```

```

417
418     builder.setView(dialogsView)
419     //Titel im Bearbeitungsfenster anzeigen
420     .setTitle(R.string.dialog_title)
421
422     //Ereignissteuerung für die Schaltfläche Eintrag ändern im Bearbeitungsfenster
423     .setPositiveButton(R.string.dialog_button_positive, (dialog, id) -> {
424         //Lesen der geänderten Daten
425         String nameString = etName_dialog.getText().toString();
426         String phoneString = etPhone_dialog.getText().toString();
427         String emailString = etEmail_dialog.getText().toString();
428
429         //Erüfung ob Eingaben in den Texteingabefeldern fehlen
430         if ((TextUtils.isEmpty(nameString) || TextUtils.isEmpty(phoneString)
431             || TextUtils.isEmpty(emailString))) {
432             Log.d(LOG_TAG, "Ein Eintrag enthielt keinen Text. Daher Abbruch der Änderung.");
433             return;
434         }
435
436

```

```

438     // An dieser Stelle schreiben wir die geänderten Daten in die SQLite Datenbank
439     Friend updatedFriend = dieDatenquelle.updateFriend(friend.getId(),
440     nameString, phoneString, emailString);
441
442     //Log-Meldung erzeugen und im Logcat ausgeben
443     Log.d(LOG_TAG, "Alter Eintrag - ID: " + friend.getId() + " Inhalt: "
444     + friend.toString());
445     Log.d(LOG_TAG, "Neuer Eintrag - ID: " + updatedFriend.getId() + " Inhalt: "
446     + updatedFriend.toString());
447
448     //Anzeige der geänderten Freundesliste
449     showWhilListEntries();
450
451     //Schließen des Bearbeitungsfensters
452     dialog.dismiss();
453
454     //Ereignissteuerung für die Schaltfläche Abbrechen im Bearbeitungsfenster
455     .setNegativeButton(R.string.dialog_button_negative, (dialog, id) -> {
456         //Abbruch des Vorgangs und Schließen des Bearbeitungsfensters
457         dialog.cancel();
458     });
459
460
461     //
462     return builder.create();
463
464
465

```

Eingabehilfe:

*/*ERGÄNZUNG: ändern eines Datensatzes. Wir erzeugen ein Bearbeitungsfenster,über nehmen die Werte für name,phone und email des ausgewählten aktuellen Freundes in die Texteingabefelder. Für den Fall, dass der Benutzer die Daten ändert und auf die Schaltfläche Eintrag ändern klickt sollen die geänderten Daten in*

```

der Datenbank gespeichert werden. Klickt der Benutzer
anderenfalls auf die Schaltfläche Abbrechen wird der
Vorgang abgebrochen und das Bearbeitungsfenster ge-
schlossen */
private AlertDialog createEditFriendDialog(
    final Friend friend) {
//Erzeugung eines Dialogfensters(Bearbeitungsfenster)

    AlertDialog.Builder builder
    = new AlertDialog.Builder(this);

//Den LayoutInflater (Befüller) initialisieren
    LayoutInflater inflater
    = getLayoutInflater();

//Initialisiert (befüllt) die View des Dialogfensters
mit dem XML-Layout
    View dialogsView
    = inflater.inflate(
        R.layout.dialog_edit_friend, null);

/*Schreibt die Werte für den ausgewählten Freund
(name, phone und email)in die Texteingabefelder im
Bearbeitungsfenster*/
    final EditText etName_dialog
    = (EditText) dialogsView
        .findViewById(R.id.etName_dialog);
    etName_dialog.setText(friend.getName());

    final EditText etPhone_dialog
    = (EditText) dialogsView
        .findViewById(R.id.etPhone_dialog);
    etPhone_dialog.setText(friend.getPhone());

    final EditText etEmail_dialog
    = (EditText) dialogsView
        .findViewById(R.id.etEmail_dialog);
    etEmail_dialog.setText(friend.getEmail());

```

```

+ " Inhalt: "
+ updatedFriend.toString());

//Anzeige der geänderten Freundesliste
showAllListEntries();

//Schließen des Bearbeitungsfensters
dialog.dismiss();
    }
}

//Ereignissteuerung für die Schaltfläche Abbrechen im
Bearbeitungsfenster
        .setNegativeButton(
            R.string.dialog_button_negative,
            new DialogInterface.OnClickListener() {
                public void onClick(
                    DialogInterface dialog, int id) {

//Abbruch des Vorgangs und Schließen des Bearbei-
tungsfensters
                    dialog.cancel();
                }
            });

    return builder.create();
}

```

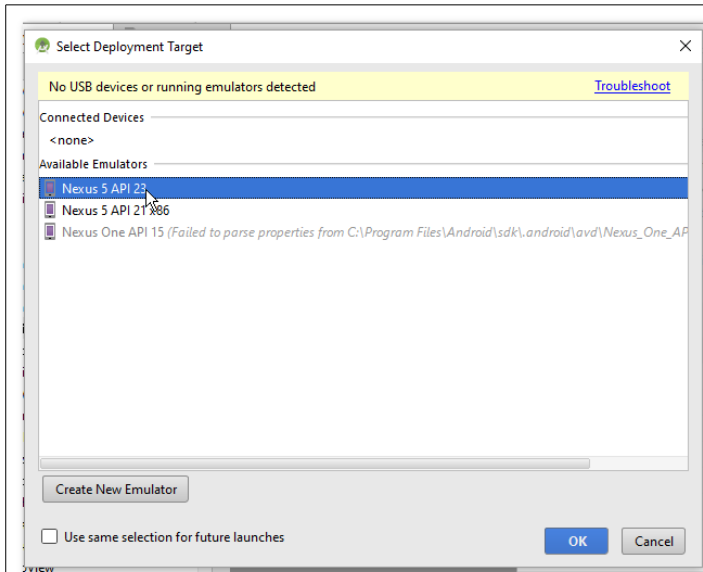
```

367
368 @Override
369 public void onDestroyActionMode(ActionMode mode) {
370     selCount = 0;
}

```

Callback-Methode `onDestroyActionMode()` implementieren.

Um sicherzustellen, dass nach der Bearbeitung die Lösch- und Änderungsoperationen wieder verfügbar sind, setzen wir mittels der Methode `onDestroyActionMode()` die Zählervariable auf 0.

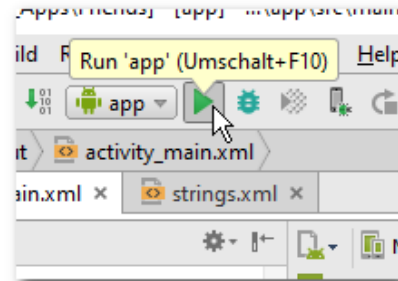


Create New Emulator:

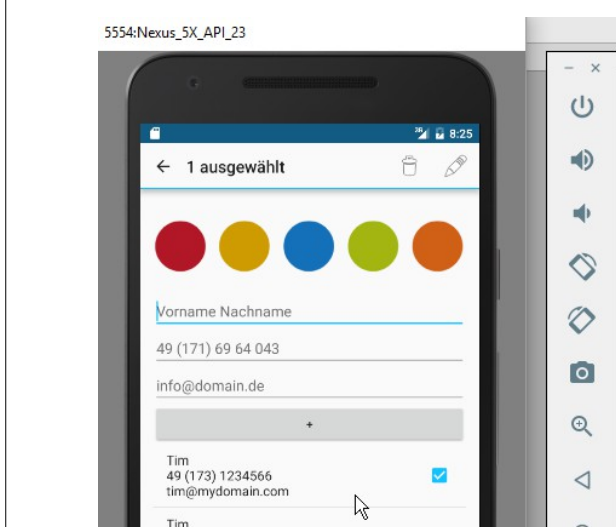
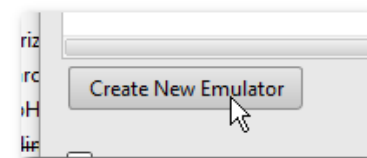
Für wenig leistungsfähige Rechner empfiehlt sich ein neues Gerät → Nexus One Device mit API 15 (SanwichIceCream) zu erzeugen:

Testen der Anwendung.

Wir starten nun den Emulator.
Emulator:



Der Emulator simuliert im vorliegenden Fall ein virtuelles Mobiltelefon vom Typ → Nexus 5 API 23.



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.

Wenn Sie nicht ungeduldig werden, startet der Emulator die App nach Abschluss des Built-Prozesses von selbst.

Bearbeiten Sie über die Benutzeroberfläche einen angezeigten Datensatz.

Aktualisierung

The sequence of screenshots illustrates the following steps:

- Initial List:** A list of contacts is shown. The contact 'Tim' (49 (173) 1234566, tim@mydomain.com) is selected, indicated by a red box and a mouse cursor.
- Select and Edit:** A red box highlights the '1 ausgewählt' header and the 'Eintrag ändern...' button. A mouse cursor clicks the edit button.
- Edit Dialog:** A dialog titled 'Listeneintrag ändern' is shown. The name 'Karl' is entered in the 'Name' field. The 'Telefon' and 'E-Mail' fields contain '49 (173) 1234566' and 'karl@mydomain.com' respectively. A red box highlights the input fields.
- Save Changes:** The 'ÄNDERN' button is clicked to save the changes. A red box highlights the 'ÄNDERN' button.
- Final List:** The contact list is updated, showing 'Karl' (49 (173) 1234566, karl@mydomain.com) instead of Tim. A red box highlights the updated contact entry.

Gratulation!