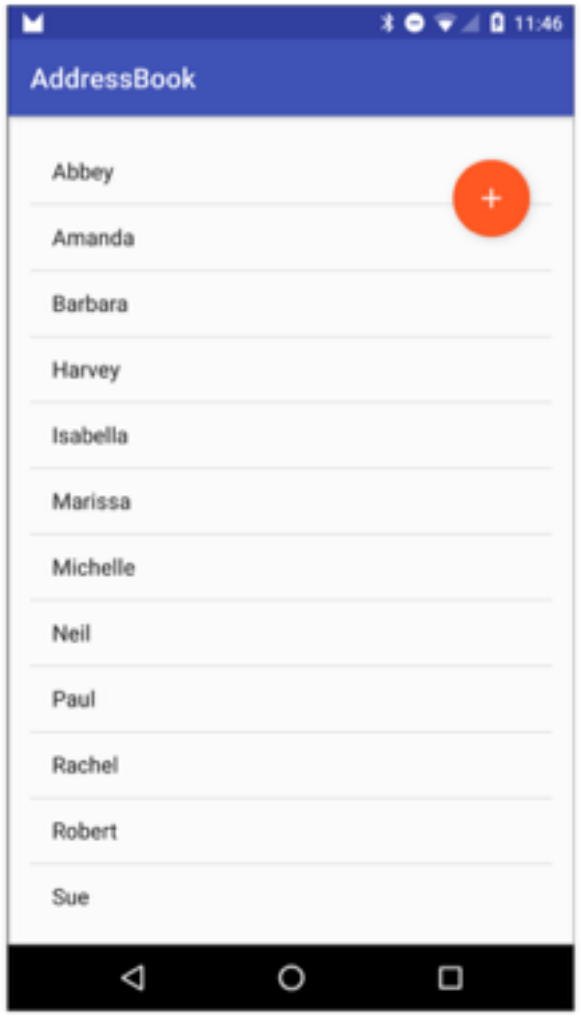
AddressBook

FragmentTransactions and the Fragment Back Stack, SQLite, SQLiteDatabase, SQLiteOpenHelper, ContentProvider, ContentResolver, Loader, LoaderManager, Cursor and GUI Styles

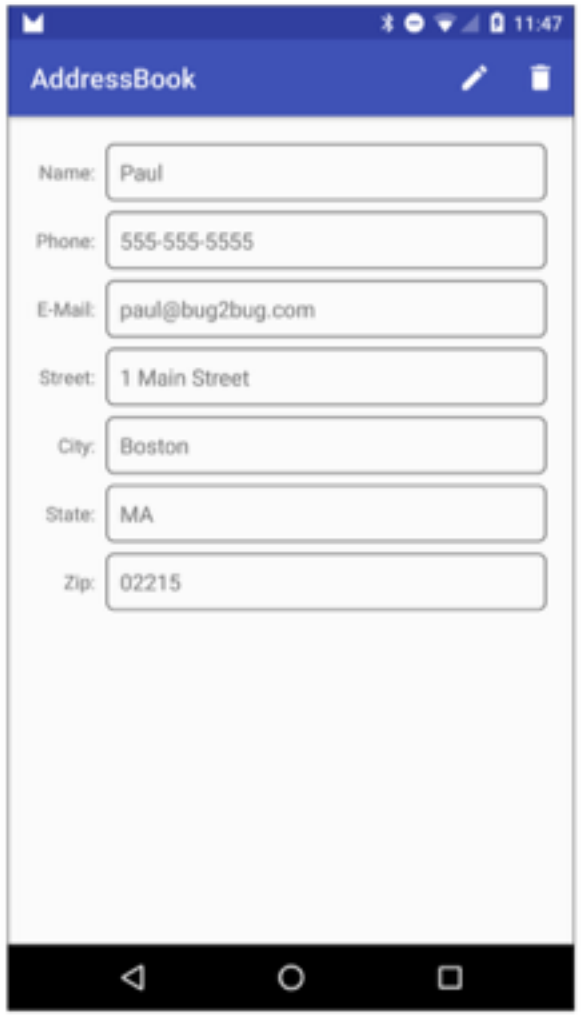
Fragments



11:46

AddressBook

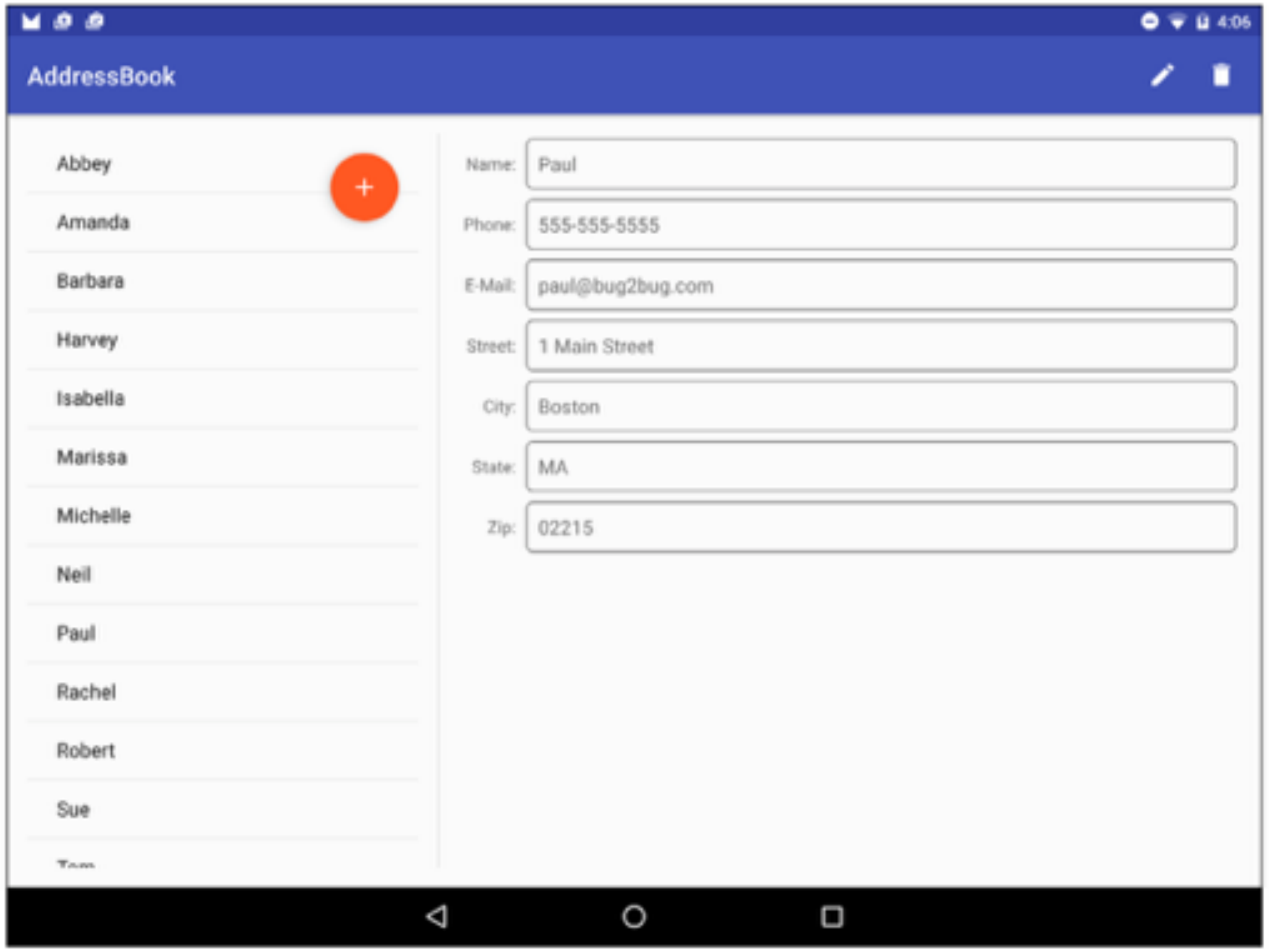
Abbey
Amanda
Barbara
Harvey
Isabella
Marissa
Michelle
Neil
Paul
Rachel
Robert
Sue



11:47

AddressBook

Name: Paul
Phone: 555-555-5555
E-Mail: paul@bug2bug.com
Street: 1 Main Street
City: Boston
State: MA
Zip: 02215

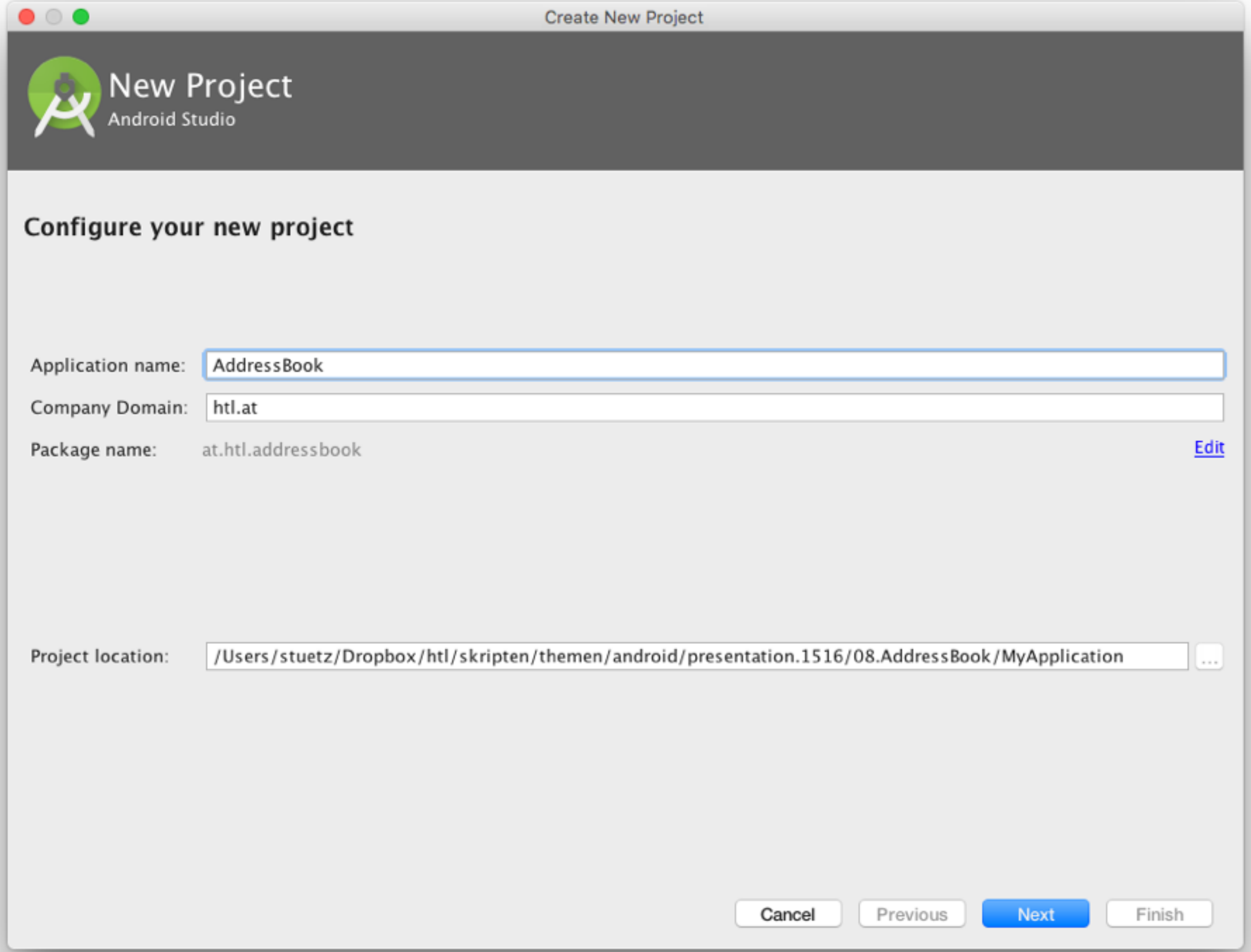


4:06

AddressBook

Abbey
Amanda
Barbara
Harvey
Isabella
Marissa
Michelle
Neil
Paul
Rachel
Robert
Sue
Tom

Name: Paul
Phone: 555-555-5555
E-Mail: paul@bug2bug.com
Street: 1 Main Street
City: Boston
State: MA
Zip: 02215



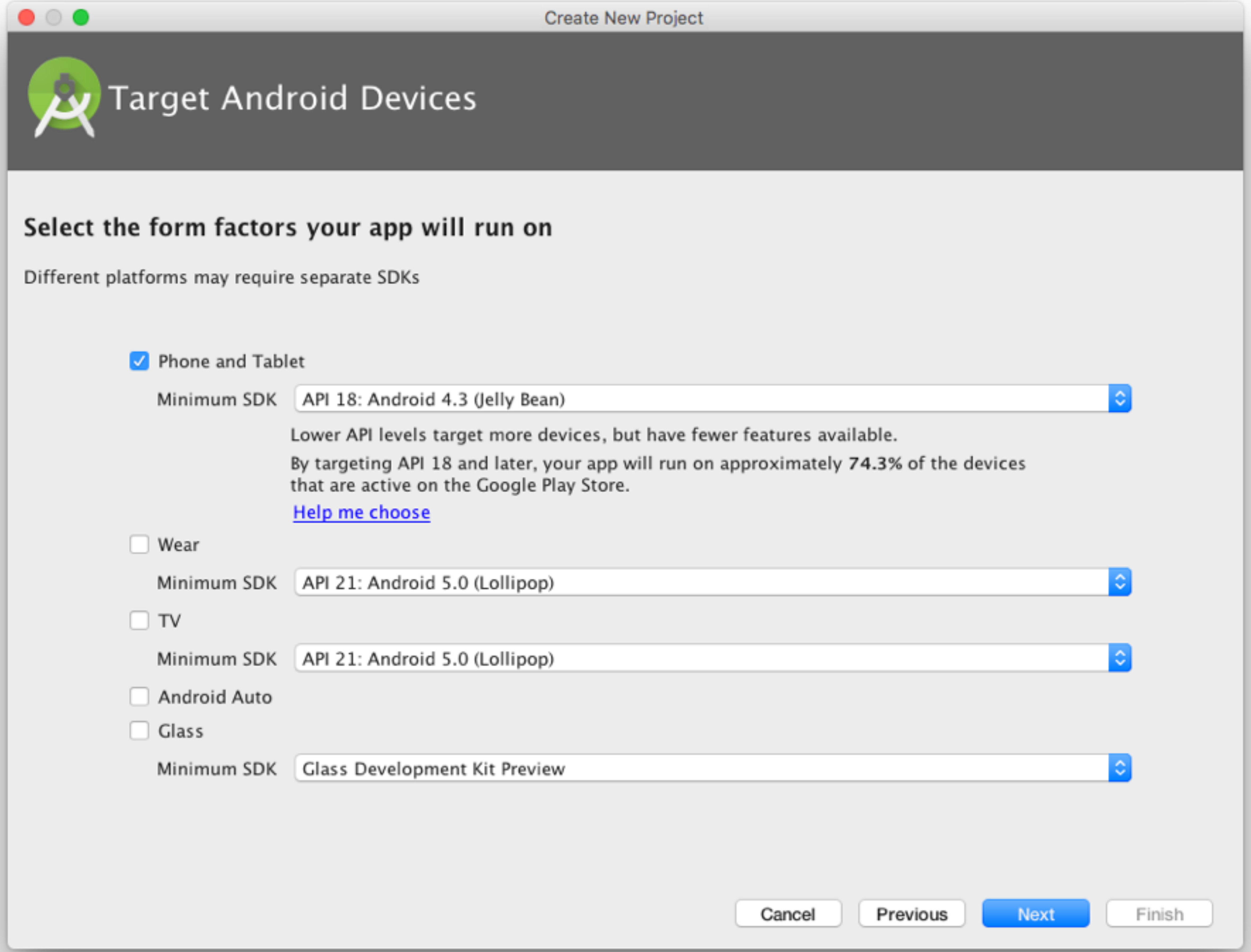
Configure your new project

Application name: AddressBook

Company Domain: htl.at

Package name: at.htl.addressbook

Project location: /Users/stuetz/Dropbox/htl/skripten/themen/android/presentation.1516/08.AddressBook/MyApplication



Create New Project

Target Android Devices

Select the form factors your app will run on

Different platforms may require separate SDKs

☑ Phone and Tablet

Minimum SDK: API 18: Android 4.3 (Jelly Bean)

Lower API levels target more devices, but have fewer features available.

By targeting API 18 and later, your app will run on approximately **74.3%** of the devices that are active on the Google Play Store.

Help me choose

□ Wear

Minimum SDK: API 21: Android 5.0 (Lollipop)

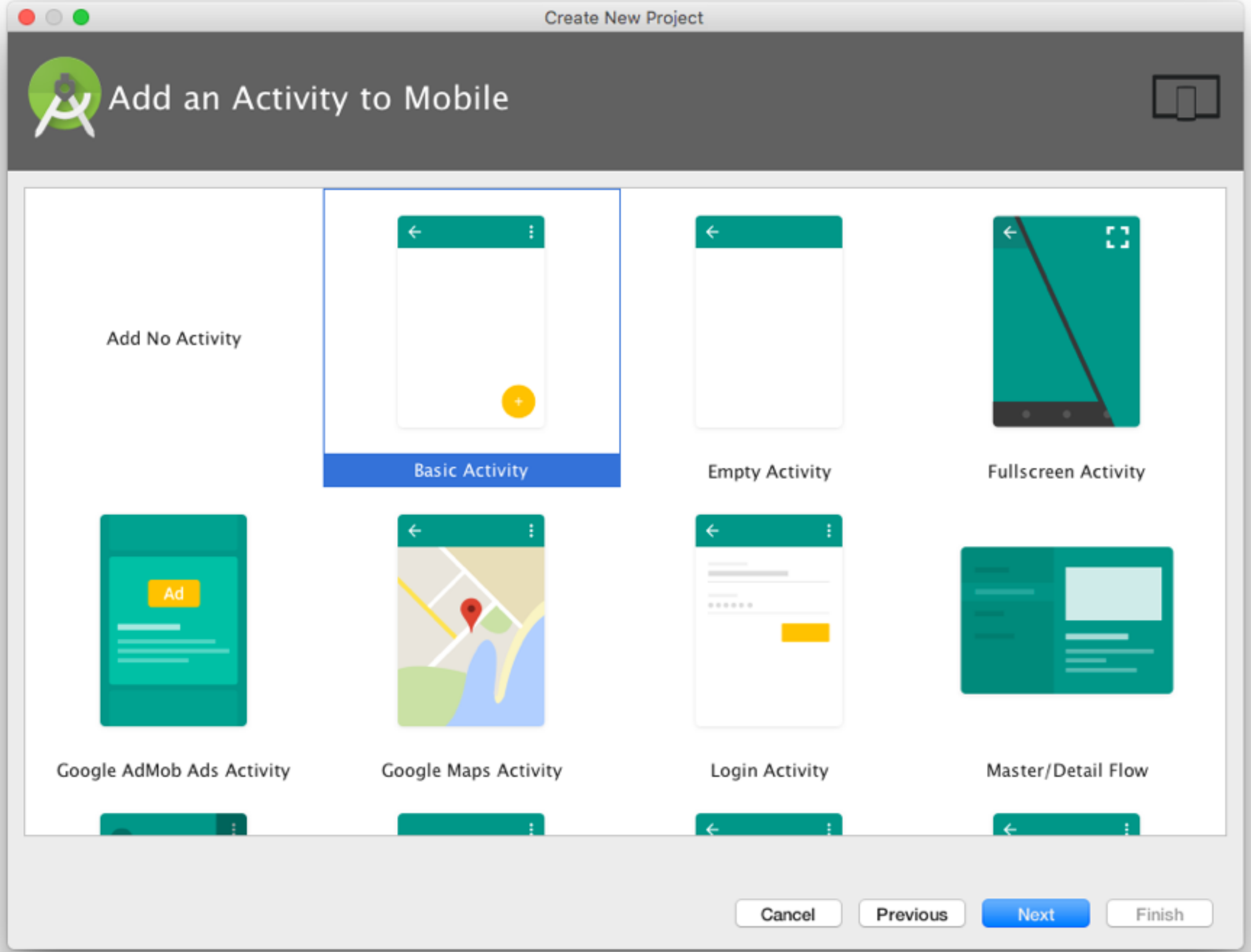
□ TV

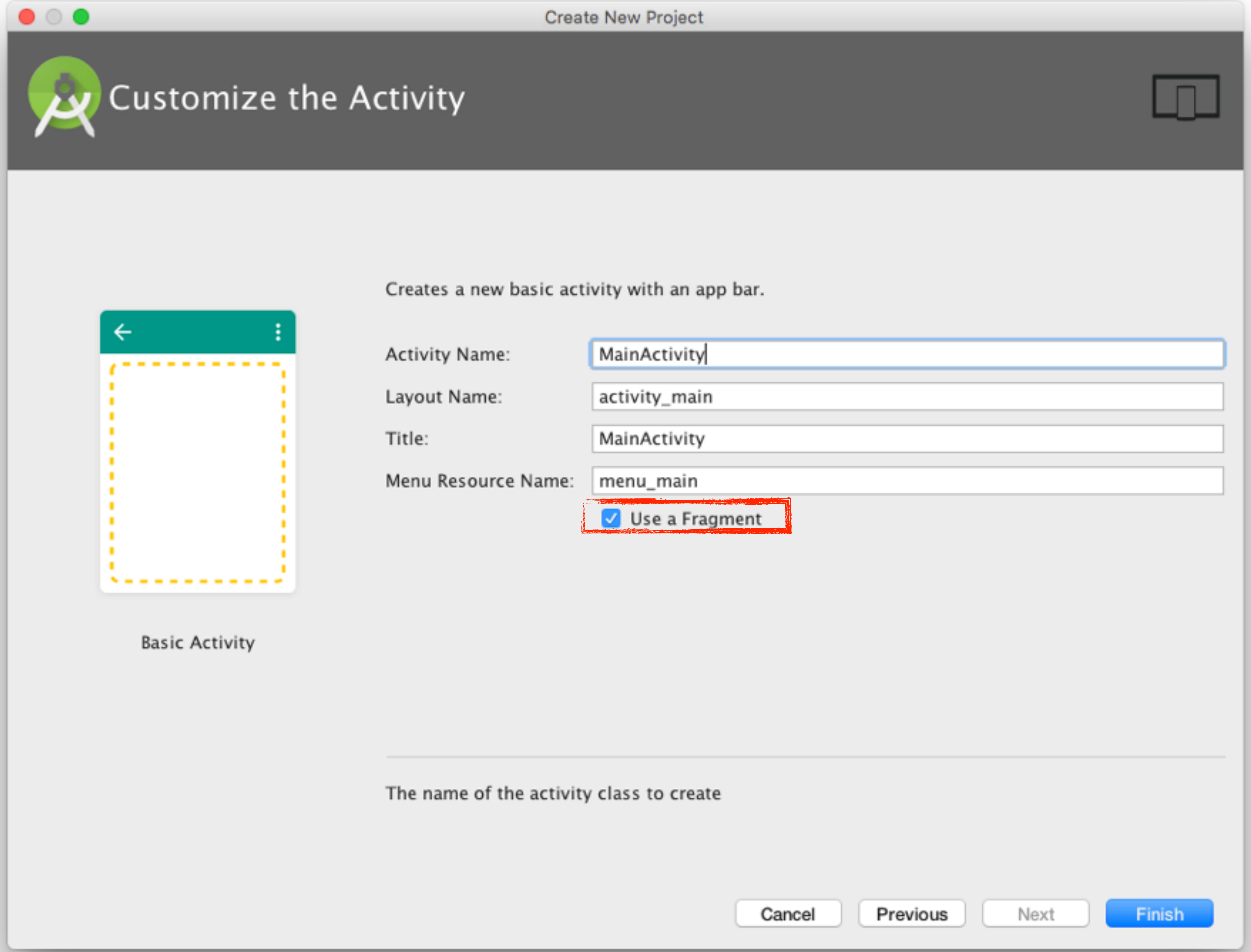
Minimum SDK: API 21: Android 5.0 (Lollipop)

□ Android Auto

□ Glass

Minimum SDK: Glass Development Kit Preview





Create New Project

Customize the Activity

Creates a new basic activity with an app bar.

Activity Name: MainActivity

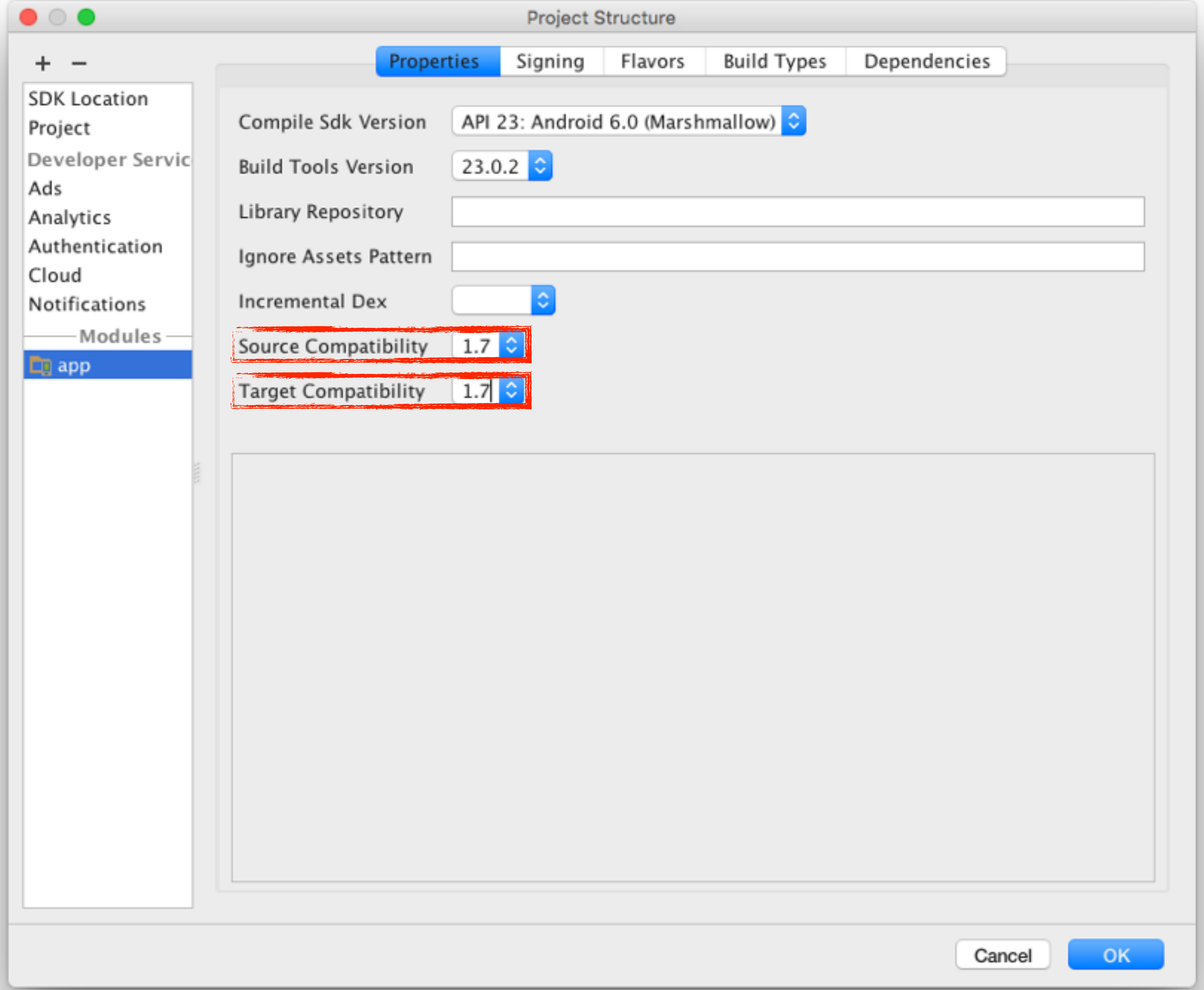
Layout Name: activity_main

Title: MainActivity

Menu Resource Name: menu_main

☑ Use a Fragment

The name of the activity class to create



Compile Sdk Version: API 23: Android 6.0 (Marshmallow)

Build Tools Version: 23.0.2

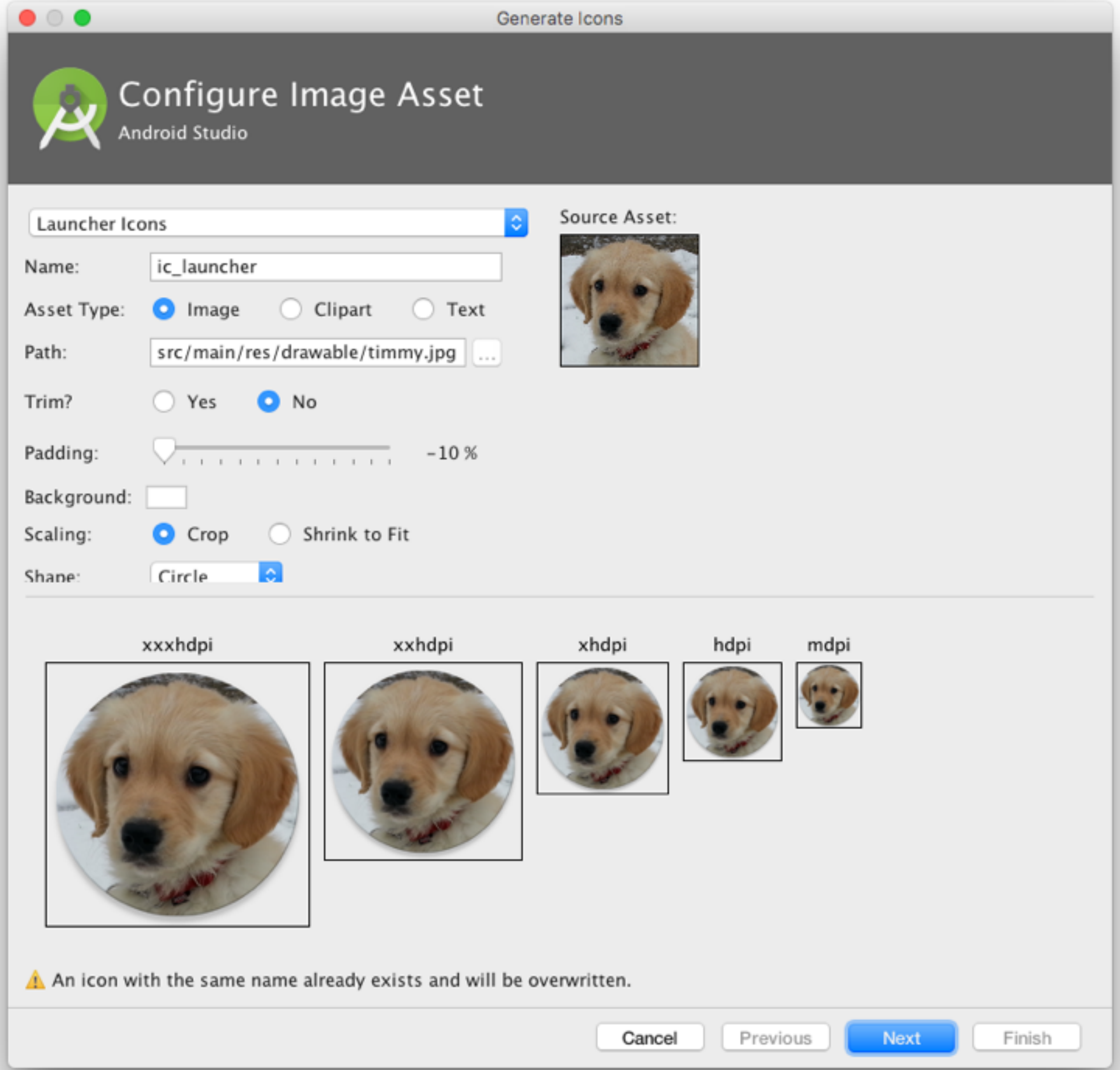
Library Repository:

Ignore Assets Pattern:

Incremental Dex:

Source Compatibility: 1.7

Target Compatibility: 1.7



Generate Icons

Configure Image Asset

Android Studio

Launcher Icons

Name: ic_launcher

Asset Type: ● Image ○ Clipart ○ Text

Path: src/main/res/drawable/timmy.jpg

Trim? ○ Yes ● No

Padding: -10 %

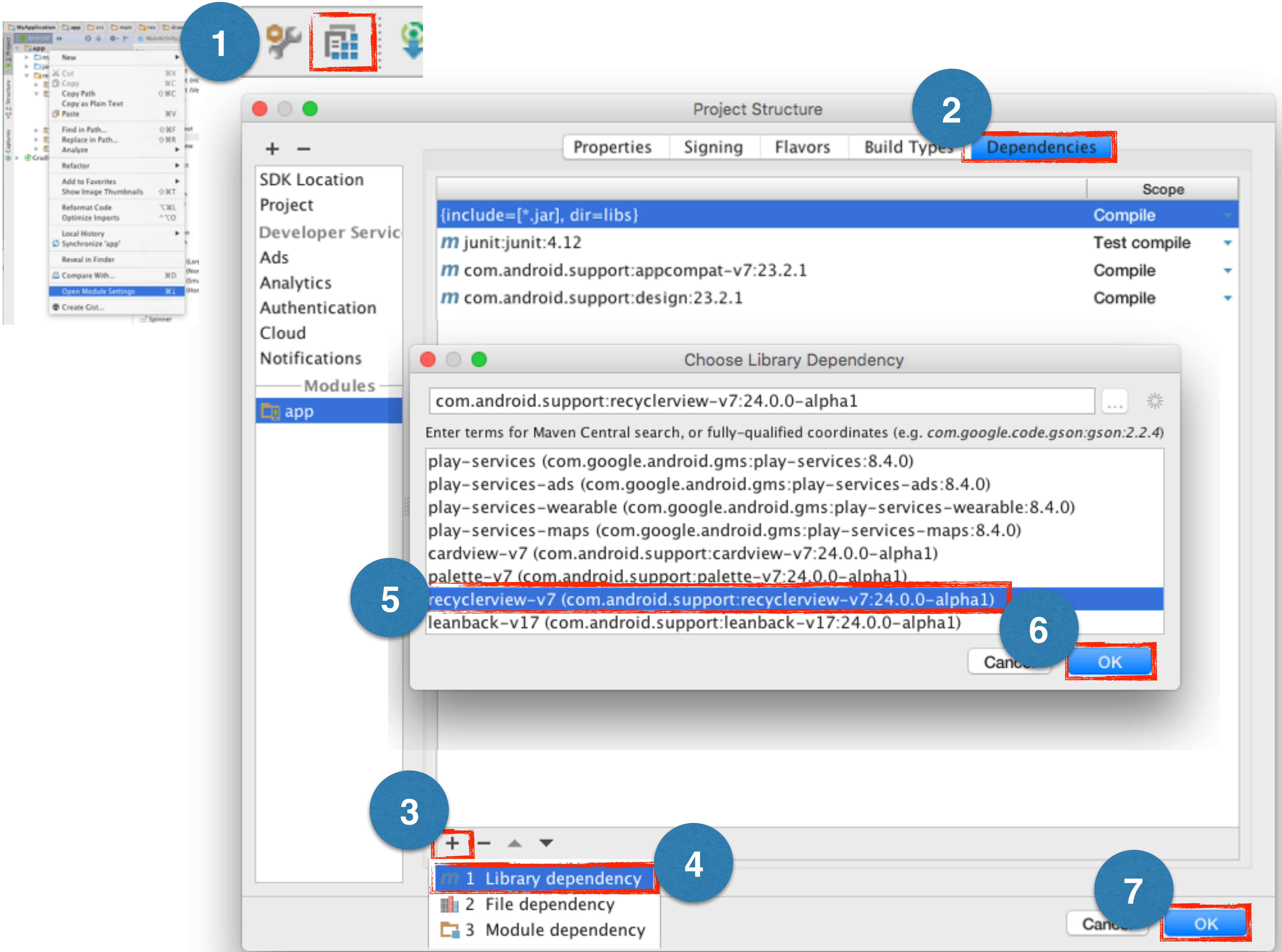
Background:

Scaling: ● Crop ○ Shrink to Fit

Shape: Circle

Source Asset:

⚠ An icon with the same name already exists and will be overwritten.



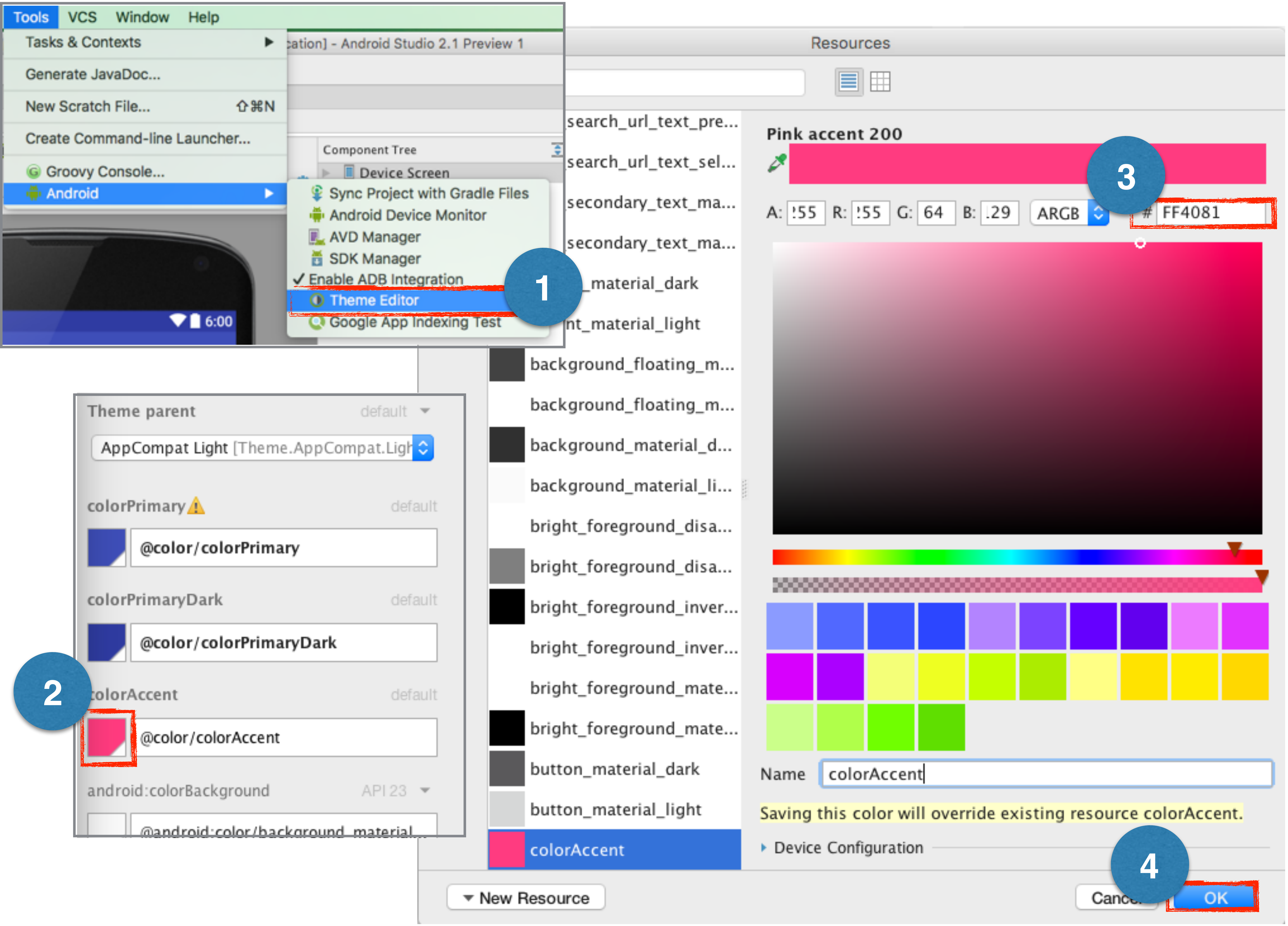
	Scope
{include=[*.jar], dir=libs}	Compile
junit:junit:4.12	Test compile
com.android.support:appcompat-v7:23.2.1	Compile
com.android.support:design:23.2.1	Compile

Choose Library Dependency

com.android.support:recyclerview-v7:24.0.0-alpha1

Enter terms for Maven Central search, or fully-qualified coordinates (e.g. *com.google.code.gson:gson:2.2.4*)

play-services (com.google.android.gms:play-services:8.4.0)
play-services-ads (com.google.android.gms:play-services-ads:8.4.0)
play-services-wearable (com.google.android.gms:play-services-wearable:8.4.0)
play-services-maps (com.google.android.gms:play-services-maps:8.4.0)
cardview-v7 (com.android.support:cardview-v7:24.0.0-alpha1)
palette-v7 (com.android.support:palette-v7:24.0.0-alpha1)
recyclerview-v7 (com.android.support:recyclerview-v7:24.0.0-alpha1)
leanback-v17 (com.android.support:leanback-v17:24.0.0-alpha1)



Theme parent default

AppCompat Light [Theme.AppCompat.Ligh

colorPrimary ⚠ default

@color/colorPrimary

colorPrimaryDark default

@color/colorPrimaryDark

colorAccent default

@color/colorAccent

android:colorBackground API 23

@android:color/background_material...

Pink accent 200

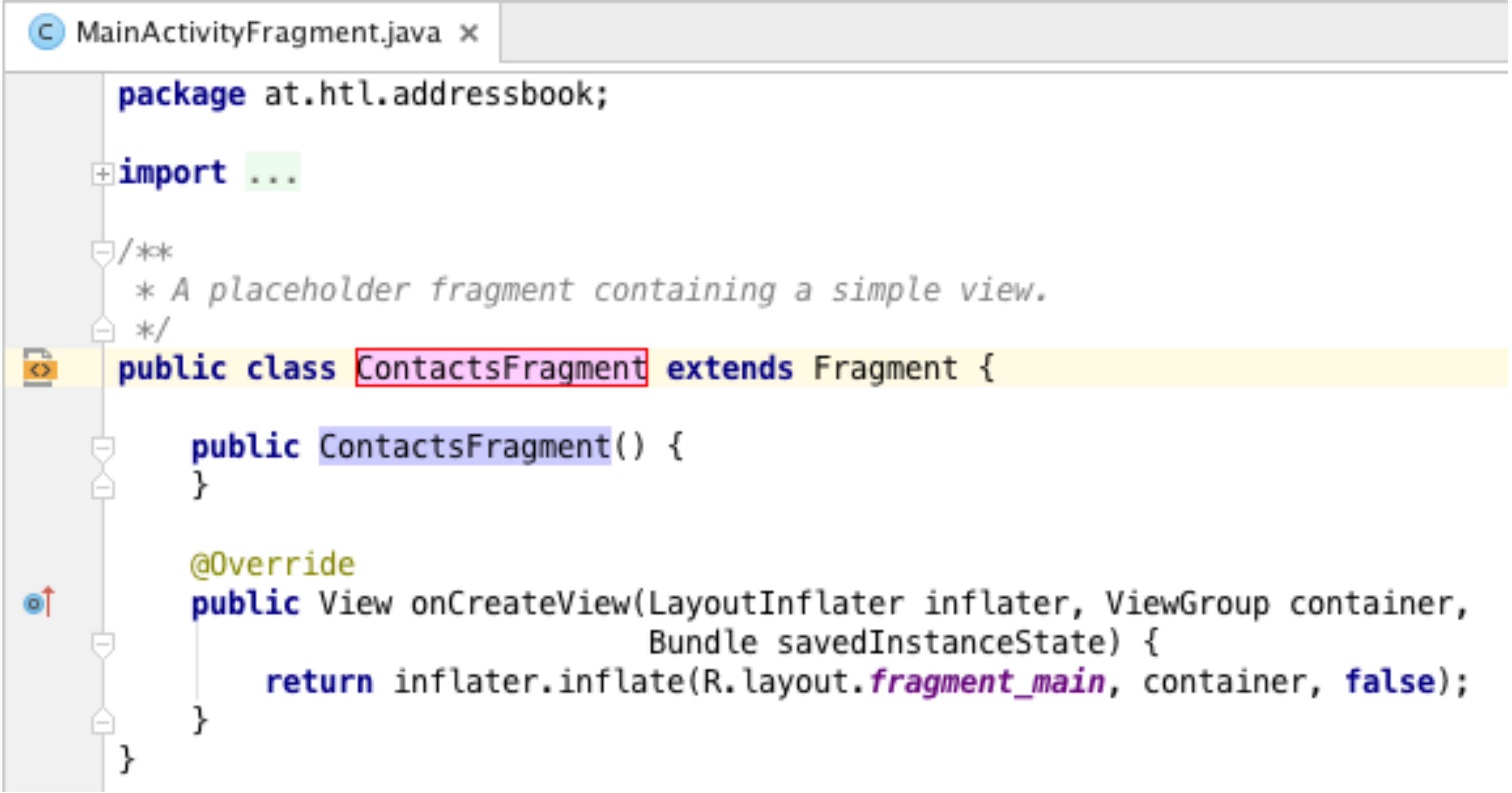
A: !55 R: !55 G: 64 B: .29 ARGB # FF4081

Name colorAccent

Saving this color will override existing resource colorAccent.

Rename MainActivityFragment.java

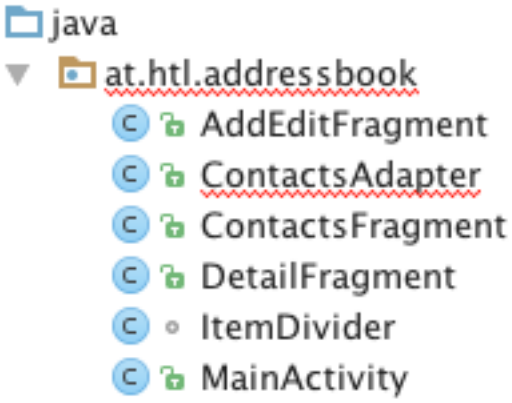
to ContactsFragment.java



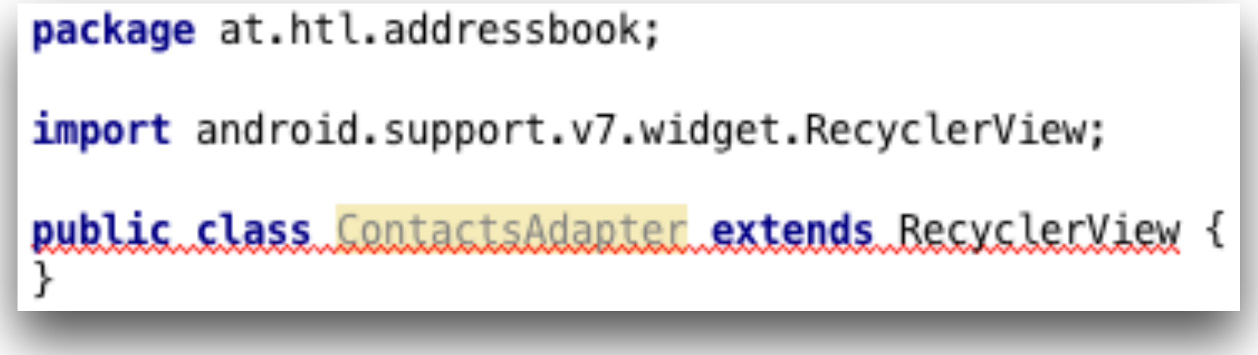
MainActivityFragment.java ×

```
package at.htl.addressbook;

import ...

/**
 * A placeholder fragment containing a simple view.
 */
public class ContactsFragment extends Fragment {

    public ContactsFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_main, container, false);
    }
}
```

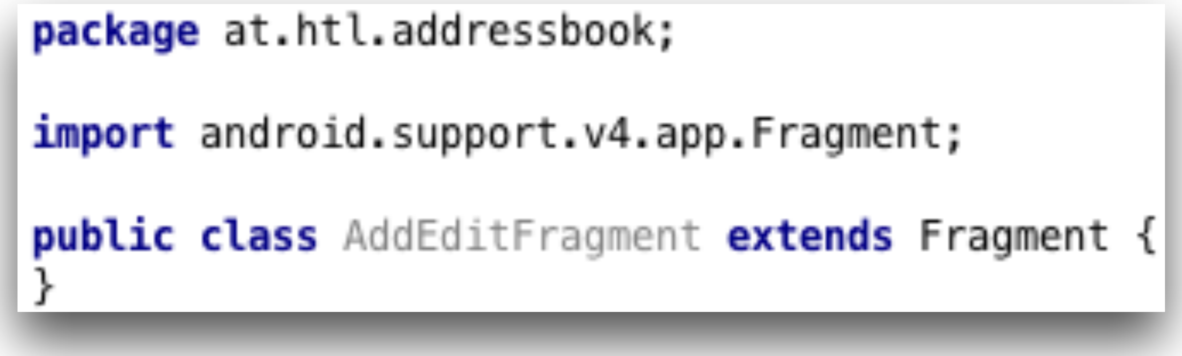
Erstellen von Klassen



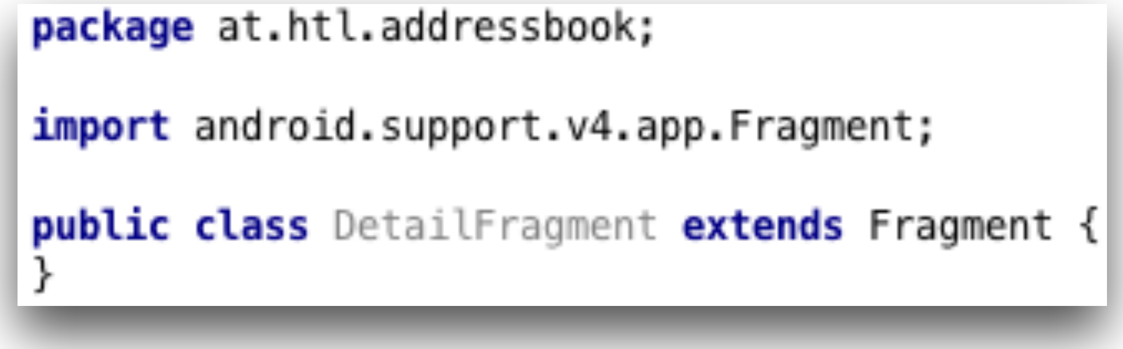
```
java
▼ at.htl.addressbook
    AddEditFragment
    ContactsAdapter
    ContactsFragment
    DetailFragment
    ItemDivider
    MainActivity
```



```java
package at.htl.addressbook;

import android.support.v7.widget.RecyclerView;

public class ContactsAdapter extends RecyclerView {
}
```



```java
package at.htl.addressbook;

import android.support.v4.app.Fragment;

public class AddEditFragment extends Fragment {
}
```



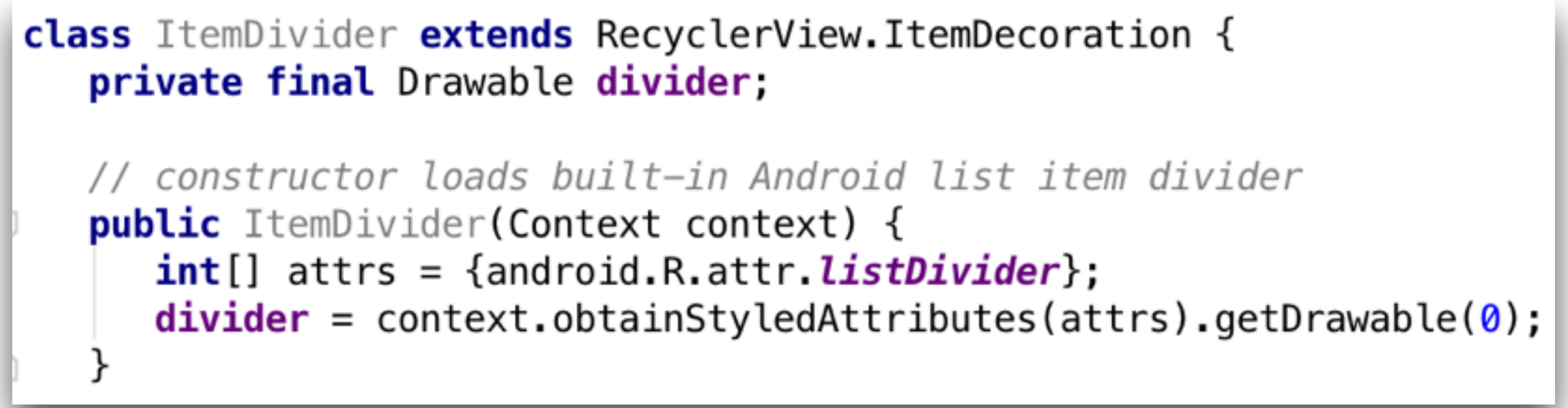
```java
package at.htl.addressbook;

import android.support.v4.app.Fragment;

public class DetailFragment extends Fragment {
}
```

ItemDecoration

- A RecyclerView.ItemDecoration object draws decorations—such as separators between items—on a RecyclerView.

- The RecyclerView.ItemDecoration subclass ItemDivider draws divider lines between list items.

- In the constructor obtain the predefined Android Drawable resource **android.R.attr.listDivider**, which is the standard Android list-item divider used by default in ListViews.



```
class ItemDivider extends RecyclerView.ItemDecoration {
   private final Drawable divider;

   // constructor loads built-in Android list item divider
   public ItemDivider(Context context) {
      int[] attrs = {android.R.attr.listDivider};
      divider = context.obtainStyledAttributes(attrs).getDrawable(0);
   }
```

onDrawOver()

- Beim Scrollen der RecyclerView werden die einzelnen Items wiederholt gezeichnet. dabei wird bei jedem Item die (hier überschriebene) Methode onDrawOver() aufgerufen

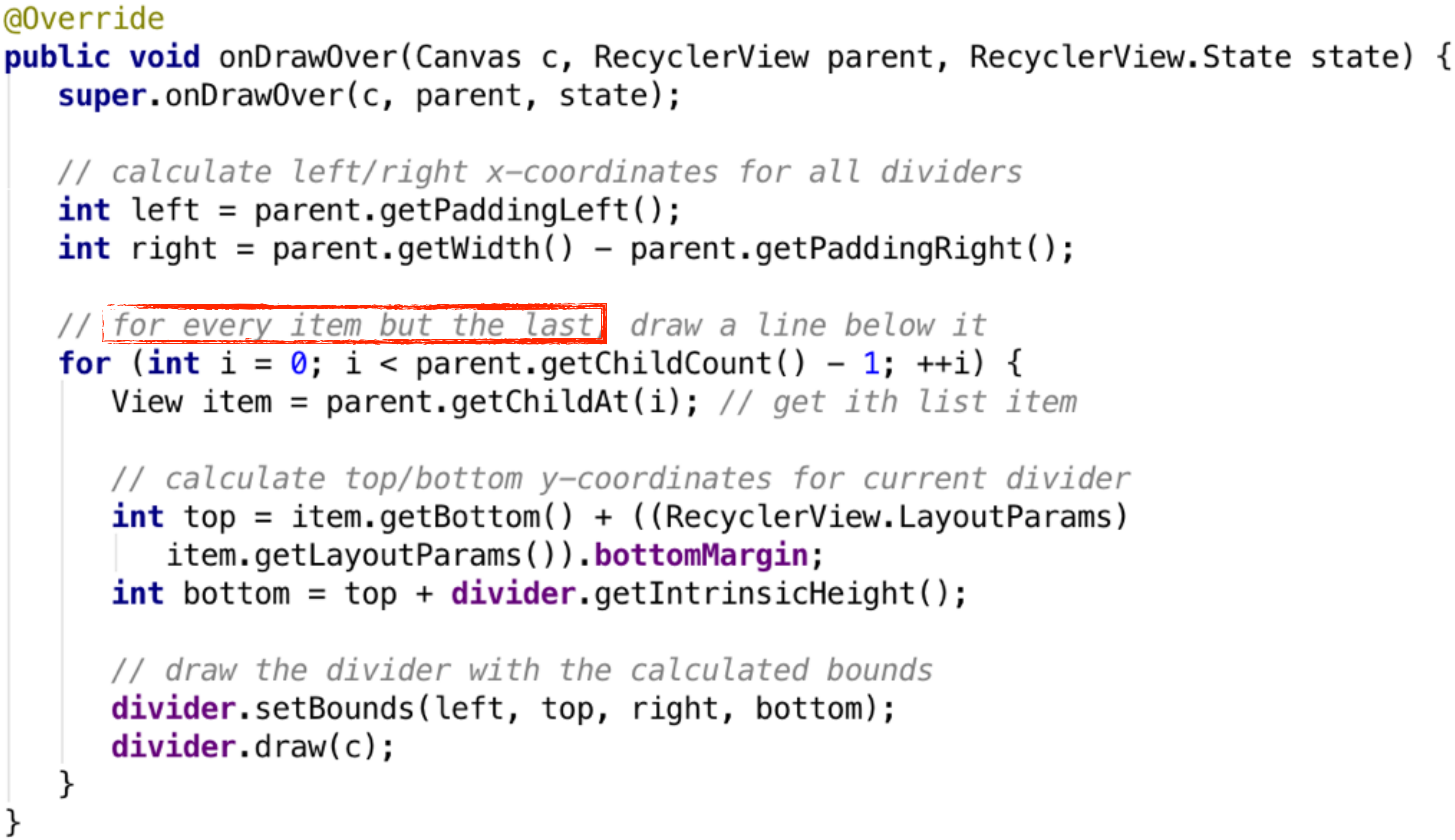
- Die Methode erhält folgende Parameter:

  - den Canvas zum Zeichnen der Dekoration

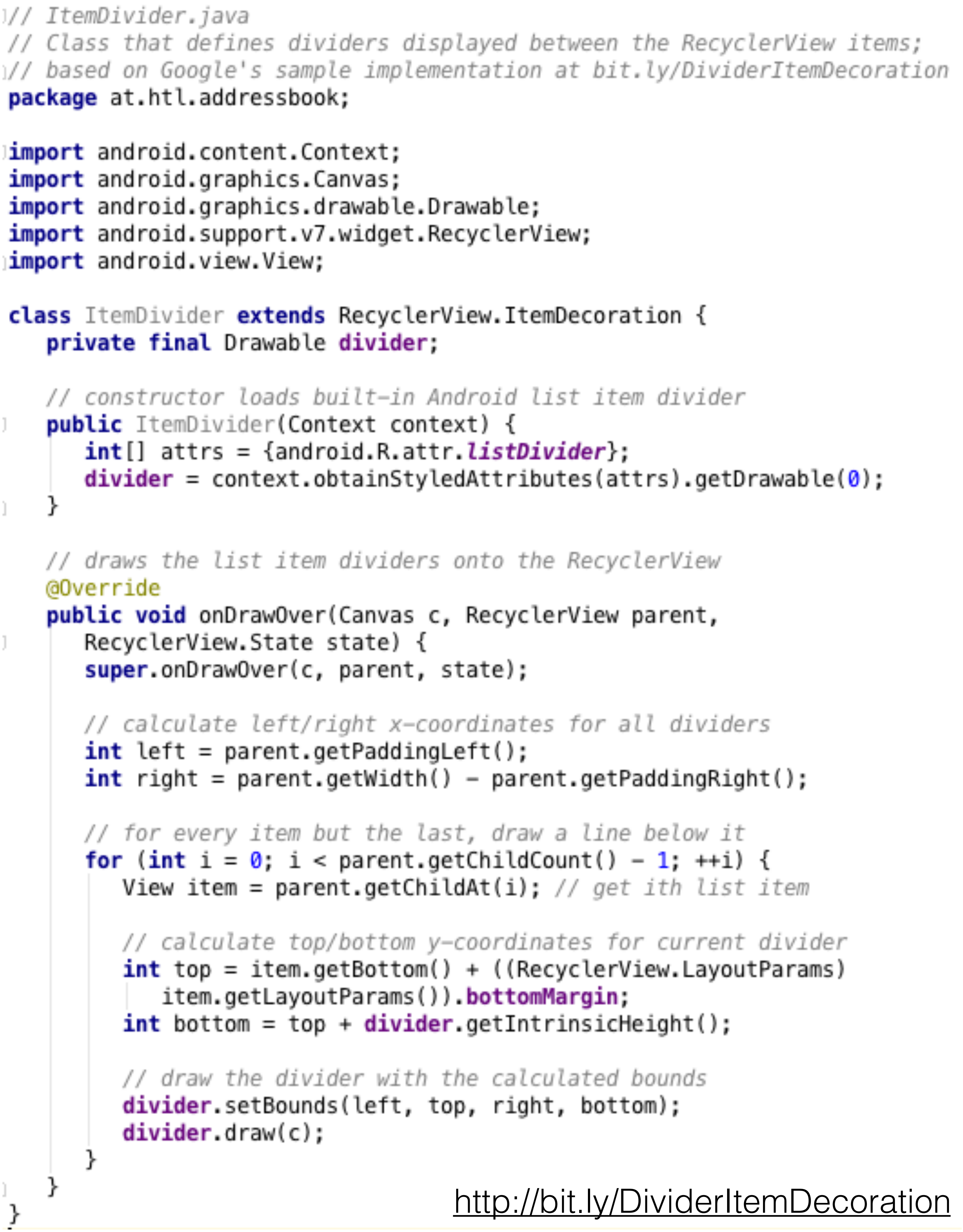
  - die RecyclerView

  - den Status der View (RecyclerView.State) zB aktuelle Scroll-Position. Der spielt hier keine Rolle und wird nur an die Superklasse weitergeleitet.

onDrawOver()



```
@Override
public void onDrawOver(Canvas c, RecyclerView parent, RecyclerView.State state) {
    super.onDrawOver(c, parent, state);

    // calculate left/right x-coordinates for all dividers
    int left = parent.getPaddingLeft();
    int right = parent.getWidth() - parent.getPaddingRight();

    // for every item but the last, draw a line below it
    for (int i = 0; i < parent.getChildCount() - 1; ++i) {
        View item = parent.getChildAt(i); // get ith list item

        // calculate top/bottom y-coordinates for current divider
        int top = item.getBottom() + ((RecyclerView.LayoutParams)
            item.getLayoutParams()).bottomMargin;
        int bottom = top + divider.getIntrinsicHeight();

        // draw the divider with the calculated bounds
        divider.setBounds(left, top, right, bottom);
        divider.draw(c);
    }
}
```



```java
// ItemDivider.java
// Class that defines dividers displayed between the RecyclerView items;
// based on Google's sample implementation at bit.ly/DividerItemDecoration
package at.htl.addressbook;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.drawable.Drawable;
import android.support.v7.widget.RecyclerView;
import android.view.View;

class ItemDivider extends RecyclerView.ItemDecoration {
   private final Drawable divider;

   // constructor loads built-in Android list item divider
   public ItemDivider(Context context) {
      int[] attrs = {android.R.attr.listDivider};
      divider = context.obtainStyledAttributes(attrs).getDrawable(0);
   }

   // draws the list item dividers onto the RecyclerView
   @Override
   public void onDrawOver(Canvas c, RecyclerView parent,
      RecyclerView.State state) {
      super.onDrawOver(c, parent, state);

      // calculate left/right x-coordinates for all dividers
      int left = parent.getPaddingLeft();
      int right = parent.getWidth() - parent.getPaddingRight();

      // for every item but the last, draw a line below it
      for (int i = 0; i < parent.getChildCount() - 1; ++i) {
         View item = parent.getChildAt(i); // get ith list item

         // calculate top/bottom y-coordinates for current divider
         int top = item.getBottom() + ((RecyclerView.LayoutParams)
            item.getLayoutParams()).bottomMargin;
         int bottom = top + divider.getIntrinsicHeight();

         // draw the divider with the calculated bounds
         divider.setBounds(left, top, right, bottom);
         divider.draw(c);
      }
   }
}
```

http://bit.ly/DividerItemDecoration

Database Overview

SQLite in Android

- **SQLiteOpenHelper** vereinfacht die Erstellung der Datenbank und gibt ein **SQLiteDatabase**-Objekt zurück, dass zur Manipulation der Daten benötigt wird
- Datenbankabfragen werden mit SQL durchgeführt
- Ergebnisse werden mittels einem **Cursor** zurückgegeben

ContentProvider 1

- Ein **ContentProvider** ermöglicht die Benutzung von Daten einer App in einer anderen App. So kann die eigene App mit Daten aus Android Contacts- und Calendar-Apps arbeiten

- Es stehen auch ContentProvider zur Verfügung, die verschiedene Telephone-Features, den Zugriff auf den MediaStore und das User Dictionary (predicitive text-input) usw. ermöglichen

- In dieser App wird mit ContentProvidern der asynchrone Zugriff auf die DB außerhalb des GUI-Threads ermöglicht. Dies ist notwendig, wenn mit Loaders and LoaderManager gearbeitet wird.

ContentProvider 2

- Wir werden in dieser App einen ContentProvider erstellen:
    - zur Abfrage der DB, um Kontakte zu finden
    - um einen neuen Kontakt einzufügen
    - einen Kontakt in der DB zu ändern
    - einen Kontakt aus der DB zu löschen
- Nach Durchführen der Änderungen wird mit Listeners die GUI aktualisiert

ContentResolver

- Mit einem ContentResolver werden die Methoden des ContentProviders (query, create, update, delete) aufgerufen

- Mit Uri’s (Uniform Resource Identifier) werden die ContentProvider adressiert auf die der jeweilige ContentResolver zugreifen möchte.

http://developer.android.com/guide/topics/providers/content-provider-basics.html

Loader and LoaderManager

- Lang laufende Operationen oder Operationen, die die weitere Ausführung behindern, bis sie abgeschlossen sind (zB Datei- und Datenbank-Zugriff) sind außerhalb des GUI-Threads auszuführen

- Dadurch wird die Antwortzeit verbessert und die ANR-Exception (Activity not responding) tritt nicht auf

- Loader und LoaderManager ermöglichen den asynchronen Datenzugriff aus Activities und Fragments

Loader

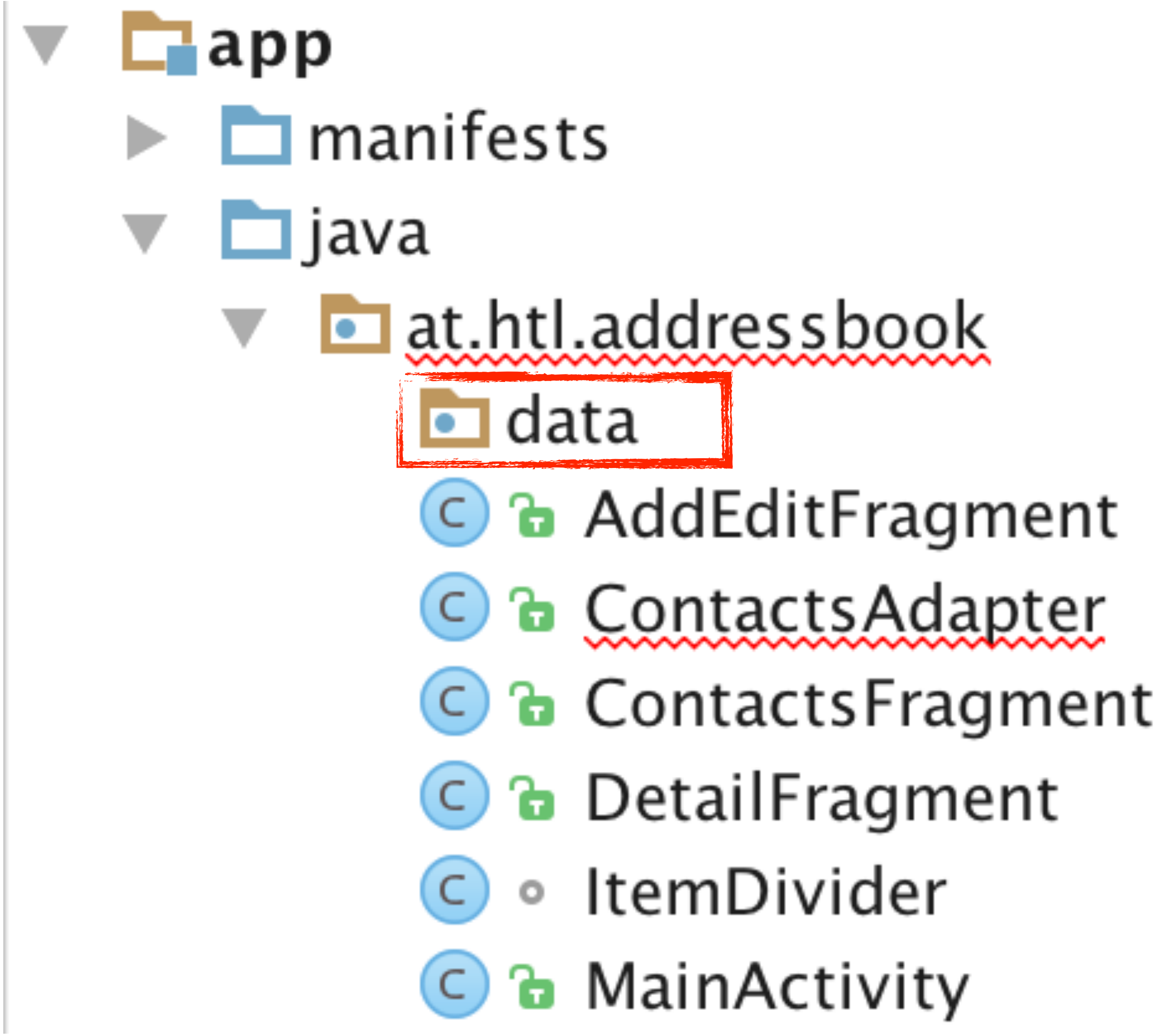
- Ein Loader führt einen asynchronen Datenzugriff durch
- Im Normalfall, bei Zugriff auf einen ContentProvider wird ein **CursorLoader** verwendet
- Der CursorLoader ist vom AsyncTaskLoader abgeleitet, der einen AsyncTask benutzt, um den Datenzugriff in einem eigenem Thread durchzuführen
- Loader beobachten auch die korrespondierenden Datenquellen und aktualisieren die Activity / Fragment
- Es ist performanter, einen bestehenden Loader zu reconnecten, als eine neue Query durchzuführen

LoaderManager

- Die Loader einer Activity oder Fragments werden durch einen LoaderManager erstellt und verwaltet
- Der Lifecycle eines Loaders ist an seine Activity / Fragment gekoppelt
- Das LoaderManager.LoaderCallbacks Interface dient zum Benachrichtigen der Activity / Fragment wenn ein Loader
  - erstellt wird
  - das Laden der Daten beendet hat
  - zurückgesetzt wurde und die Daten nicht länger verfügbar sind

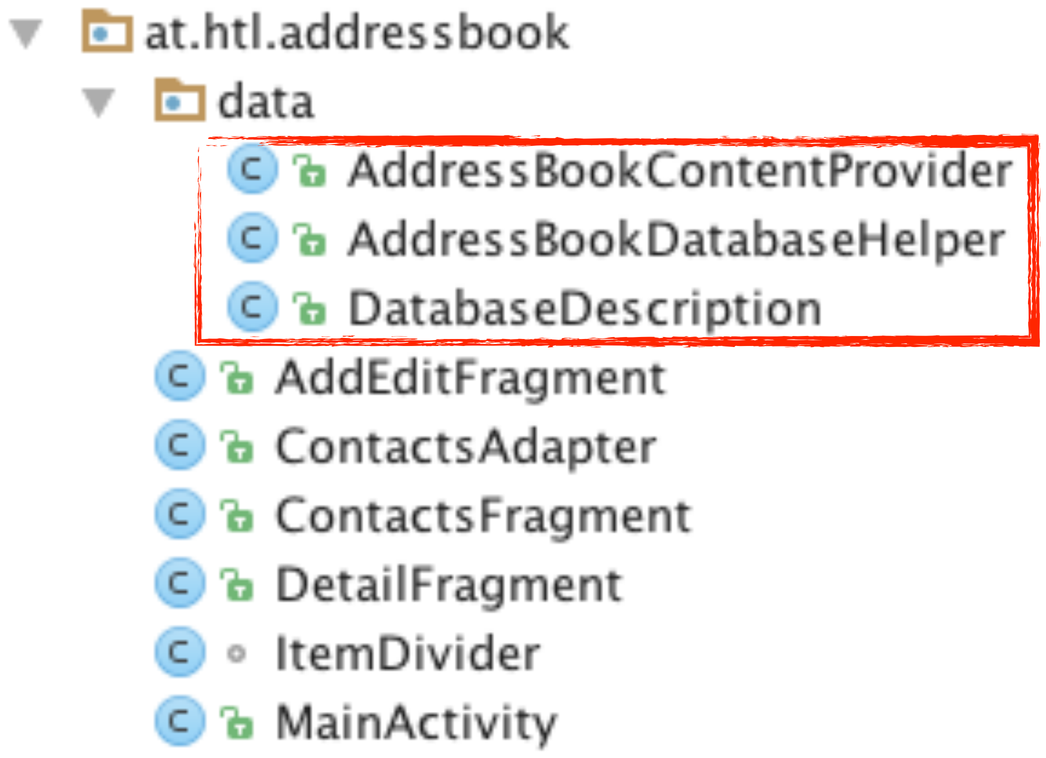
http://developer.android.com/guide/components/loaders.html

Package data



app
manifests
java
at.htl.addressbook
data
AddEditFragment
ContactsAdapter
ContactsFragment
DetailFragment
ItemDivider
MainActivity

addressbook.data

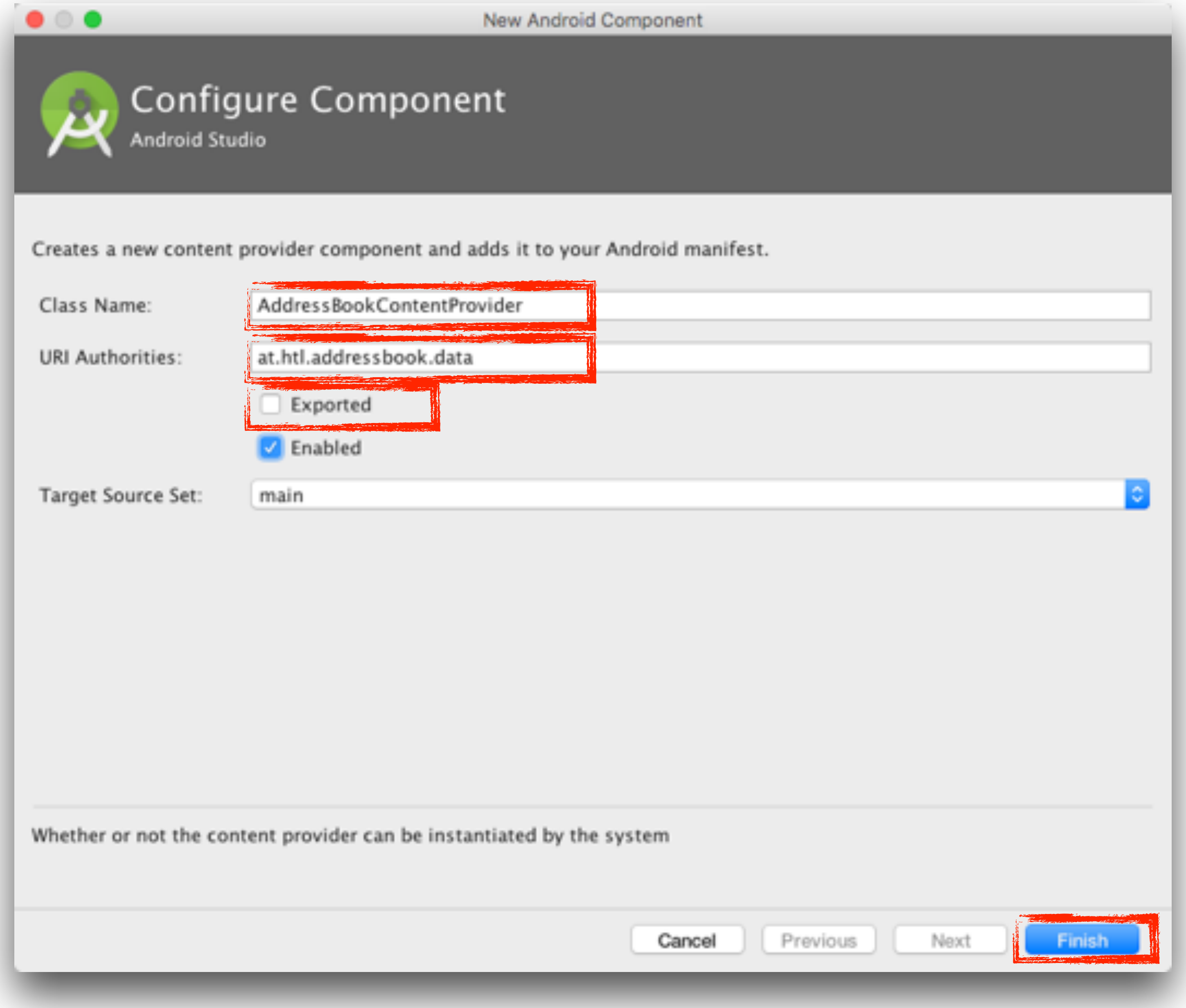


▼ at.htl.addressbook
  ▼ data
    AddressBookContentProvider
    AddressBookDatabaseHelper
    DatabaseDescription
  AddEditFragment
  ContactsAdapter
  ContactsFragment
  DetailFragment
  ItemDivider
  MainActivity

- DatabaseDescription: Struktur der contact-Tabelle in der Datenbank
- AddressBookDatabaseHelper: abgeleitet von SQLiteOpenHelper zur Erstellung der DB und den Zugriff darauf
- AddressBookContentProvider: abgeleitet von ContentProvider. Ist für Änderungen (DML) in der Datenbank zuständig

AddressBookContentProvider erstellen

New - Other - ContentProvider



Creates a new content provider component and adds it to your Android manifest.

Class Name: AddressBookContentProvider

URI Authorities: at.htl.addressbook.data

☐ Exported

☑ Enabled

Target Source Set: main

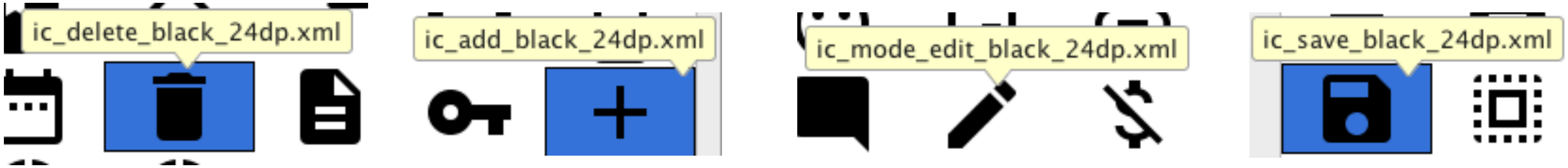
Whether or not the content provider can be instantiated by the system

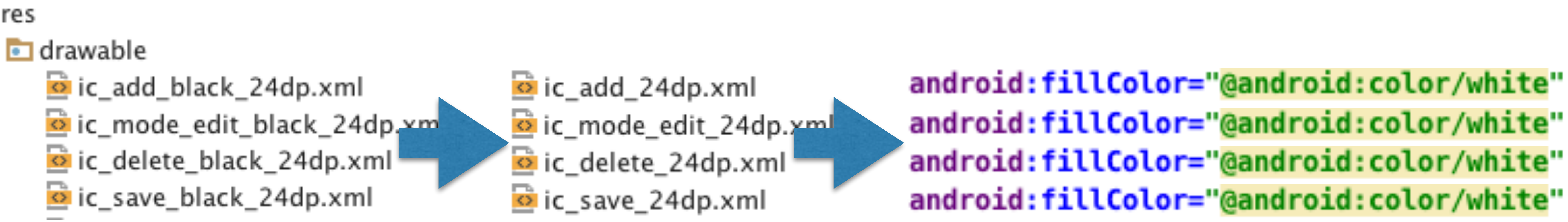
Beachte: Dieser ContentProvider wird im Manifest eingetragen

```
<provider
    android:name=".data.AddressBookContentProvider"
    android:authorities="at.htl.addressbook.data"
    android:enabled="true"
    android:exported="false">
</provider>
```

Icons’s hinzufügen

Vector Asset Studio



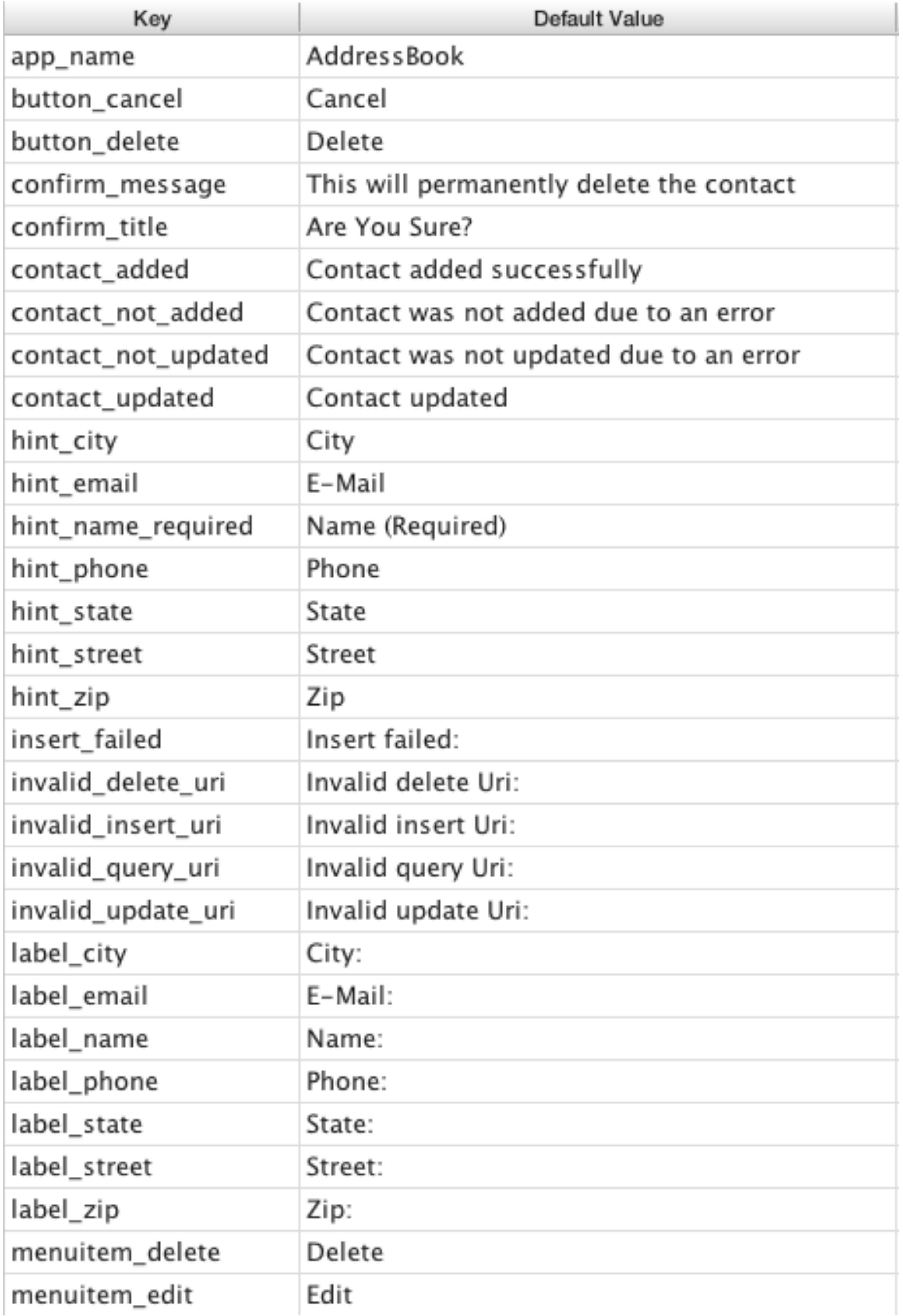


res
drawable
- ic_add_black_24dp.xml
- ic_mode_edit_black_24dp.xm
- ic_delete_black_24dp.xml
- ic_save_black_24dp.xml

- ic_add_24dp.xml
- ic_mode_edit_24dp.xml
- ic_delete_24dp.xml
- ic_save_24dp.xml

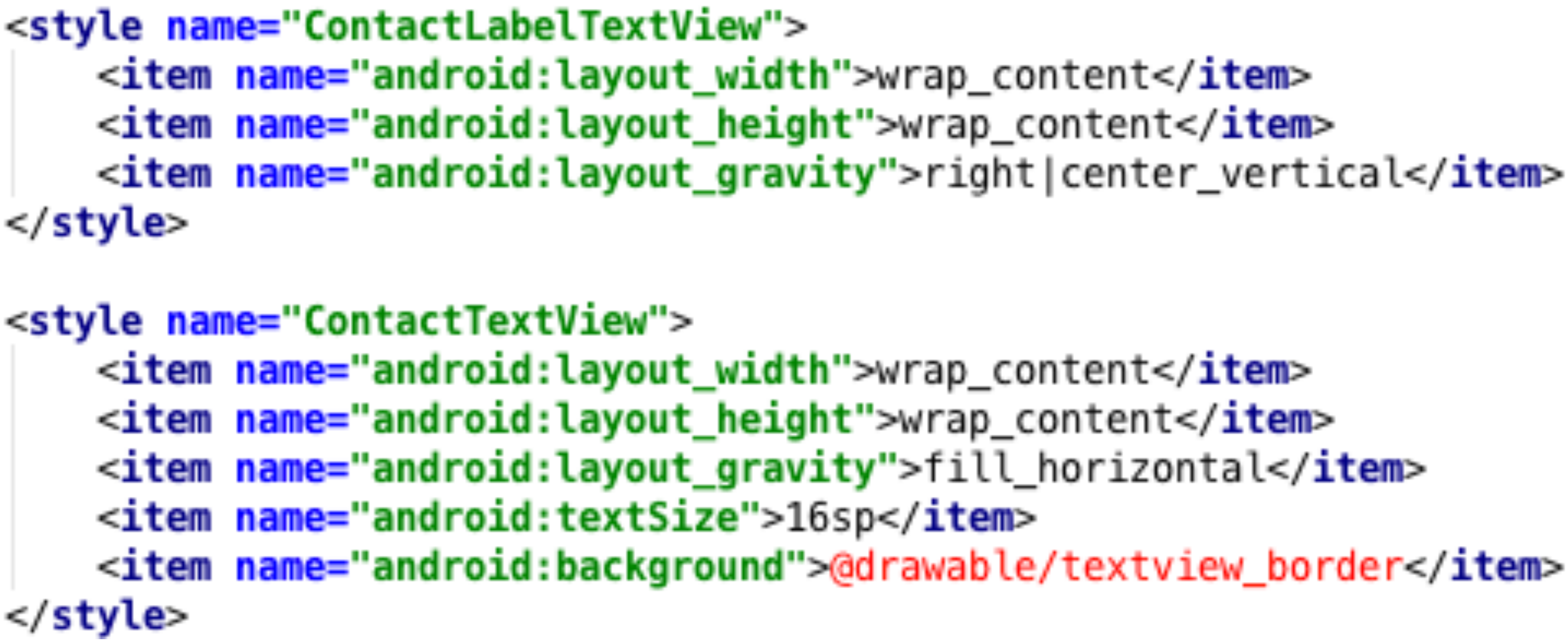
```
android:fillColor="@android:color/white"
android:fillColor="@android:color/white"
android:fillColor="@android:color/white"
android:fillColor="@android:color/white"
```

strings.xml



Key	Default Value
app_name	AddressBook
button_cancel	Cancel
button_delete	Delete
confirm_message	This will permanently delete the contact
confirm_title	Are You Sure?
contact_added	Contact added successfully
contact_not_added	Contact was not added due to an error
contact_not_updated	Contact was not updated due to an error
contact_updated	Contact updated
hint_city	City
hint_email	E-Mail
hint_name_required	Name (Required)
hint_phone	Phone
hint_state	State
hint_street	Street
hint_zip	Zip
insert_failed	Insert failed:
invalid_delete_uri	Invalid delete Uri:
invalid_insert_uri	Invalid insert Uri:
invalid_query_uri	Invalid query Uri:
invalid_update_uri	Invalid update Uri:
label_city	City:
label_email	E-Mail:
label_name	Name:
label_phone	Phone:
label_state	State:
label_street	Street:
label_zip	Zip:
menuitem_delete	Delete
menuitem_edit	Edit

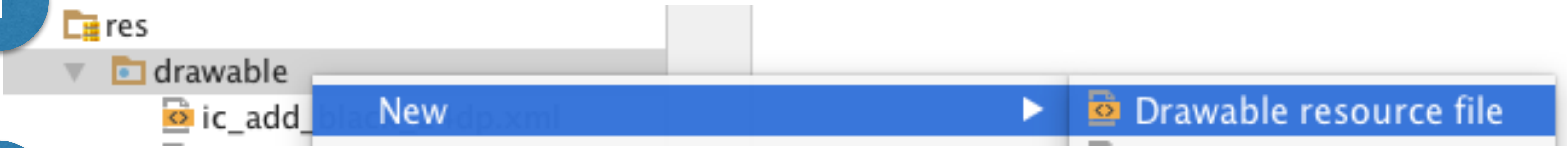
styles.xml



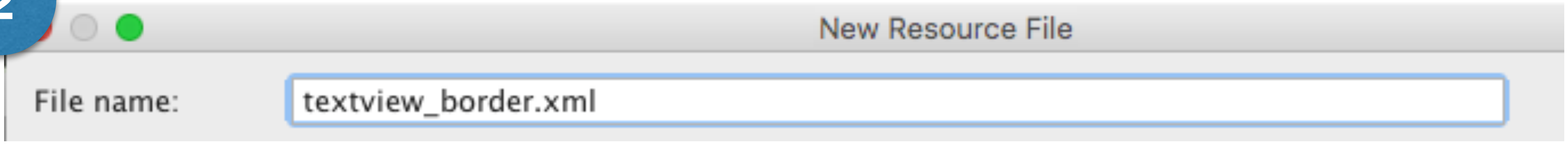
```xml
<style name="ContactLabelTextView">
    <item name="android:layout_width">wrap_content</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:layout_gravity">right|center_vertical</item>
</style>

<style name="ContactTextView">
    <item name="android:layout_width">wrap_content</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:layout_gravity">fill_horizontal</item>
    <item name="android:textSize">16sp</item>
    <item name="android:background">@drawable/textview_border</item>
</style>
```

textview_border.xml

1

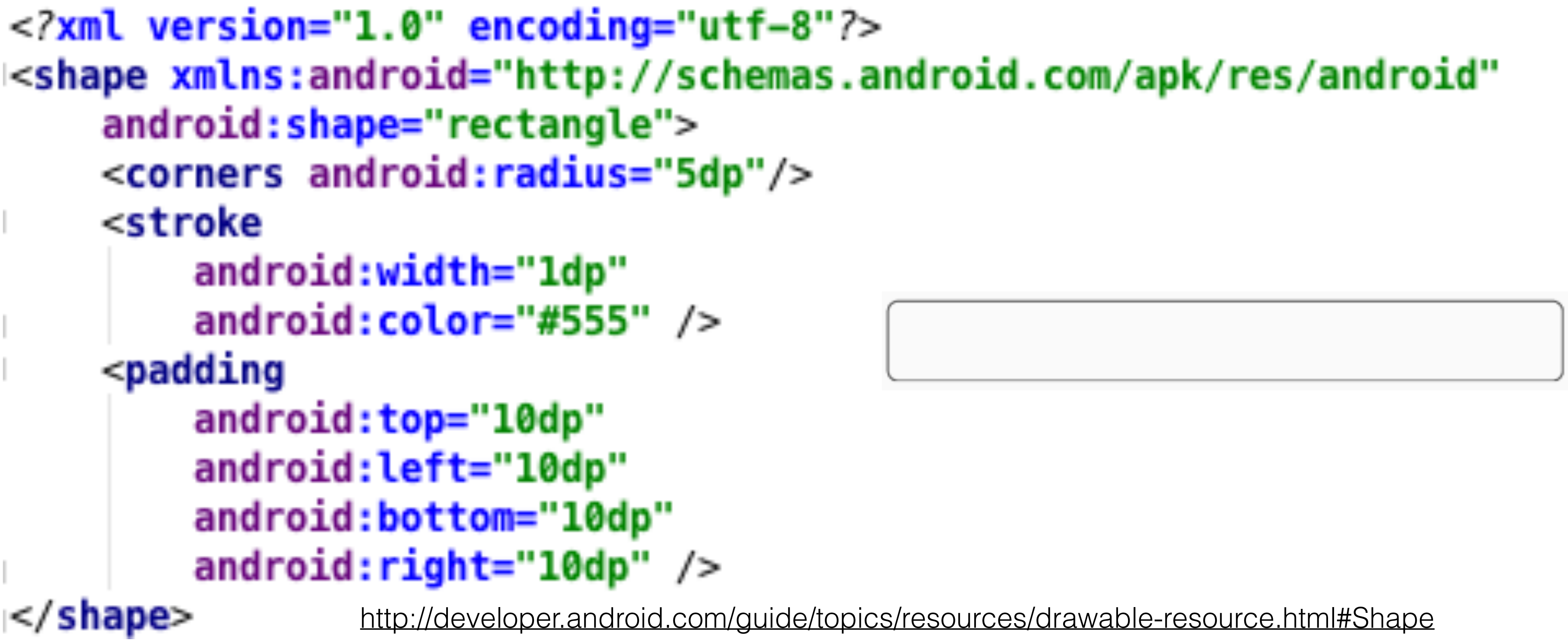


2


New Resource File

File name: textview_border.xml

3


```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <corners android:radius="5dp"/>
    <stroke
        android:width="1dp"
        android:color="#555" />
    <padding
        android:top="10dp"
        android:left="10dp"
        android:bottom="10dp"
        android:right="10dp" />
</shape>
```

http://developer.android.com/guide/topics/resources/drawable-resource.html#Shape

MainActivity layout

- Per default beinhaltet die MainActivity einen FloatingActionButton. In dieser App stellen wir den FloatingActionButton nur zur Verfügung, wenn er gebraucht wird. Daher wird der FAB in activity_main.xml gelöscht.

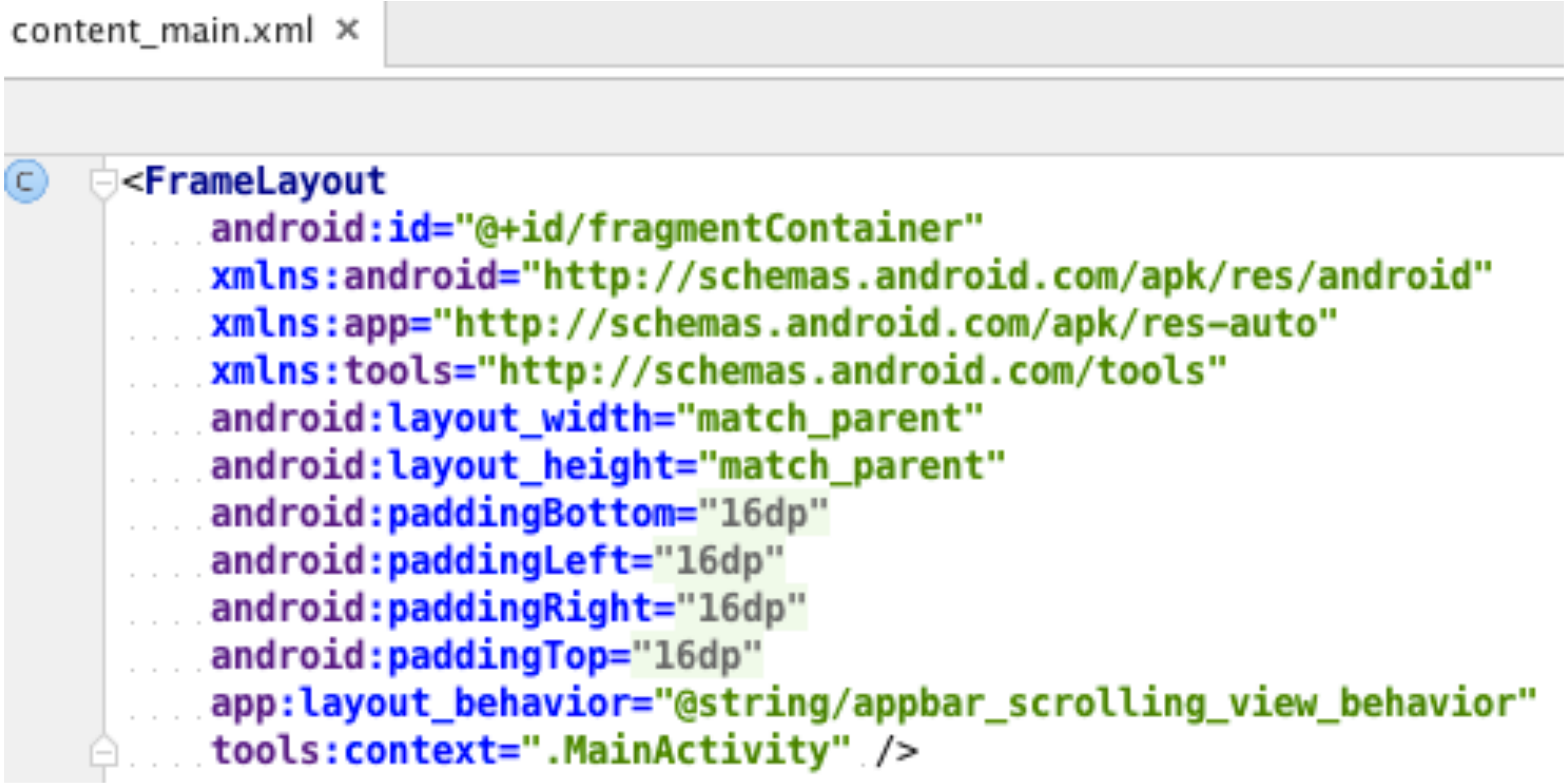
- Die id des CoordinatorLayout wird auf **coordinatorLayout** gesetzt. Bei Verwendung von Snackbars greifen wir auf diese id zu

- Lösche den Code des FAB

content_main.xml

for phone-sized devices

ersetze den Inhalt von content_main.xml

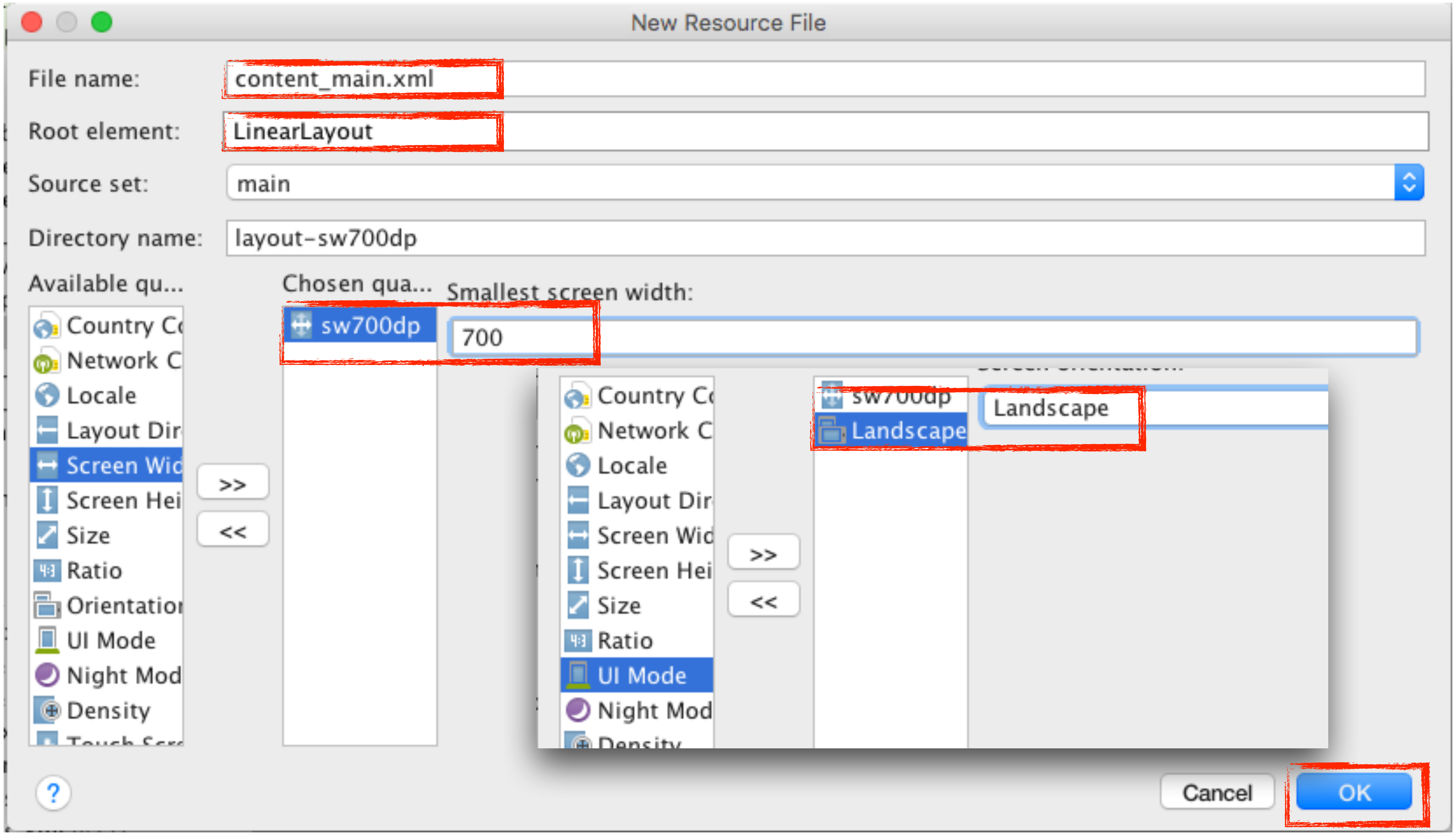


```
<FrameLayout
    android:id="@+id/fragmentContainer"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context=".MainActivity" />
```

content_main.xml

res/layout - New Layout File

for table-sized devices



New Resource File

File name: content_main.xml

Root element: LinearLayout

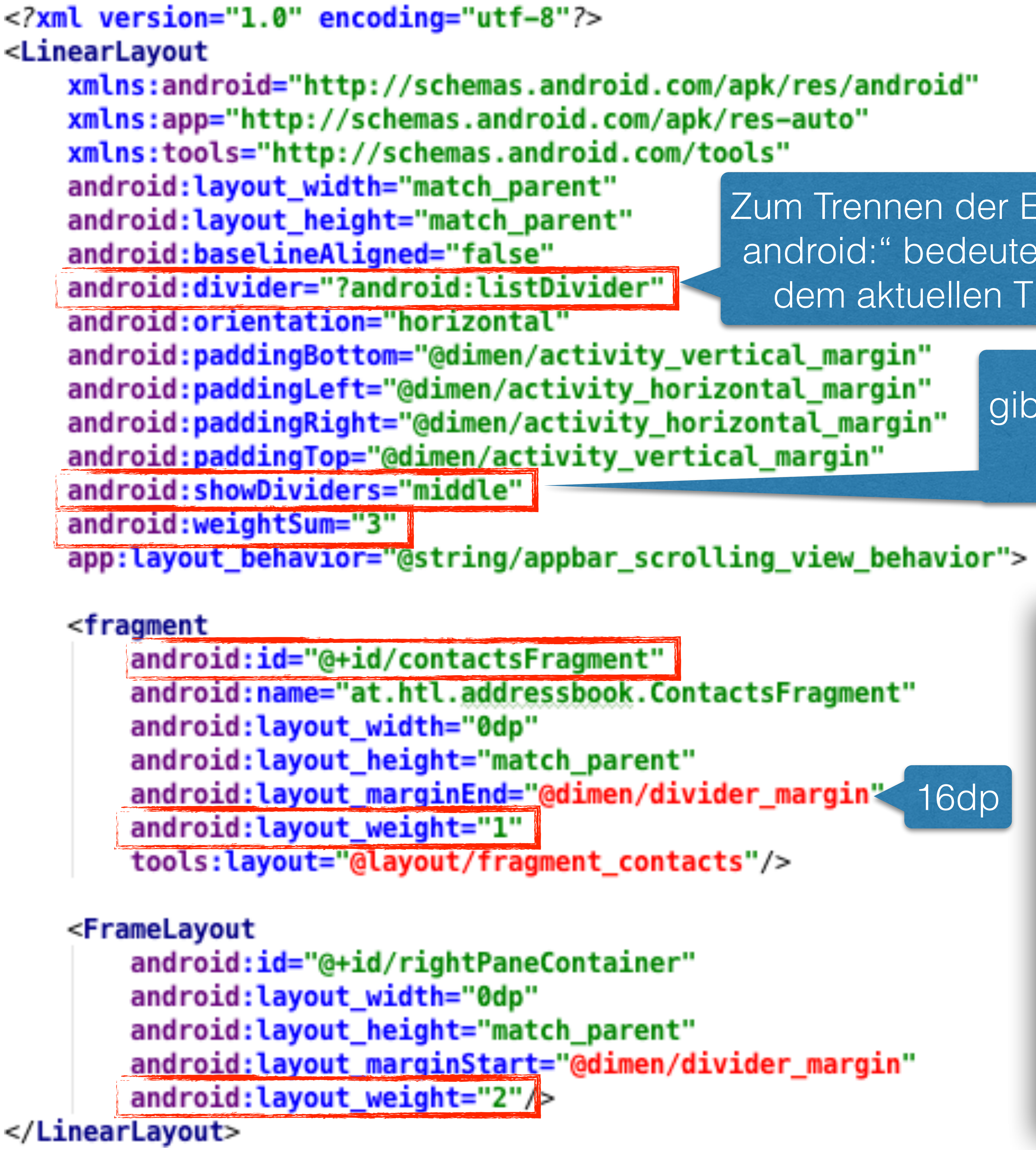
Source set: main

Directory name: layout-sw700dp

Chosen qua...: sw700dp

Smallest screen width: 700

Screen Orientation: Landscape

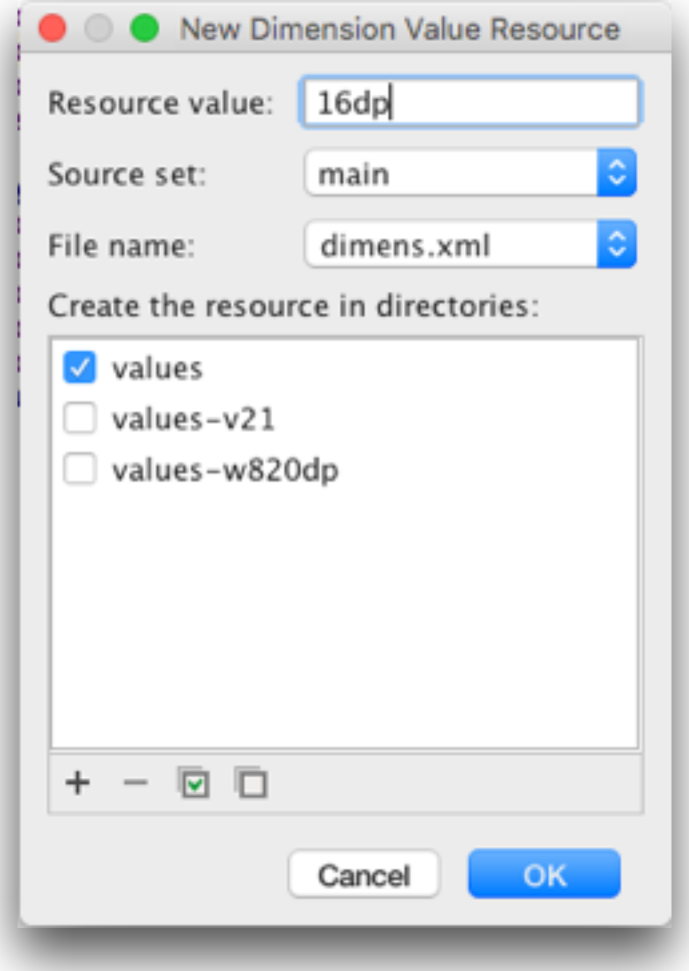


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:baselineAligned="false"
    android:divider="?android:listDivider"
    android:orientation="horizontal"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:showDividers="middle"
    android:weightSum="3"
    app:layout_behavior="@string/appbar_scrolling_view_behavior">

    <fragment
        android:id="@+id/contactsFragment"
        android:name="at.htl.addressbook.ContactsFragment"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_marginEnd="@dimen/divider_margin"
        android:layout_weight="1"
        tools:layout="@layout/fragment_contacts"/>

    <FrameLayout
        android:id="@+id/rightPaneContainer"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_marginStart="@dimen/divider_margin"
        android:layout_weight="2"/>
</LinearLayout>
```

Zum Trennen der Elemente in der Liste. „?android:“ bedeutet, dass der Trennstrich dem aktuellen Theme entnommen ist

gibt an, wo der Divider angezeigt wird

16dp



New Dimension Value Resource

Resource value: 16dp

Source set: main

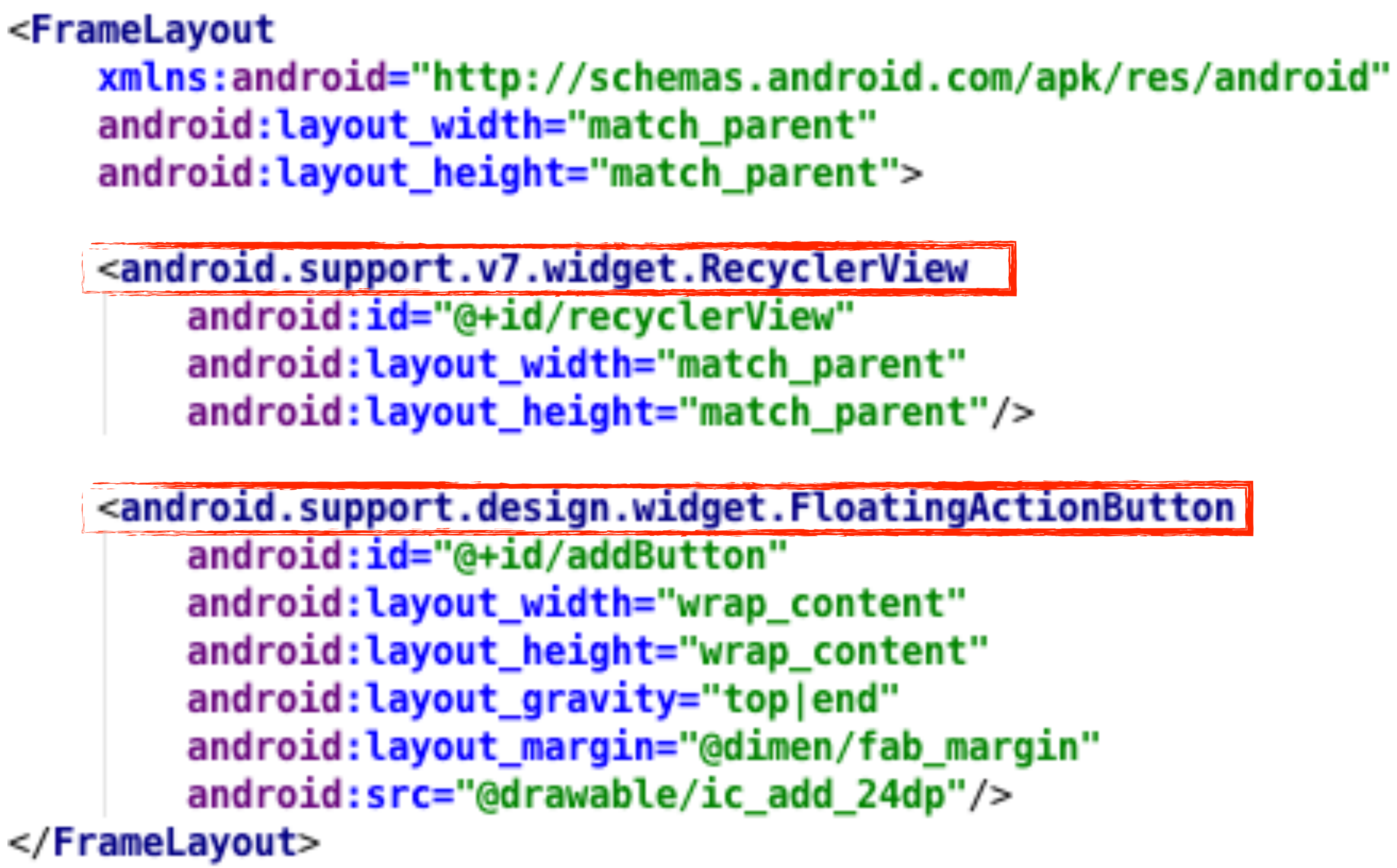
File name: dimens.xml

Create the resource in directories:

☑ values
□ values-v21
□ values-w820dp

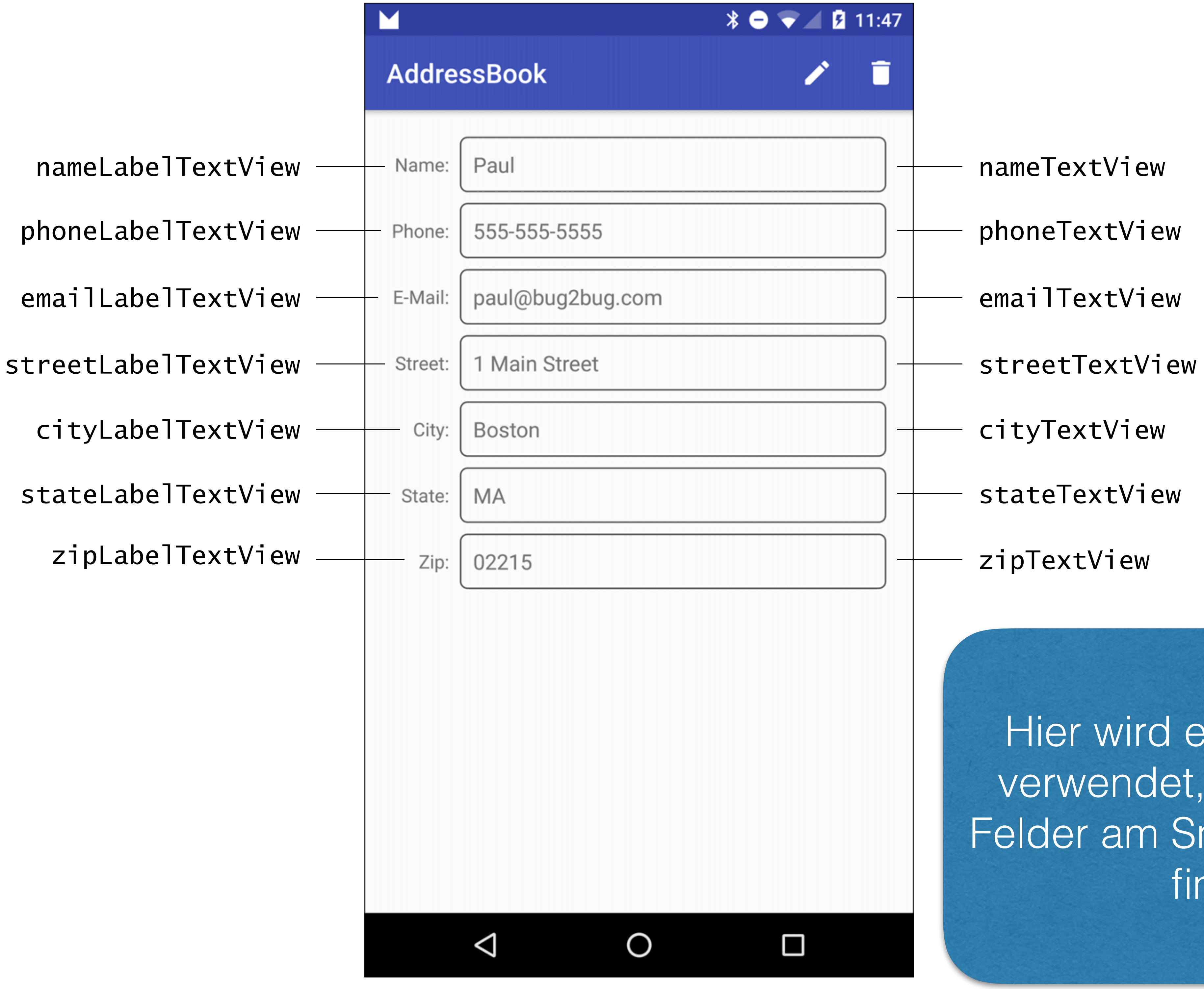
fragment_contacts.xml

res/layout - New - Layout Resource File



```
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <android.support.v7.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/addButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="top|end"
        android:layout_margin="@dimen/fab_margin"
        android:src="@drawable/ic_add_24dp"/>
</FrameLayout>
```

DetailFragment layout



11:47

AddressBook

nameLabelTextView — Name: Paul — nameTextView

phoneLabelTextView — Phone: 555-555-5555 — phoneTextView

emailLabelTextView — E-Mail: paul@bug2bug.com — emailTextView

streetLabelTextView — Street: 1 Main Street — streetTextView

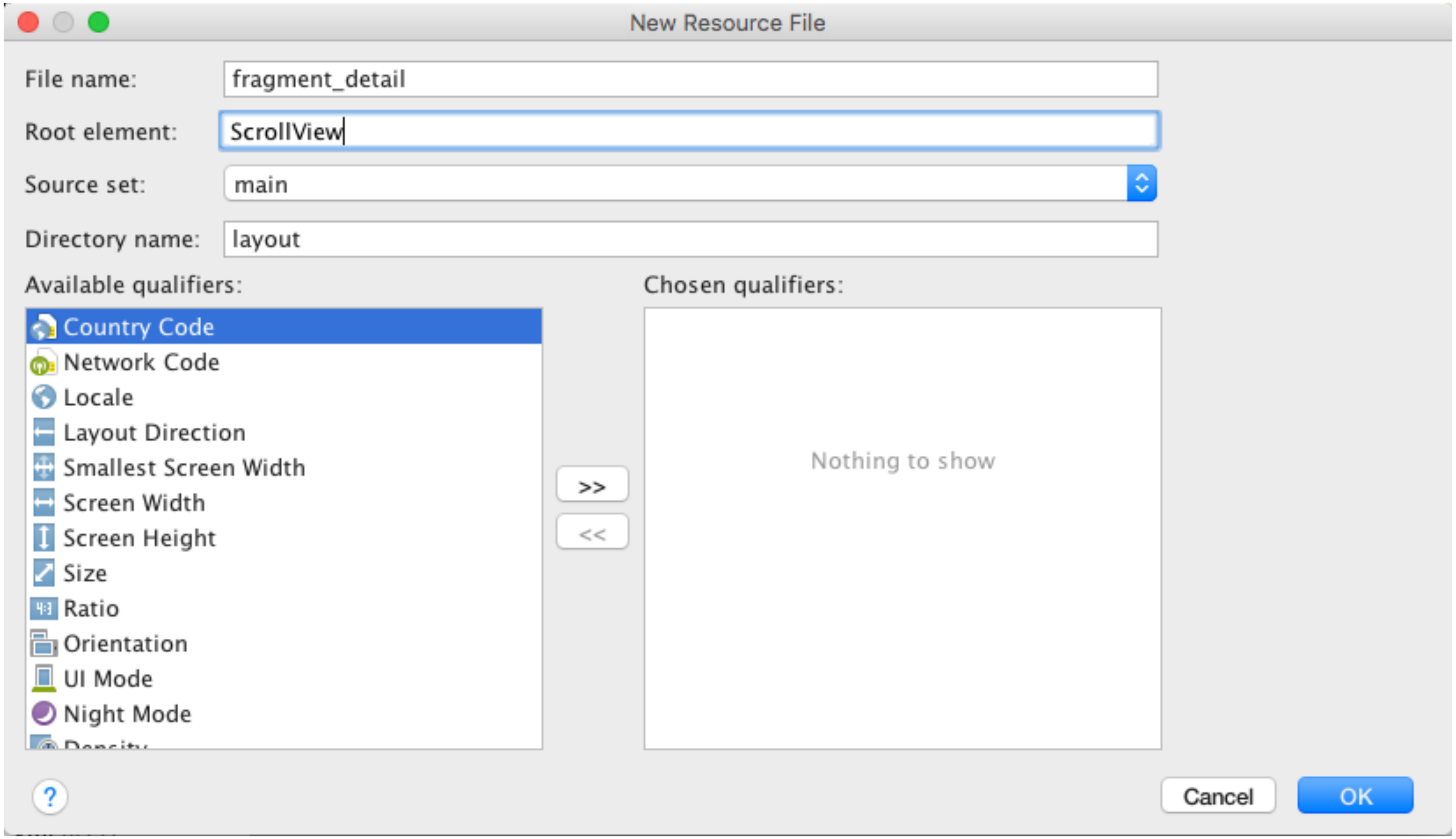
cityLabelTextView — City: Boston — cityTextView

stateLabelTextView — State: MA — stateTextView

zipLabelTextView — Zip: 02215 — zipTextView

Hier wird eine ScrollView verwendet, falls nicht alle Felder am Smartphone Platz finden.

fragment_detail.xml



New Resource File

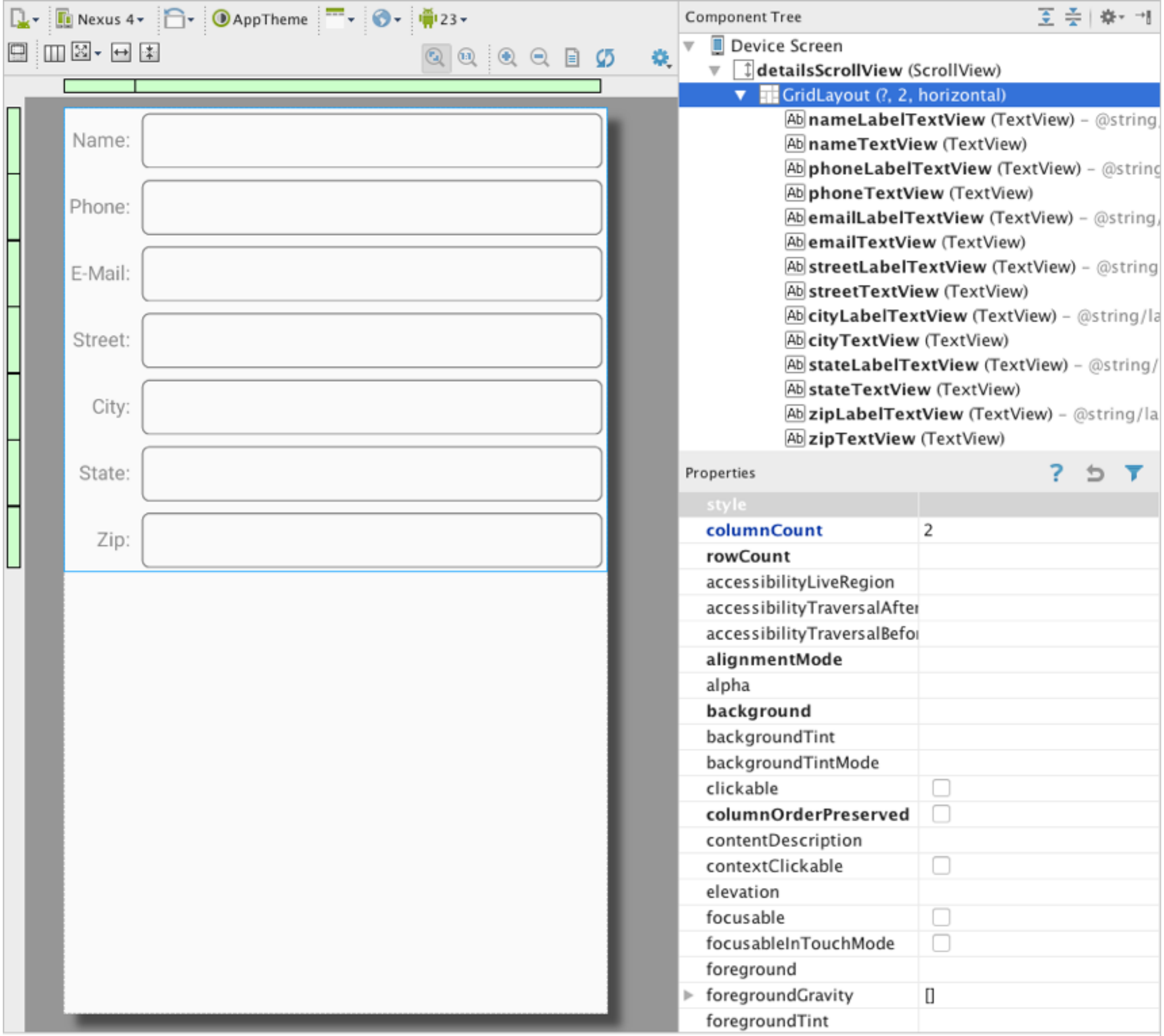
File name: fragment_detail

Root element: ScrollView

Source set: main

Directory name: layout

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

</ScrollView>
```



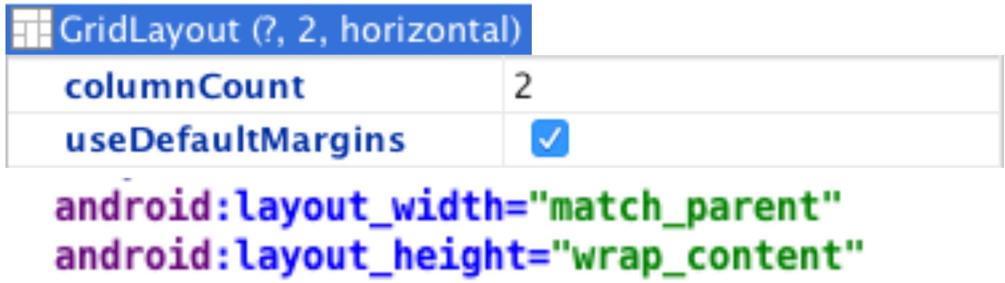
Component Tree

- Device Screen
  - detailsScrollView (ScrollView)
    - GridLayout (?, 2, horizontal)
      - nameLabelTextView (TextView) – @string
      - nameTextView (TextView)
      - phoneLabelTextView (TextView) – @string
      - phoneTextView (TextView)
      - emailLabelTextView (TextView) – @string/
      - emailTextView (TextView)
      - streetLabelTextView (TextView) – @string
      - streetTextView (TextView)
      - cityLabelTextView (TextView) – @string/la
      - cityTextView (TextView)
      - stateLabelTextView (TextView) – @string/
      - stateTextView (TextView)
      - zipLabelTextView (TextView) – @string/la
      - zipTextView (TextView)

Properties

style	
columnCount	2
rowCount	
accessibilityLiveRegion	
accessibilityTraversalAfter	
accessibilityTraversalBefo	
alignmentMode	
alpha	
background	
backgroundTint	
backgroundTintMode	
clickable	☐
columnOrderPreserved	☐
contentDescription	
contextClickable	☐
elevation	
focusable	☐
focusableInTouchMode	☐
foreground	
foregroundGravity	[]
foregroundTint	

fragment_detail.xml



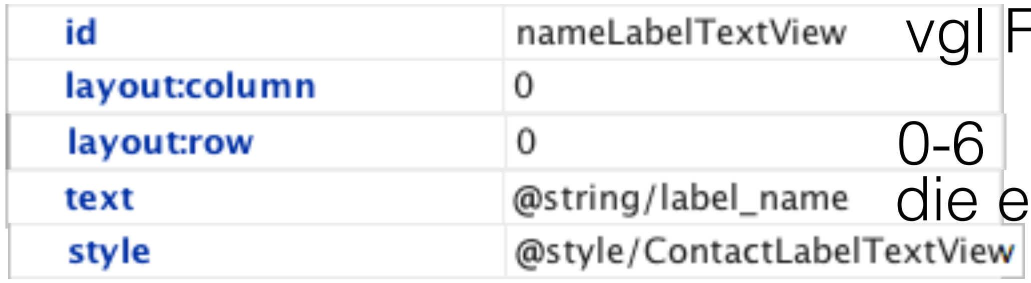
GridLayout (?, 2, horizontal)

columnCount	2
useDefaultMargins	☑

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
```

The layout:height value enables the parent ScrollView to determine the GridLayout’s actual height and decide whether to provide scrolling

TextViews der linken Spalte



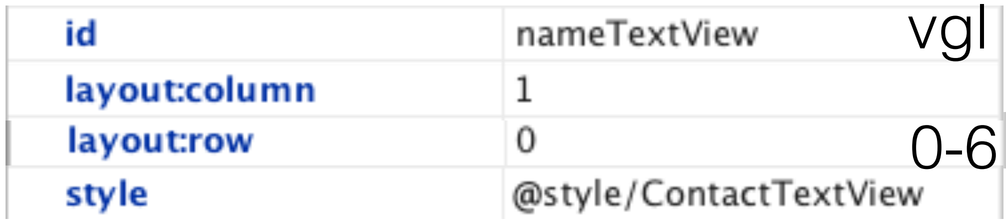
id	nameLabelTextView
layout:column	0
layout:row	0
text	@string/label_name
style	@style/ContactLabelTextView

vgl Folie 37

0-6

die entsprechende Textressource

TextViews der rechten Spalte

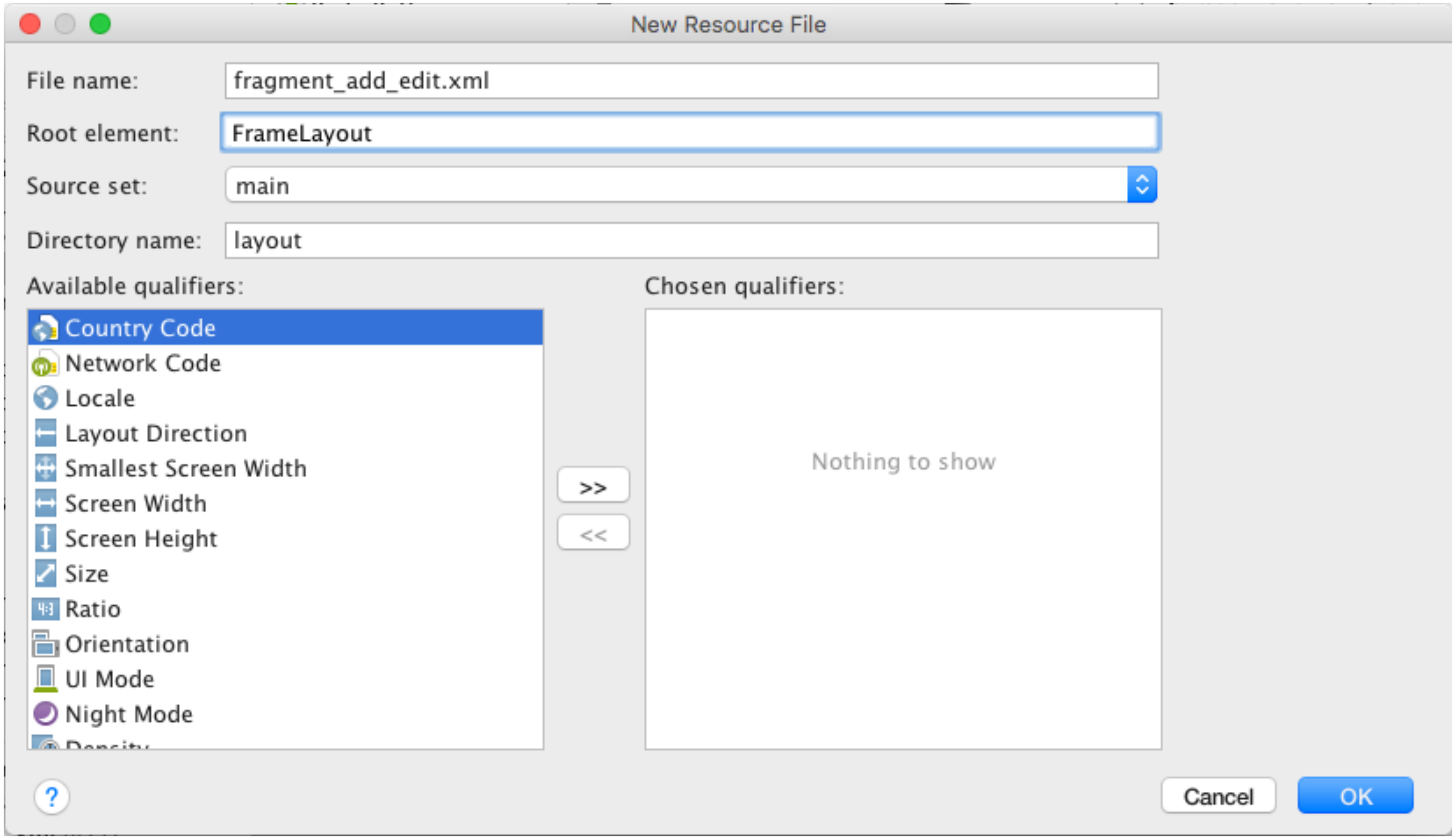


id	nameTextView
layout:column	1
layout:row	0
style	@style/ContactTextView

vgl Folie 37

0-6

AddEditFragment layout



New Resource File

File name: fragment_add_edit.xml

Root element: FrameLayout

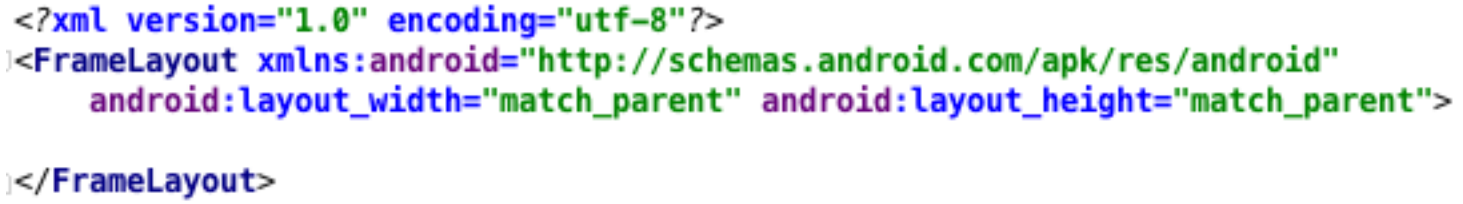
Source set: main

Directory name: layout

Available qualifiers:

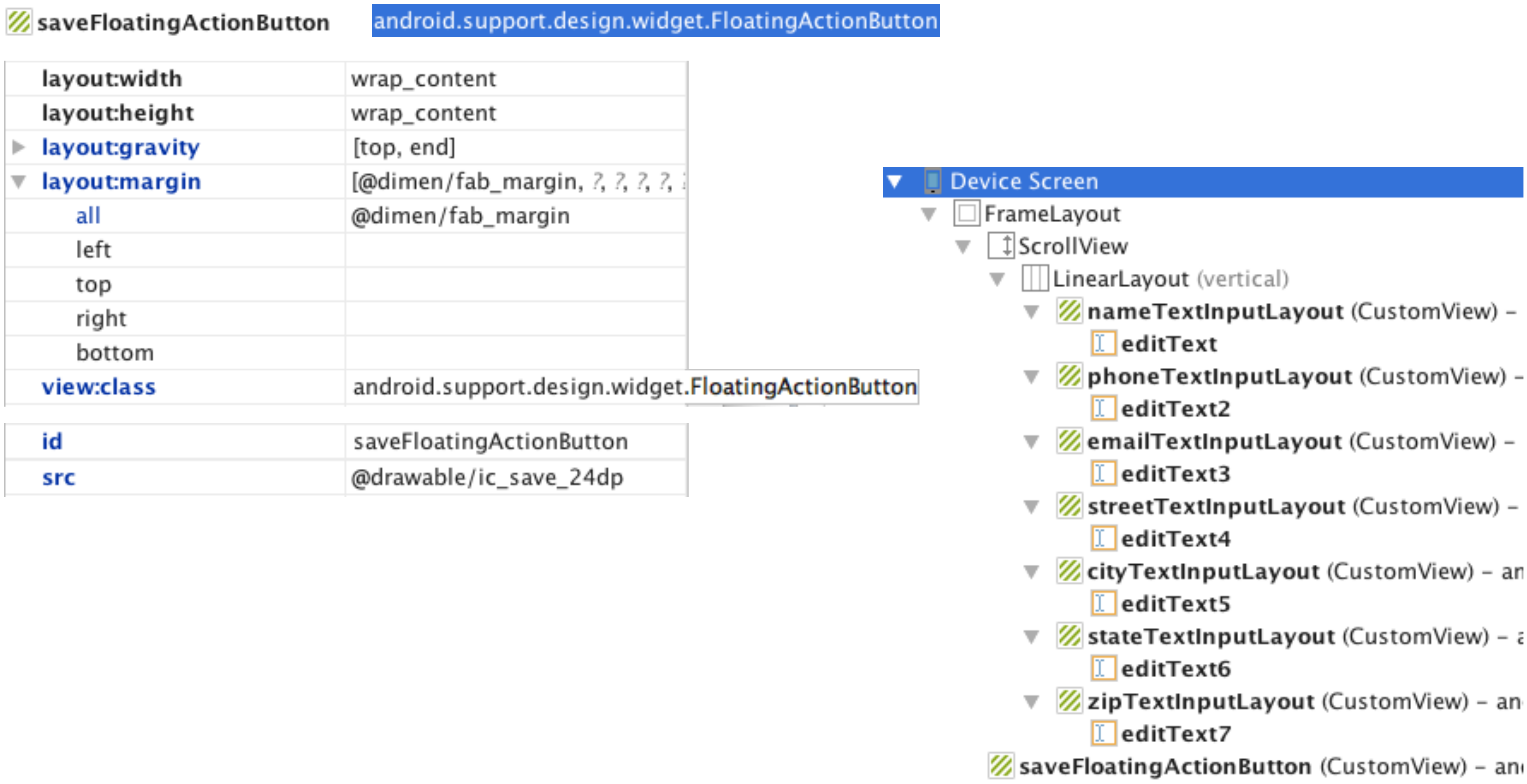
Chosen qualifiers:

Nothing to show



```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent">

</FrameLayout>
```

saveFloatingActionButton



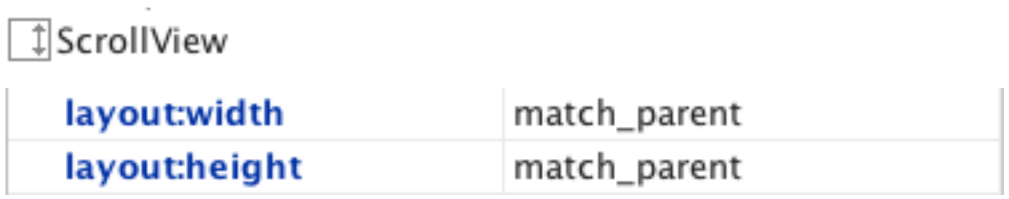
saveFloatingActionButton android.support.design.widget.FloatingActionButton

layout:width	wrap_content
layout:height	wrap_content
▶ layout:gravity	[top, end]
▼ layout:margin	[@dimen/fab_margin, ?, ?, ?, ?,
all	@dimen/fab_margin
left	
top	
right	
bottom	
view:class	android.support.design.widget.FloatingActionButton

id	saveFloatingActionButton
src	@drawable/ic_save_24dp

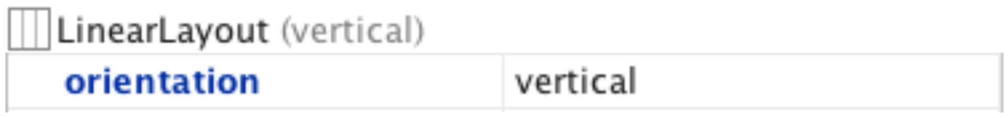
```
▼ Device Screen
  ▼ FrameLayout
    ▼ ScrollView
      ▼ LinearLayout (vertical)
        ▼ nameTextInputLayout (CustomView) –
            editText
        ▼ phoneTextInputLayout (CustomView) –
            editText2
        ▼ emailTextInputLayout (CustomView) –
            editText3
        ▼ streetTextInputLayout (CustomView) –
            editText4
        ▼ cityTextInputLayout (CustomView) – ar
            editText5
        ▼ stateTextInputLayout (CustomView) – a
            editText6
        ▼ zipTextInputLayout (CustomView) – an
            editText7
    saveFloatingActionButton (CustomView) – an
```

fragment_add_edit.xml



ScrollView

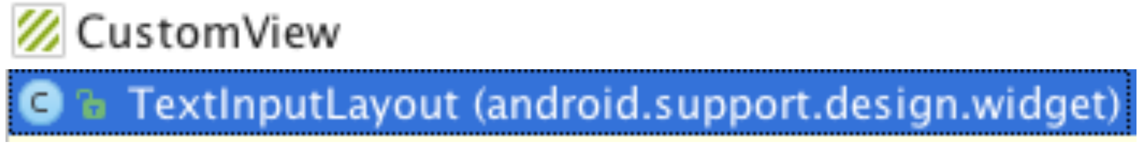
layout:width	match_parent
layout:height	match_parent



LinearLayout (vertical)

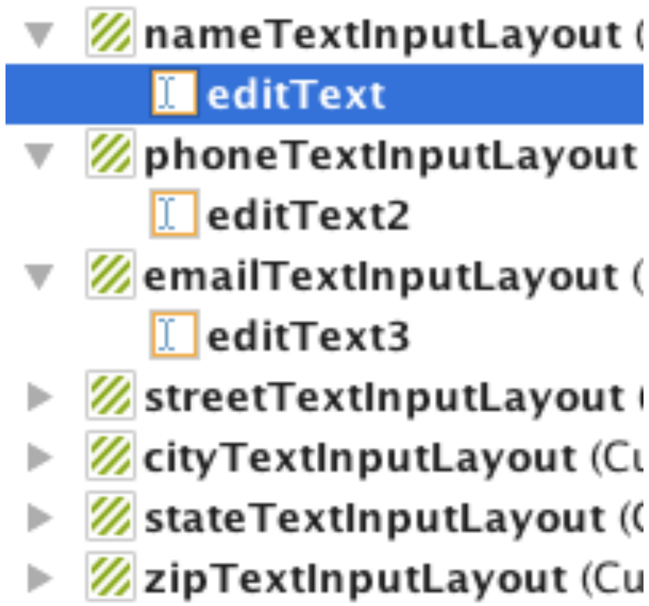
orientation	vertical

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
```

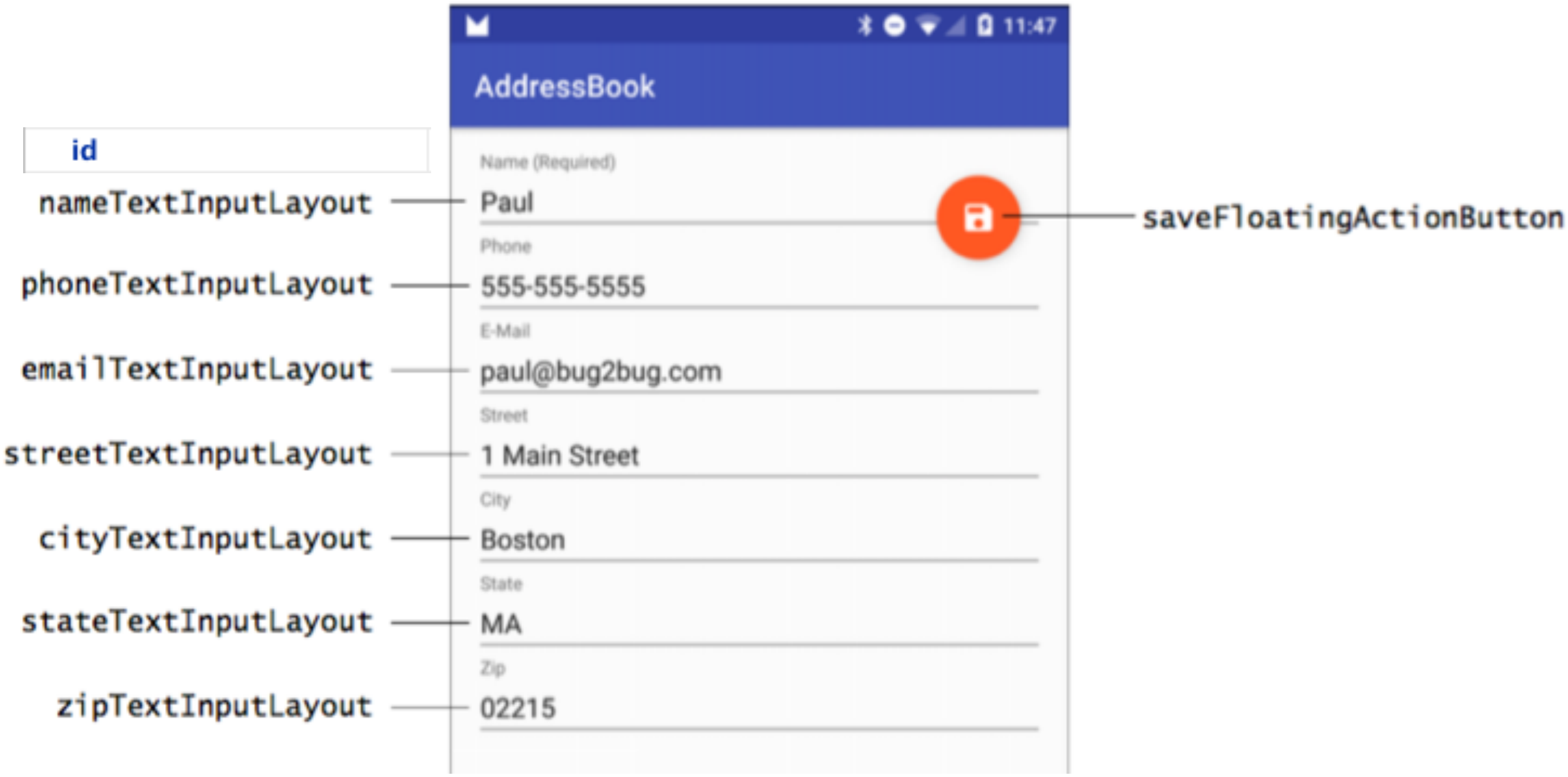


CustomView

TextInputLayout (android.support.design.widget)



```
▼ nameTextInputLayout (
      editText
▼ phoneTextInputLayout
      editText2
▼ emailTextInputLayout (
      editText3
▶ streetTextInputLayout (
▶ cityTextInputLayout (Cu
▶ stateTextInputLayout ((
▶ zipTextInputLayout (Cu
```



Name (Required)
Paul

Phone
555-555-5555

E-Mail
paul@bug2bug.com

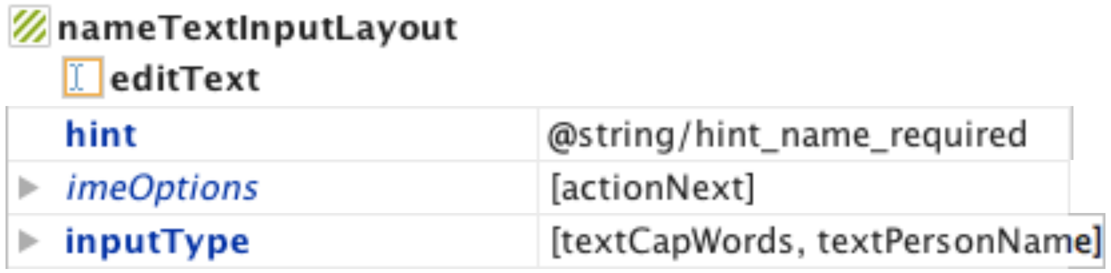
Street
1 Main Street

City
Boston

State
MA

Zip
02215

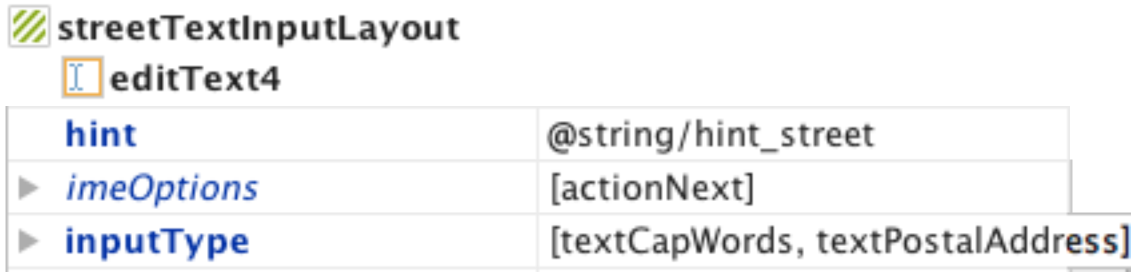
editText - editText7



nameTextInputLayout

editText

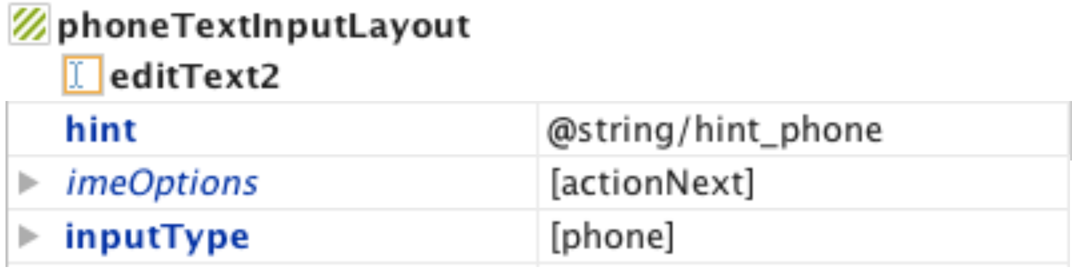
hint	@string/hint_name_required
imeOptions	[actionNext]
inputType	[textCapWords, textPersonName]



streetTextInputLayout

editText4

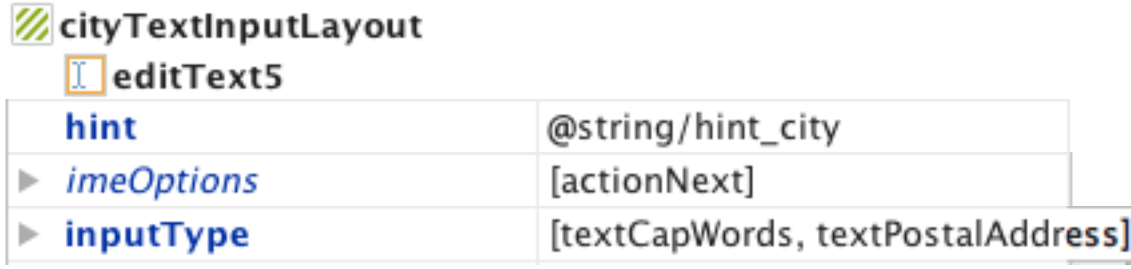
hint	@string/hint_street
imeOptions	[actionNext]
inputType	[textCapWords, textPostalAddress]



phoneTextInputLayout

editText2

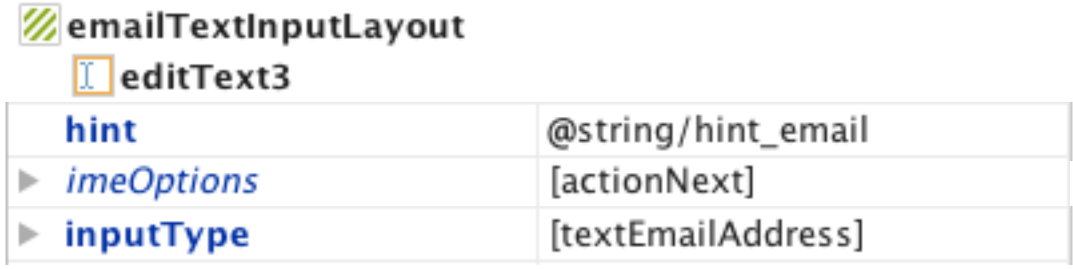
hint	@string/hint_phone
imeOptions	[actionNext]
inputType	[phone]



cityTextInputLayout

editText5

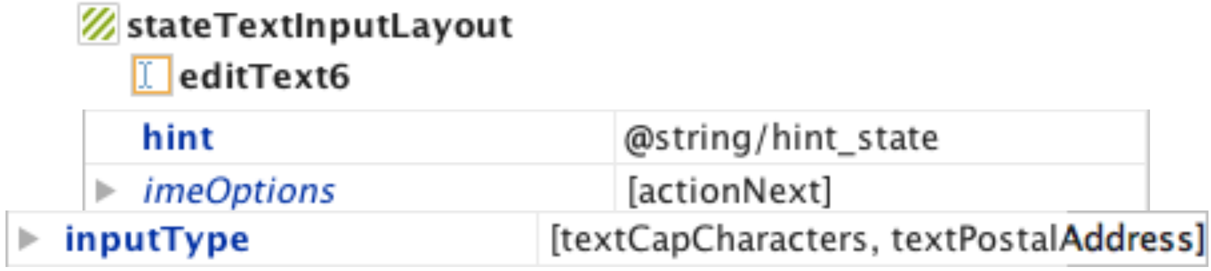
hint	@string/hint_city
imeOptions	[actionNext]
inputType	[textCapWords, textPostalAddress]



emailTextInputLayout

editText3

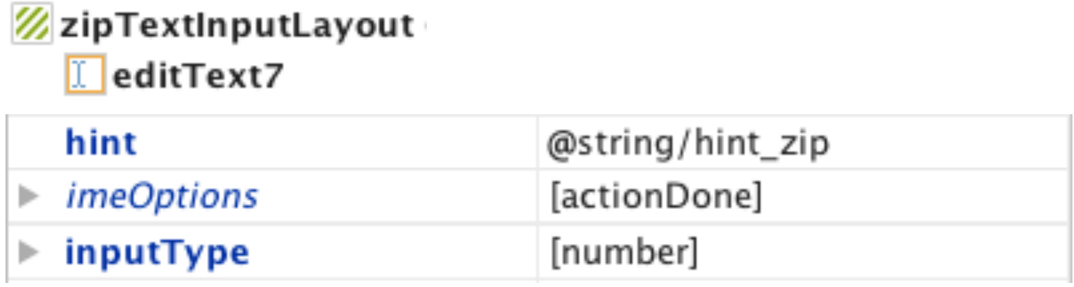
hint	@string/hint_email
imeOptions	[actionNext]
inputType	[textEmailAddress]



stateTextInputLayout

editText6

hint	@string/hint_state
imeOptions	[actionNext]
inputType	[textCapCharacters, textPostalAddress]

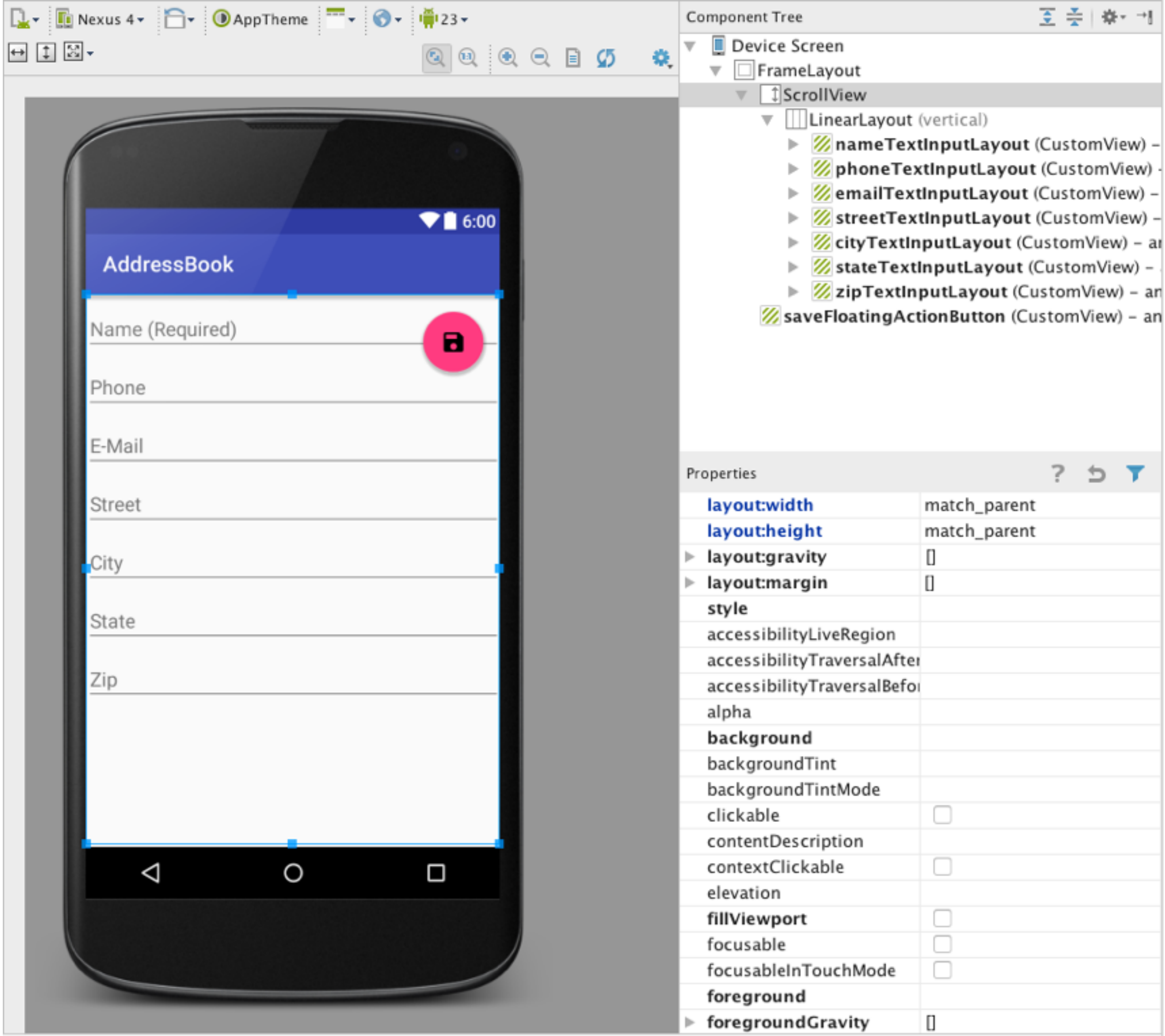


zipTextInputLayout

editText7

hint	@string/hint_zip
imeOptions	[actionDone]
inputType	[number]

input method editor (IME) … der Button rechts unten auf der Soft-Tastatur



Component Tree

- Device Screen
  - FrameLayout
    - ScrollView
      - LinearLayout (vertical)
        - nameTextInputLayout (CustomView) –
        - phoneTextInputLayout (CustomView) -
        - emailTextInputLayout (CustomView) –
        - streetTextInputLayout (CustomView) –
        - cityTextInputLayout (CustomView) – ar
        - stateTextInputLayout (CustomView) –
        - zipTextInputLayout (CustomView) – an
    - saveFloatingActionButton (CustomView) – an

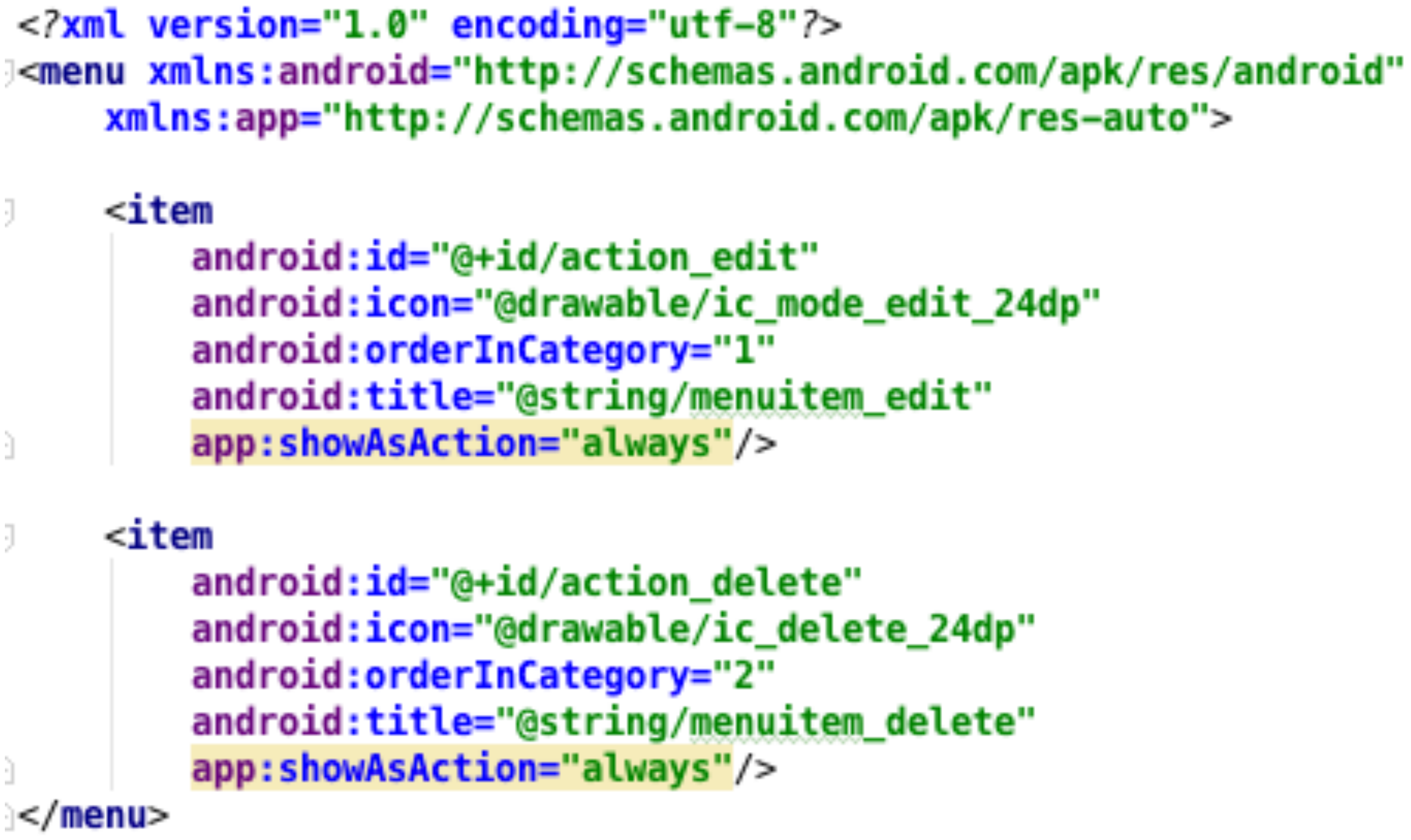
Properties

layout:width	match_parent
layout:height	match_parent
layout:gravity	[]
layout:margin	[]
style	
accessibilityLiveRegion	
accessibilityTraversalAfter	
accessibilityTraversalBefo	
alpha	
background	
backgroundTint	
backgroundTintMode	
clickable	□
contentDescription	
contextClickable	□
elevation	
fillViewport	□
focusable	□
focusableInTouchMode	□
foreground	
foregroundGravity	[]

DetailFragment Menu

- Lösche aus MainActivity: onCreateOptionsMenu- und onOptionsItemSelected-Methoden. In der Main Activity wird kein Menü benötigt

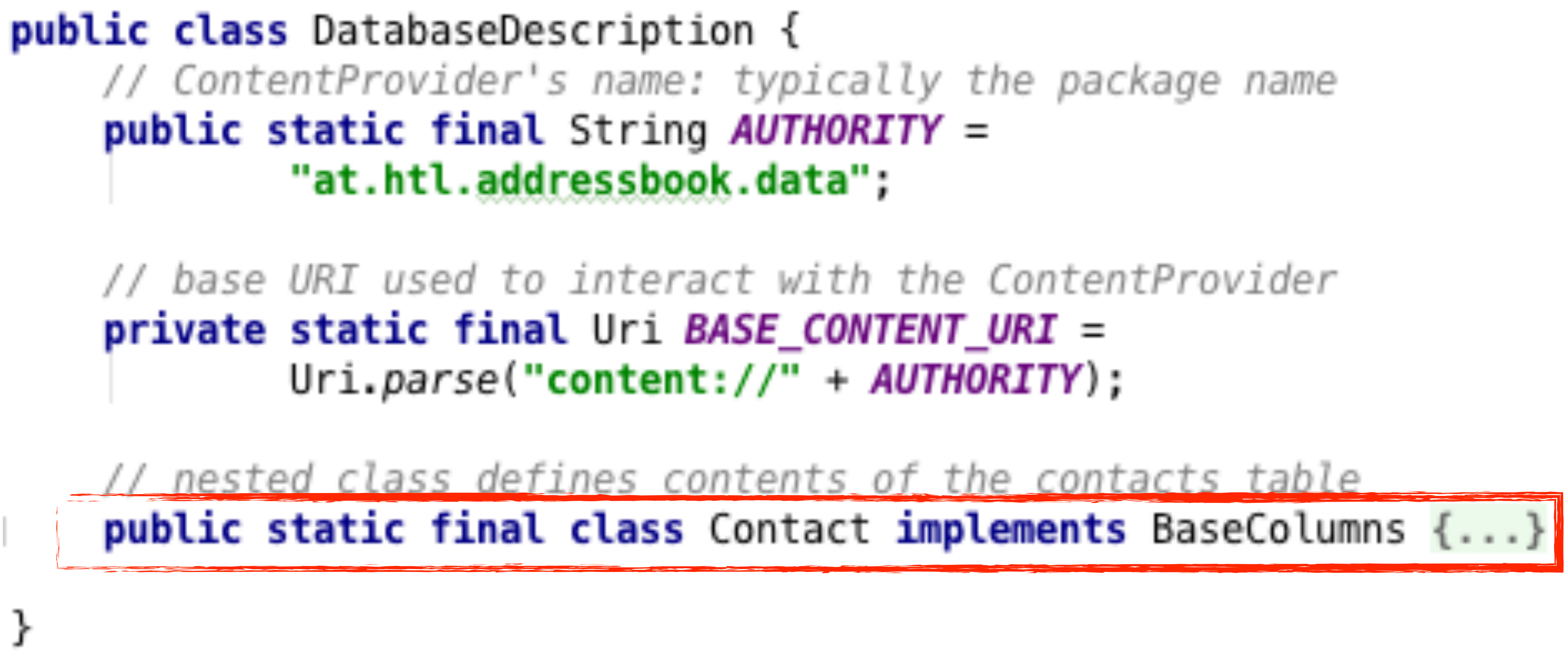
- Benenne menu_main.xml in fragment_details_menu.xml um



```xml
<?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">

    <item
        android:id="@+id/action_edit"
        android:icon="@drawable/ic_mode_edit_24dp"
        android:orderInCategory="1"
        android:title="@string/menuitem_edit"
        app:showAsAction="always"/>

    <item
        android:id="@+id/action_delete"
        android:icon="@drawable/ic_delete_24dp"
        android:orderInCategory="2"
        android:title="@string/menuitem_delete"
        app:showAsAction="always"/>
</menu>
```

Database coden

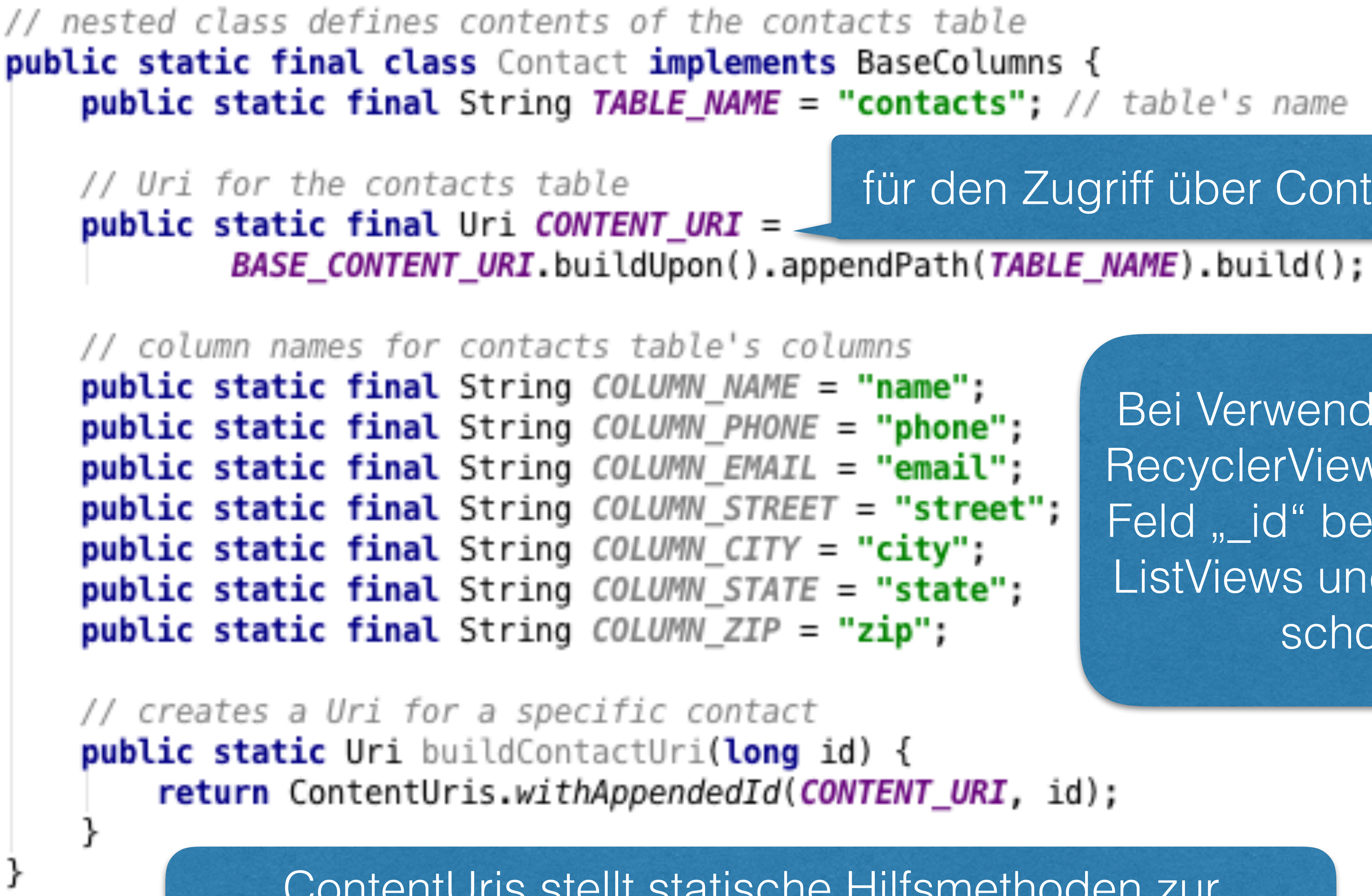
DatabaseDescription.java



```
public class DatabaseDescription {
    // ContentProvider's name: typically the package name
    public static final String AUTHORITY =
            "at.htl.addressbook.data";

    // base URI used to interact with the ContentProvider
    private static final Uri BASE_CONTENT_URI =
            Uri.parse("content://" + AUTHORITY);

    // nested class defines contents of the contacts table
    public static final class Contact implements BaseColumns {...}

}
```

Contact

durch implementieren von BaseColumns wird automatisch eine PK-Spalte _ID erstellt
-> Konstante _ID mit Wert „_id“



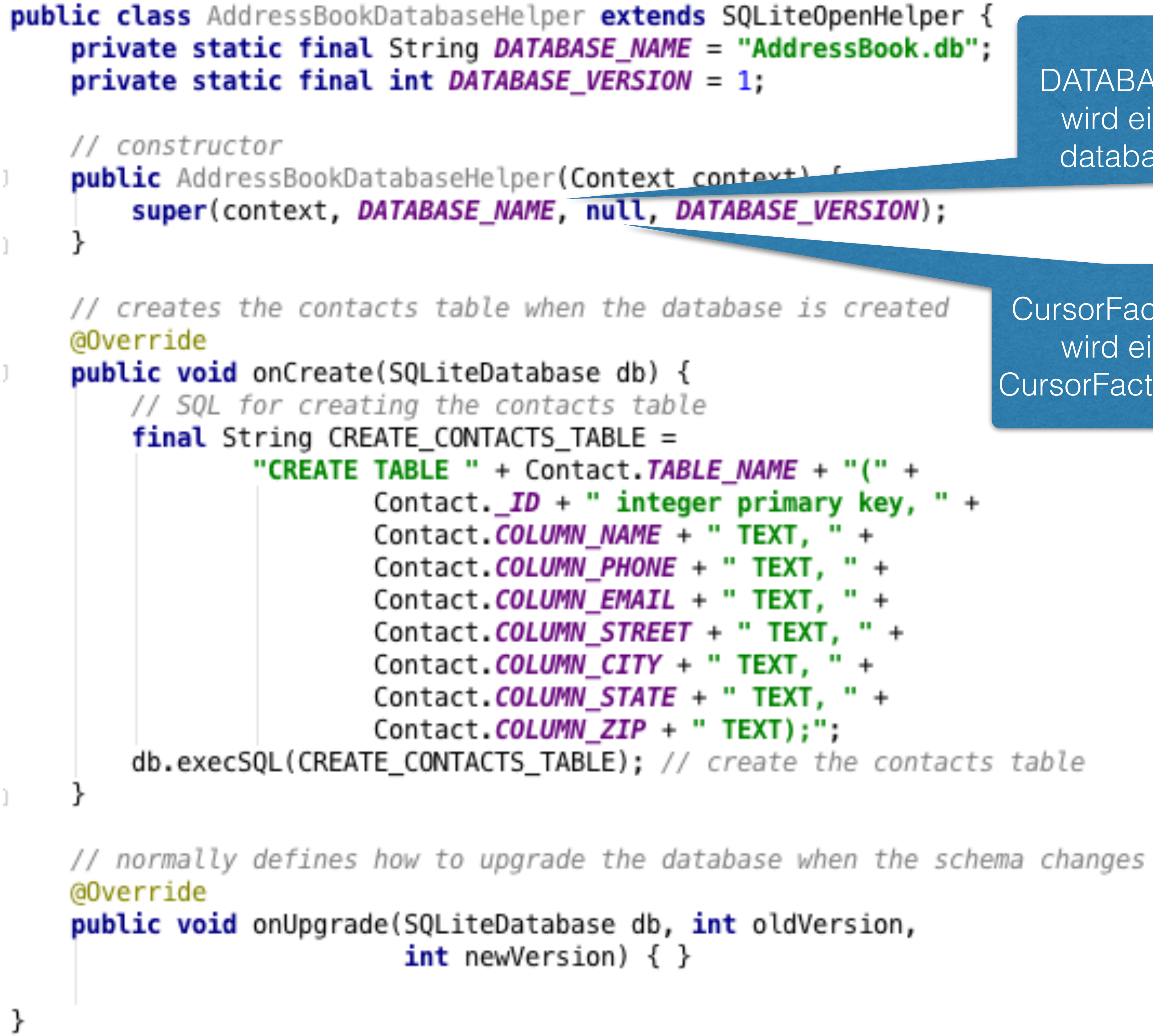
```
// nested class defines contents of the contacts table
public static final class Contact implements BaseColumns {
    public static final String TABLE_NAME = "contacts"; // table's name

    // Uri for the contacts table
    public static final Uri CONTENT_URI =
            BASE_CONTENT_URI.buildUpon().appendPath(TABLE_NAME).build();

    // column names for contacts table's columns
    public static final String COLUMN_NAME = "name";
    public static final String COLUMN_PHONE = "phone";
    public static final String COLUMN_EMAIL = "email";
    public static final String COLUMN_STREET = "street";
    public static final String COLUMN_CITY = "city";
    public static final String COLUMN_STATE = "state";
    public static final String COLUMN_ZIP = "zip";

    // creates a Uri for a specific contact
    public static Uri buildContactUri(long id) {
        return ContentUris.withAppendedId(CONTENT_URI, id);
    }
}
```

für den Zugriff über ContentProvider

Bei Verwendung einer RecyclerView wird kein Feld „_id“ benötigt, bei ListViews und Cursors schon

ContentUris stellt statische Hilfsmethoden zur Verfügung, um „content://„-Uris zu ändern. Hier wird ein / und eine Id hinten angefügt

AddressBookDatabaseHelper



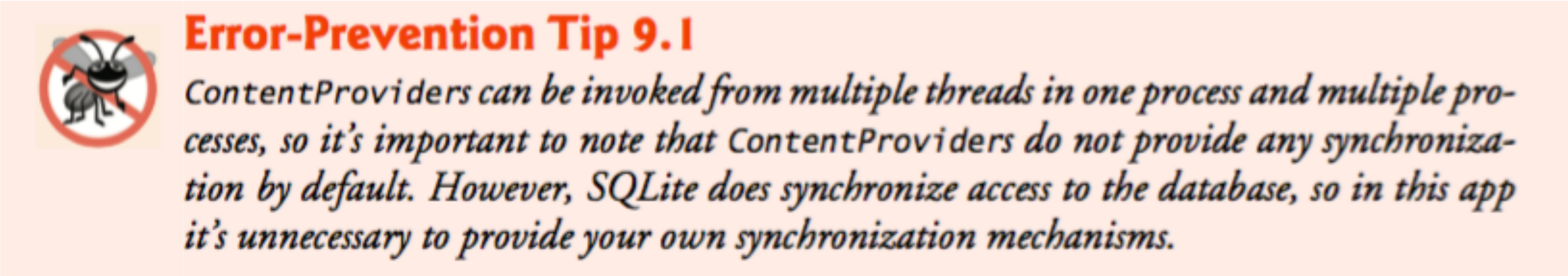
```java
public class AddressBookDatabaseHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "AddressBook.db";
    private static final int DATABASE_VERSION = 1;

    // constructor
    public AddressBookDatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    // creates the contacts table when the database is created
    @Override
    public void onCreate(SQLiteDatabase db) {
        // SQL for creating the contacts table
        final String CREATE_CONTACTS_TABLE =
                "CREATE TABLE " + Contact.TABLE_NAME + "(" +
                        Contact._ID + " integer primary key, " +
                        Contact.COLUMN_NAME + " TEXT, " +
                        Contact.COLUMN_PHONE + " TEXT, " +
                        Contact.COLUMN_EMAIL + " TEXT, " +
                        Contact.COLUMN_STREET + " TEXT, " +
                        Contact.COLUMN_CITY + " TEXT, " +
                        Contact.COLUMN_STATE + " TEXT, " +
                        Contact.COLUMN_ZIP + " TEXT);";
        db.execSQL(CREATE_CONTACTS_TABLE); // create the contacts table
    }

    // normally defines how to upgrade the database when the schema changes
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion,
                          int newVersion) { }
}
```

Ist der DATABASE_NAME null, wird eine in-memory database verwendet

CursorFactory - bei null wird eine SQLite CursorFactory verwendet

AddressBookContentProvider

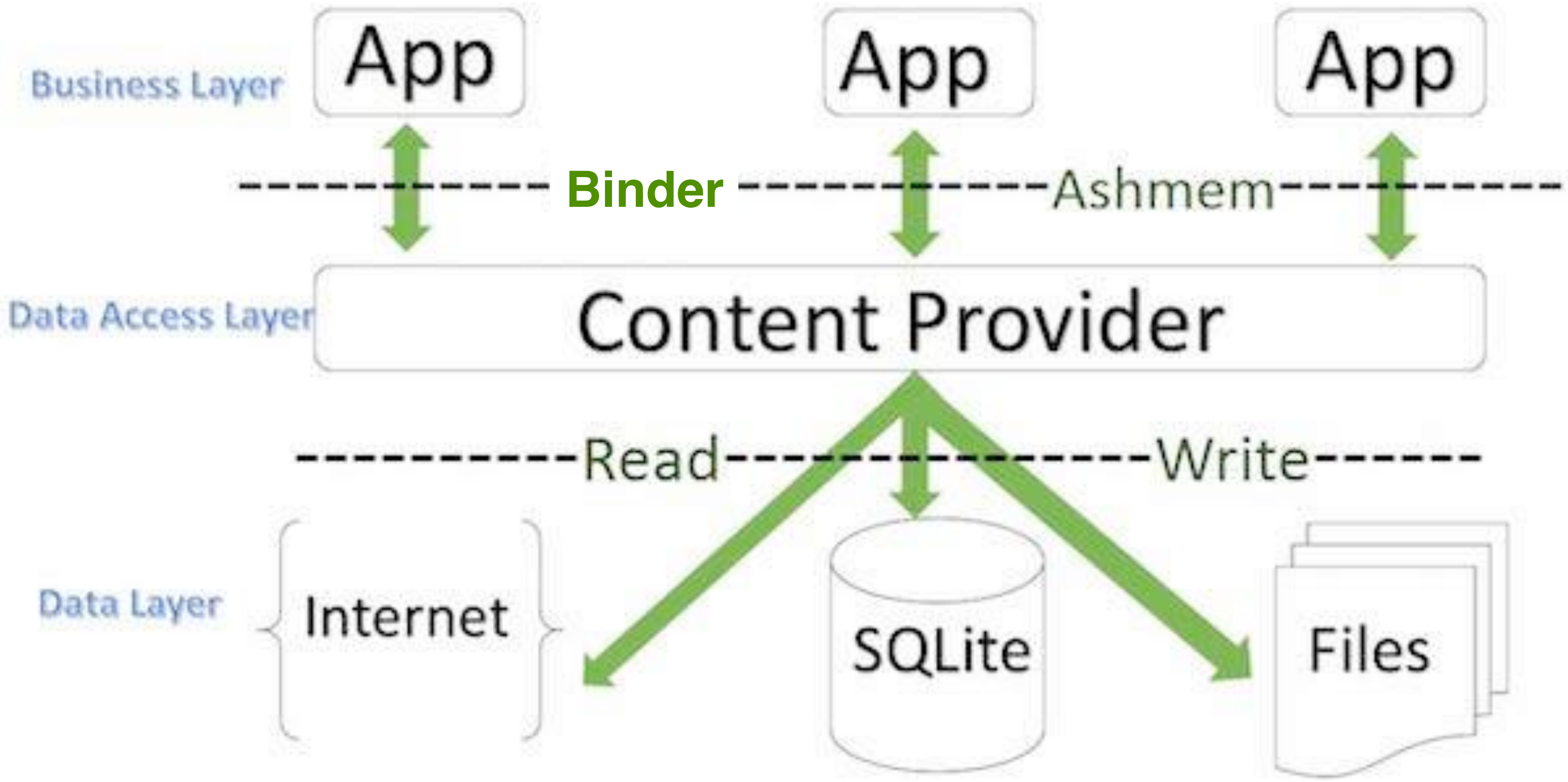
- The AddressBookContentProvider subclass of ContentProvider defines how to perform query, insert, update and delete operations on this app's database.



Error-Prevention Tip 9.1

`ContentProviders` can be invoked from multiple threads in one process and multiple processes, so it's important to note that `ContentProviders` do not provide any synchronization by default. However, SQLite does synchronize access to the database, so in this app it's unnecessary to provide your own synchronization mechanisms.

ContentProvider

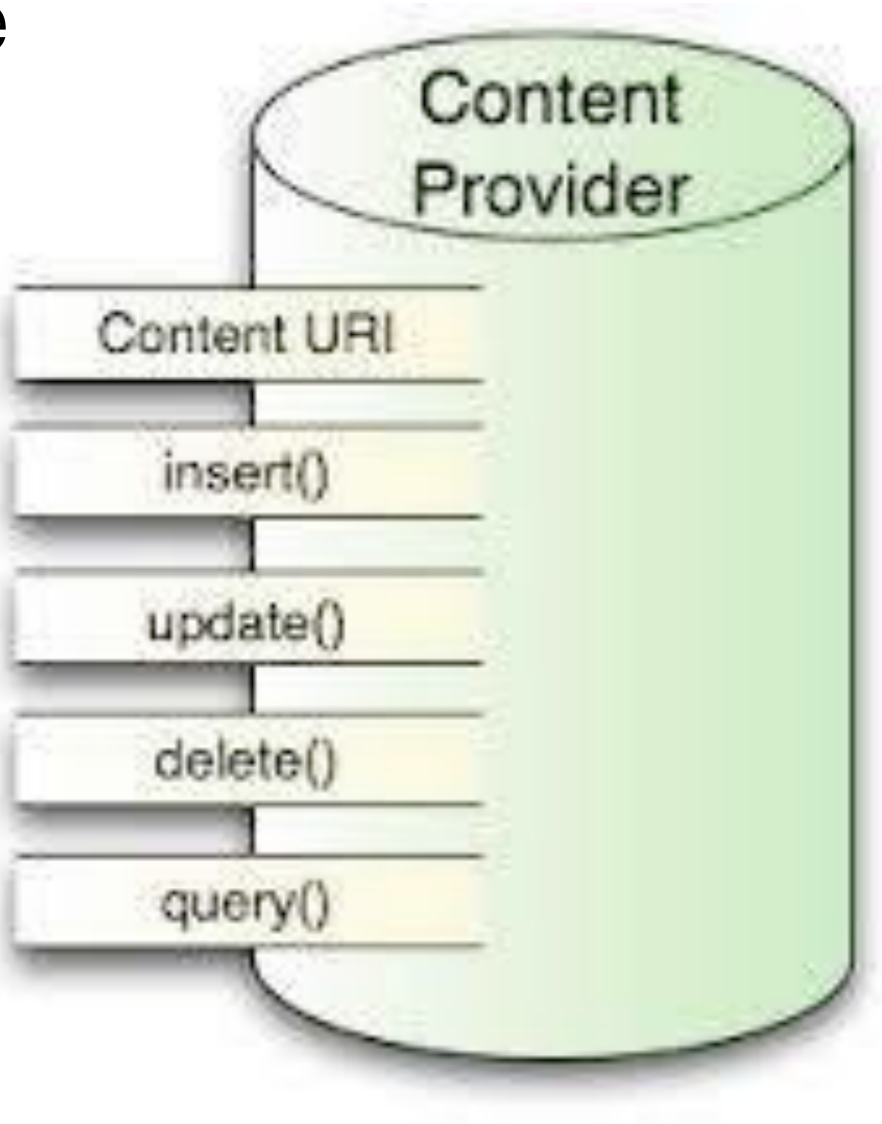


http://www.tutorialspoint.com/android/android_content_providers.htm

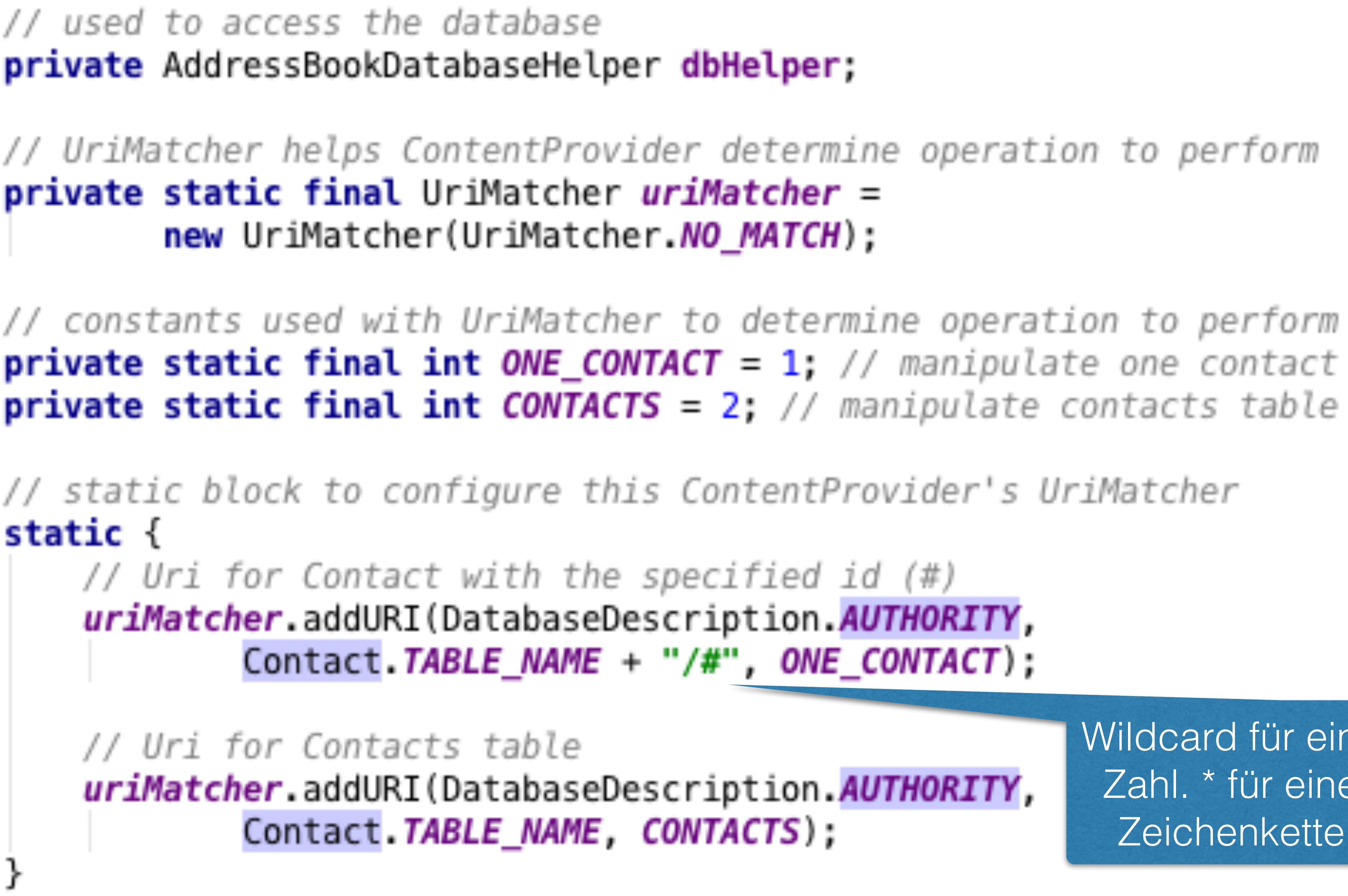
AddressBookContentProvider

- authority … This specifies the name of the content provider, for example contacts, browser etc. For third-party content providers, this could be the fully qualified name, such as at.htl.addressbook.data

- Ein ContentProvider verwendet einen UriMatcher, um zu bestimmen, wie die query-, insert-, update- und delete-Methoden implementiert sind



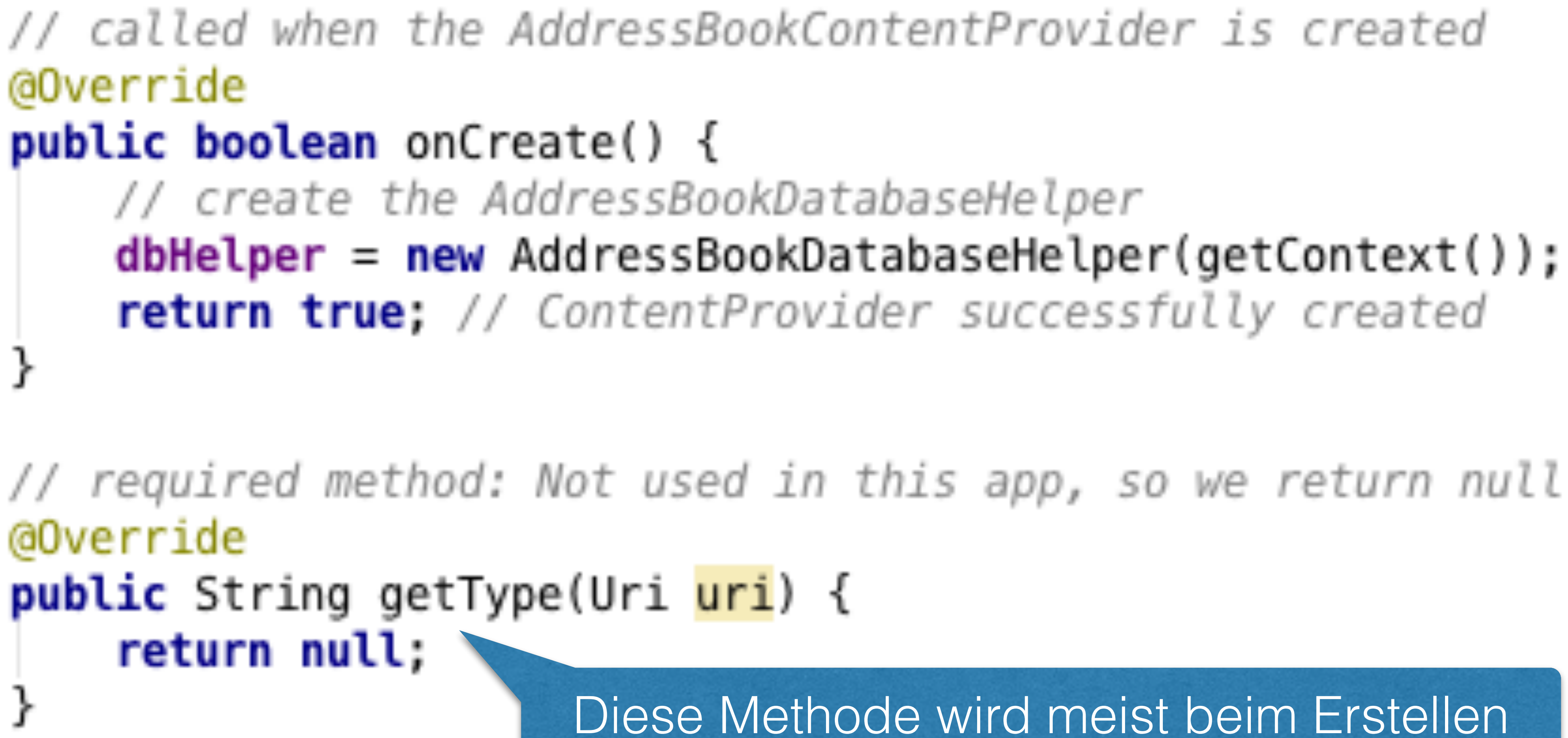
AddressBookContentProvider



```java
// used to access the database
private AddressBookDatabaseHelper dbHelper;

// UriMatcher helps ContentProvider determine operation to perform
private static final UriMatcher uriMatcher =
        new UriMatcher(UriMatcher.NO_MATCH);

// constants used with UriMatcher to determine operation to perform
private static final int ONE_CONTACT = 1; // manipulate one contact
private static final int CONTACTS = 2; // manipulate contacts table

// static block to configure this ContentProvider's UriMatcher
static {
    // Uri for Contact with the specified id (#)
    uriMatcher.addURI(DatabaseDescription.AUTHORITY,
            Contact.TABLE_NAME + "/#", ONE_CONTACT);

    // Uri for Contacts table
    uriMatcher.addURI(DatabaseDescription.AUTHORITY,
            Contact.TABLE_NAME, CONTACTS);
}
```

Wildcard für eine Zahl. * für eine Zeichenkette

onCreate(), getType()



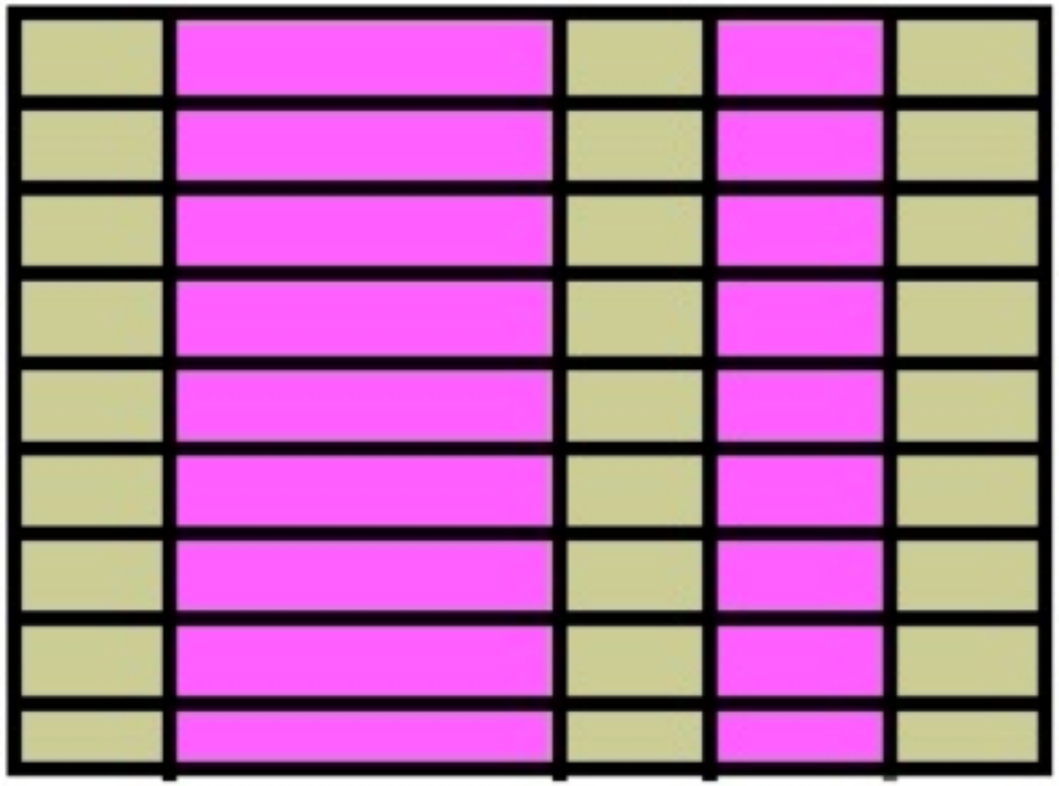
```java
// called when the AddressBookContentProvider is created
@Override
public boolean onCreate() {
    // create the AddressBookDatabaseHelper
    dbHelper = new AddressBookDatabaseHelper(getContext());
    return true; // ContentProvider successfully created
}

// required method: Not used in this app, so we return null
@Override
public String getType(Uri uri) {
    return null;
}
```

Diese Methode wird meist beim Erstellen und Starten von Intents mit unterschiedlichen MIME-Typen verwendet

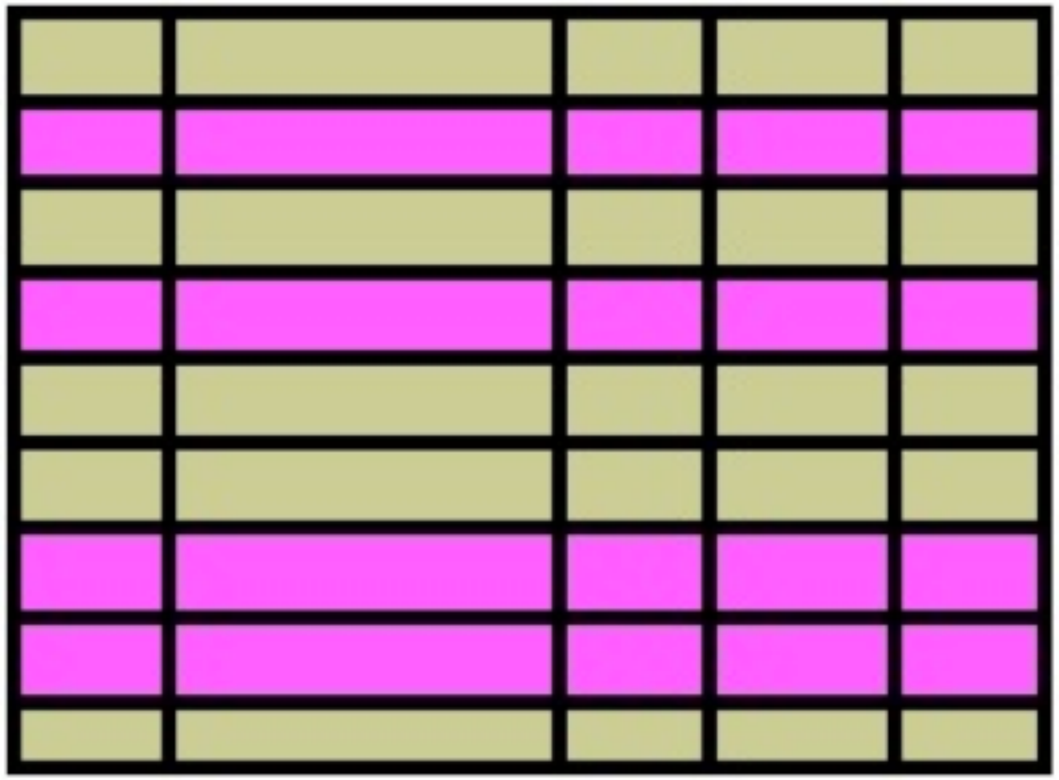
MIME … Multipurpose Internet Mail Extension

http://www.freeformatter.com/mime-types-list.html

Projection



Selection



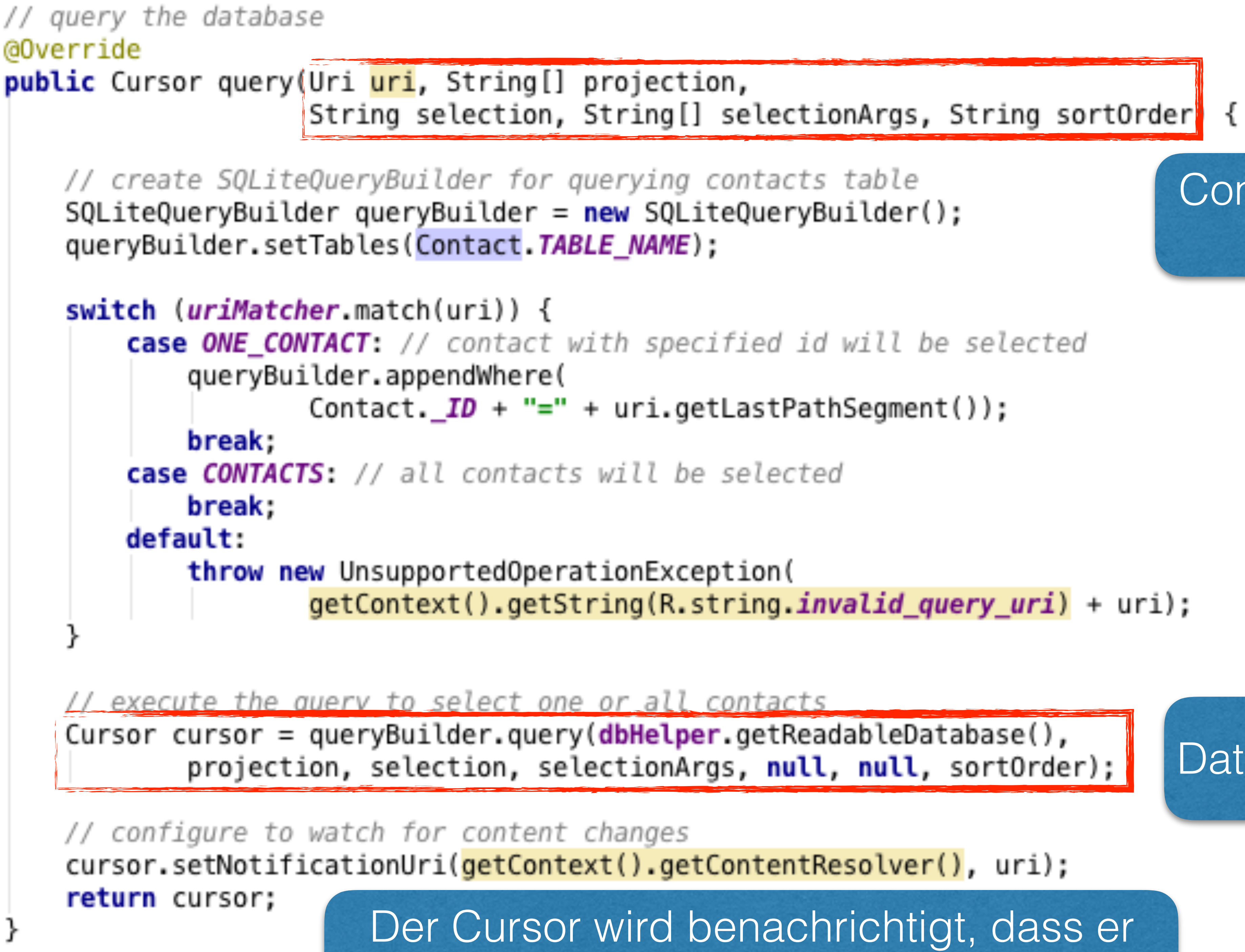
Umsetzung in SQL ?

SELECT

WHERE-Klausel

query()

Die Uri repräsentiert die erhaltenen Daten



```java
// query the database
@Override
public Cursor query(Uri uri, String[] projection,
                    String selection, String[] selectionArgs, String sortOrder) {

    // create SQLiteQueryBuilder for querying contacts table
    SQLiteQueryBuilder queryBuilder = new SQLiteQueryBuilder();
    queryBuilder.setTables(Contact.TABLE_NAME);

    switch (uriMatcher.match(uri)) {
        case ONE_CONTACT: // contact with specified id will be selected
            queryBuilder.appendWhere(
                    Contact._ID + "=" + uri.getLastPathSegment());
            break;
        case CONTACTS: // all contacts will be selected
            break;
        default:
            throw new UnsupportedOperationException(
                    getContext().getString(R.string.invalid_query_uri) + uri);
    }

    // execute the query to select one or all contacts
    Cursor cursor = queryBuilder.query(dbHelper.getReadableDatabase(),
            projection, selection, selectionArgs, null, null, sortOrder);

    // configure to watch for content changes
    cursor.setNotificationUri(getContext().getContentResolver(), uri);
    return cursor;
}
```

ContentProvider -query

Database-query

Der Cursor wird benachrichtigt, dass er aktualisiert werden kann

Beispiel für ContentProvider-query()

```
... .query(uri,
            new String[] {
            KEY_ROWID,
            KEY_COMPANY,
            KEY_ORDER,
            KEY_DESCRIPTION,
            KEY_QUANTITY,
            KEY_PRICE},
            KEY_COMPANY + "=?" + " and "  +
            KEY_ORDER + "=?",
            new String[] {company,orderNumber},
            null
            );
```

projection

selection

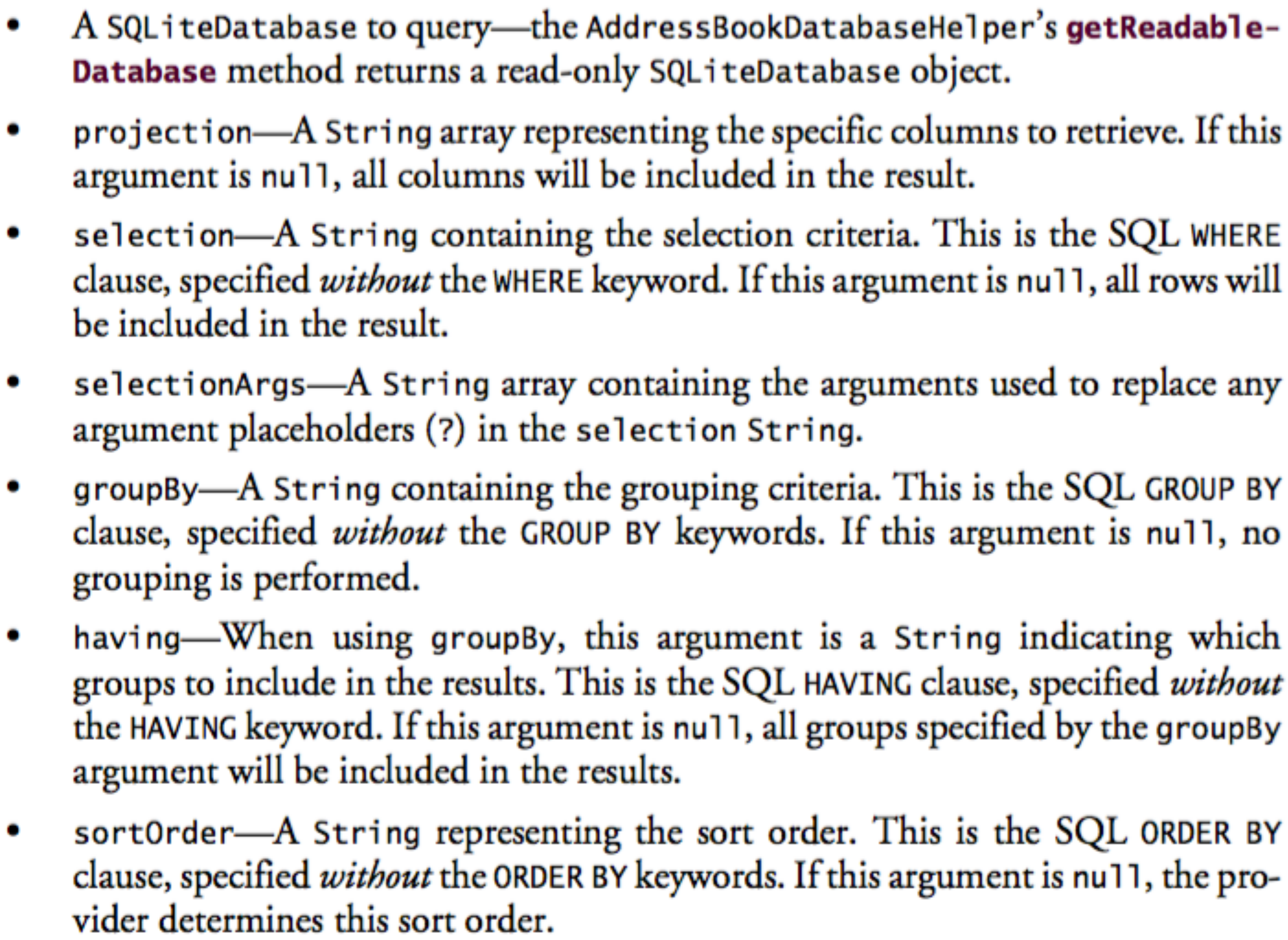
selectionArgs

sortOrder

ContentProvider query

database query

Querying the database



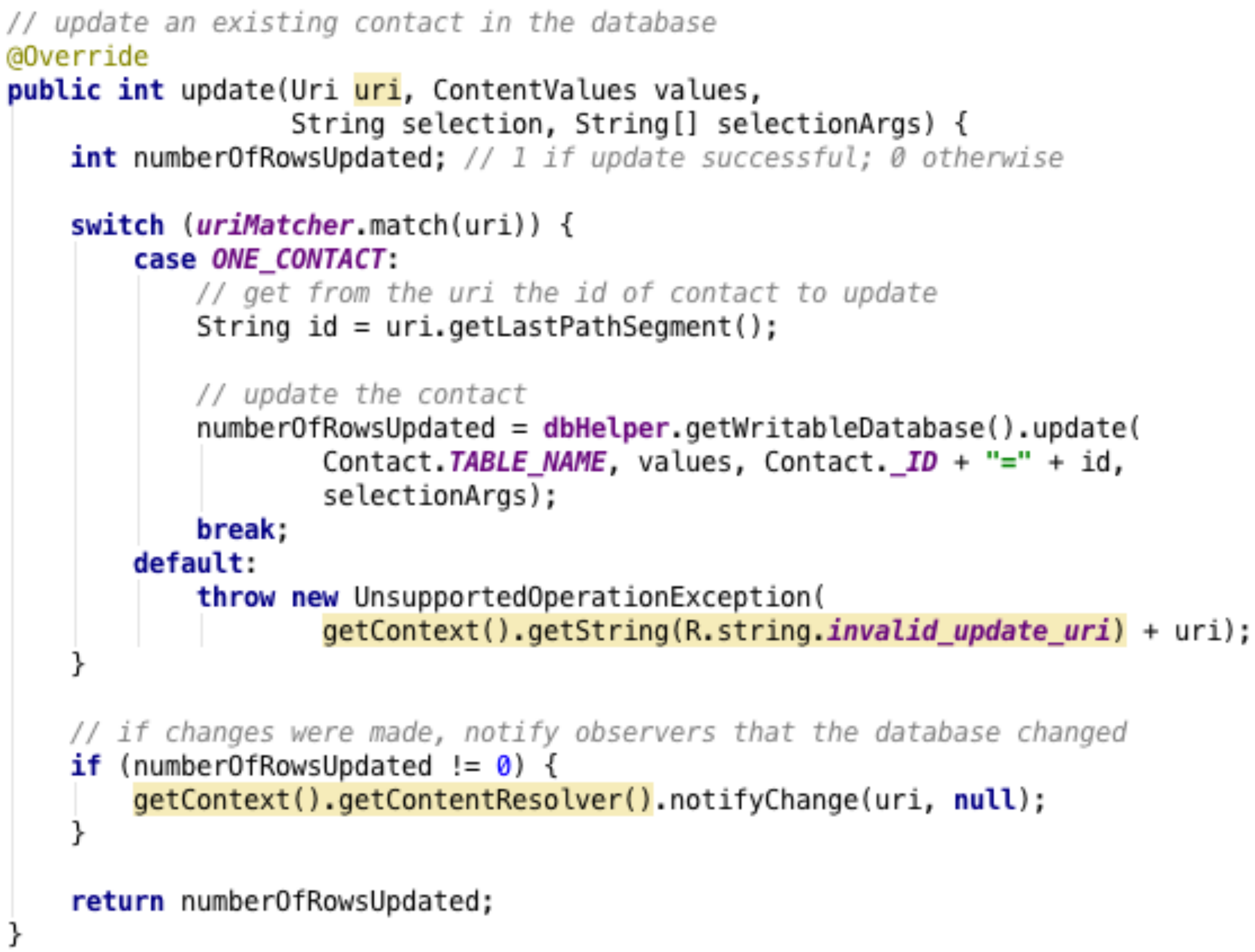
- A SQLiteDatabase to query—the AddressBookDatabaseHelper's **getReadableDatabase** method returns a read-only SQLiteDatabase object.
- projection—A String array representing the specific columns to retrieve. If this argument is null, all columns will be included in the result.
- selection—A String containing the selection criteria. This is the SQL WHERE clause, specified *without* the WHERE keyword. If this argument is null, all rows will be included in the result.
- selectionArgs—A String array containing the arguments used to replace any argument placeholders (?) in the selection String.
- groupBy—A String containing the grouping criteria. This is the SQL GROUP BY clause, specified *without* the GROUP BY keywords. If this argument is null, no grouping is performed.
- having—When using groupBy, this argument is a String indicating which groups to include in the results. This is the SQL HAVING clause, specified *without* the HAVING keyword. If this argument is null, all groups specified by the groupBy argument will be included in the results.
- sortOrder—A String representing the sort order. This is the SQL ORDER BY clause, specified *without* the ORDER BY keywords. If this argument is null, the provider determines this sort order.

insert()

uri … Tabelle, in der die Zeile eingefügt wird

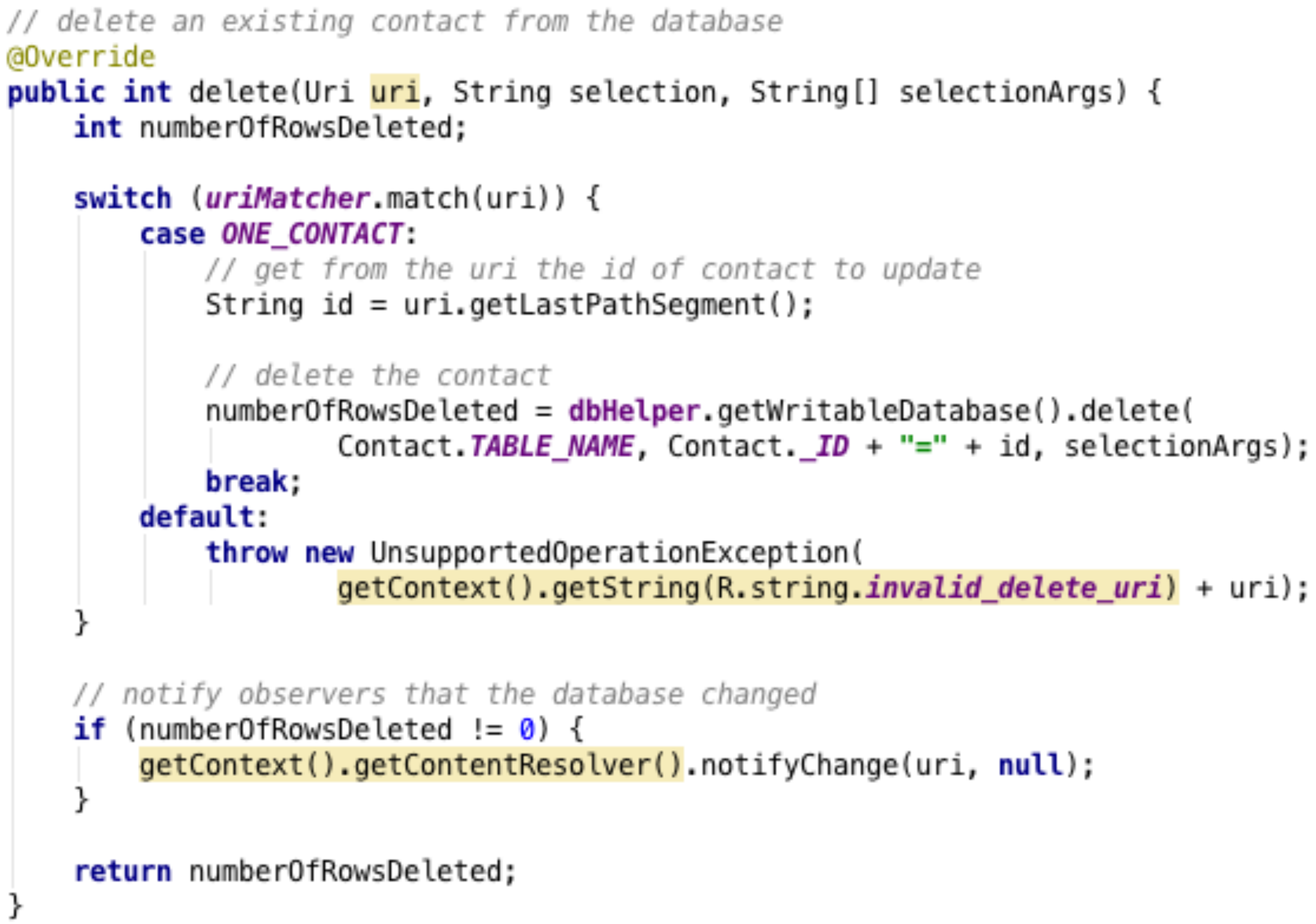
```java
// insert a new contact in the database
@Override
public Uri insert(Uri uri, ContentValues values) {
    Uri newContactUri = null;

    switch (uriMatcher.match(uri)) {
        case CONTACTS:
            // insert the new contact--success yields new contact's row id
            long rowId = dbHelper.getWritableDatabase().insert(
                    Contact.TABLE_NAME, null, values);

            // if the contact was inserted, create an appropriate Uri;
            // otherwise, throw an exception
            if (rowId > 0) { // SQLite row IDs start at 1
                newContactUri = Contact.buildContactUri(rowId);

                // notify observers that the database changed
                getContext().getContentResolver().notifyChange(uri, null);
            } else {
                throw new SQLException(
                        getContext().getString(R.string.insert_failed) + uri);
            }
            break;
        default:
            throw new UnsupportedOperationException(
                    getContext().getString(R.string.invalid_insert_uri) + uri);
    }

    return newContactUri;
}
```

update()



```
// update an existing contact in the database
@Override
public int update(Uri uri, ContentValues values,
                  String selection, String[] selectionArgs) {
    int numberOfRowsUpdated; // 1 if update successful; 0 otherwise

    switch (uriMatcher.match(uri)) {
        case ONE_CONTACT:
            // get from the uri the id of contact to update
            String id = uri.getLastPathSegment();

            // update the contact
            numberOfRowsUpdated = dbHelper.getWritableDatabase().update(
                    Contact.TABLE_NAME, values, Contact._ID + "=" + id,
                    selectionArgs);
            break;
        default:
            throw new UnsupportedOperationException(
                    getContext().getString(R.string.invalid_update_uri) + uri);
    }

    // if changes were made, notify observers that the database changed
    if (numberOfRowsUpdated != 0) {
        getContext().getContentResolver().notifyChange(uri, null);
    }

    return numberOfRowsUpdated;
}
```

delete()



```java
// delete an existing contact from the database
@Override
public int delete(Uri uri, String selection, String[] selectionArgs) {
    int numberOfRowsDeleted;

    switch (uriMatcher.match(uri)) {
        case ONE_CONTACT:
            // get from the uri the id of contact to update
            String id = uri.getLastPathSegment();

            // delete the contact
            numberOfRowsDeleted = dbHelper.getWritableDatabase().delete(
                    Contact.TABLE_NAME, Contact._ID + "=" + id, selectionArgs);
            break;
        default:
            throw new UnsupportedOperationException(
                    getContext().getString(R.string.invalid_delete_uri) + uri);
    }

    // notify observers that the database changed
    if (numberOfRowsDeleted != 0) {
        getContext().getContentResolver().notifyChange(uri, null);
    }

    return numberOfRowsDeleted;
}
```

MainActivity.java

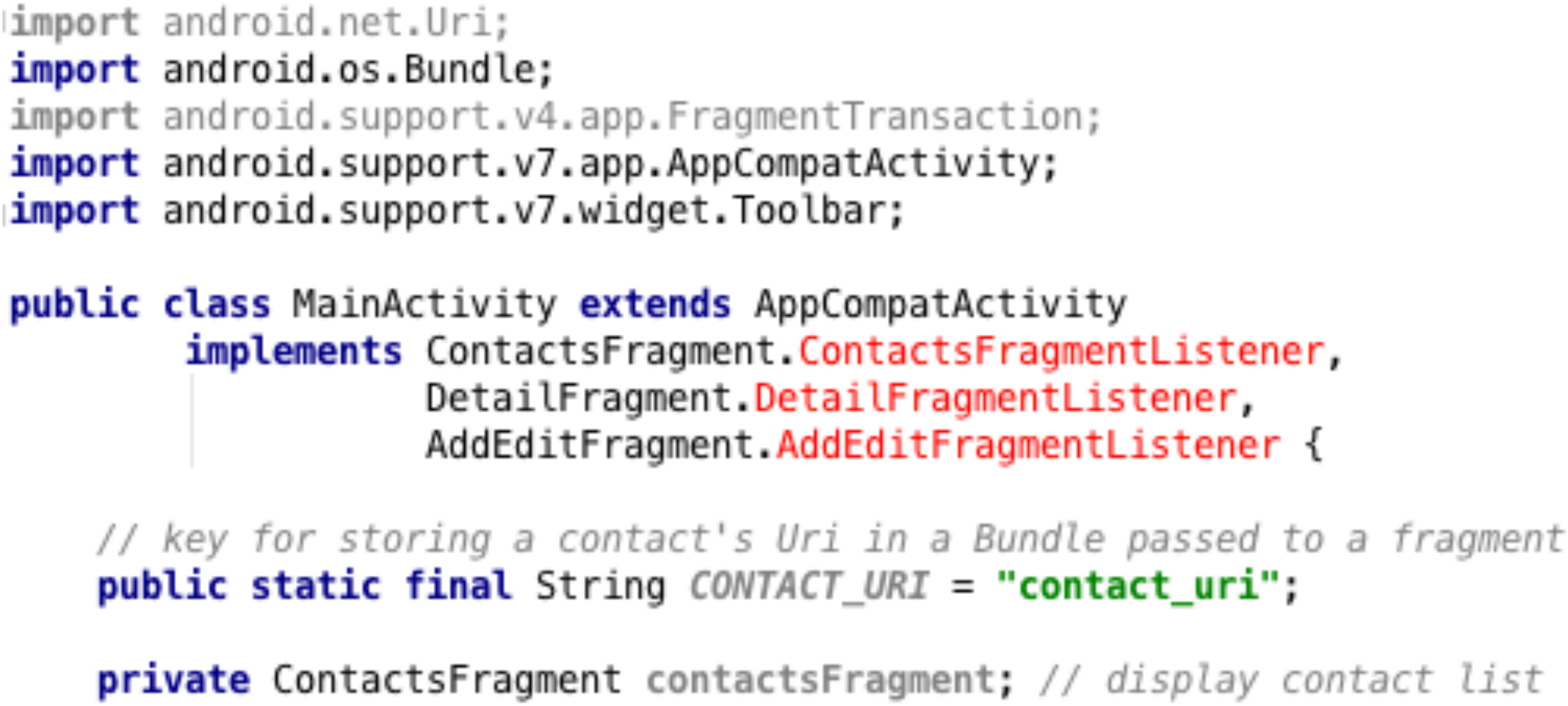
Kommunikation zwischen Fragment und Activity

- To communicate data between Fragments and a host Activity or the Activity’s other Fragments, it’s considered best practice to do so through the host Activity—this makes the Fragments more reusable, because they do not refer to one another directly. Typically, each Fragment defines an interface of callback methods that are implemented in the host Activity. We’ll use this technique to enable this app’s MainActivity to be notified when the user:

- selects a contact to display,
- touches the contact-list Fragment’s add (+) FloatingActionButton,
- touches the contact details Fragment’s ✎ or 🗑 actions,
- or touches 💾 to finish editing an existing contact or adding a new one.

MainActivity.java

zuständig für die Kommunikation zwischen den Fragments



```
import android.net.Uri;
import android.os.Bundle;
import android.support.v4.app.FragmentTransaction;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;

public class MainActivity extends AppCompatActivity
        implements ContactsFragment.ContactsFragmentListener,
                   DetailFragment.DetailFragmentListener,
                   AddEditFragment.AddEditFragmentListener {

    // key for storing a contact's Uri in a Bundle passed to a fragment
    public static final String CONTACT_URI = "contact_uri";

    private ContactsFragment contactsFragment; // display contact list
```

Interfaces

- **ContactsFragment**. ContactsFragmentListener contains call-back methods that the ContactsFragment uses to tell the MainActivity **when the user selects a contact in the contact list or adds a new contact**.

- **DetailFragment**. DetailFragmentListener contains callback methods that the DetailFragment uses to tell the MainActivity **when the user deletes a contact or wishes to edit an existing contact**.

- **AddEditFragment**. AddEditFragmentListener contains a call-back method that the AddEditFragment uses to tell the MainActivity **when the user saves a new contact or saves changes to an existing contact**.

FragmentTransactions

! https://www.youtube.com/watch?v=Nv24t2CJ6yw !

- In earlier apps that used Fragments, you declared each Fragment in an Activity's layout or, for a DialogFragment, called its show method to create it. The Flag Quiz app demonstrated how to use multiple activities to host each of the app's Fragments on a phone device, and a single Activity to host multiple Fragments on a tablet device.

- In this app, you'll use only one Activity to host all of the app's Fragments. On a phone-sized device, you'll display one Fragment at a time. On a tablet, you'll always display the Fragment containing the contact list and display the Fragments for viewing, adding and editing contacts as they're needed. To do this, you'll use the FragmentManager and FragmentTransactions to dynamically display Fragments. In addition, you'll use Android's Fragment back stack—a data structure that stores Fragments in last-in-first-out (LIFO) order—to provide automatic support for Android's back button ( ). This enables users to go back to prior Fragments via the back button.

http://developer.android.com/guide/components/fragments.html

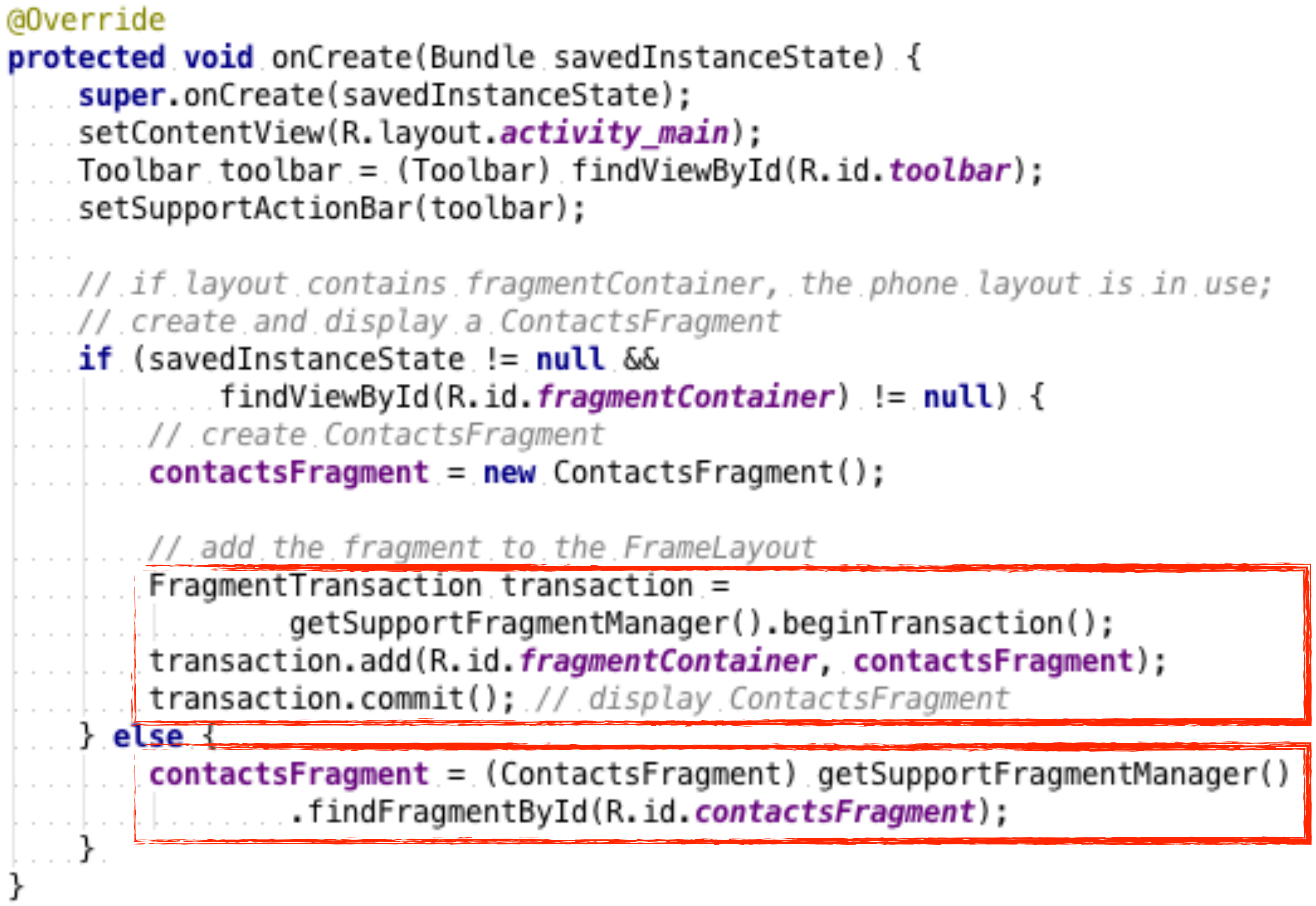
Übung

https://www.youtube.com/watch?v=cDNoJc7azMs

- Arbeiten Sie obiges Beispiel von Vivz in einem eigenen Projekt aus

MainActivity.java

onCreate()



```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    // if layout contains fragmentContainer, the phone layout is in use;
    // create and display a ContactsFragment
    if (savedInstanceState != null &&
            findViewById(R.id.fragmentContainer) != null) {
        // create ContactsFragment
        contactsFragment = new ContactsFragment();

        // add the fragment to the FrameLayout
        FragmentTransaction transaction =
                getSupportFragmentManager().beginTransaction();
        transaction.add(R.id.fragmentContainer, contactsFragment);
        transaction.commit(); // display ContactsFragment
    } else {
        contactsFragment = (ContactsFragment) getSupportFragmentManager()
                .findFragmentById(R.id.contactsFragment);
    }
}
```

MainActivity.java

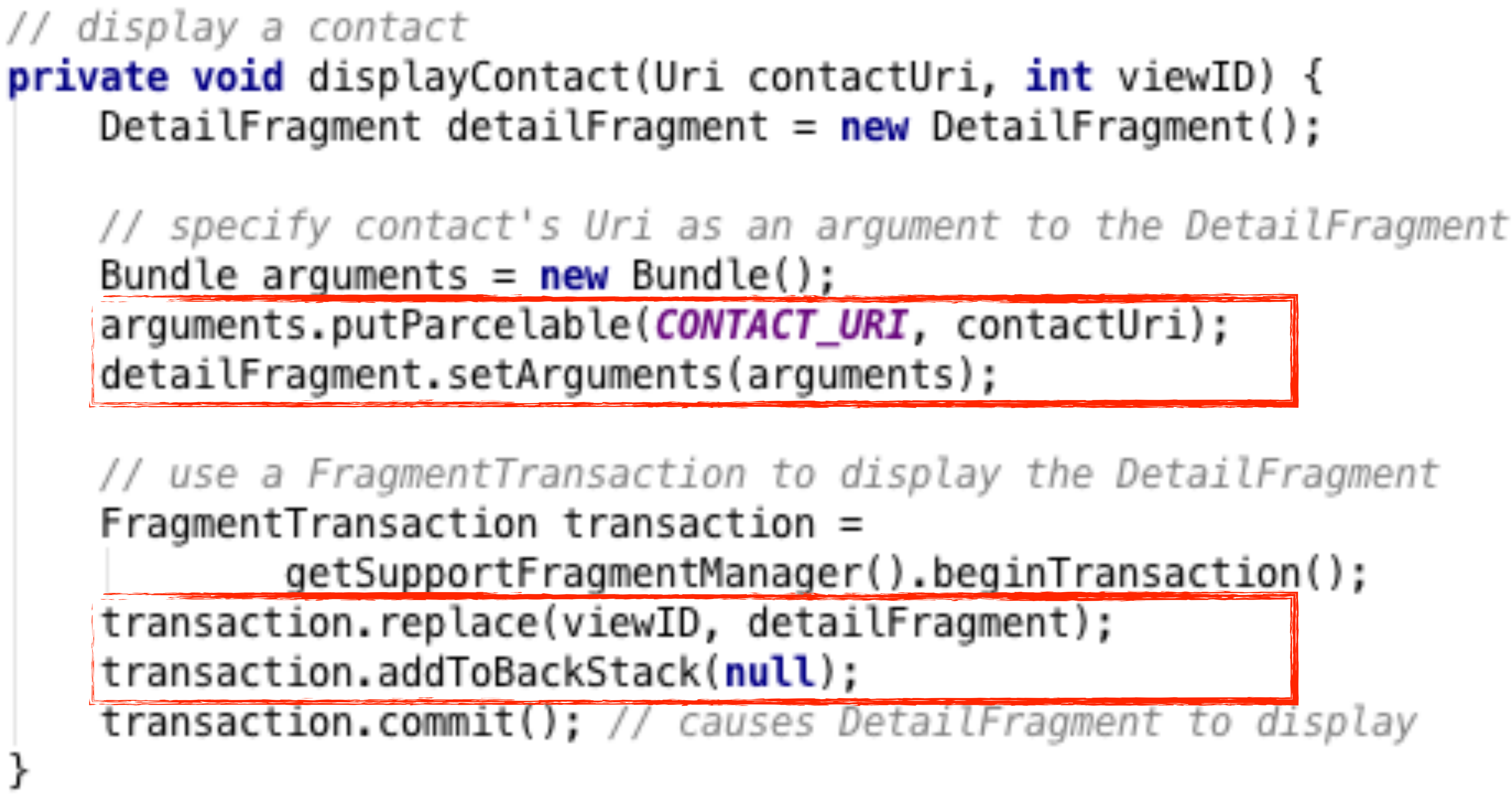
onContactsSelected(), onAddContact

Das Interface wird erst später erstellt

```
@Override
public void onContactSelected(Uri contactUri) {
    if (findViewById(R.id.fragmentContainer) != null) { // phone
        displayContact(contactUri, R.id.fragmentContainer);
    }
    else { // tablet
        // removes top of back stack
        getSupportFragmentManager().popBackStack();

        displayContact(contactUri, R.id.rightPaneContainer);
    }
}

// display AddEditFragment to add a new contact
@Override
public void onAddContact() {
    if (findViewById(R.id.fragmentContainer) != null) {// phone
        displayAddEditFragment(R.id.fragmentContainer, null);
    } else {// tablet
        displayAddEditFragment(R.id.rightPaneContainer, null);
    }
}
```

MainActivity.java

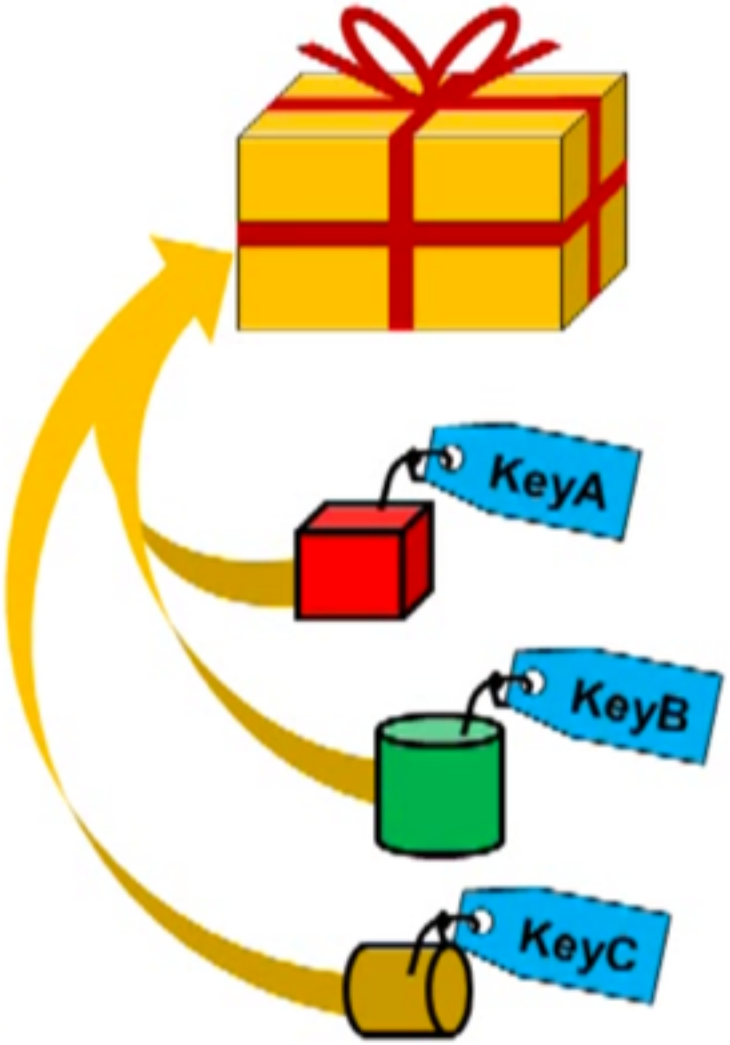
displayContact()

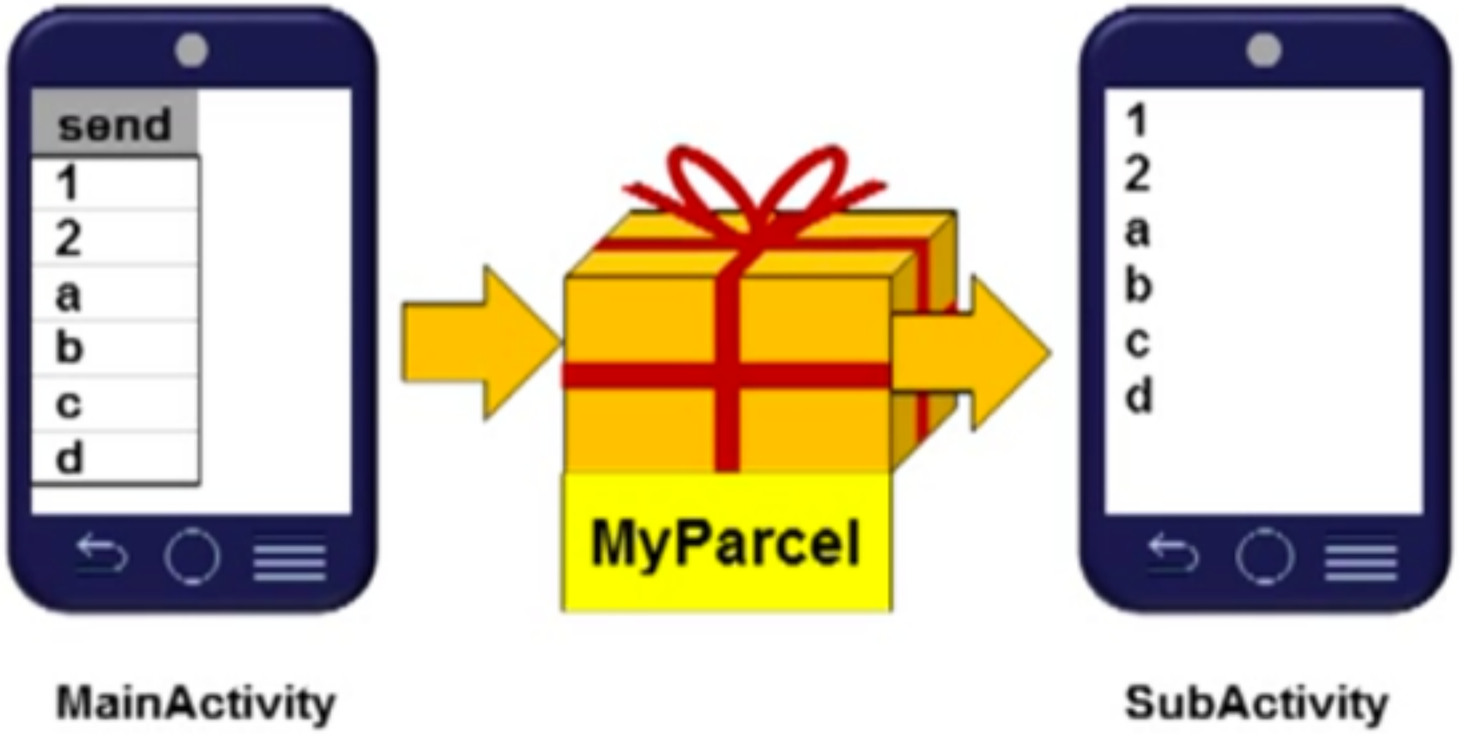


```java
// display a contact
private void displayContact(Uri contactUri, int viewID) {
    DetailFragment detailFragment = new DetailFragment();

    // specify contact's Uri as an argument to the DetailFragment
    Bundle arguments = new Bundle();
    arguments.putParcelable(CONTACT_URI, contactUri);
    detailFragment.setArguments(arguments);

    // use a FragmentTransaction to display the DetailFragment
    FragmentTransaction transaction =
            getSupportFragmentManager().beginTransaction();
    transaction.replace(viewID, detailFragment);
    transaction.addToBackStack(null);
    transaction.commit(); // causes DetailFragment to display
}
```

Parcelable

https://www.youtube.com/watch?v=kx2WlZksWk8





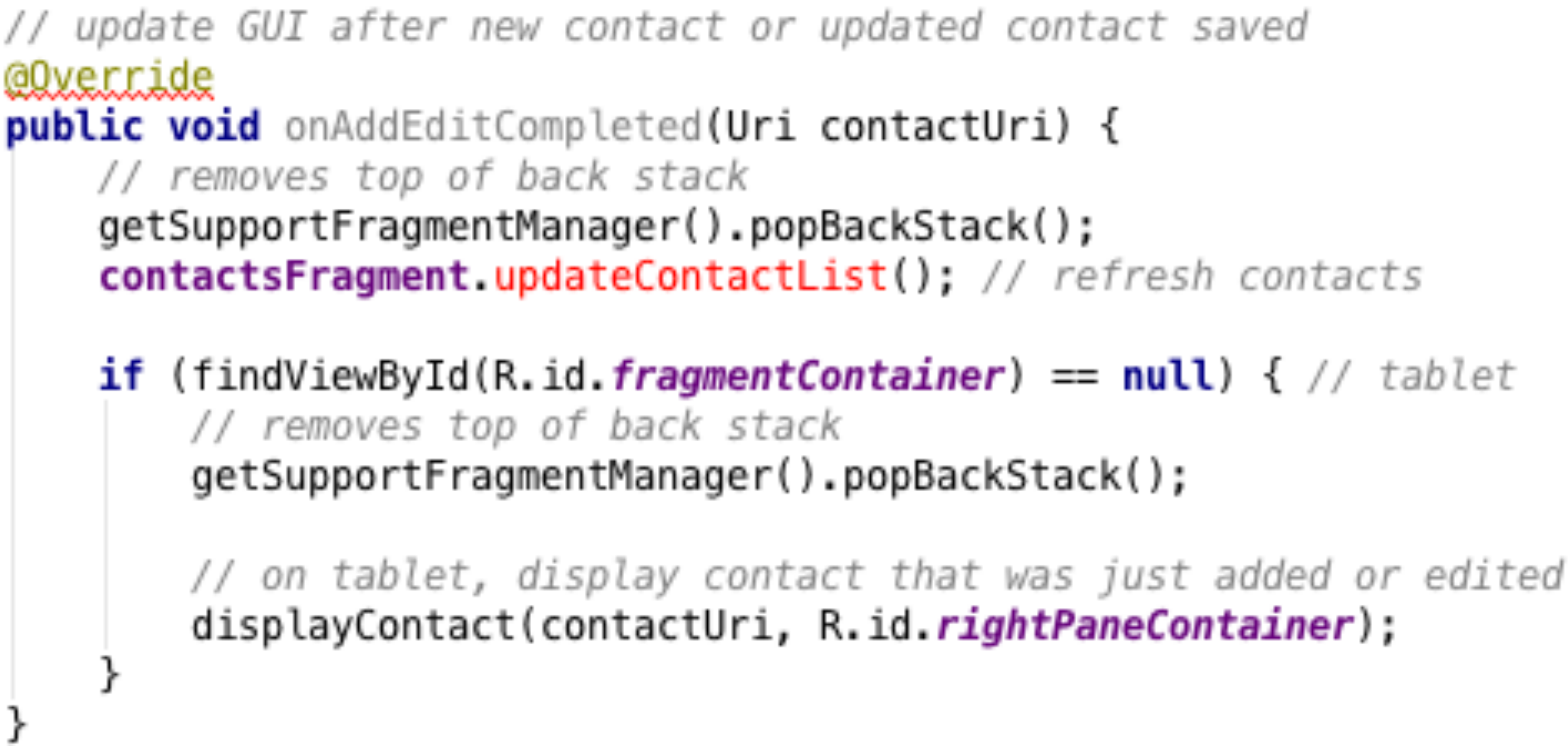
MainActivity.java

onContactDeleted(), onEditContact()

```
// return to contact list when displayed contact deleted
@Override
public void onContactDeleted() {
    // removes top of back stack
    getSupportFragmentManager().popBackStack();
    contactsFragment.updateContactList(); // refresh contacts
}

// display the AddEditFragment to edit an existing contact
@Override
public void onEditContact(Uri contactUri) {
    if (findViewById(R.id.fragmentContainer) != null) {// phone
        displayAddEditFragment(R.id.fragmentContainer, contactUri);
    } else {// tablet
        displayAddEditFragment(R.id.rightPaneContainer, contactUri);
    }
}
```

MainActivity.java

onAddEditCompleted()



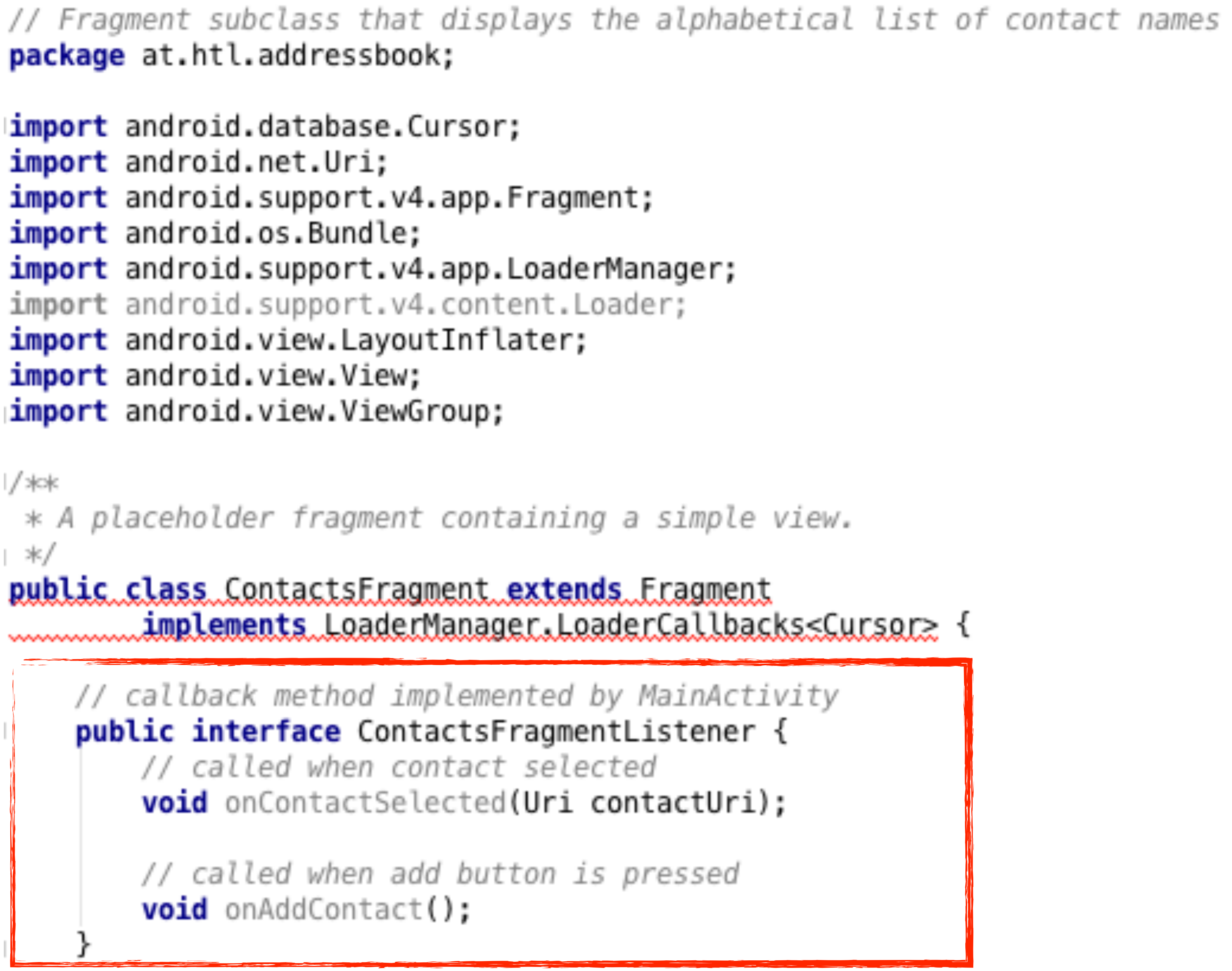
```
// update GUI after new contact or updated contact saved
@Override
public void onAddEditCompleted(Uri contactUri) {
    // removes top of back stack
    getSupportFragmentManager().popBackStack();
    contactsFragment.updateContactList(); // refresh contacts

    if (findViewById(R.id.fragmentContainer) == null) { // tablet
        // removes top of back stack
        getSupportFragmentManager().popBackStack();

        // on tablet, display contact that was just added or edited
        displayContact(contactUri, R.id.rightPaneContainer);
    }
}
```

ContactFragment.java

Zeigt die Kontaktliste in einer RecyclerView an und stellt den FloatingActionButton zur Verfügung

ContactsFragment.java

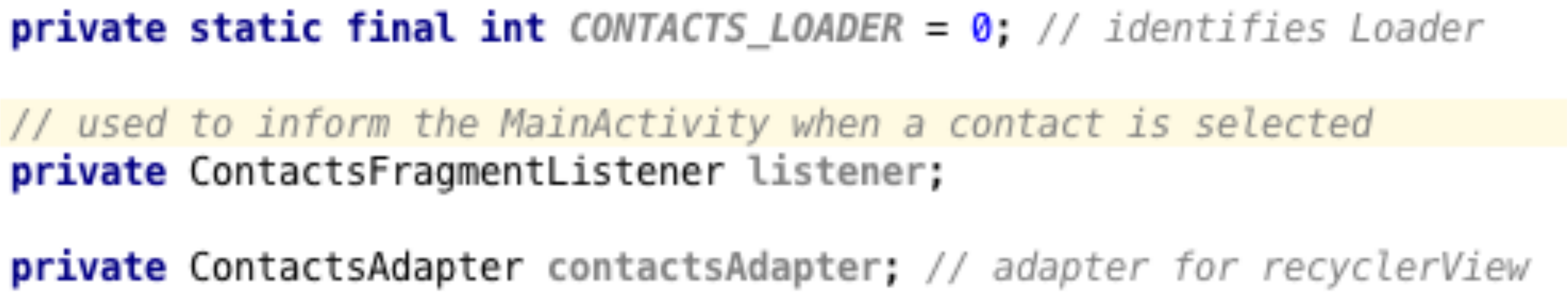
Interface ContactsFragmentListener



```
// Fragment subclass that displays the alphabetical list of contact names
package at.htl.addressbook;

import android.database.Cursor;
import android.net.Uri;
import android.support.v4.app.Fragment;
import android.os.Bundle;
import android.support.v4.app.LoaderManager;
import android.support.v4.content.Loader;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

/**
 * A placeholder fragment containing a simple view.
 */
public class ContactsFragment extends Fragment
        implements LoaderManager.LoaderCallbacks<Cursor> {

    // callback method implemented by MainActivity
    public interface ContactsFragmentListener {
        // called when contact selected
        void onContactSelected(Uri contactUri);

        // called when add button is pressed
        void onAddContact();
    }
```

ContactsFragment.java

fields



```
private static final int CONTACTS_LOADER = 0; // identifies Loader

// used to inform the MainActivity when a contact is selected
private ContactsFragmentListener listener;

private ContactsAdapter contactsAdapter; // adapter for recyclerView
```

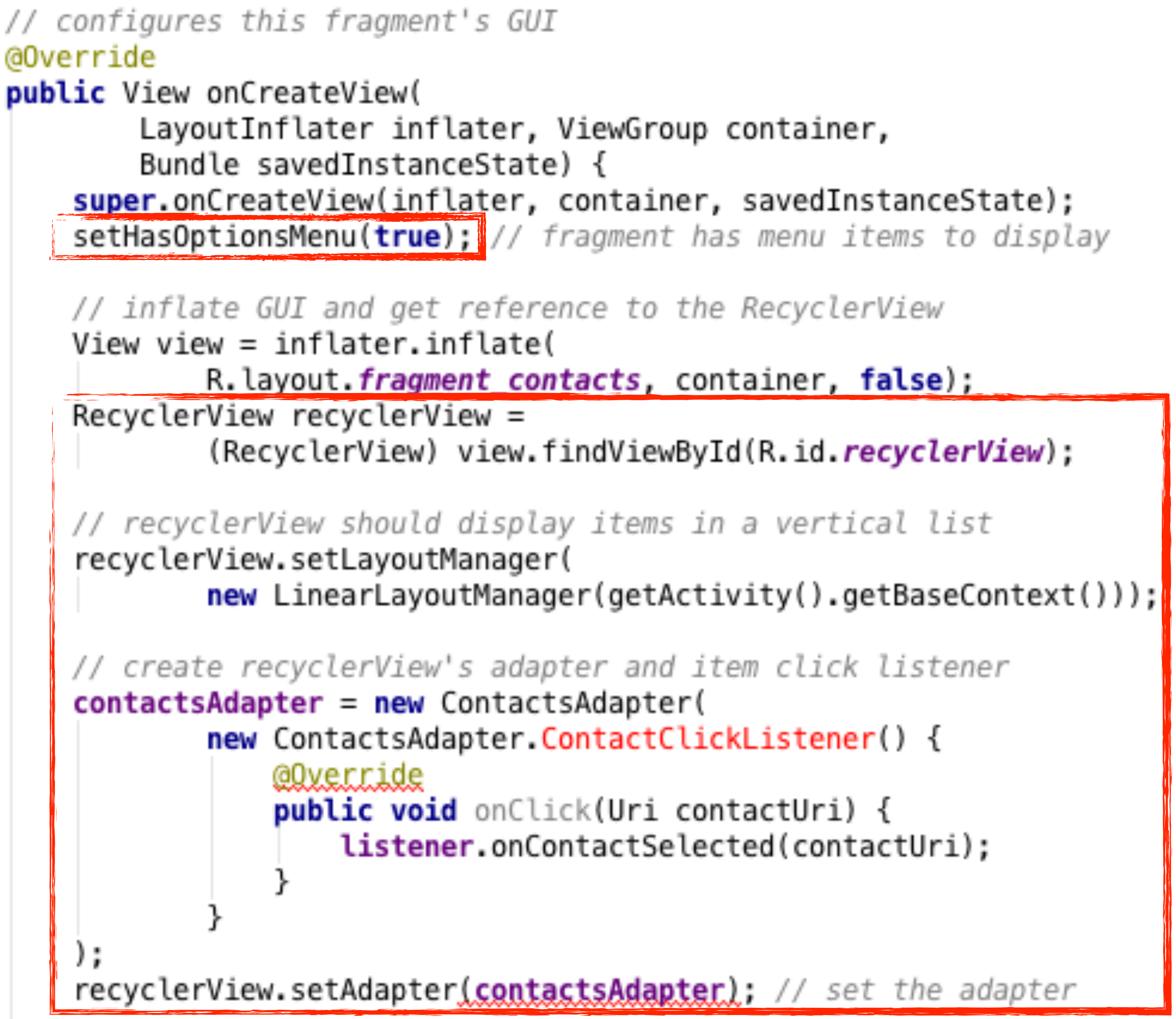
- **CONTACTS_LOADER** declares a constant that’s used to identify the Loader when processing the results returned from the AddressBookCon- tentProvider. In this case, we have only one Loader—if a class uses more than one Loader, each should have a constant with a unique integer value so that you can identify which Loader to manipulate in the LoaderManager.LoaderCallbacks<Cursor> callback methods.

- The instance variable **listener** will refer to the object that implements the interface (MainActivity).

- Instance variable **contactsAdapter** will refer to the ContactsAdapter that binds data to the RecyclerView.

ContactsFragment.java

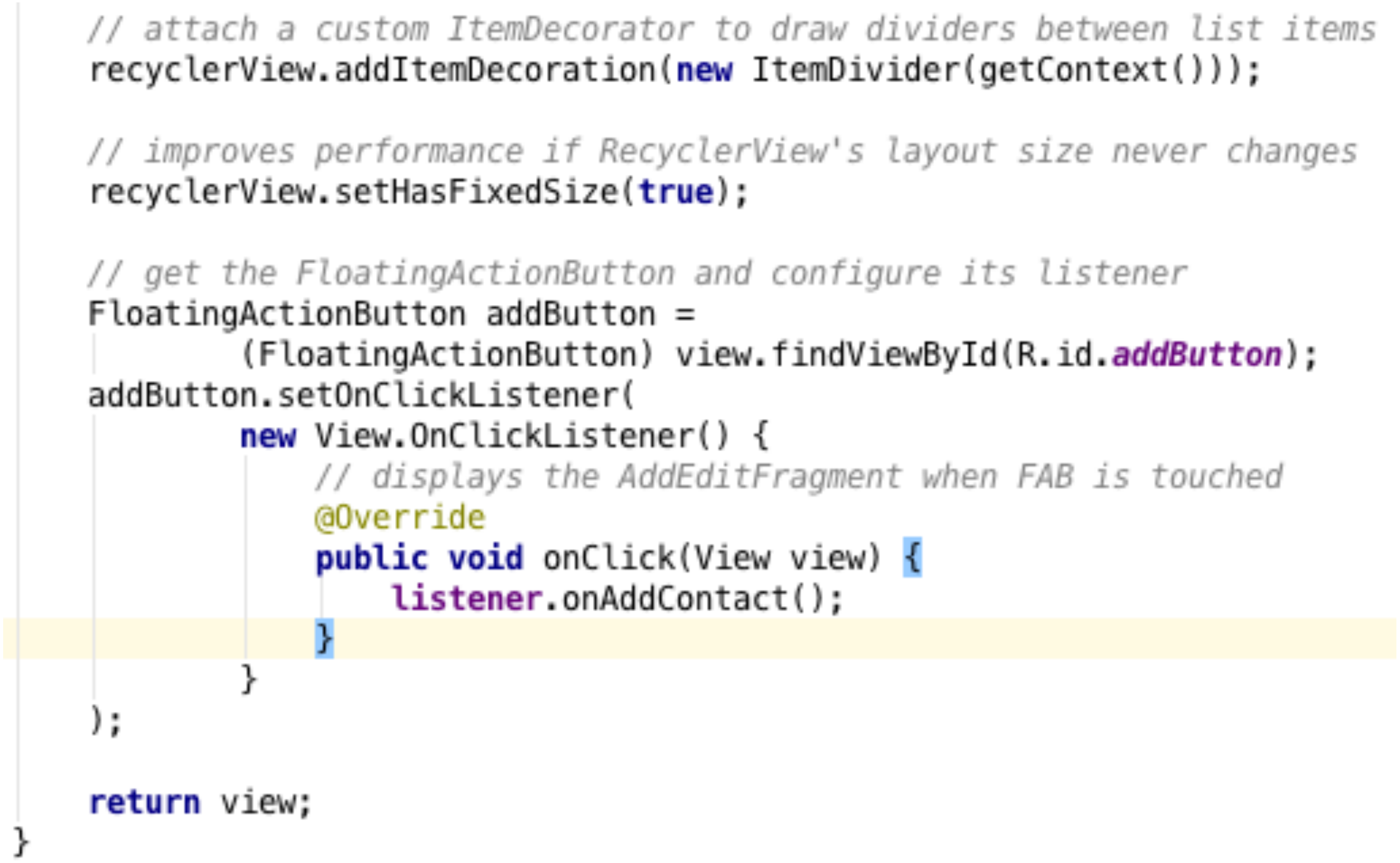
onCreateView() 1 / 2



```
// configures this fragment's GUI
@Override
public View onCreateView(
        LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
    super.onCreateView(inflater, container, savedInstanceState);
    setHasOptionsMenu(true); // fragment has menu items to display

    // inflate GUI and get reference to the RecyclerView
    View view = inflater.inflate(
            R.layout.fragment_contacts, container, false);
    RecyclerView recyclerView =
            (RecyclerView) view.findViewById(R.id.recyclerView);

    // recyclerView should display items in a vertical list
    recyclerView.setLayoutManager(
            new LinearLayoutManager(getActivity().getBaseContext()));

    // create recyclerView's adapter and item click listener
    contactsAdapter = new ContactsAdapter(
            new ContactsAdapter.ContactClickListener() {
                @Override
                public void onClick(Uri contactUri) {
                    listener.onContactSelected(contactUri);
                }
            }
    );
    recyclerView.setAdapter(contactsAdapter); // set the adapter
```

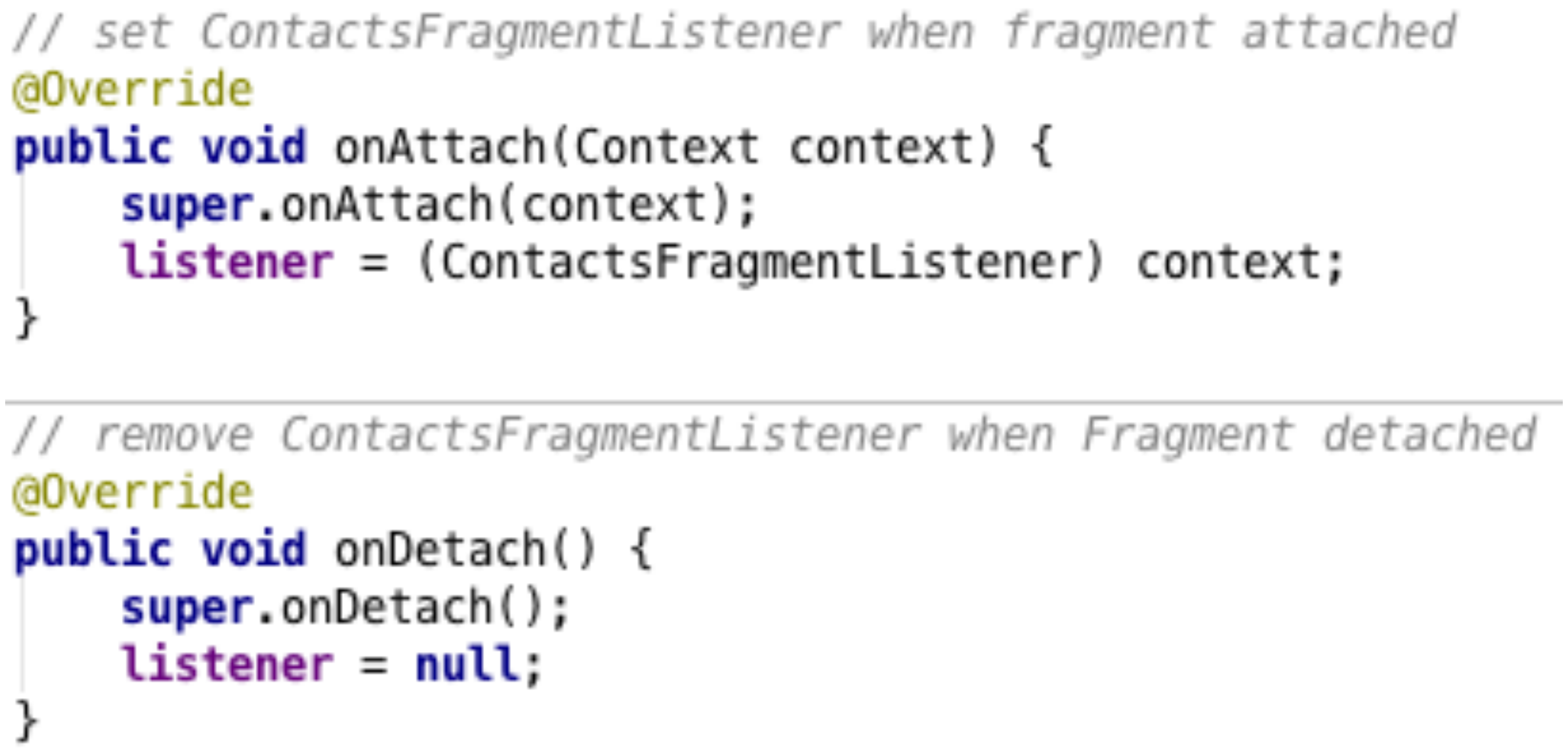
ContactsFragment.java

onCreateView() 2 / 2



```
    // attach a custom ItemDecorator to draw dividers between list items
    recyclerView.addItemDecoration(new ItemDivider(getContext()));

    // improves performance if RecyclerView's layout size never changes
    recyclerView.setHasFixedSize(true);

    // get the FloatingActionButton and configure its listener
    FloatingActionButton addButton =
            (FloatingActionButton) view.findViewById(R.id.addButton);
    addButton.setOnClickListener(
            new View.OnClickListener() {
                // displays the AddEditFragment when FAB is touched
                @Override
                public void onClick(View view) {
                    listener.onAddContact();
                }
            }
    );

    return view;
}
```

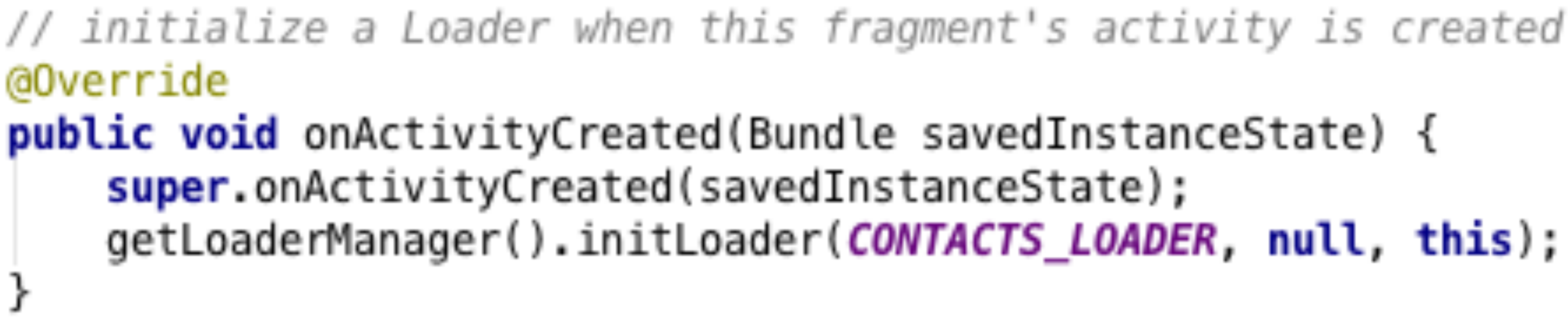
ContactsFragment.java



```java
// set ContactsFragmentListener when fragment attached
@Override
public void onAttach(Context context) {
    super.onAttach(context);
    listener = (ContactsFragmentListener) context;
}
```

```java
// remove ContactsFragmentListener when Fragment detached
@Override
public void onDetach() {
    super.onDetach();
    listener = null;
}
```

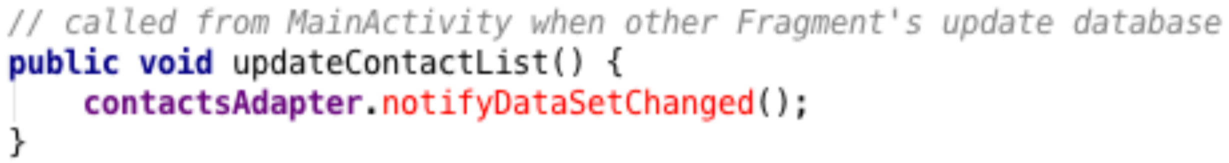
ContactsFragment.java



```java
// initialize a Loader when this fragment's activity is created
@Override
public void onActivityCreated(Bundle savedInstanceState) {
    super.onActivityCreated(savedInstanceState);
    getLoaderManager().initLoader(CONTACTS_LOADER, null, this);
}
```

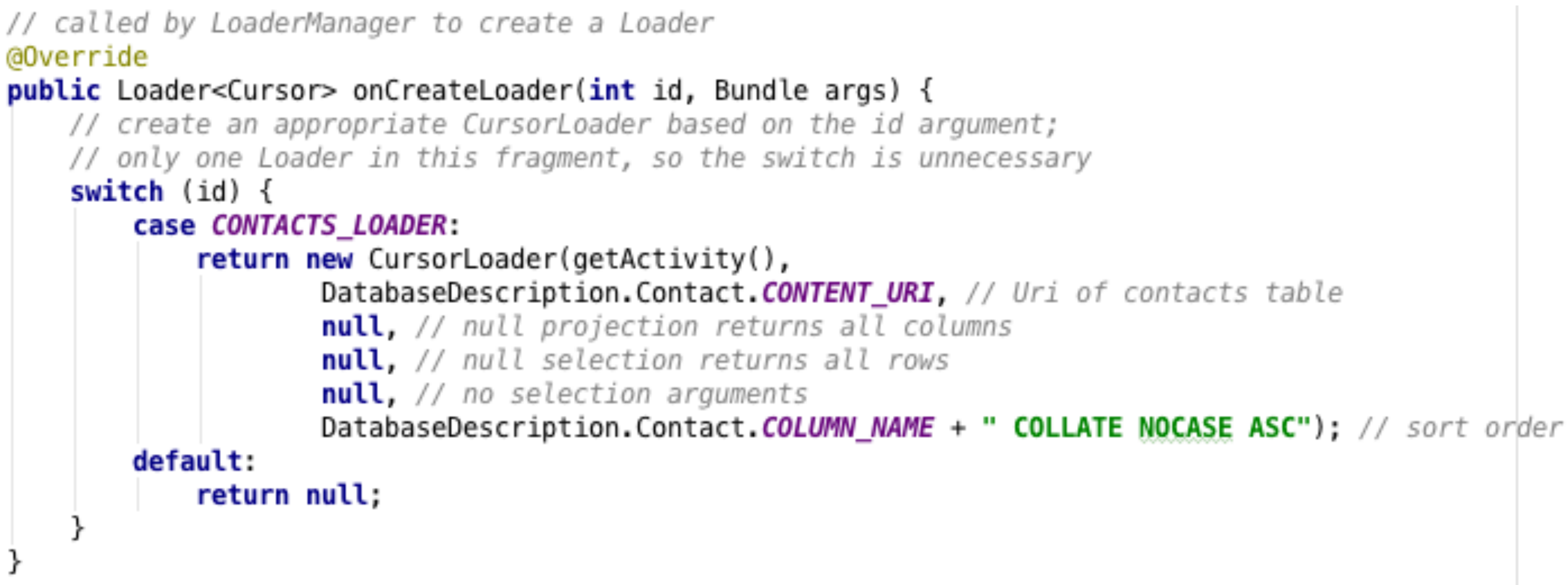
ContactsFragment.java

updateContactList()



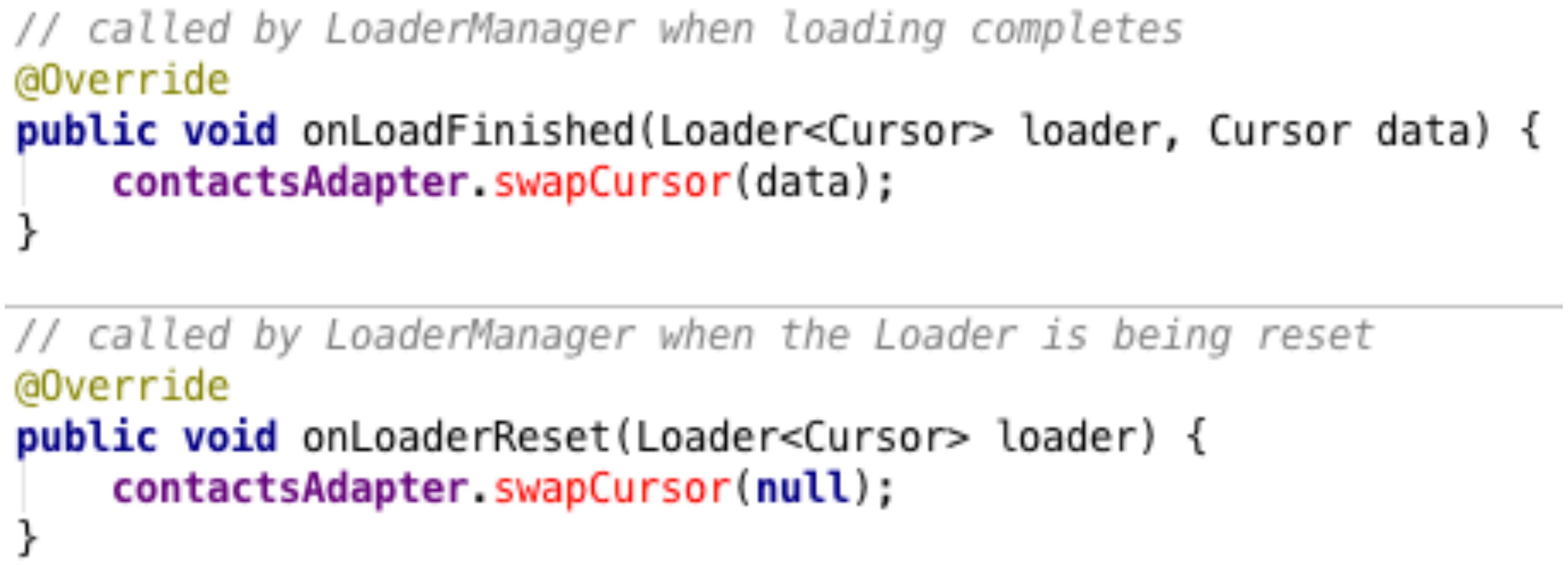
```
// called from MainActivity when other Fragment's update database
public void updateContactList() {
    contactsAdapter.notifyDataSetChanged();
}
```

ContactsFragment.java



```
// called by LoaderManager to create a Loader
@Override
public Loader<Cursor> onCreateLoader(int id, Bundle args) {
    // create an appropriate CursorLoader based on the id argument;
    // only one Loader in this fragment, so the switch is unnecessary
    switch (id) {
        case CONTACTS_LOADER:
            return new CursorLoader(getActivity(),
                    DatabaseDescription.Contact.CONTENT_URI, // Uri of contacts table
                    null, // null projection returns all columns
                    null, // null selection returns all rows
                    null, // no selection arguments
                    DatabaseDescription.Contact.COLUMN_NAME + " COLLATE NOCASE ASC"); // sort order
        default:
            return null;
    }
}
```

ContactsFragment.java



```java
// called by LoaderManager when loading completes
@Override
public void onLoadFinished(Loader<Cursor> loader, Cursor data) {
    contactsAdapter.swapCursor(data);
}

// called by LoaderManager when the Loader is being reset
@Override
public void onLoaderReset(Loader<Cursor> loader) {
    contactsAdapter.swapCursor(null);
}
```

Noch Fragen?