Paixon

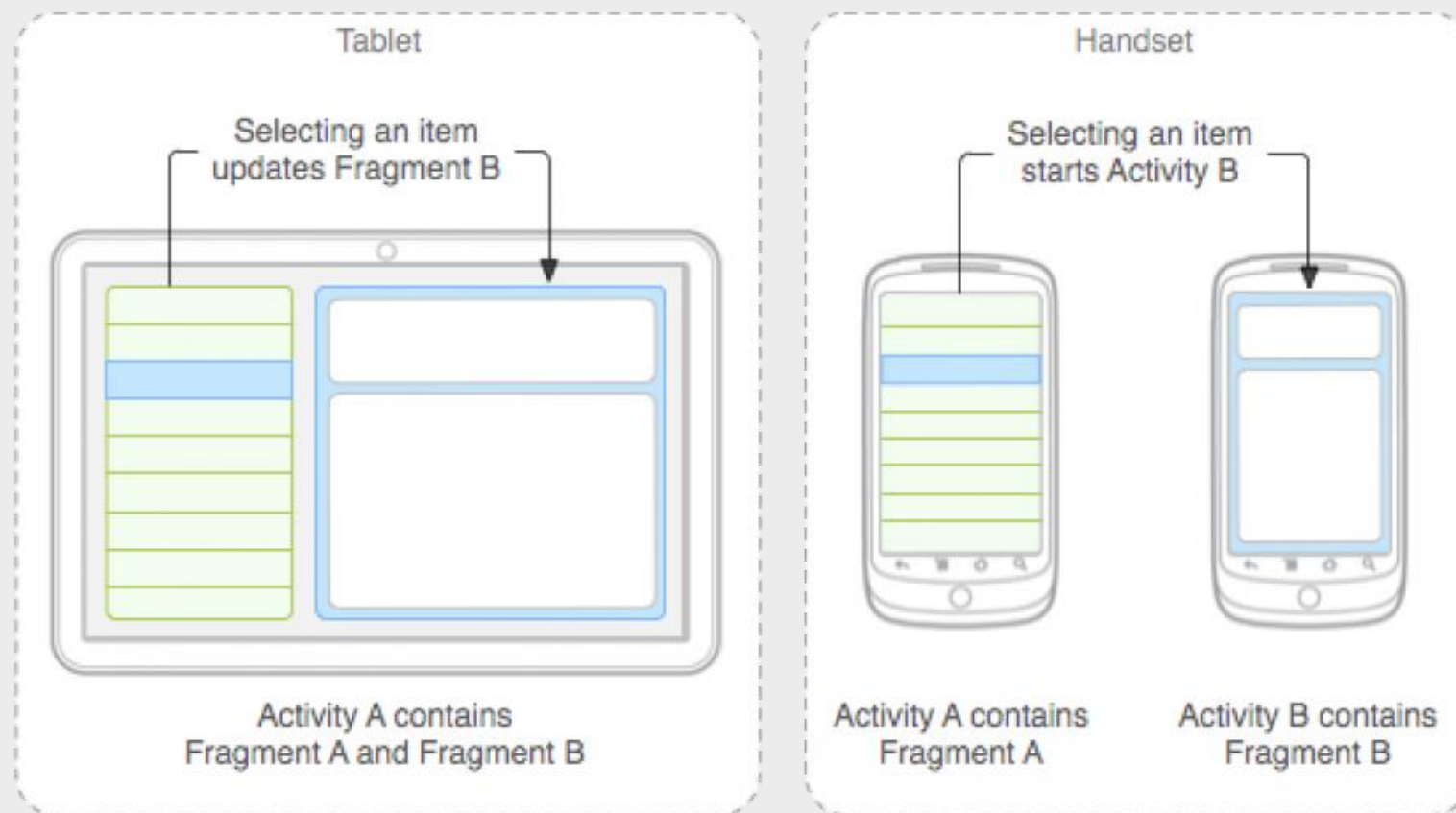# Fragments

Paixon

Fragments – Why?

Paixon

Paixon

Show fragments depending on screen size

Paixon

- Introduced with Android Honeycomb 3.0
- Part of the screen
- Always running inside an activity (Activity must inherit FragmentActivity)
- Own lifecycle (onAttach() / onCreate() / onCreateView() / ...)

- Backward compatibility given by Support Library
  - AppCompatActivity

Class inheriting from Fragment

```java
public static class ExampleFragment extends Fragment {

    @Override
    public View onCreateView( LayoutInflater inflater,
                              ViewGroup container,
                              Bundle savedInstanceState) {

        return inflater.inflate(R.layout.mylayout, container, false);
    }
}
```

Paixon

## How to add a fragment using XML?

```xml
<fragment android:name="com.example.news.ArticleListFragment"
          android:id="@+id/theIdOfTheFragment"
          android:layout_width="match_parent"
          android:layout_height="match_parent" />
```

How to add a fragment using code?

```java
FragmentManager fragmentManager = getFragmentManager();
FragmentTransaction fragmentTransaction =
fragmentManager.beginTransaction();

ExampleFragment fragment = new ExampleFragment();
fragmentTransaction.add(R.id.fragment_container, fragment);
fragmentTransaction.commit();
```

Fragments – Use Case Smartphone

Paixon
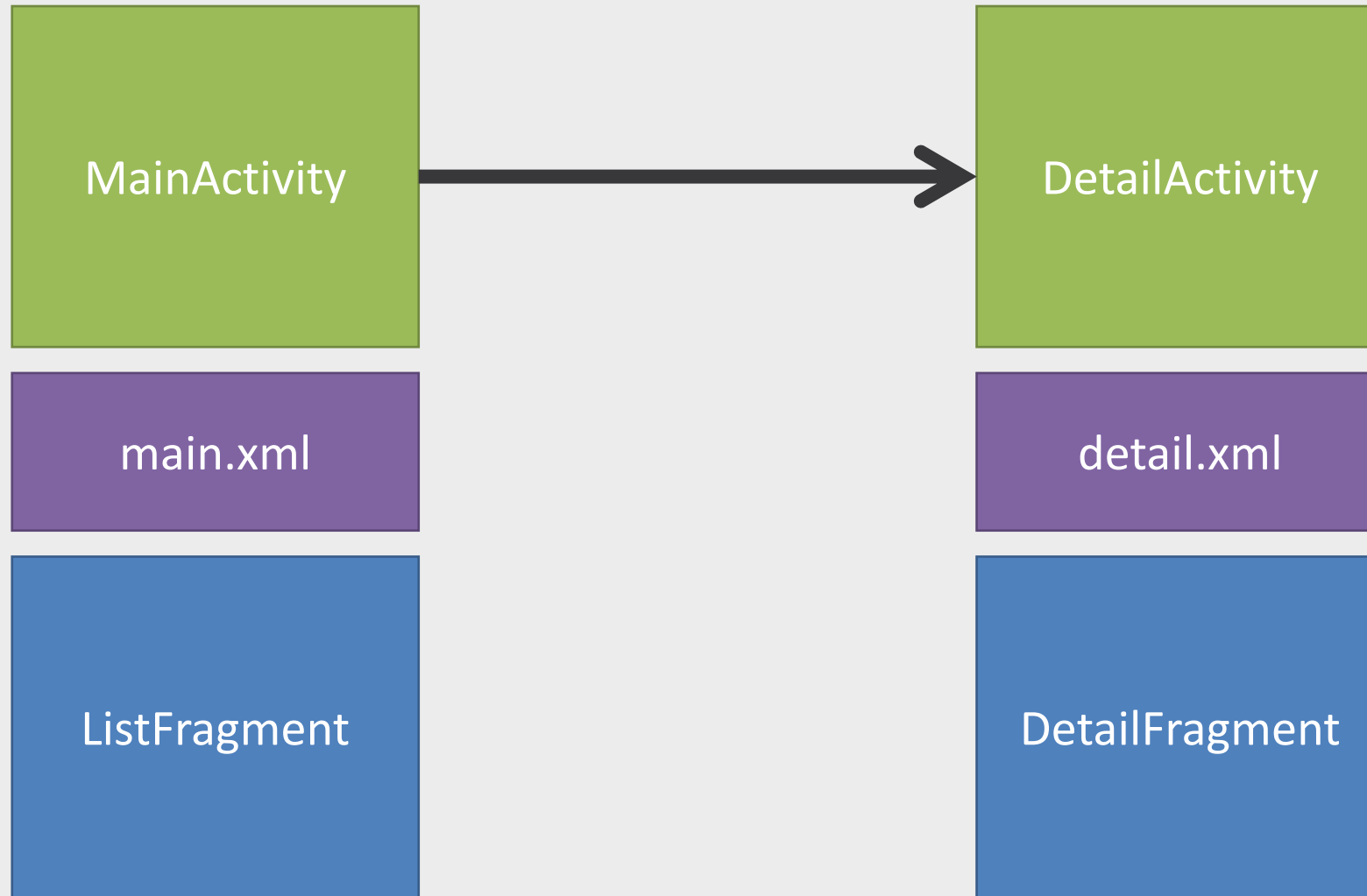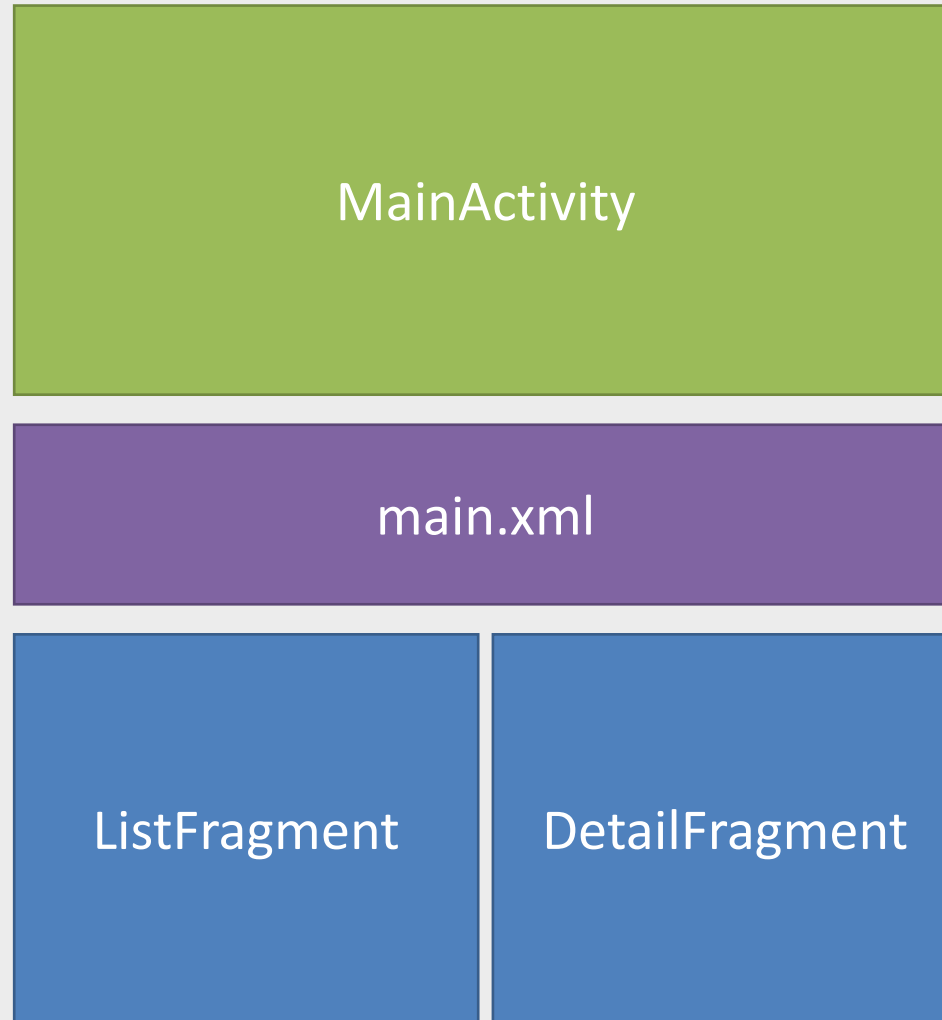
Build a tablet-aware app

- Define usage of fragments in layout files
- Use different resource qualifiers to decide which layout to choose
  - Smartphone – Portrait   Show List → Click navigates to other Activity
  - Smartphone – Landscape: Show List and Detail Fragment
  - Tablet   Show List and Detail Fragment
- Use findViewById() to find out whether a fragment is available on the current configuration

Paixon

Paixon
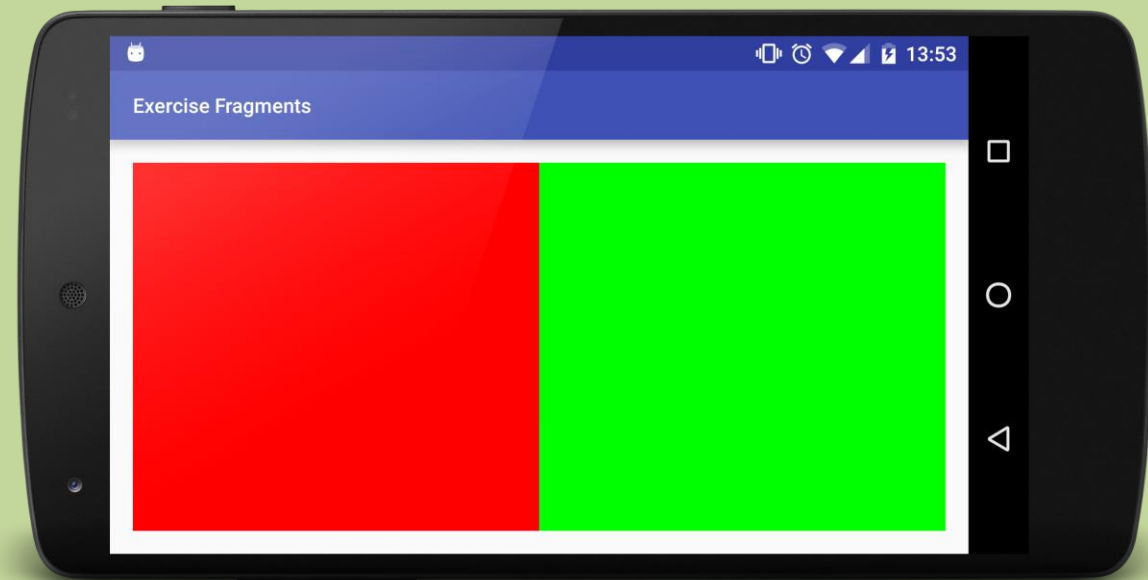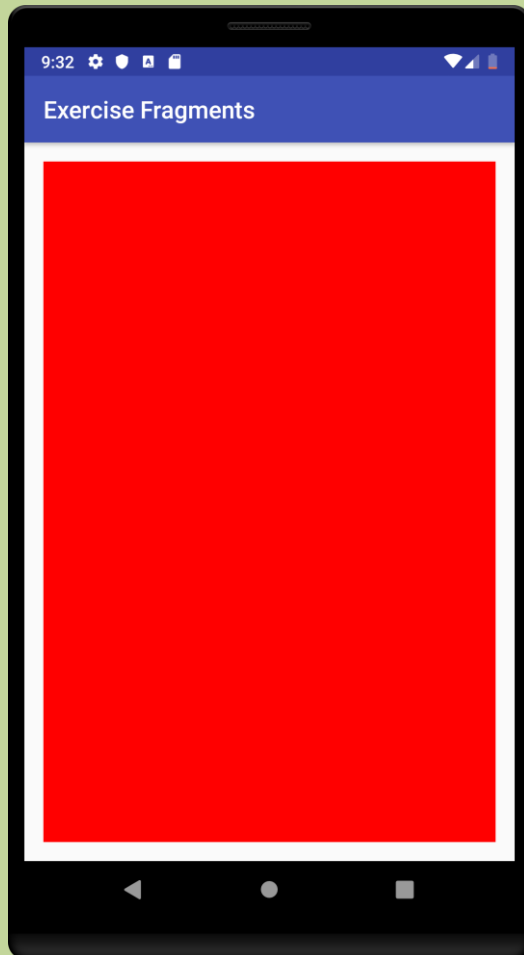
Aufgaben

Analyse der bestehenden Dateien
Fragment1.java Verstehe den Zusammenhang mit dem entsprechenden layout file
fragment1.xml   Layout für das Fragment1. dabei handelt es sich um ein normales Layout wie es auch von Activities verwendet wird

Zeige Fragment1 in der MainActivity an
Füge dazu das Fragment am richtigen Ort im entsprechenden XML-file für das MainActivity-layout ein

Erstelle ein zweites Fragment "Fragment2"
Erstelle die entsprechende Java Klasse sowie das Layout file. Stelle sicher dass in der onCreateView-Methode das richtige Layout geladen wird

Unterscheide zwischen Portrait- und Landscape-Mode
Wenn das Gerät im Portrait ist, zeige die Fragments untereinander an. Im "landscape"-mode sollen die Fragments nebeneinander angezeigt werden.

Projekt: Exercise_Fragements

Paixon

BroadcastReceiver

Paixon

System notifications

Low energy, connection lost, SMS received, System Ready, WiFi Connection Lost, ...

Custom Broadcasts

Communicate between Apps or Components

sendBroadcast(...) or sendOrderedBroadcast(...) on context

Paixon

Extend abstract «BroadcastReceiver» class
- onReceive(Intent intent)
- Intent contains data

Paixon

Register using Manifest

```xml
<receiver android:name=".MyBroadcastReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
    </intent-filter>
</receiver>
```

BroadcastReceiver – Technical Implementation

Paixon

```java
// Create new Receiver Object
BroadcastReceiver receiver = new MyBroadcastReceiver();

// Create Intent Filter for requested Event
IntentFilter filter = new IntentFilter();
filter.addAction(Intent.ACTION_AIRPLANE_MODE_CHANGED);

// register Receiver
this.registerReceiver(receiver, filter);

// unregister Receiver
this.unregisterReceiver(receiver);
```
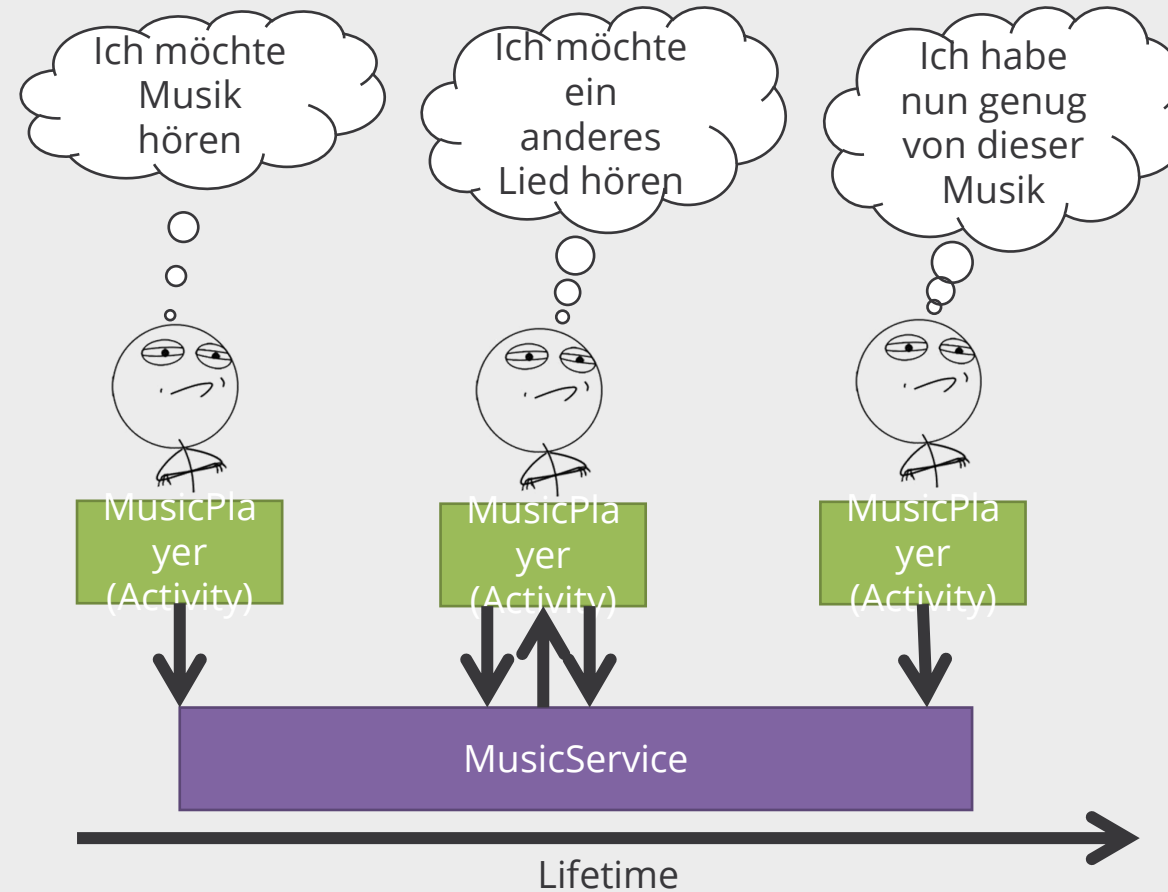
- Registration in manifest is very restricted
- Do not block thread in receiver
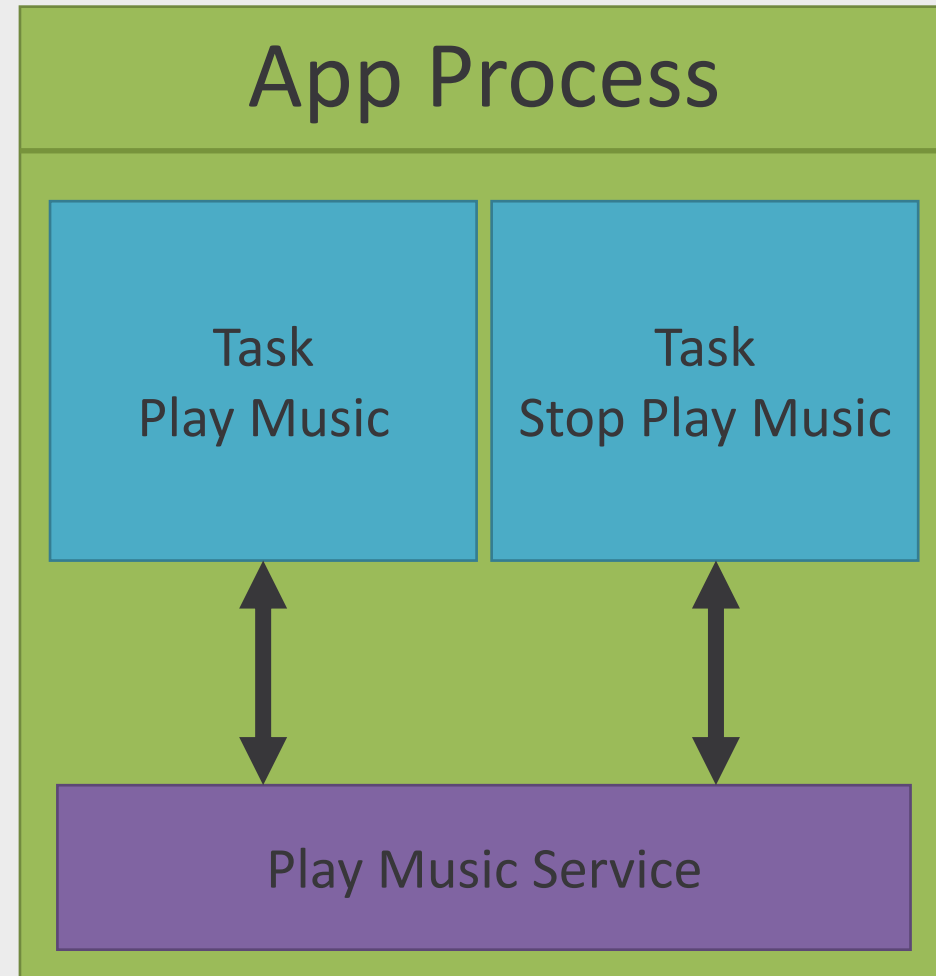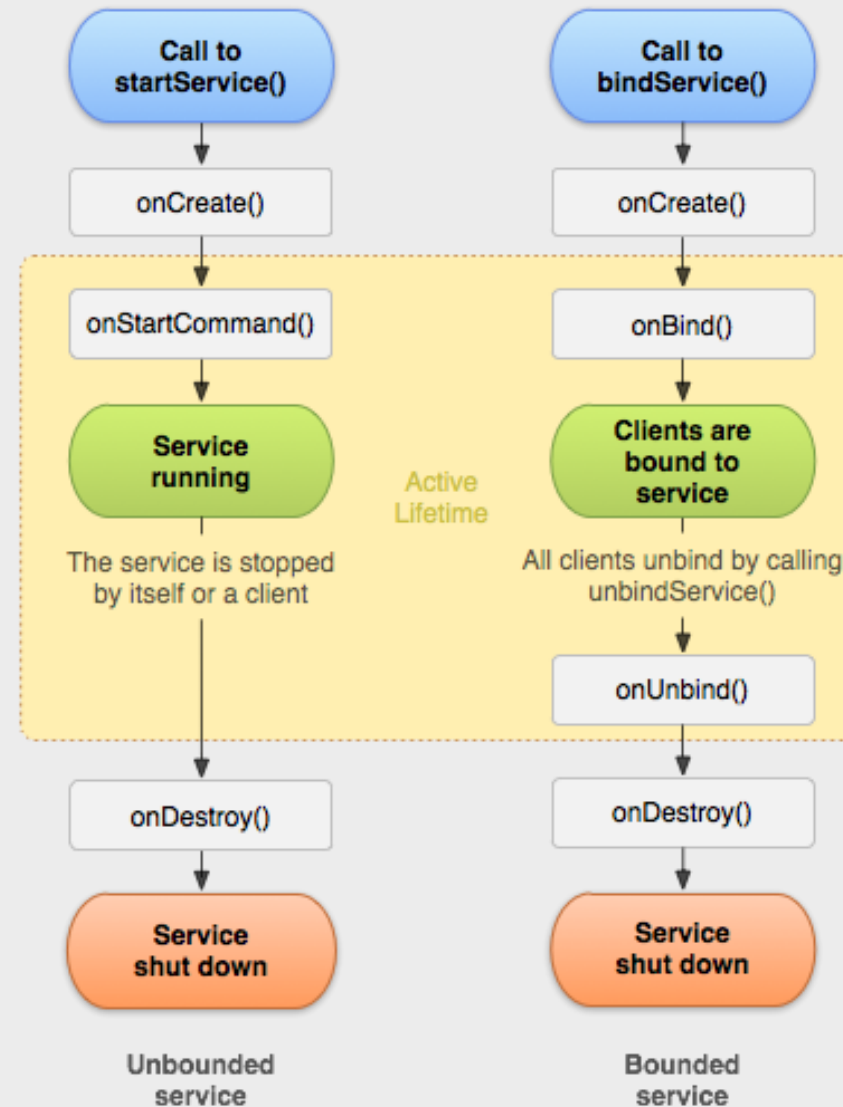- To complicated for inter-component communication

Service

Paixon

Component for long running operations

Services Types

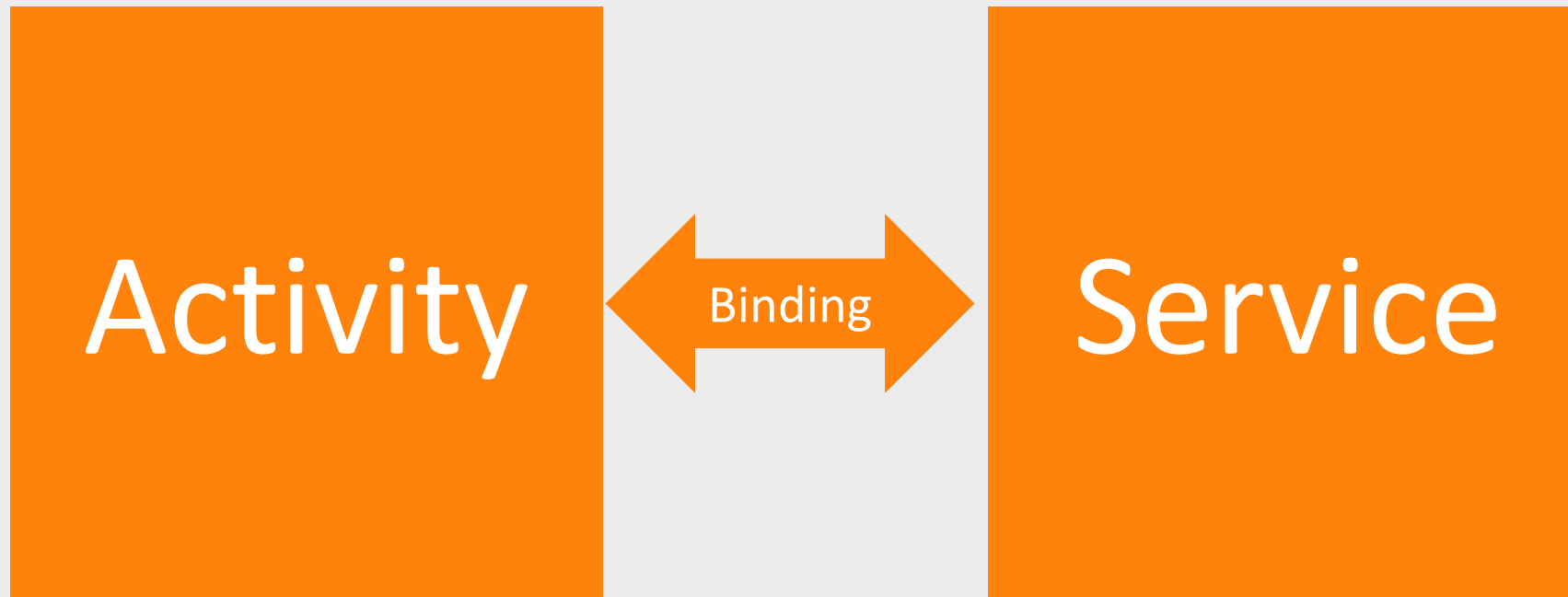| Bound Services	Started Services
Exists as long as any other component is bound to it	Exists as long as nobody stops it
Start use *bindService(...)*	Start use *startService(...)*
Communication throught RPC	Communication through Broadcasts

Combined Services
Start with startService → Services stays forever
Connect to Service with bindService(...) to communicate

Paixon

- Class extending Service
- Register in Manifest
- Runs on UI Thread
- Service runs only once

- Stop the service
  - stopService(Intent)
  - stopSelf() in the service

App Process

Task
Play Music

Task
Stop Play Music

Play Music Service

Paixon

Paixon

Paixon

Paixon
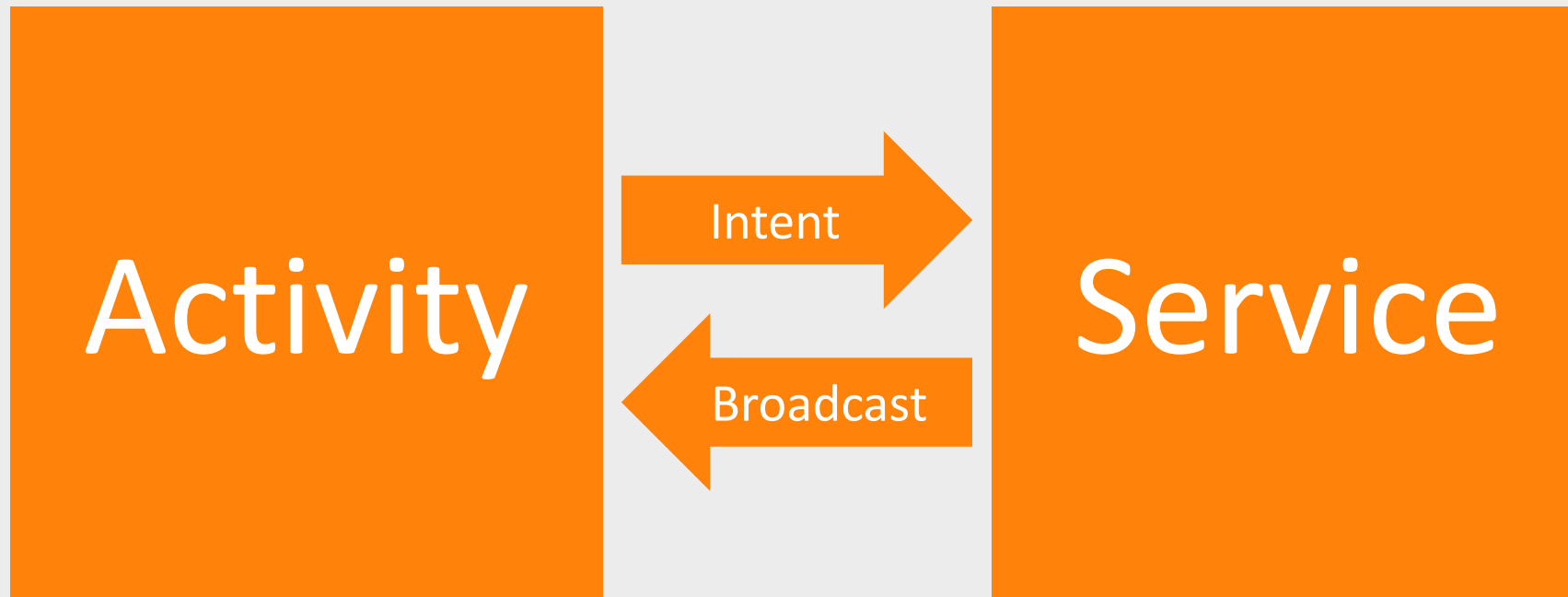
Activity

```
ServiceConnection        (3)
{
  onServiceConnected(Binder binder){
    Service ser = binder.getService();
    ser.doSomethingFancy();
  }
}


// Create the binding
bindService(ServiceConnection);    (4)
```

Service

```
Binder        (1)
{
  getService(){
    return Service.this;
  }
}


onBind{
  return Binder;      (2)
}

// Methods
doSomethingFancy();
```

Paixon

Create a Binder in the Service

```java
public class MyServiceBinder extends Binder {
        public MyBindableService getService() {
                return MyBindableService.this;
        }
}


private final IBinder binder = new MyServiceBinder();

@Override
public IBinder onBind(Intent intent) {
        return binder;
}
```

Paixon

Create a ServiceConnection in your Activity

```java
private ServiceConnection serConn= new ServiceConnection() {

@Override
public void onServiceConnected(ComponentName name, IBinder binder){
        MyServiceBinder customBinder = (MyServiceBinder)binder;
        MyBindableService service = customBinder.getService();
}

@Override
public void onServiceDisconnected(ComponentName name) {

}
};
```

Paixon

Use bindServce() and unbindService() to connect/disconnect

```java
@Override
protected void onResume() {
        super.onResume();
        Intent intent = new Intent(this, MyBindableService.class);
        // Notice BIND_AUTO_CREATE will automatically start
        // your service. You do not need to call startService(…)
        bindService(intent, serCon, Context.BIND_AUTO_CREATE);
}


@Override
protected void onPause() {
        super.onPause();
        unbindService(serCon);
}
```

Paixon

Aufgaben

Eine simple App mit zwei Buttons. Wenn "Increase service counter" gedrückt wird wird der Zähler um 1 erhöht. Wenn "Read cuttent counter" gedrückt wird, wird der aktuelle Wert ausgelesen.

Wenn der User die App über den Back-Button verlässt und die App neu auf dem Gerät startet muss "Read current counter" den alten Wert zurück geben.

Projekt: Exercise_Service

Paixon
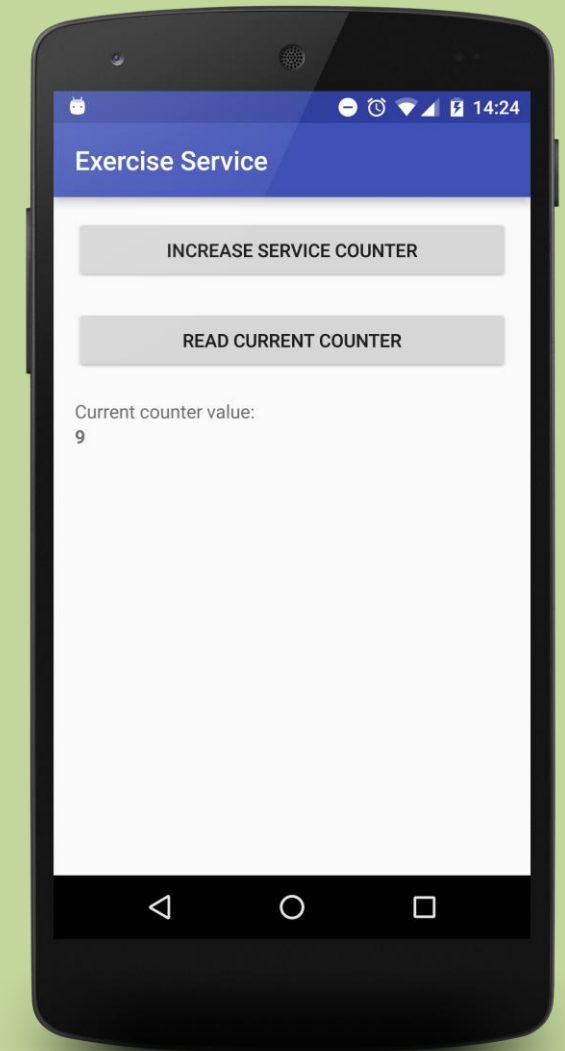
Aufgaben

1. Analysiere das AndroidManifest.xml und verstehe wie Services definiert werden

2. Implementation Bindable Service
   - In der CounterService-Klasse muss erst ein Binder implementiert werden der in onBind(…) zurückgegeben werden kann (siehe slides)
   - Erstelle eine ServiceConnection in der MainActivity. In der onServiceConnected-Methode müssen wir uns nun die Service-Instanz welche wir über den Binder erhalten in der MainActivity-Klasse merken.
   - Benutze nun die lifecycle-Methoden der Activity um den Service zu binden:
     - I. **onCreate()** → startService()
     - II. **onResume()** → bindService(…)
       Benutze das Context.BIND_AUTO_CREATE-flag damit der service gestartet wird falls er noch nicht vorhanden ist.
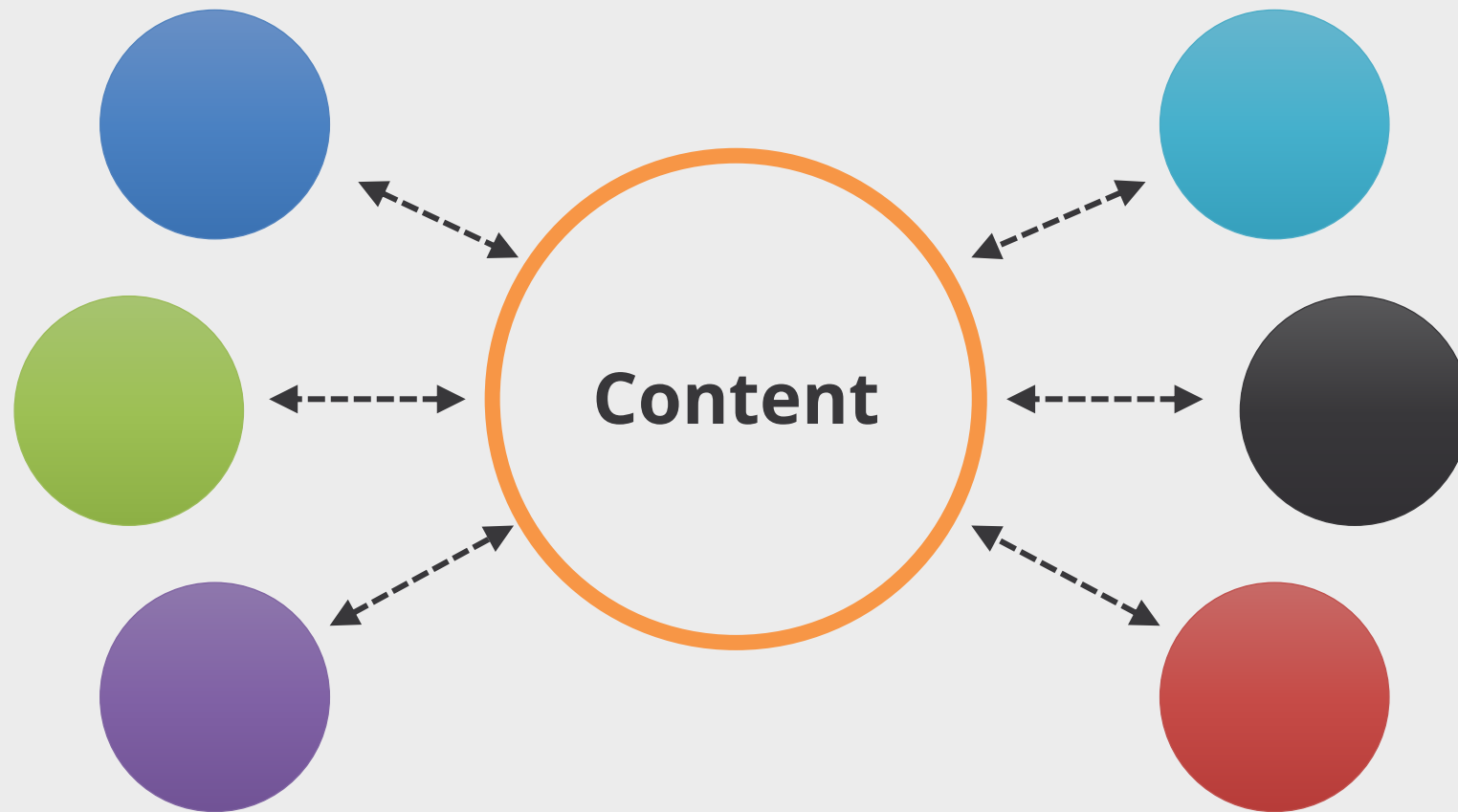   - III. **onPause()** → unbindService(…)

ContentProvider

Paixon

Was ist Content?

Kontakte   Bilder   Daten

Paixon

- Internal REST Service
  - Request/Insert/Update data based on URI's
- Share data with another application
- Manage data security
- Registered in the manifest file
- CRUD-Methods available
  - insert(Uri uri, …)
  - query(Uri uri, …)
  - update(Uri uri,…)
  - delete(Uri uri, …)

Paixon

Identifies the ContentProvider and the path to find the data in it

content://ch.schoeb.notesdemo/notes

Schema   Authority   Path

http://google.ch/playstore

Paixon
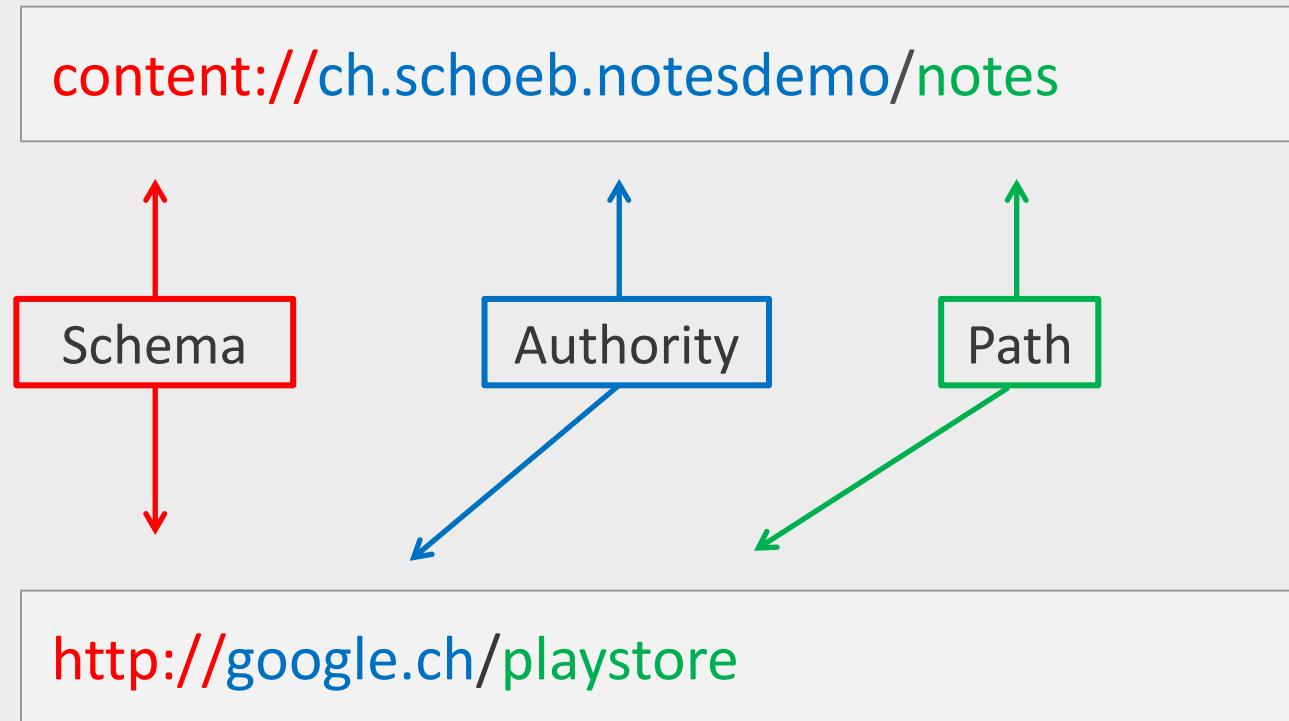
```java
// Run query
Cursor cur = null;
ContentResolver cr = getContentResolver();
Uri uri = Calendars.CONTENT_URI;
String selection = "((" + Calendars.ACCOUNT_NAME + " = ?) AND ("
                        + Calendars.ACCOUNT_TYPE + " = ?) AND ("
                        + Calendars.OWNER_ACCOUNT + " = ?))";

String[] selectionArgs = new String[] {"hera@example.com", "com.example", "hera@example.com"};

// Submit the query and get a Cursor object back.
cur = cr.query(uri, EVENT_PROJECTION, selection, selectionArgs, null);
```

https://developer.android.com/guide/topics/providers/calendar-provider.html

Paixon

Components

Activity	**Service**	**ContentProvider**	**BroadcastReceiver**
UI	Background	Provide data	Get notified about events

Threading

Paixon

DEMO ANR

Application not Responding

Paixon

- Single process

- Single UI Thread

- Every component is running on the UI Thread

- Create thread like in normal java

```java
Thread t = new Thread(new Runnable() {

        @Override
        public void run() {
                // Do long running operation here
        }
});

t.start();
```

10-24 14:25:32.195: E/AndroidRuntime(1906): FATAL EXCEPTION: Thread-8310-24
14:25:32.195: E/AndroidRuntime(1906):
android.view.ViewRootImpl$CalledFromWrongThreadException: Only the original thread that
created a view hierarchy can touch its views.10-24 14:25:32.195: E/AndroidRuntime(1906):
        at android.view.ViewRootImpl.checkThread(ViewRootImpl.java:4746)10-24
14:25:32.195: E/AndroidRuntime(1906):   at
android.view.ViewRootImpl.requestLayout(ViewRootImpl.java:823)10-24 14:25:32.195:
E/AndroidRuntime(1906):   at android.view.View.requestLayout(View.java:15468)10-24
14:25:32.195: E/AndroidRuntime(1906):   at
android.view.View.requestLayout(View.java:15468)10-24 14:25:32.195:
E/AndroidRuntime(1906):   at android.view.View.requestLayout(View.java:15468)10-24
14:25:32.195: E/AndroidRuntime(1906):   at
android.view.View.requestLayout(View.java:15468)10-24 14:25:32.195:
E/AndroidRuntime(1906):   at
android.widget.RelativeLayout.requestLayout(RelativeLayout.java:318)10-24 14:25:32.195:
E/AndroidRuntime(1906):   at android.view.View.requestLayout(View.java:15468)10-24
14:25:32.195: E/AndroidRuntime(1906):   at
android.widget.TextView.checkForRelayout(TextView.java:6313)10-24 14:25:32.195:
E/AndroidRuntime(1906):   at android.widget.TextView.setText(TextView.java:3567)10-24
14:25:32.195: E/AndroidRuntime(1906):   at
android.widget.TextView.setText(TextView.java:3425)10-24 14:25:32.195:
E/AndroidRuntime(1906):   at android.widget.TextView.setText(TextView.java:3400)10-24
14:25:32.195: E/AndroidRuntime(1906):   at
ch.schoeb.day3_demo_threading.MainActivity$1$1.run(MainActivity.java:25)

Paixon

Access to UI Elements from background thread not allowed
• Even if it seems to work sometimes!

Possibilities to "dispatch"
• Handler.post(Message)
• runOnUiThread(new Runnable(…)) on Context

- Simplified background-worker

- Executed in own thread (from thread pool)

- Callback-Methods on UI-Thread

- Abstract basic class

Paixon

```java
public class Worker extends AsyncTask<Params, Progress, Result>
{
        protected Result doInBackground(Params... param1) {
                return theResult;
        }

        protected void onPreExecute() {
        }

        protected void onPostExecute(Result result) {
        }

        protected void onProgressUpdate(Progress... values) {
        }
}

new Worker().execute(Params param1);
```

UI THREAD

Paixon

- AsyncTask does survive Activity destroy (orientation change...)
- Order of Execution
  - Donat – Honeycomb → ThreadPool
  - \>= Honeycomb → Single Thread
  - Use executeOnExecutor() to define
- No error handling
- Compose multiple WebService calls

Recommended:

- Do run AsyncTasks in a Service
- Use RxJava, RxAndroid

Paixon

Aufgabe

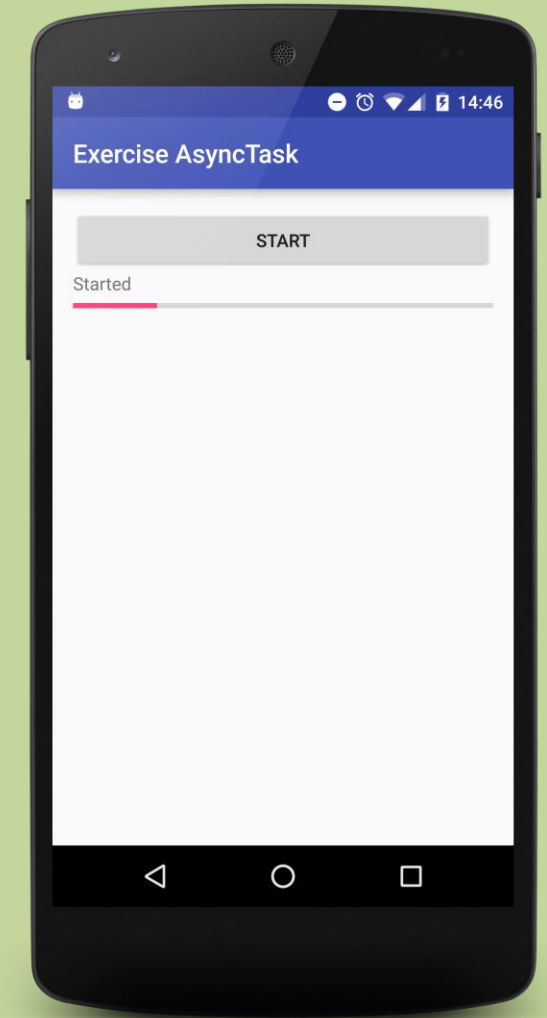- UI with Button, TextView and ProgressBar available
- When clicking on the Button:
  - Set progress to 0
  - Set textView to "Running"
  - Every second increase 10% up to 100%
  - When finished execute AsyncTask and write "finished" to the given TextView

Tipps
Use AsyncTask / Use callback-Methods to access UI from background-thread

Projekt: Exercise_AsyncTask

Paixon

External Libraries
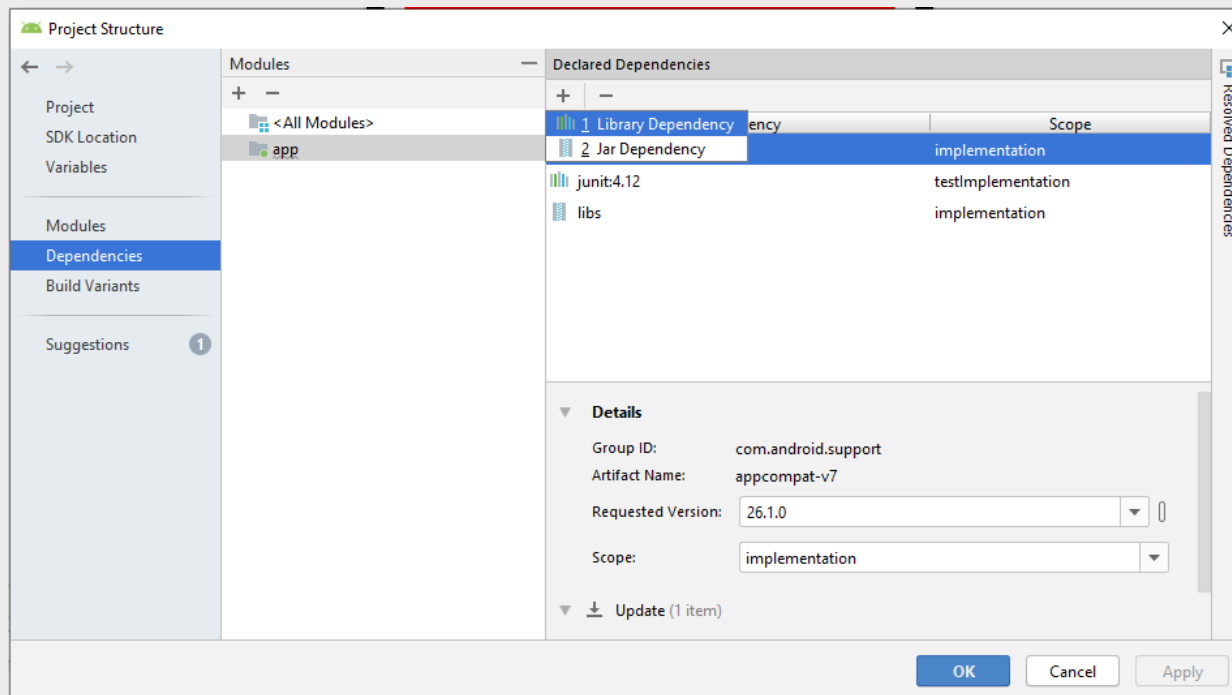
Paixon

Google

• Support Library

• Android Architecture Components

• Android Testing Support Library

• ...

Community

• Development Helper (Message-Bus, Dependency-Injection, ...)

• UI-Components (Calendar, Swiper, Navigation, ...)

• Data-Management (ORM, Rest-Libraries, ...)

• ...

Über die Benutzeroberfläche

File → Project Structure → Dependencies → Module(app)

External Library hinzufügen

Im Code

Eintrag in build.gradle

```
dependencies {
    // Füge alle jar-Files aus dem libs-Ordner hinzu
    implementation fileTree(dir: 'libs', include: ['*.jar'])

    // Füge die App-Support-Library hinzu
    implementation 'com.android.support:appcompat-v7:26.1.0'

    // Wenn für die Test kompiliert wird, füge JUnit hinzu
    testImplementation 'junit:junit:4.12'
}
```

Hinweis:
In älteren Gradle-Versionen wurde anstatt "implementation" das Keyword "compile" verwendet.

Paixon

- Multiple libraries provided by Google (Support Library / AndroidX)

Foundation	Architecture	Behavior	UI

Find the correct library you need

https://developer.android.com/jetpack/

Reduziert Boilerplate-Code in der Activity

• Keine findViewById-Aufrufe mehr nötig

• Keine Anonymen Inner-Classes für Click-Handler

```java
class ExampleActivity extends Activity {
        @BindView(R.id.user) EditText username;

        @BindString(R.string.login_error) String loginErrorMessage;

        @OnClick(R.id.submit) void submit() { // TODO call server... }

        @Override public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.simple_activity);
                ButterKnife.bind(this);
                // TODO Use fields...
        }
}
```
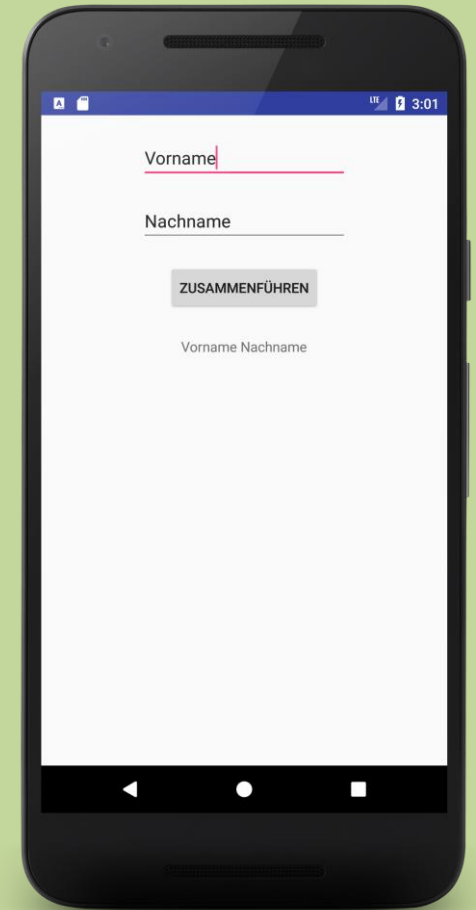
Paixon

Aufgabe

- Füge die "Butterknife"-Library hinzu
- Baue den Code so um, dass "findViewById" nicht mehr verwendet werden muss.

Library-Dokumentation

http://jakewharton.github.io/butterknife/

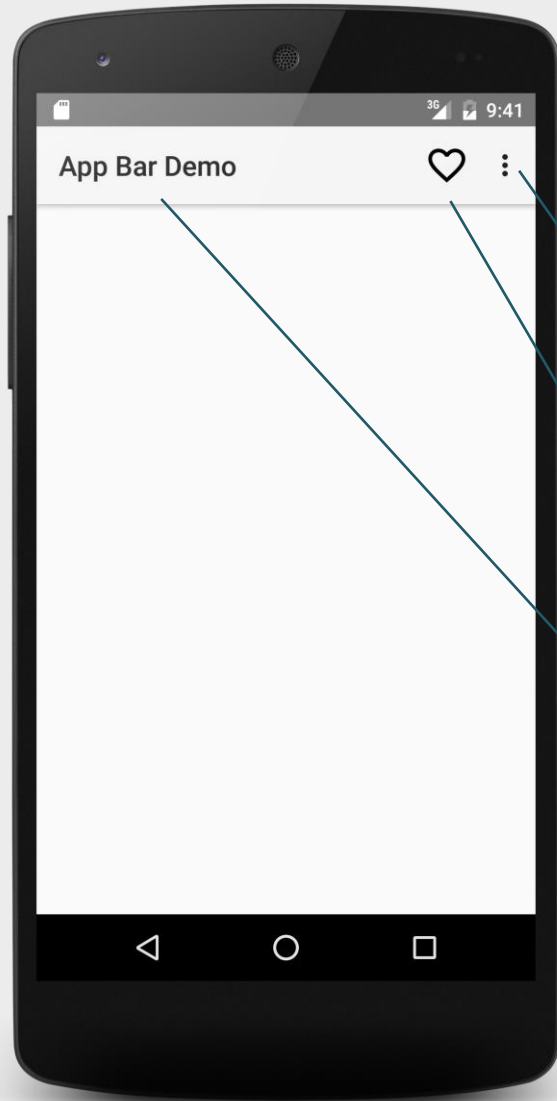## Projekt: Exercise_Butterknife

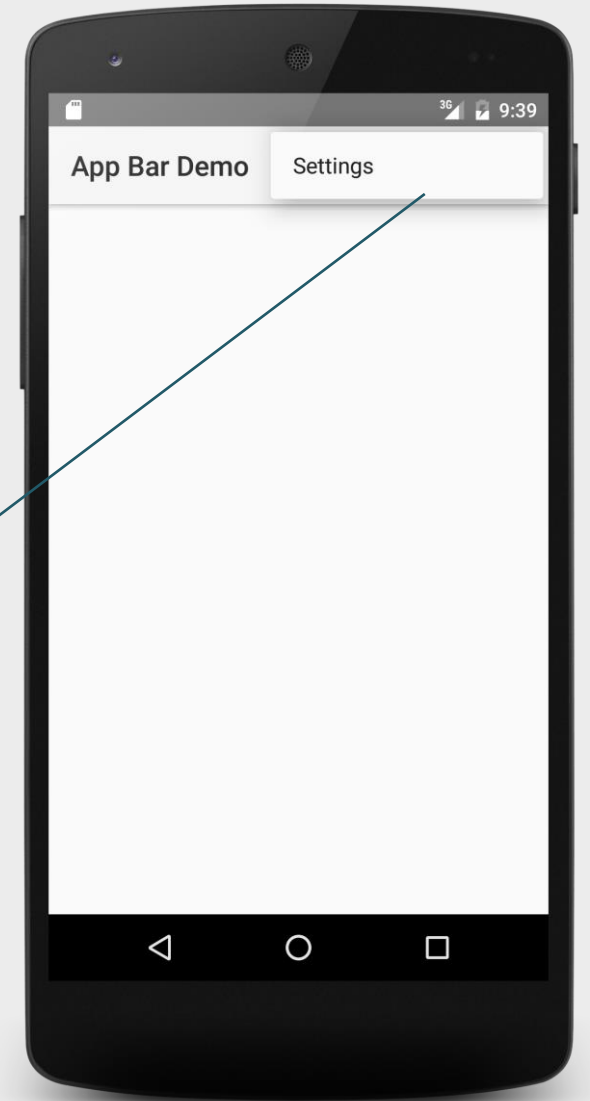# AppBar

aka ActionBar

AppBar

Paixon

Overflow

Menu Item

Label

Expanded Overflow

Paixon

- Available as ActionBar since Android 3.0

- Features added on every release
  - ActionBar behaves differently depending on what version of the android system

Use the Support-Library Toolbar to have consistent behaviour accross different android versions.

Paixon

Extend your Activity from AppCompatActivity

```java
public class MyActivity extends AppCompatActivity {
    // ...
}
```

Make sure your Activity does not already have an ActionBar

```xml
<application android:theme="@style/Theme.AppCompat.Light.NoActionBar" />
```

Paixon

Add a Toolbar to your layout

```xml
<android.support.v7.widget.Toolbar
    android:id="@+id/my_toolbar"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    android:background="?attr/colorPrimary"
    android:elevation="4dp"
    android:theme="@style/ThemeOverlay.AppCompat.ActionBar„ />
```

Tell the Activity where to find the Toolbar

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_my);
    Toolbar myToolbar = (Toolbar) findViewById(R.id.my_toolbar);
    setSupportActionBar(myToolbar);
}
```

Paixon

Define a Label

```
<activity android:name=".AppBarDemo"
    android:label="App Bar Demo">
</activity>
```

Paixon
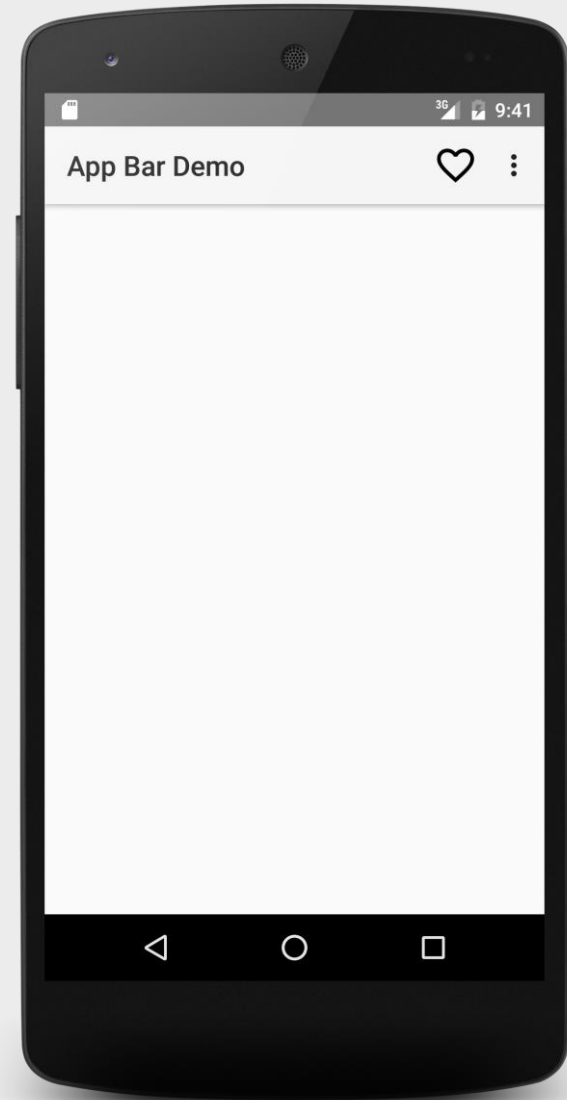
Define the menu

```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">

    <!-- "Mark Favorite", should appear as action button if possible -->
    <item
        android:id="@+id/action_favorite"
        android:icon="@drawable/ic_favorite_border_black_48dp"
        android:title="Favorite"
        app:showAsAction="ifRoom"/>

    <!-- Settings, should always be in the overflow -->
    <item android:id="@+id/action_settings"
        android:title="@string/action_settings"
        app:showAsAction="never"/>

</menu>
```

Paixon

Create & Handle menu in your Activity

```java
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_appbardemo, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_settings:
            // User chose the "Settings" item, show the app settings UI...
            return true;

        default:
            // If we got here, the user's action was not recognized.
            // Invoke the superclass to handle it.
            return super.onOptionsItemSelected(item);

    }
}
```

Paixon

Provide Up-Navigation

```xml
<application ... >
    ...

    <!-- The main/home activity (it has no parent activity) -->

    <activity
        android:name="com.example.myfirstapp.MainActivity" ...>
        ...
    </activity>

    <!-- A child of the main activity -->
    <activity
        android:name="com.example.myfirstapp.MyChildActivity"
        android:label="@string/title_activity_child"
        android:parentActivityName="com.example.myfirstapp.MainActivity" >

        <!-- Parent activity meta-data to support 4.0 and lower -->
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="com.example.myfirstapp.MainActivity" />
    </activity>
</application>
```

Paixon

Enable the Up Button

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_my_child);

    // my_child_toolbar is defined in the layout file
    Toolbar myChildToolbar = (Toolbar) findViewById(R.id.my_child_toolbar);
    setSupportActionBar(myChildToolbar);

    // Get a support ActionBar corresponding to this toolbar
    ActionBar ab = getSupportActionBar();

    // Enable the Up button
    ab.setDisplayHomeAsUpEnabled(true);
}
```

Paixon

Aufgabe

Die Übung beinhaltet bereits zwei Activities. Stelle sicher, dass über die ActionBar zur zweiten Activity navigiert werden kann und entsprechend der Up-Button zurück auf die erste Activity verweist.

Projekt: Exercise_AppBar