

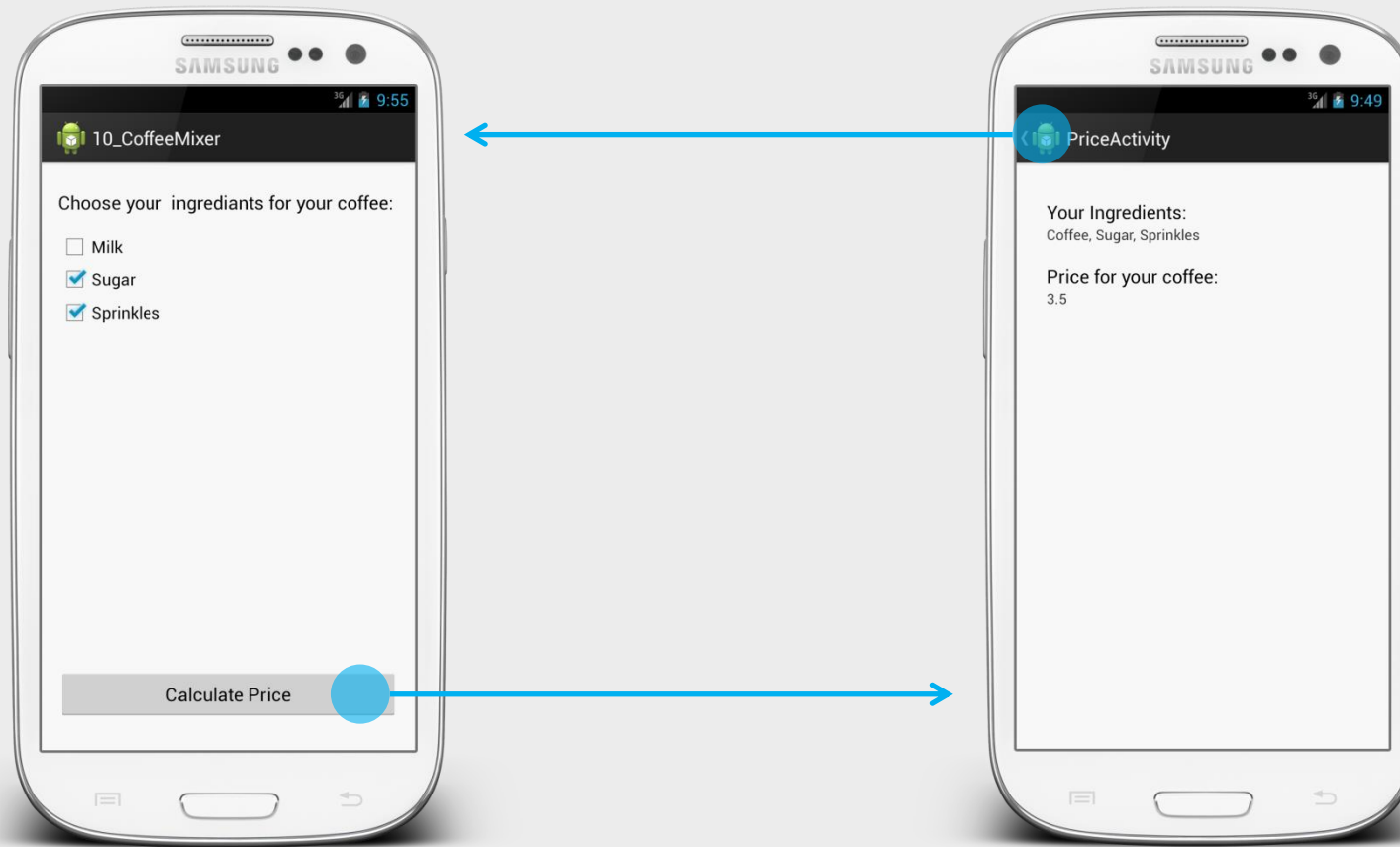
Recap

Activities	Einzelner Screen
Intents	Absicht etwas zu machen (z.B. Navigation A nach B)
LayoutParameter	Wie werden Views dargestellt im Layout
ViewGroups:	Platzierung von mehreren Views
LinearLayout	Horizontale/Vertikale Darstellung
RelativeLayout	Platzierung relative zu anderen Views oder dem Layout
AdapterViews	Datengetriebene Layouts

Fragments

Fragments – Why?

Paixon

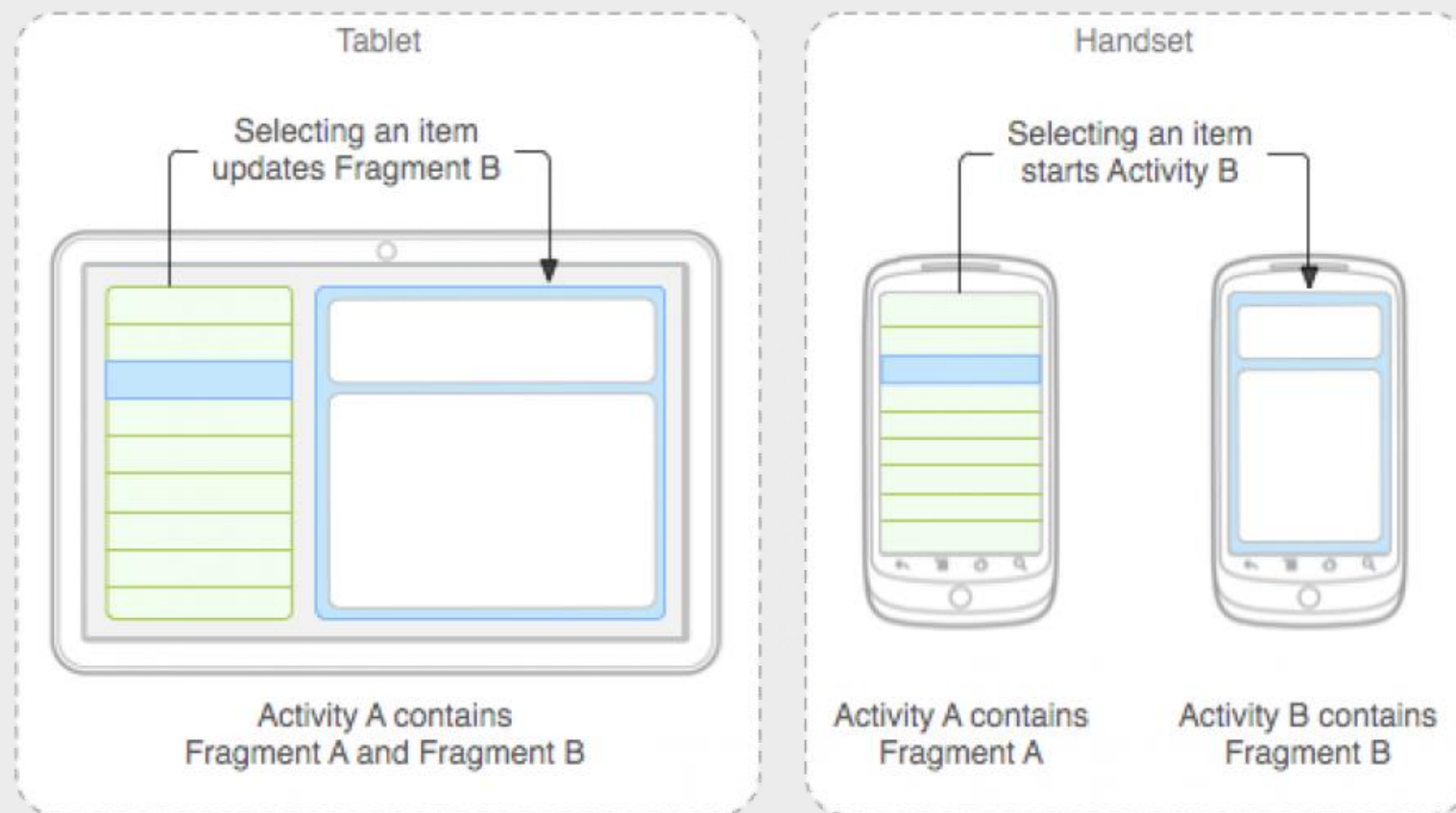


Fragments – Why?

Paixon



Show fragments depending on screen size



- 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

```
public static class ExampleFragment extends Fragment {  
  
    @Override  
    public View onCreateView( LayoutInflater inflater,  
                             ViewGroup container,  
                             Bundle savedInstanceState) {  
  
        return inflater.inflate(R.layout.mylayout, container, false);  
    }  
}
```

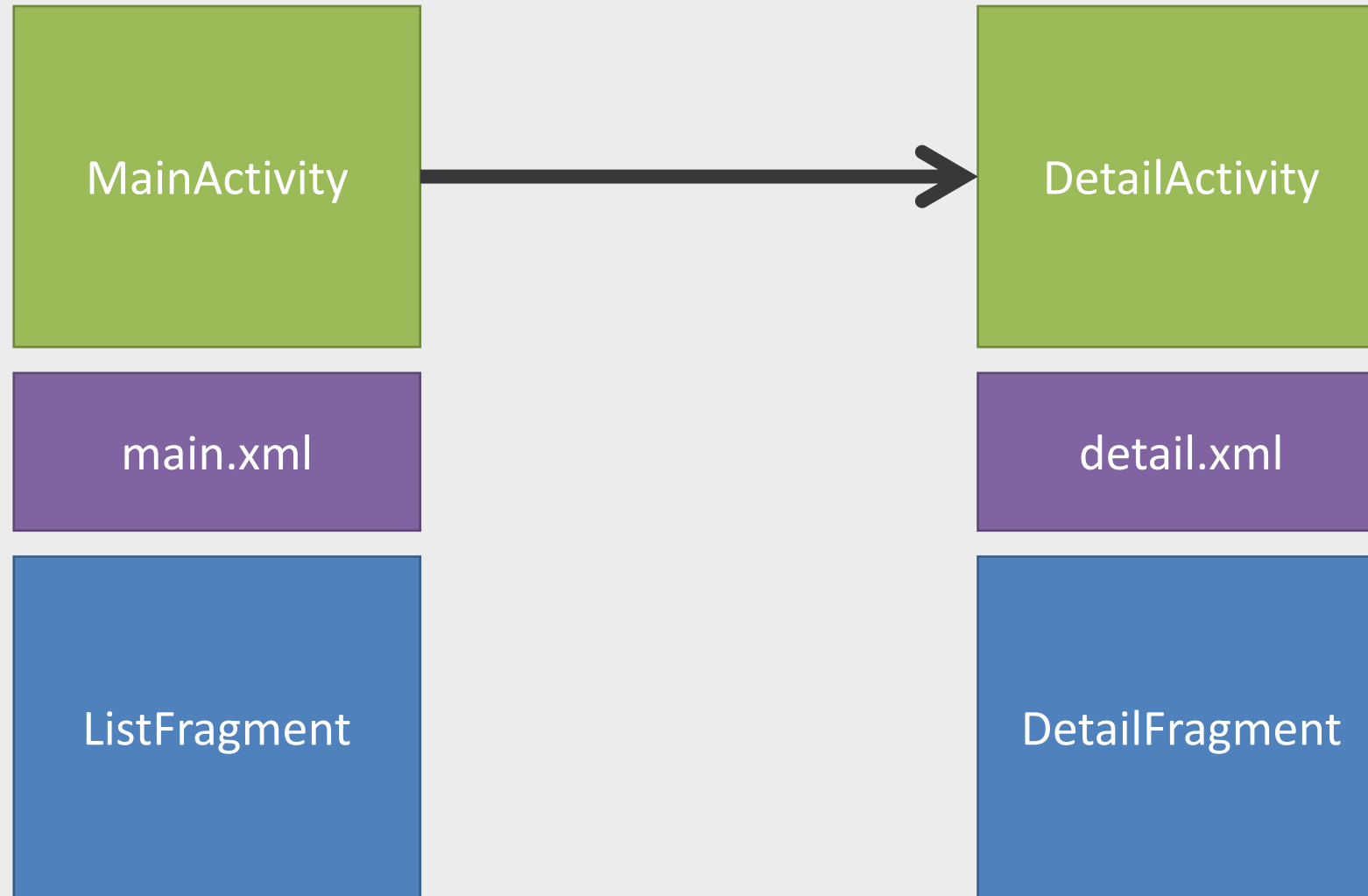

How to add a fragment using XML?

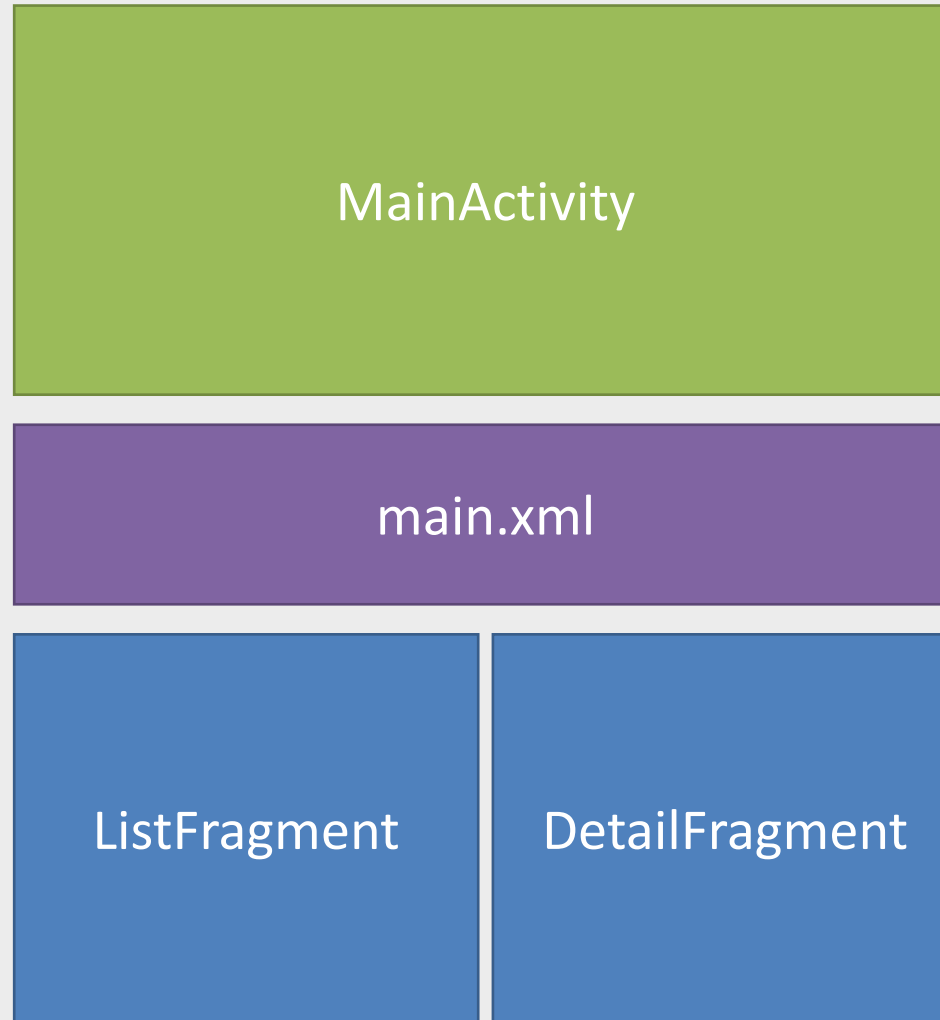
```
<fragment android:name="com.example.news.ArticleListFragment"
          android:id="@+id/theIdOfTheFragment"
          android:layout_width="match_parent"
          android:layout_height="match_parent" />
```

How to add a fragment using code?

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

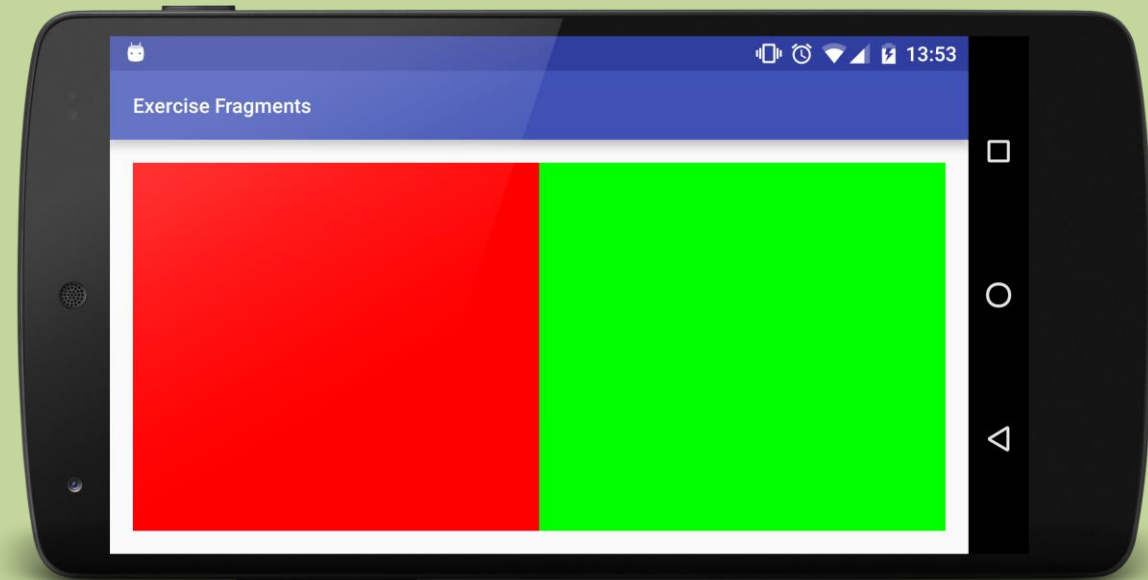
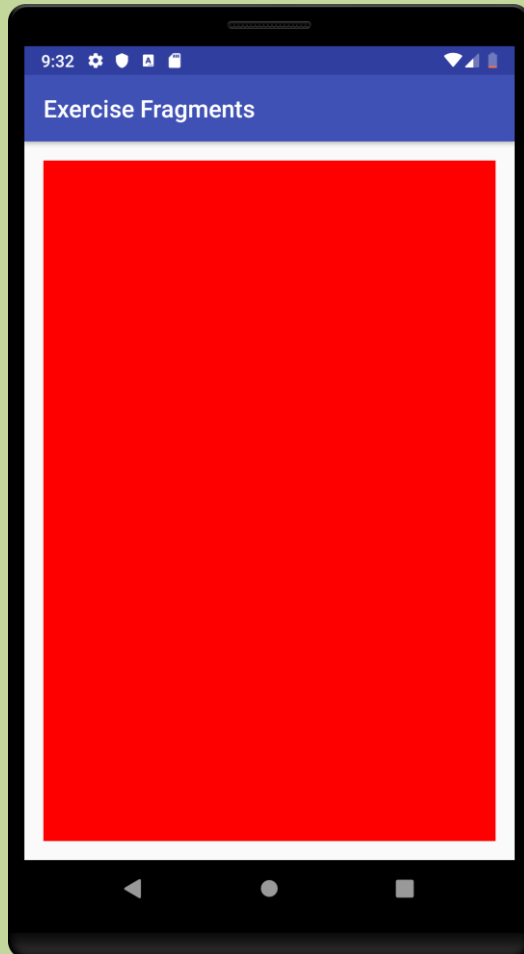
ExampleFragment fragment = new ExampleFragment();
fragmentTransaction.add(R.id.fragment_container, fragment);
fragmentTransaction.commit();
```





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



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.

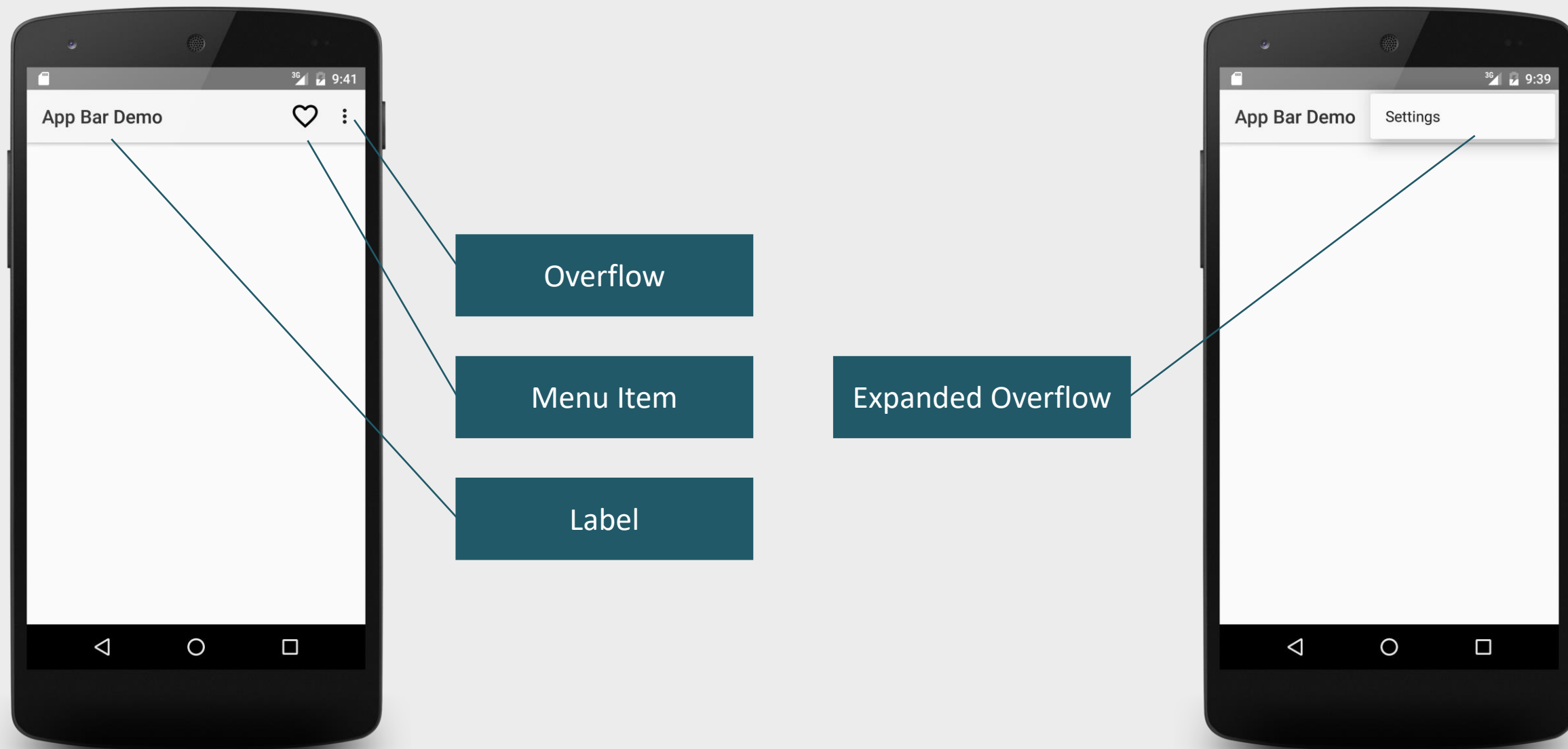
Projekt: Exercise_Fragments

Toolbar

aka ActionBar aka AppBar

AppBar

Paixon



Verfügbarkeit

- Android 5.0+ (API Level 21+)
- Ältere Versionen über Support Library

Extend your Activity from AppCompatActivity

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

Make sure your Activity does not already have an ActionBar

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

Add a Toolbar to your layout

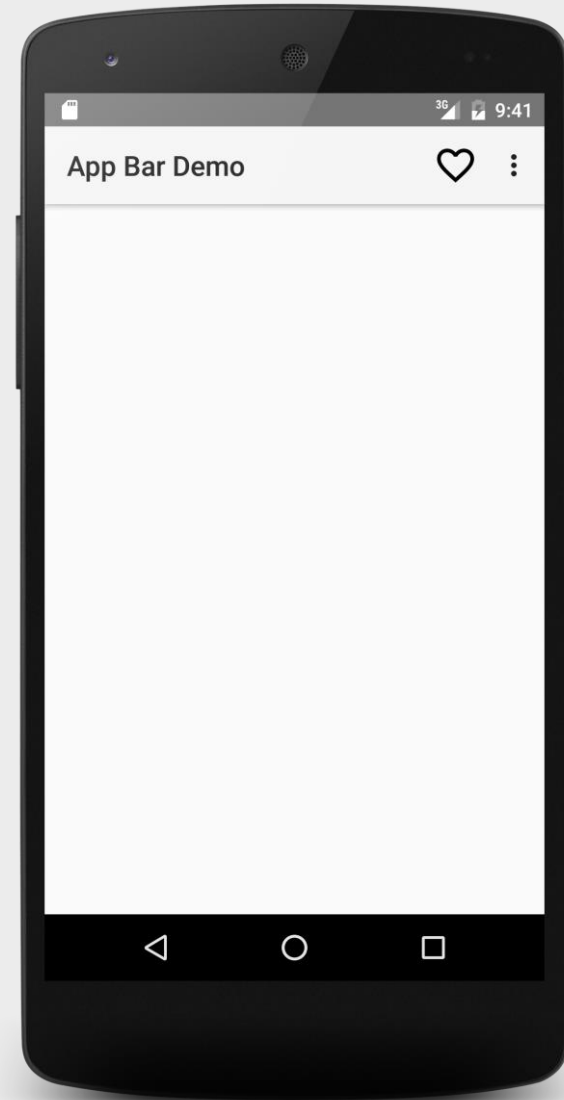
```
<android.support.v7.widget.Toolbar
    android:id="@+id/my_toolbar"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    android:background="?attr/colorPrimary"
    android:elevation="4dp"
    android:theme="@style/ThemeOverlay.AppCompat.ActionBar" />
```

Tell the Activity where to find the Toolbar

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

Define a Label

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



Define the menu

```
<?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>
```

Create & Handle menu in your Activity

```
@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);
    }
}
```

Provide Up-Navigation

```
<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>
```

Enable the Up Button

```
@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);
}
```


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

BroadcastReceiver

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

Extend abstract «BroadcastReceiver» class

- onReceive(Intent intent)
- Intent contains data

Register BroadcastReceiver in manifest-file

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

Register using Manifest

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

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

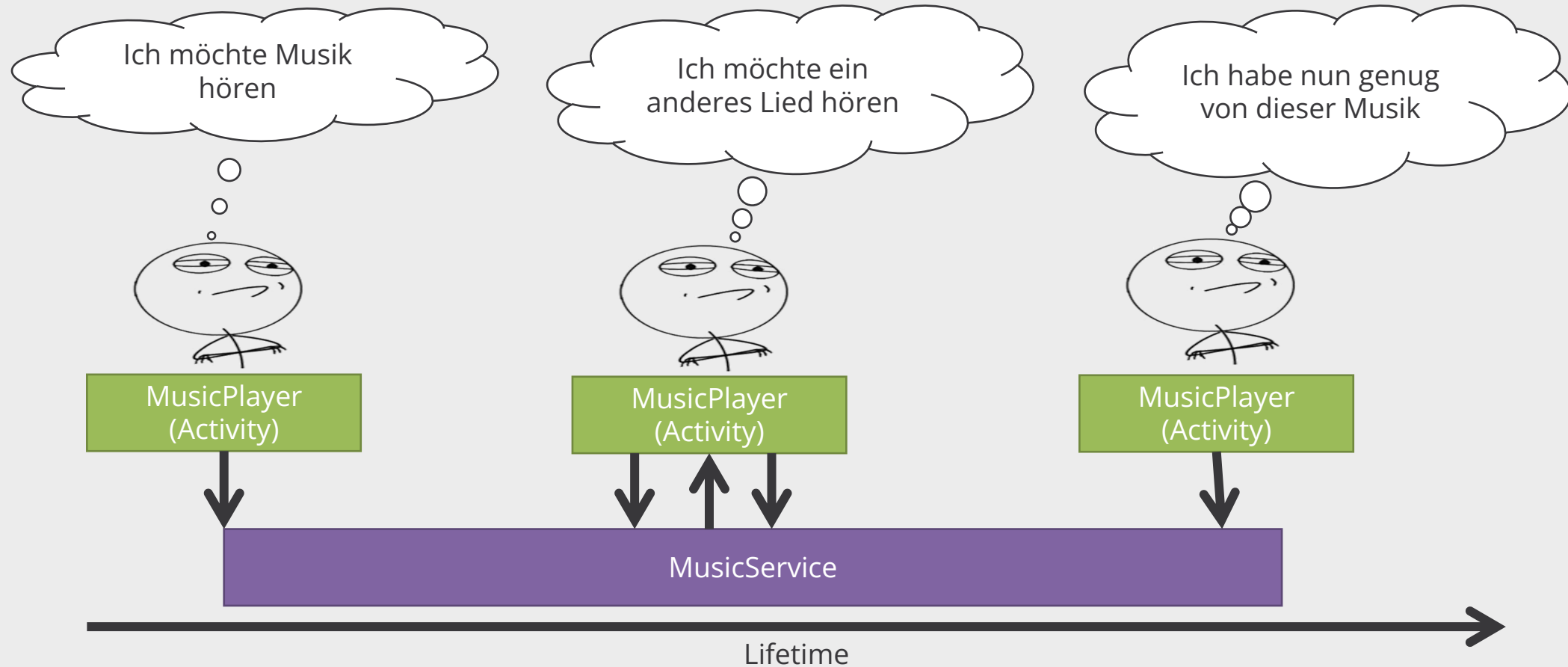
// Create Intent Filter for requested Event
IntentFilter filter = new IntentFilter();
filter.addAction(Intent.ACTION_AIRPLANE_MODE_CHANGED);

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

// unregister Receiver
this.unregisterReceiver(receiver);
```

- Registration in manifest is very restricted
- Do not block thread in receiver
- Too complicated for inter-component communication

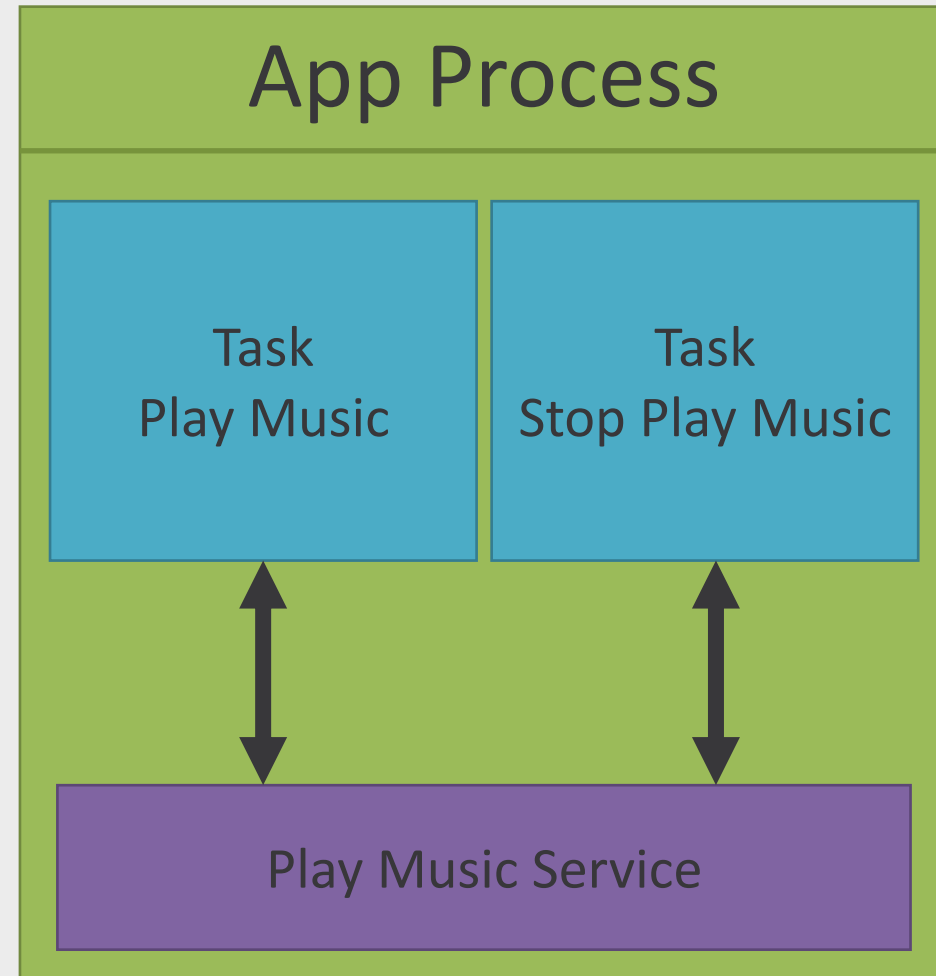
Service

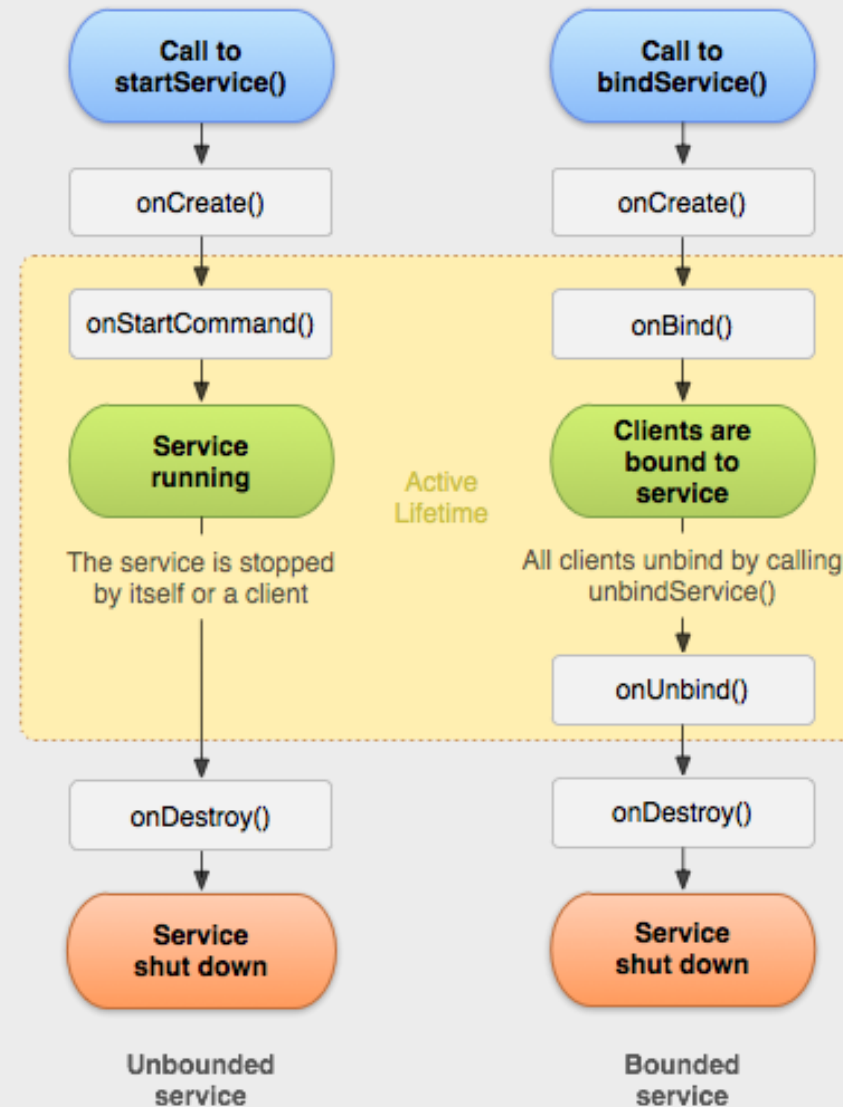


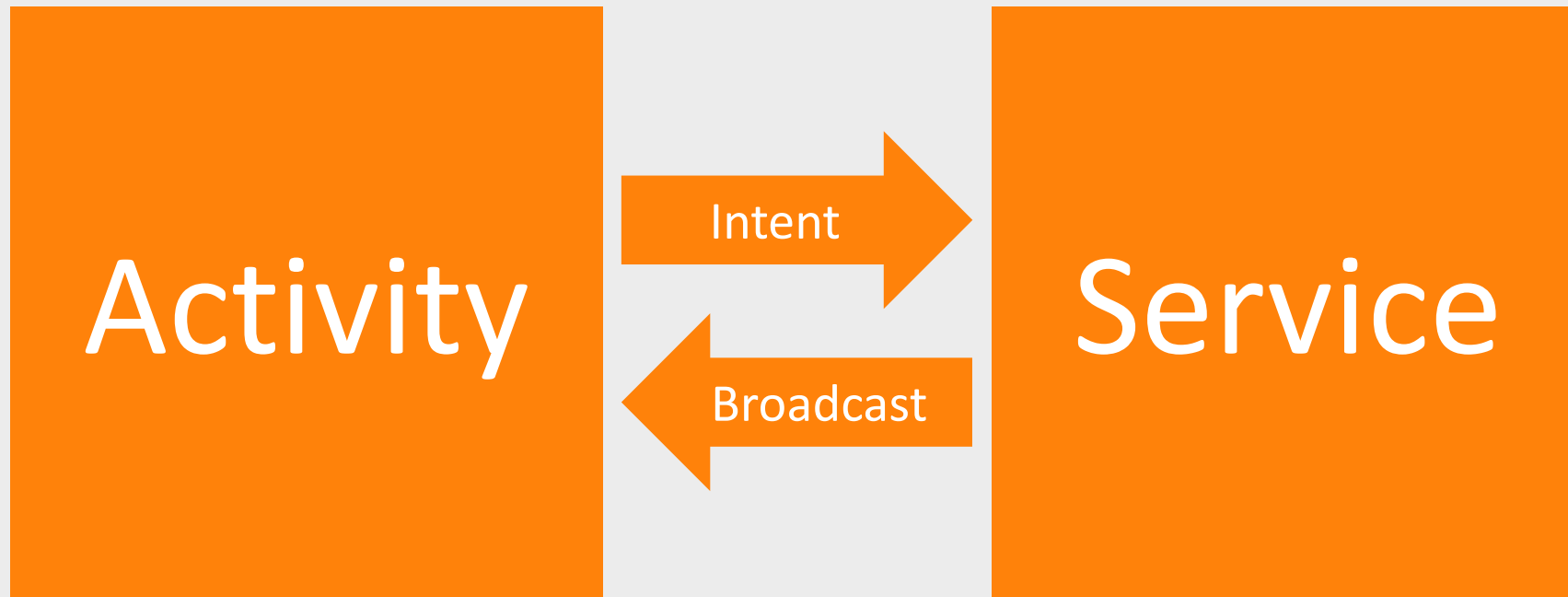
Component for long running operations

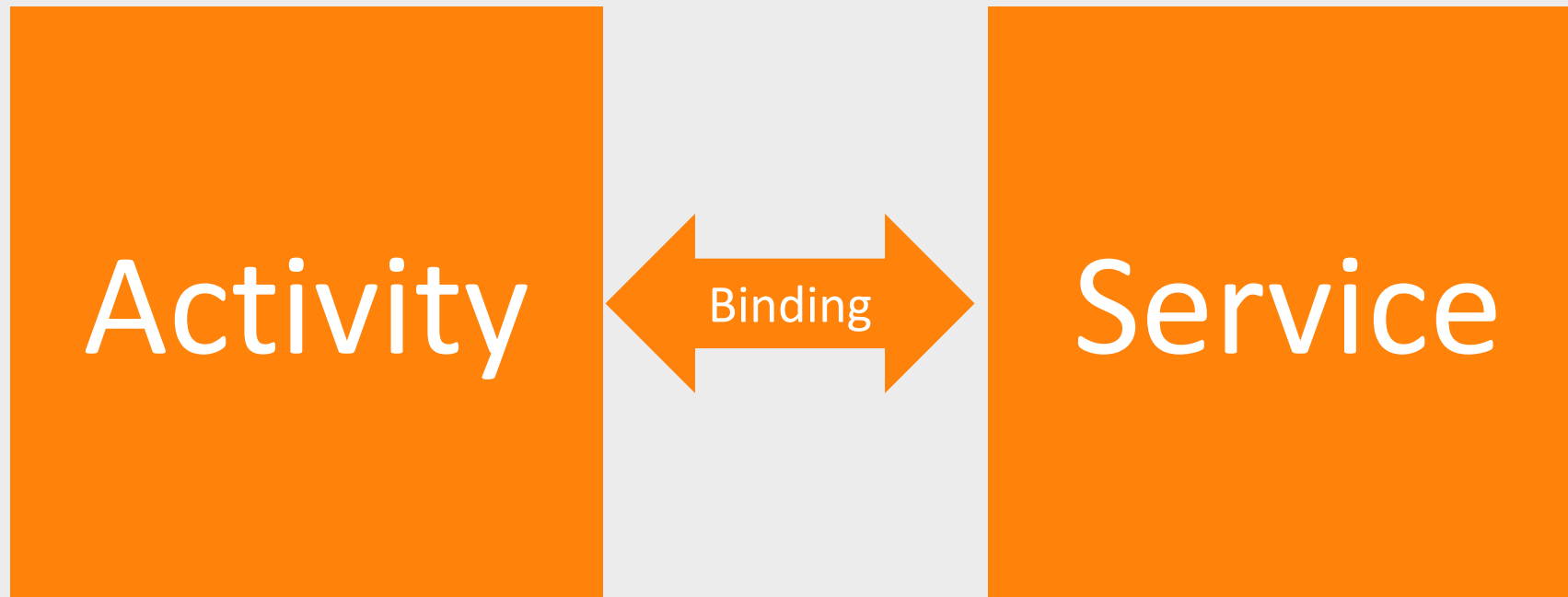
Bound Services	Started Services
Exists as long as any other component is bound to it	Exists as long as nobody stops it
Start use <i>bindService(...)</i>	Start use <i>startService(...)</i>
Communication through RPC	Communication through Broadcasts
Combined Services	
Start with <i>startService</i> → Services stays forever	
Connect to Service with <i>bindService(...)</i> to communicate	

- Class extending Service
- Register in Manifest
- Runs on UI Thread
- Service runs only once
- Stop the service
 - `stopService(Intent)`
 - `stopSelf()` in the service









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();
```

Create a Binder in the Service

```
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;  
}
```


Create a ServiceConnection in your Activity

```
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) {  
  
    }  
};
```

Use `bindService()` and `unbindService()` to connect/disconnect

```
@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);
}

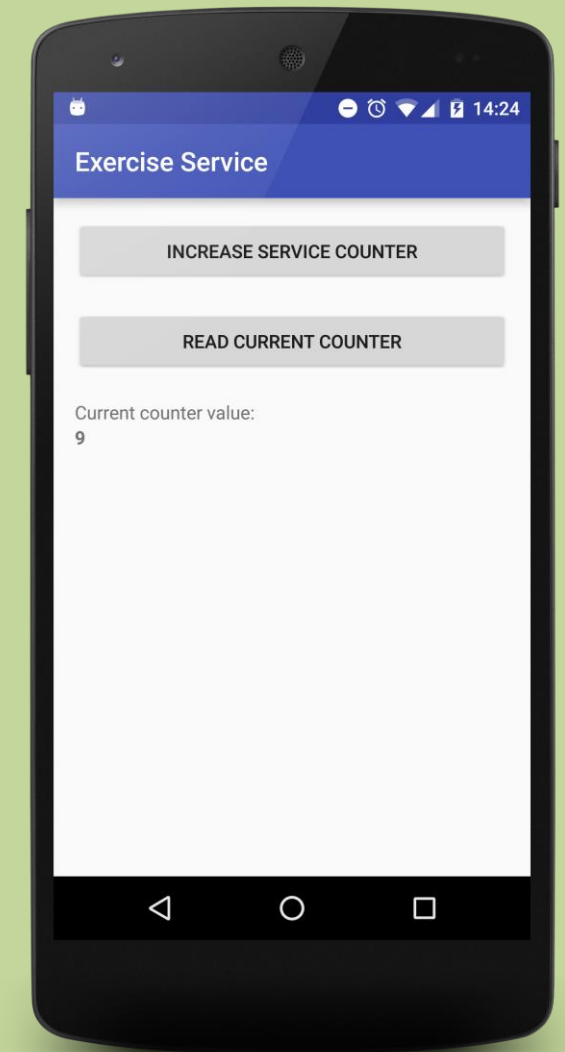
@Override
protected void onPause() {
    super.onPause();
    unbindService(serCon);
}
```

Aufgaben

Eine simple App mit zwei Buttons. Wenn “Increase service counter” gedrückt wird wird der Zähler um 1 erhöht. Wenn “Read current 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



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

Was ist Content?



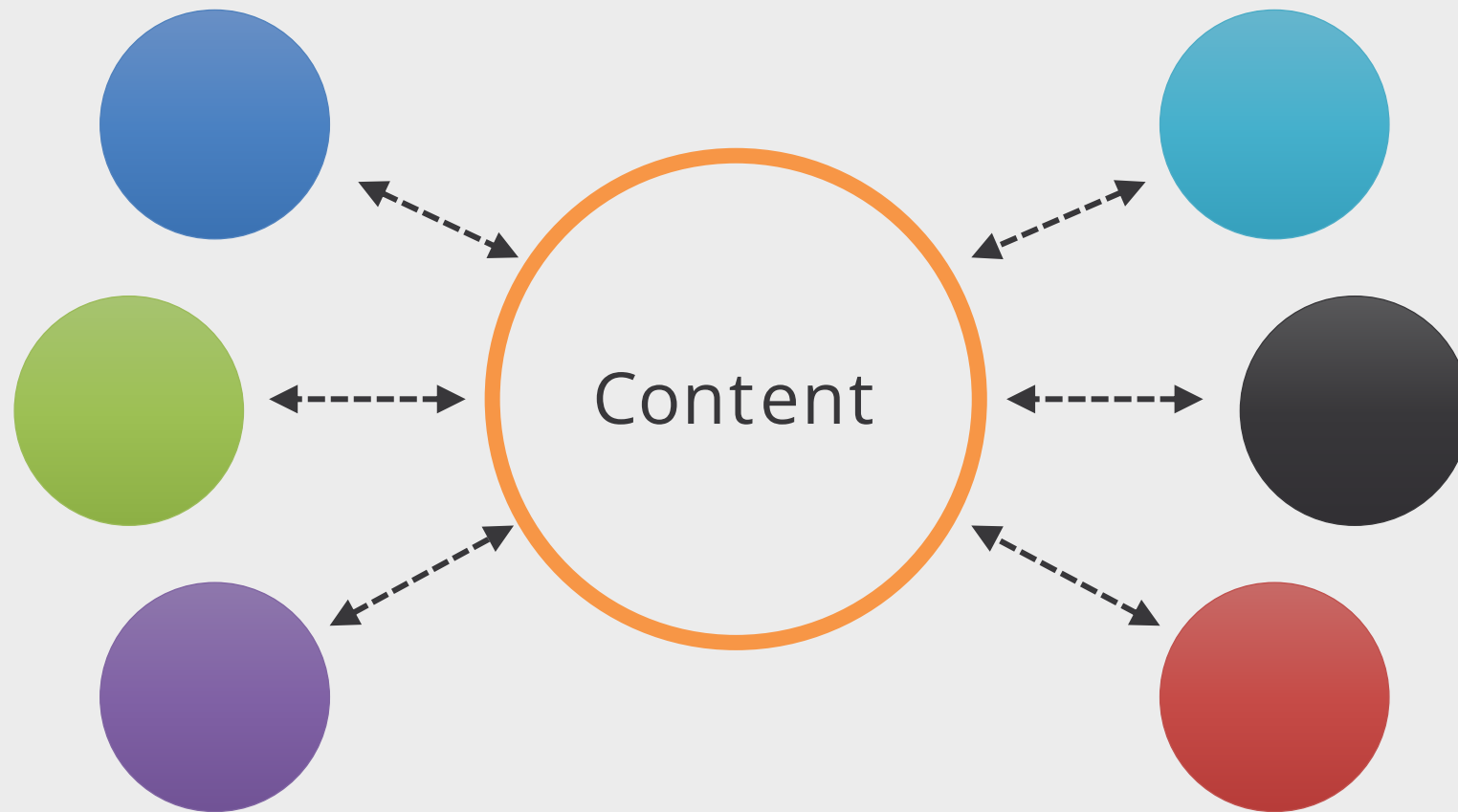
Kontakte



Bilder

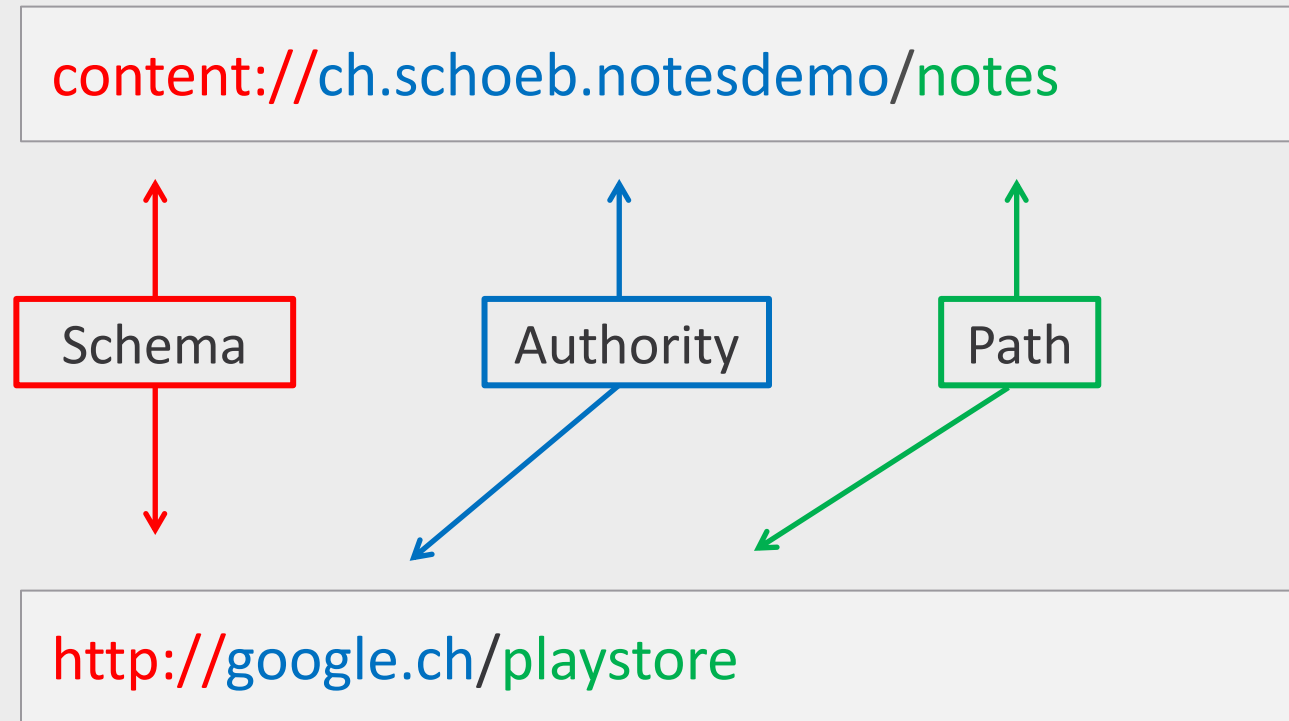
Glasprodukt	Lichtwerte (%)		Energie (%)				U _g -Wert (W/m ² K)
	Transmission	Reflexion	Transmission	Reflexion	Absorption	Gesamtenergie- durchlässigkeit	Argonfüllung
Isolierglasscheinheit (6 mm Außenscheibe – 16 mm Argon – 4 mm Pilkington Optifloat™ klar)							
Pilkington Suncool™ 70/40	71	10	39	28	33	43	1,1
Pilkington Suncool™ 70/35	70	16	35	35	31	37	1,0
Pilkington Suncool™ 66/33	66	16	33	35	32	36	1,0
Pilkington Suncool™ 60/30	60	19	30	36	34	32	1,0
Pilkington Suncool™ 50/25	50	19	24	33	43	27	1,0
Pilkington Suncool™ Blue 50/27	50	19	25	35	40	28	1,1
Pilkington Suncool™ Silver 50/30	50	39	29	43	29	32	1,0
Pilkington Suncool™ 40/22	40	20	20	35	45	23	1,1
Pilkington Suncool™ 30/17	30	26	16	37	47	19	1,1
Dreifachisolierglas (6 mm Außenscheibe – 12 mm Argon – 4 mm Pilkington Optifloat™ klar – 12 mm Argon – 4 mm Pilkington Optitherm™ S3)							
Pilkington Suncool™ 70/40	63	13	32	30	38	38	0,7
Pilkington Suncool™ 70/35	63	19	29	36	35	34	0,7
Pilkington Suncool™ 66/33	59	19	28	36	36	32	0,7
Pilkington Suncool™ 60/30	54	21	25	37	38	29	0,7
Pilkington Suncool™ 50/25	45	20	21	33	46	24	0,7
Pilkington Suncool™ Blue 50/27	45	20	21	36	43	25	0,7
Pilkington Suncool™ Silver 50/30	45	40	24	44	33	28	0,7
Pilkington Suncool™ 40/22	36	21	17	35	48	20	0,7
Pilkington Suncool™ 30/17	27	26	13	37	50	16	0,7

Daten



- 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, ...)

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



```
// 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>

Components

Activity

UI

Service

Background

ContentProvider

Provide data

BroadcastReceiver

Get notified about events

Threading

DEMO ANR

Application not Responding


- Single process
- Single UI Thread
- Every component is running on the UI Thread
- Create thread like in normal 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)

Access to UI Elements from
background thread is NOT allowed
Even if it seems to work sometimes!

- Simplified background-work
- Executed in own thread (from thread pool)
- Callback-Methods on UI-Thread
- Abstract basic class



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

    protected void onPreExecute() {
    }

    protected void onPostExecute(Result result) {
    }

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

Handler.post

```
Handler handler = new Handler(Looper.getMainLooper());  
handler.post(new Runnable() {  
    @Override  
    public void run() {  
        // Executed on UI Thread  
    }  
});
```

runOnUiThread

```
runOnUiThread(new Runnable() {  
    @Override  
    public void run() {  
        // Executed on UI Thread  
    }  
});
```

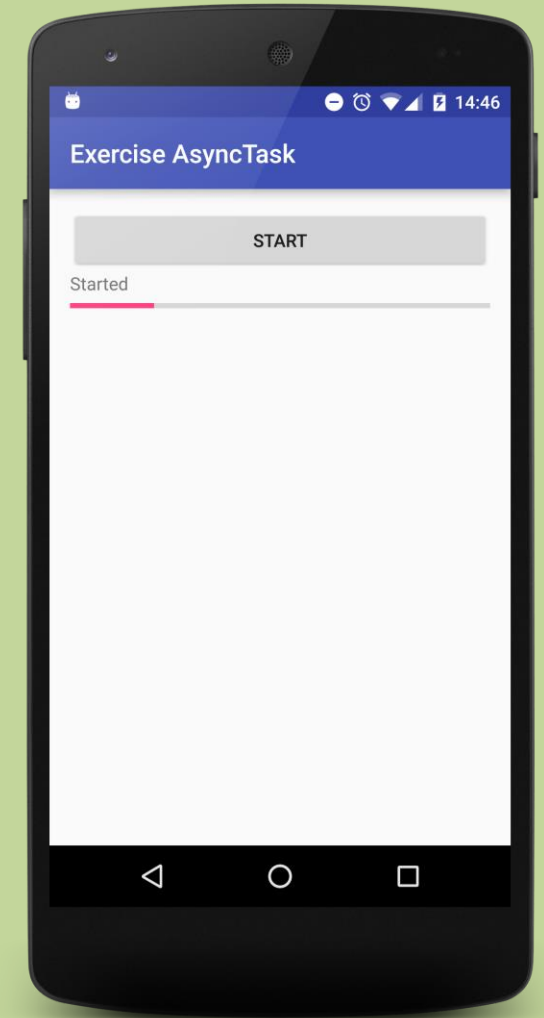
How to stop a thread?

```
Thread thread = new Thread(new Runnable() {  
    @Override  
    public void run() {  
        if(!Thread.currentThread().isInterrupted()){  
            // Do your stuff  
        }  
    }  
});  
thread.interrupt();
```

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 → write "finished" to the given TextView

Projekt: Exercise_AsyncTask



External Libraries

Google

- Support Library
- Android Architecture Components
- Android Testing Support Library
- ...

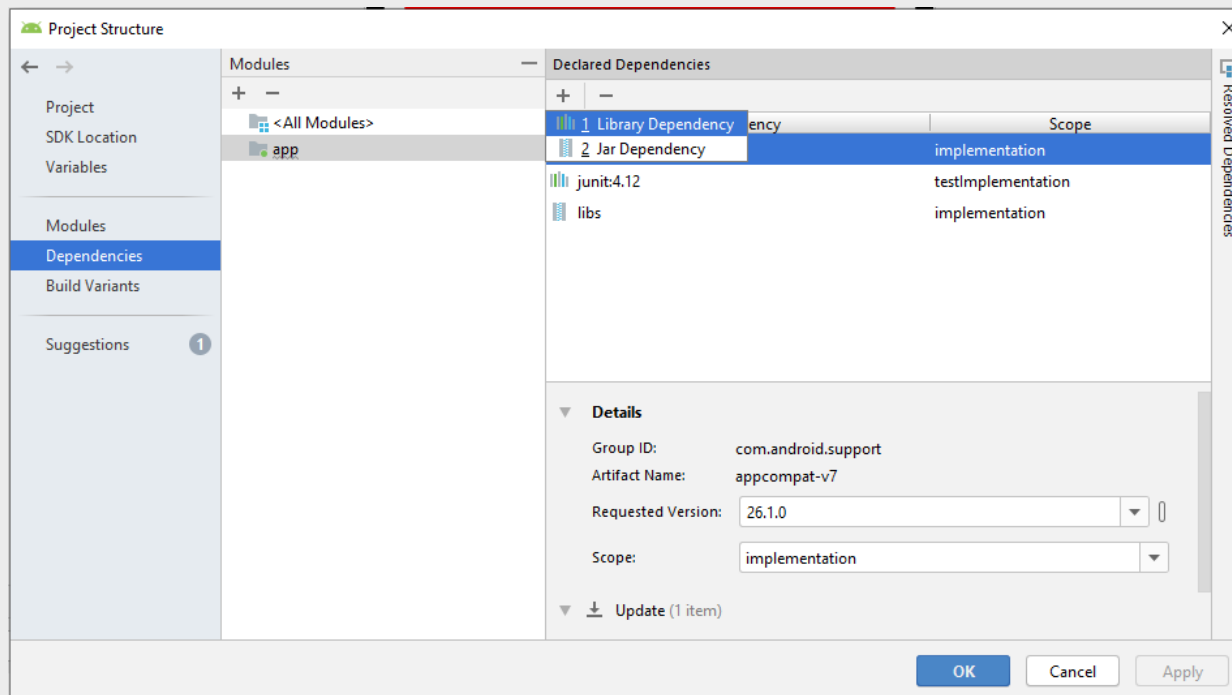
Community

- Development Helper (Message-Bus, Dependency-Injection, ...)
- UI-Components (Calendar, Swiper, Navigation, ...)
- Data-Management (ORM, Rest-Libraries, ...)
- ...

External Library hinzufügen

Über die Benutzeroberfläche

File → Project Structure → Dependencies → Module(app)



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.

Networking

Achtung:

Sämtliche Netzwerk-Operationen müssen in jedem Fall in einem Hintergrund-Thread ausgeführt werden.

Android bietet HttpURLConnection und HttpsURLConnection

```
URL url = new URL("http://paixon.ch");
HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
try {
    InputStream in = new BufferedInputStream(urlConnection.getInputStream());
    readStream(in);
} finally {
    urlConnection.disconnect();
}
```

- Raw Daten → Objekt
- Async-Issues
- Cancel von Aufrufen
- Wiederholende/Ersetzende Aufrufe
- Priorisierung
- Retry-Mechanismen
- Effizienz (Gzip, HTTP2, ...)

Es gibt Libraries die euer Leben vereinfachen

- OkHTTP
- Retrofit
- Volley

Retrofit

“A type-safe HTTP client for Android and Java”

- Open-Source Projekt entwickelt von Square
- Vom JSON-API zum typisierten Service
- Baisert auf OkHTTP (GZip, Caching, HTTP2, ...)

<http://square.github.io/retrofit/>

Schritt 1: Resource analysieren

Resource URL

<http://transport.opendata.ch/v1/connections>

----- Base URL-----/---- Path ----

Request-Parameters

Request Parameters

Name	Required	Description	Example
from	required	Specifies the departure location of the connection	Lausanne
to	required	Specifies the arrival location of the connection	Genève
via	optional	Specifies up to five via locations. When specifying several vias, array notation (<code>via[]=via1&via[]=via2</code>) is required.	Bern
date	optional	Date of the connection, in the format YYYY-MM-DD	2012-03-25

Schritt 2: Data Transfer Object erstellen

```
public class ConnectionDto {  
  
    public String from;  
    public Date fromTime;  
  
    public ArrivalLocationDto arrivalLocationInfos;  
    public List<LocationDto> stopOvers;  
}
```

- Aufzählungen werden als List<>'s dargestellt
- Verschachtelungen für Objekte

Schritt 3: Service-Interface für die Resource erstellen

```
public interface ConnectionService {  
  
    @GET("connections")  
    Call<ConnectionDto> searchConnections(@Query("from") String from, @Query("to") String to);  
  
}
```

Aufruf an:

<BASEURL>/connections?from={String from}&to={String to}

Schritt 4: Bootstrap Retrofit

```
Retrofit retrofit = new Retrofit.Builder()  
    .baseUrl("http://transport.opendata.ch/v1/")  
    .addConverterFactory(GsonConverterFactory.create())  
    .build();
```

Schritt 5: Service intanzieren

```
ConnectionService service = retrofit.create(ConnectionService.class);
```

Schritt 6: Call absetzen

```
service.searchConnections("Zürich", "Bern").enqueue(new Callback<ConnectionDto>() {  
    @Override  
    public void onResponse(Call<ConnectionDto> call, Response<ConnectionDto> response) {  
        if (response.isSuccessful()) {  
            ConnectionDto connection = response.body();  
            // Handle result...  
        }  
    }  
  
    @Override  
    public void onFailure(Call<ConnectionDto> call, Throwable t) {  
    }  
});
```

Async Call → Handler on Main Thread

Aufgaben

Part 1:

Verstehe wie die externe Library "Retrofit" eingebunden ist

Part 2:

Zeige die nächste Abfahrts und Ankunfts-Zeit der Verbindung Zürich → Bern an

Part 3:

Überlege dir, welche Informationen du für deine App benötigst. Baue die DTO's entsprechend auf.

Projekt: Exercise_Network

