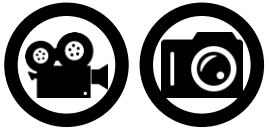
Android-Basics

Wintersemester 2018/2019

Verena Ebert

You can

copy, share and change,



film and photograph,

blog, live-blog and tweet

this presentation given that you attribute it to its author and respect the rights and licenses of its parts.

Allgemeines

- https://developer.android.com/guide/index.html
- Übungen im Ilias für Android-Grundlagen
- Laptop für die Übungen mitbringen

Lernziele

- Nach dieser Vorlesung sollten Sie…
  - … den prinzipiellen Aufbau einer Android-App kennen.
  - … ein Grundverständnis für die App-Entwicklung auf Android mit Java erlangt haben.

Schätzfrage

Auf wie viel Prozent der 2018 verkaufen Smartphones läuft Android?

a) 49%

b) 61%

c) 86%

d) 95%

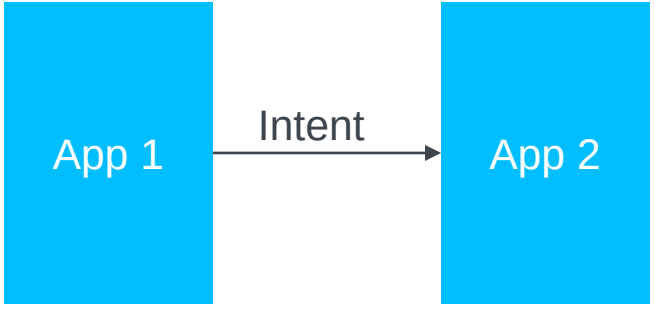
Was ist eigentlich Android?

- Betriebssystem für mobile Endgeräte basierend auf dem Linux Kernel
- Hauptsächlich entwickelt von Google
- Benutzerschnittstelle fokussiert auf direkte Manipulation (Touchscreens)
- Spezialisierte Versionen für
  - Fernseher (Android TV)
  - Automotive (Android Car)
  - Armbanduhren und wearable computing (Android Wear)

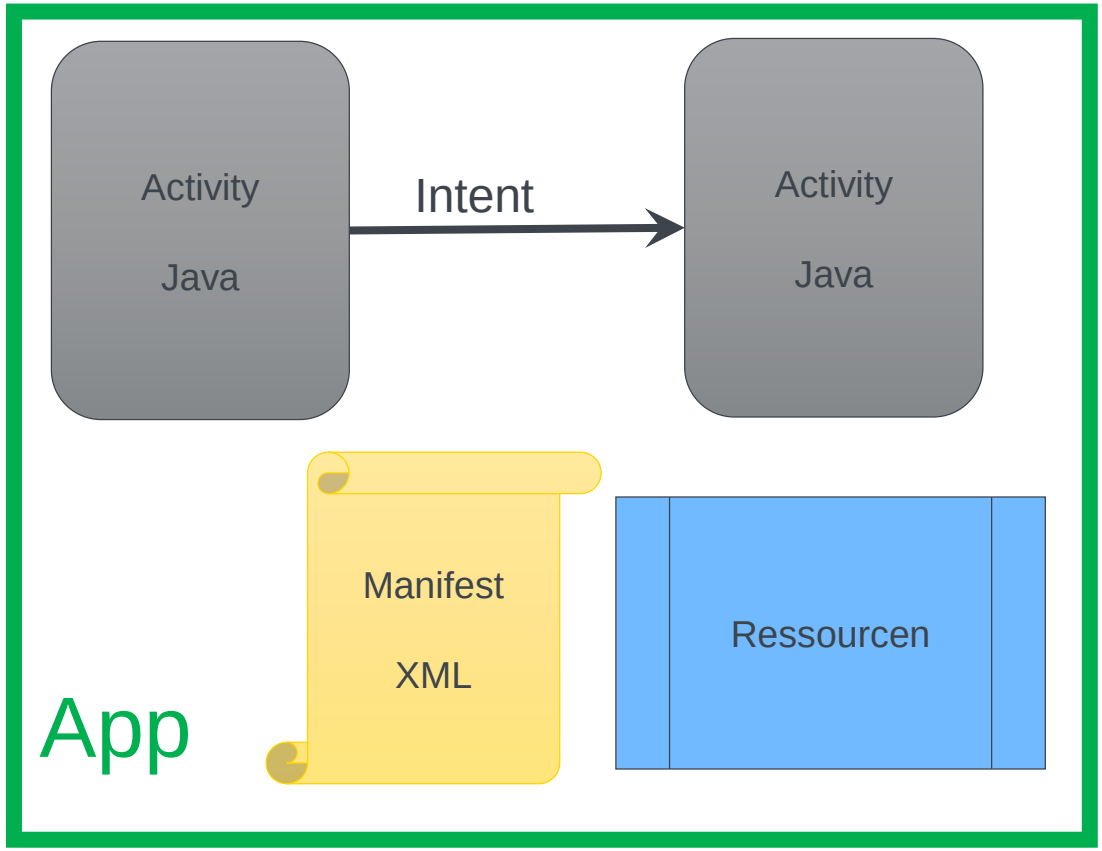
Der Aufbau von Android-Apps

- Mehrere Apps kommunizieren über Intents

➢ Prinzip der losen Kopplung



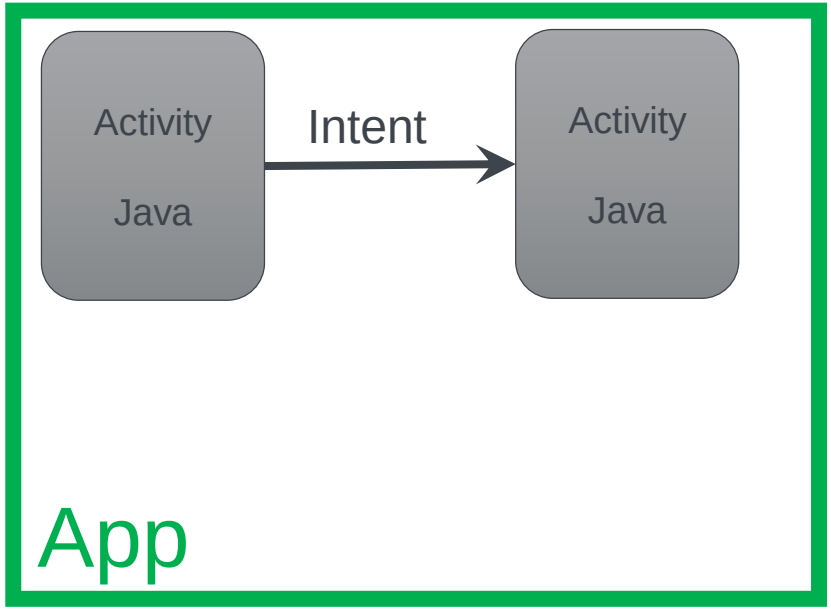
Der Aufbau von Android-Apps



Der Aufbau von Android-Apps

Activities

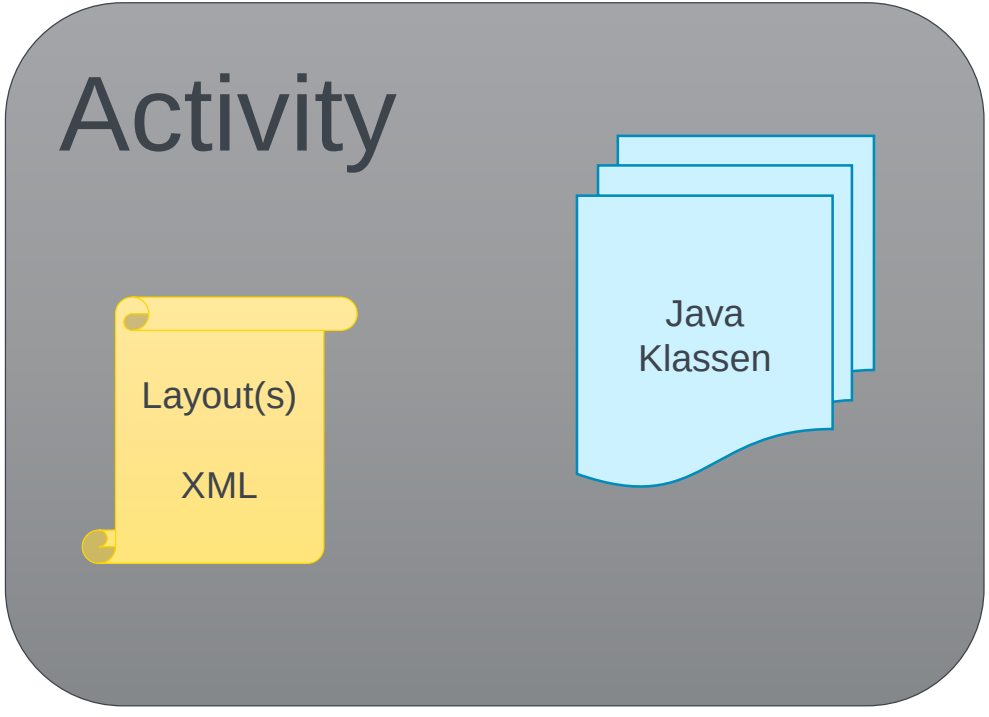
- Activitys sind die Einstiegspunkte in Apps
  - Für Benutzerinteraktion sowie als Ausführungseinheit
- Activities rufen sich über Intents auf.
  - Intent = Abstrakte Beschreibung, was getan werden soll (z.B. Foto machen)
- Es kann mehrere Activities geben, die die gewünschte Funktionalität bereitstellen.
- Bsp: Eine Activity zeigt den Posteingang, eine andere erzeugt eine neue E-Mail.



Der Aufbau von Android-Apps

Activities

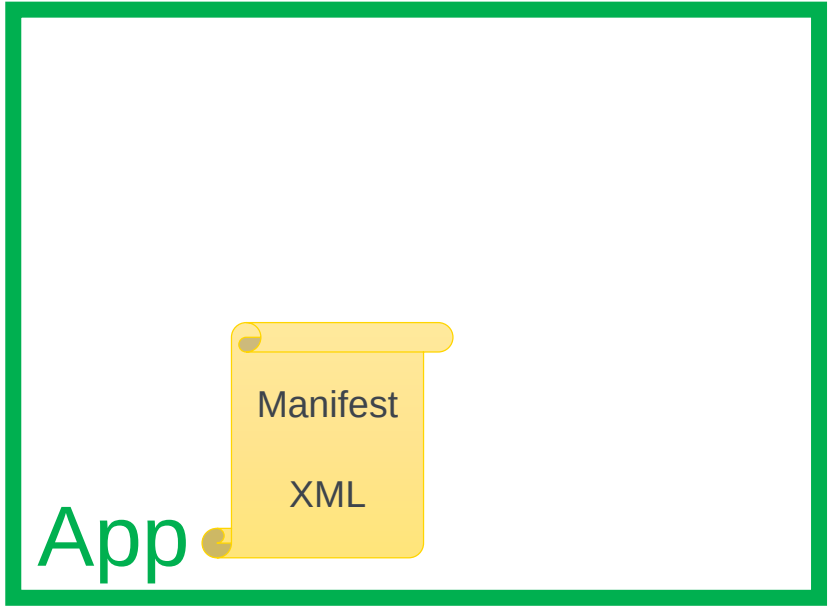
- Layout der Activity
- Java-Klassen für Verhalten



Der Aufbau von Android-Apps

Manifest

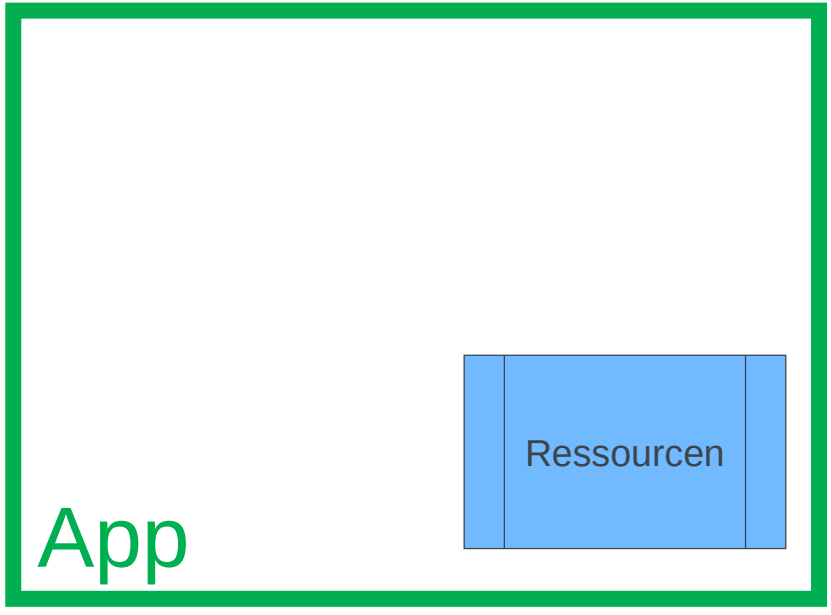
- Voraussetzungen / Anforderungen an die Ziel-Plattform
- API-Level
- Liste der Activities
- Berechtigungen
- …



Der Aufbau von Android-Apps

Ressourcen

- Layout-Dateien
- Bilder
- …

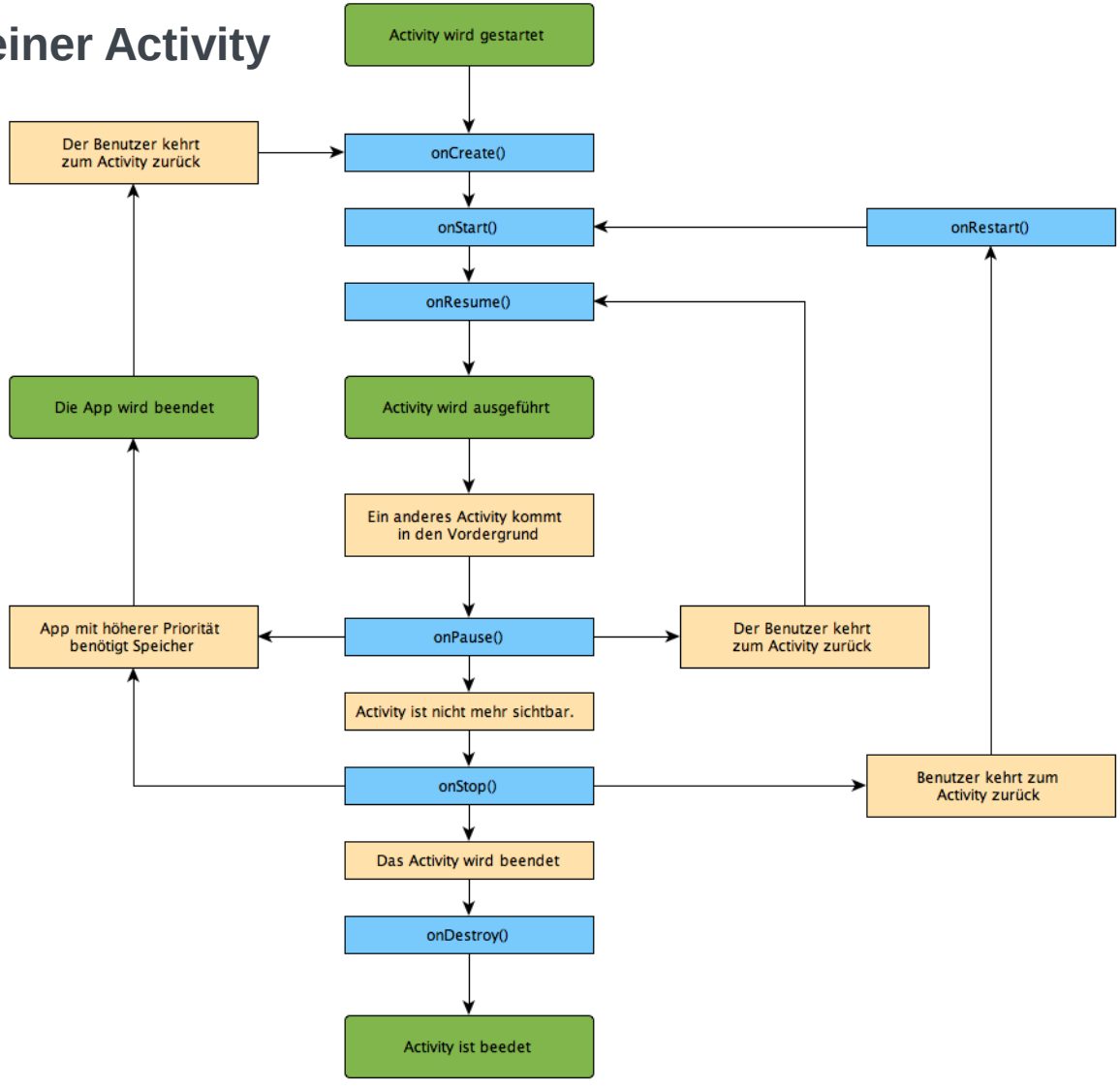


Der Aufbau von Android-Apps

Weitere Komponenten von Android

- Service
  - Für Hintergrund-Tasks oder lang laufende Aktitivitäten
- Content Provider
  - Für den Zugriff auf anwendungsübergreifende Datenspeicher
- Broadcast Receiver
  - Activity ohne Benutzerschnittstelle
  - Erhalten alle Messages der registrierten Intents, zB. Flugmodus wird eingeschaltet

Lebenszyklus einer Activity



Quelle: https://developer.android.com/guide/components/activities/activity-lifecycle.html

Aufgabe (5 min)

Überlegen Sie sich, was an den verschiedenen Stellen des Lebenszyklus in den Call-Backs beispielsweise passieren kann bzw. sinnvollerweise passieren sollte.

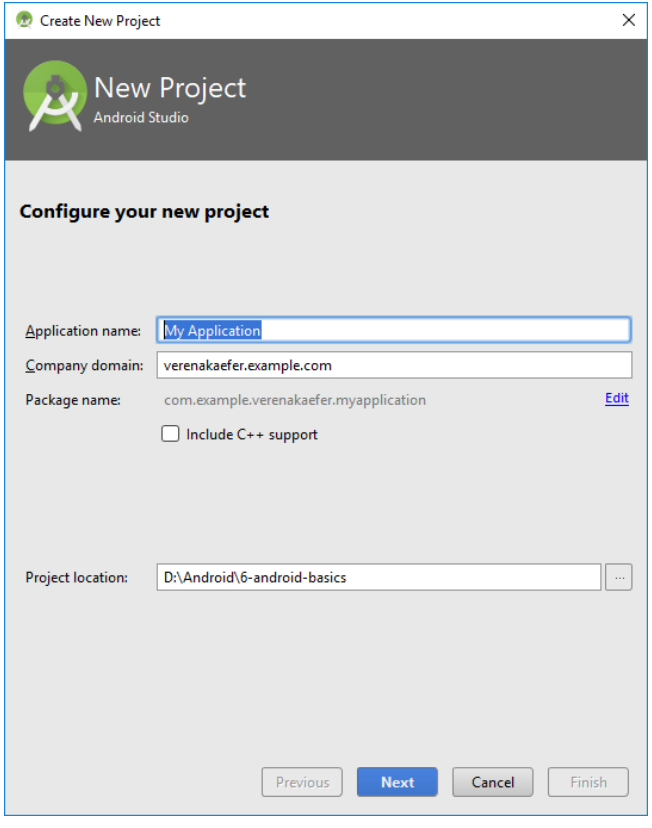
Meine erste App

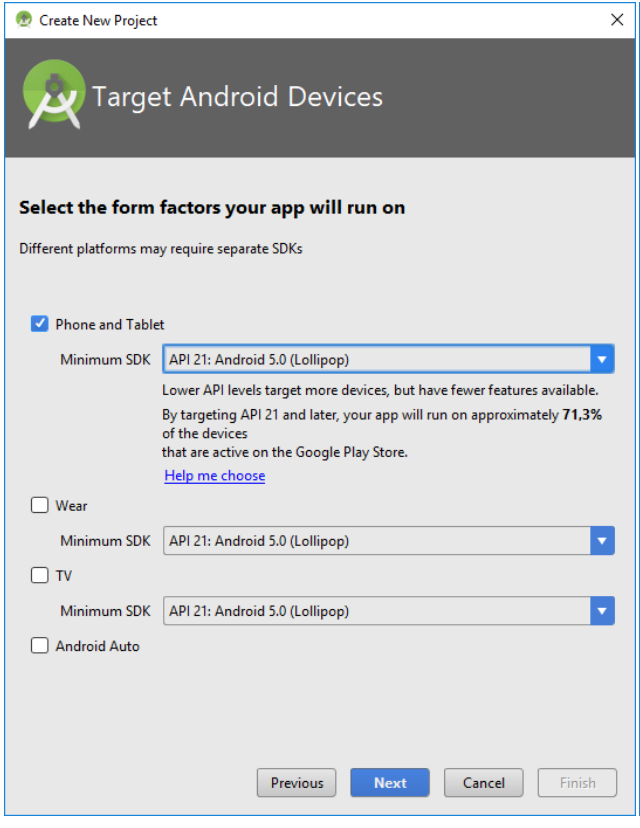
Android Studio 3.0

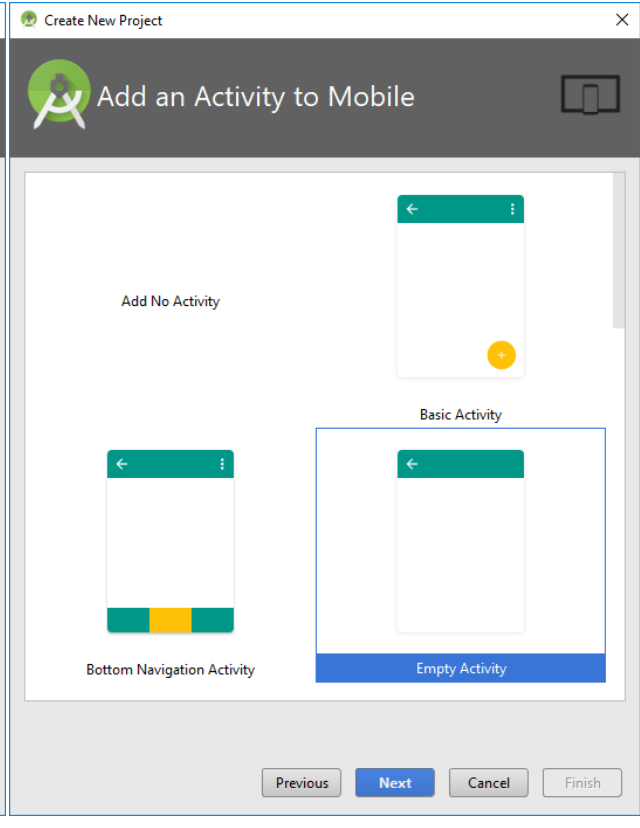
- In der Vorlesung wird Android Studio 3.2.1 verwendet
- **Achtung:** Projekte, die in AS 3.0 erstellt wurden, machen unter AS 2.3.3 Probleme

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:26.1.0'
    compile 'com.android.support.constraint:constraint-layout:1.0.2'
    compile 'com.android.support:design:26.1.0'
    testCompile 'junit:junit:4.12'
    androidTestCompile 'com.android.support.test:runner:1.0.1'
    androidTestCompile 'com.android.support.test.espresso:espresso-core:3.0.1'
    testCompile "org.mockito:mockito-core:+"
    androidTestCompile "org.mockito:mockito-android:+"
    androidTestCompile 'com.android.support:support-annotations:26.1.0'
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:27.0.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    implementation 'com.android.support:design:27.0.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.1'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'
    testImplementation "org.mockito:mockito-core:+"
    androidTestImplementation "org.mockito:mockito-android:+"
    androidTestImplementation 'com.android.support:support-annotations:27.0.0'
}
```

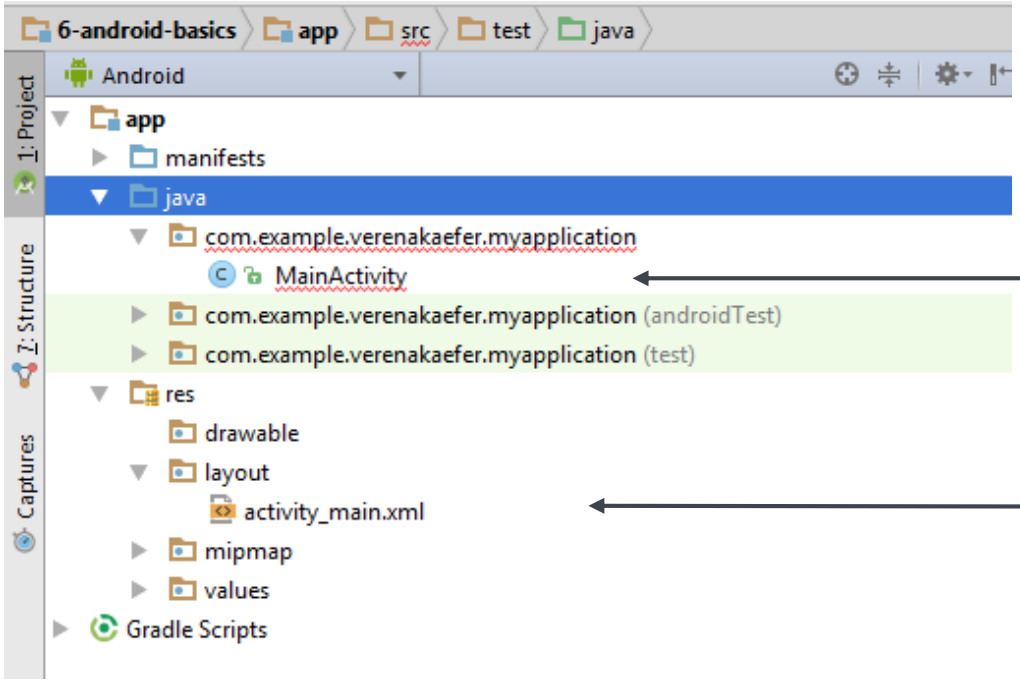
Meine erste App







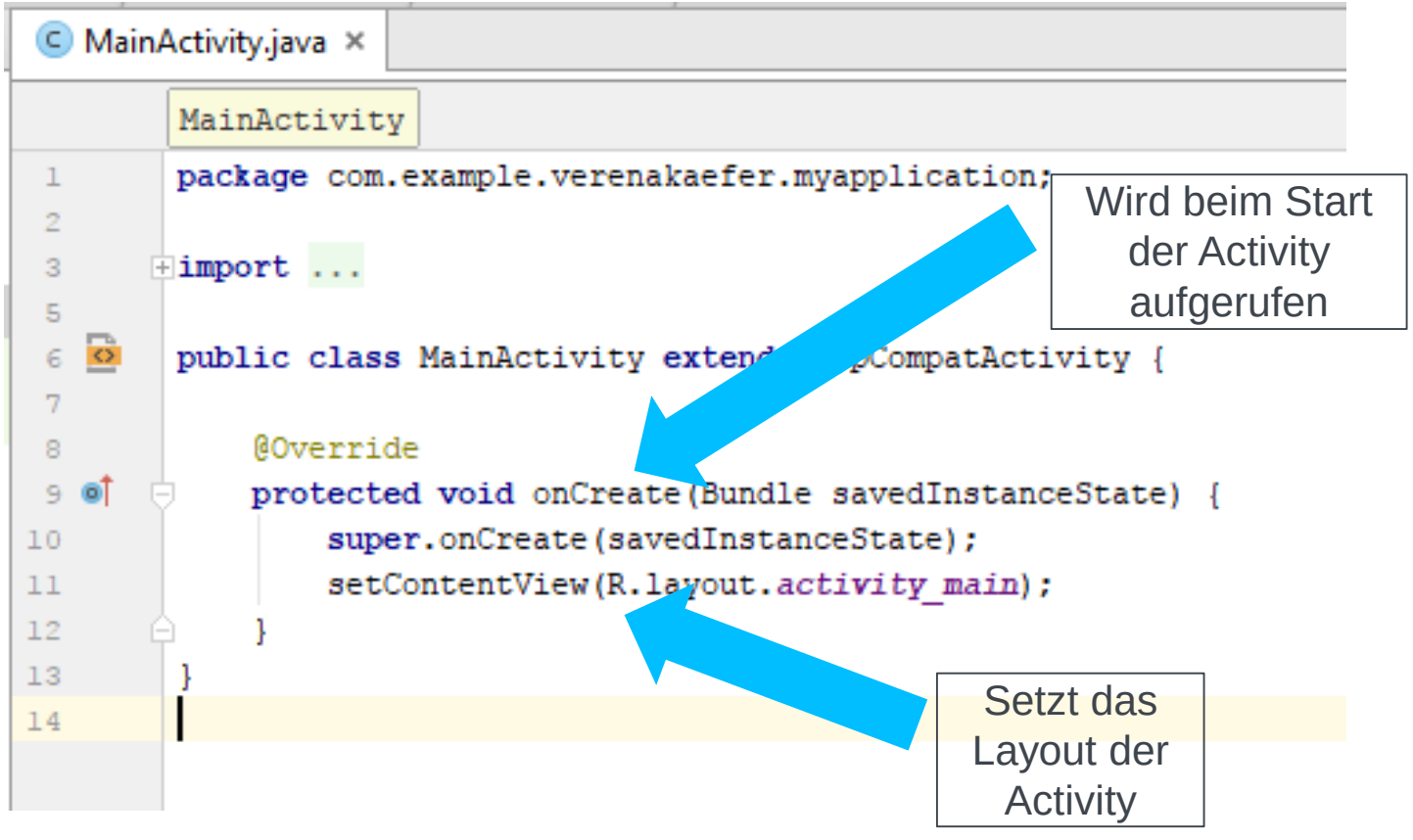
Meine erste App



Java-Code der Activity
-> Was soll die Activity tun?

XML_code der Activity
-> Wie soll die Activity aussehen?

Meine erste App



Meine erste App – Layout der Activity

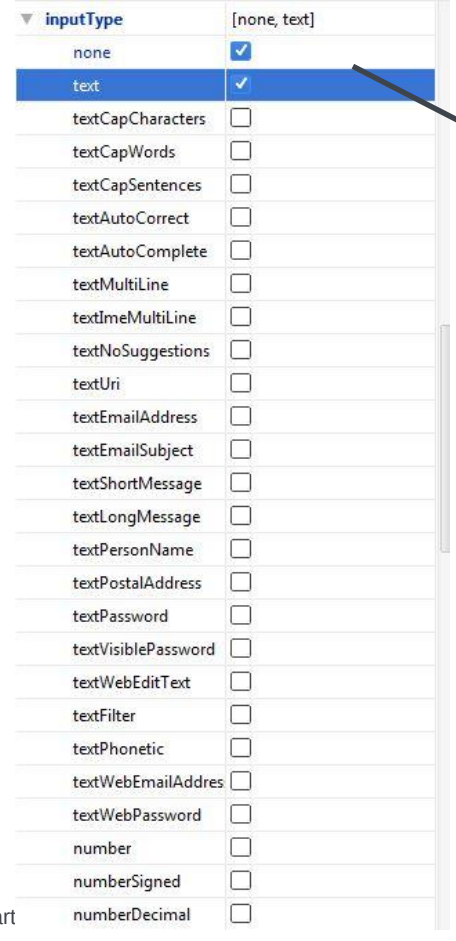
Meine erste App – Beispiel Text senden

Vorsicht: Kein Eingabetyp für Textfeld definiert und hart codierte Benamung des Buttons

```
    <EditText
        android:id="@+id/editText1"
        android:layout_width="163dp"
        android:layout_height="45dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="88dp"
        android:ems="10" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editText1"
        android:layout_centerHorizontal="true"
        android:text="Button" />

</RelativeLayout>
```

Meine erste App – Beispiel Text senden

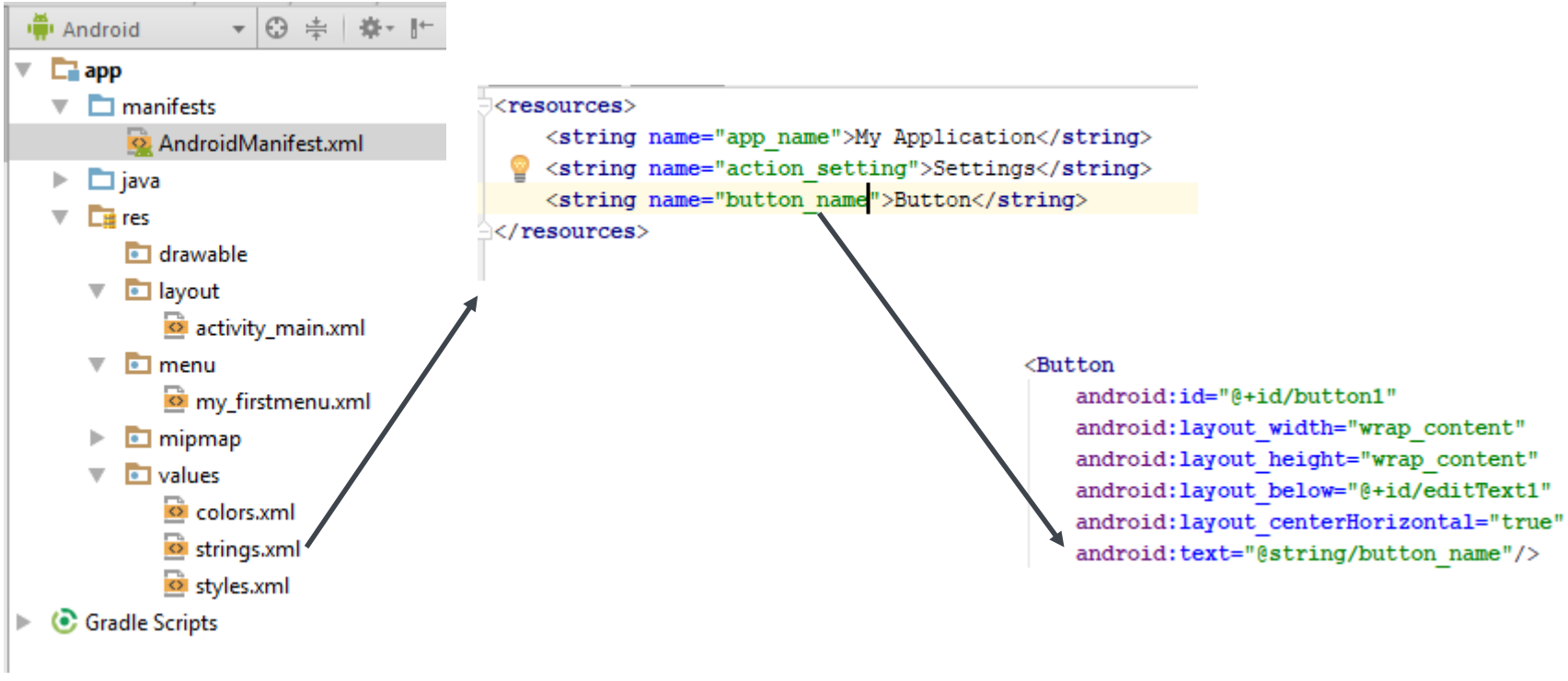
Eingabetyp für Textfelder




```
<EditText
    android:id="@+id/editText1"
    android:layout_width="163dp"
    android:layout_height="45dp"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="88dp"
    android:ems="10"
    android:inputType="none|text" />
```

Meine erste App – Beispiel Text senden

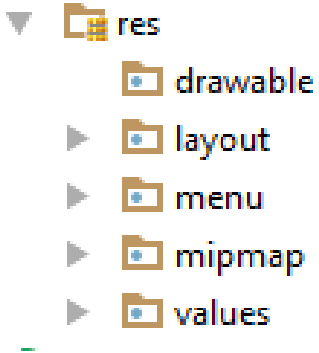
String-Lokalisation



Einfache Lokalisation durch Ersetzen der XML möglich

Meine erste App – Beispiel Text senden

Exkurs: Ressourcen



- Drawable
  - Bilder, Grafiken & Icons
- Layout(-land)
  - XML-Dateien mit Layout-Informationen (ggf für gedrehte Geräte)
- Menu
  - Kontext -und Options-Menüeinträge im XML Format
- Mipmap
  - Icon der App in verschiedenen Auflösungen
- Values
  - Alle Arten von Schlüssel-/Wert-Definitionen (zB Strings)

Meine erste App – Beispiel Text senden

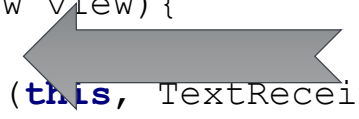
Schritt 2: Klick auf Button

```
public static final String EXTRA_MSG =
"com.example.verenakaefer.my_unique_message_name";

public void sendText(View view){
    Intent myIntent
            = new Intent(this, TextReceiver.class);
    EditText textView
            = (EditText)   findViewById(R.id.editText1);
    String msg = textView.getText().toString();
    myIntent.putExtra(EXTRA_MSG,msg);
    startActivity(myIntent);

}
```

Meine erste App – Beispiel Text senden

Schritt 2: Klick auf Button

```
public static final String EXTRA_MSG =
"com.example.verenakaefer.my_unique_message_name";

public void sendText(View view){
    Intent myIntent
            = new Intent(this, TextReceiver.
    EditText textView
            = (EditText)   findViewById(R.ic
    String msg = textView.getText().toStrin
    myIntent.putExtra(EXTRA_MSG,msg);
    startActivity(myIntent);

}
```



Dienstleistungen von anderen Apps werden über Intents abgerufen bzw. Daten zur weiteren Verarbeitung übergeben.

Meine erste App – Beispiel Text senden

Schritt 2: Klick auf Button

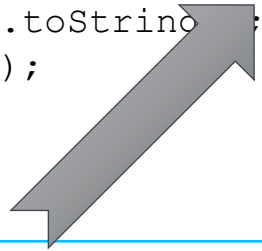
```
public static final String EXTRA_MSG =
"com.example.verenakaefer.my_unique_message_name";

public void sendText(View view){
    Intent myIntent
            = new Intent(this, TextReceiver.class);
    EditText textView
            = (EditText)   findViewById(R.id.editText1);
    String msg = textView.getText().toString();
    myIntent.putExtra(EXTRA_MSG,msg);
    startActivity(myIntent);

}
```

Hier wird die benutzte / aufgerufene Klasse angegeben, falls eine eigene Klasse verwendet wird, sonst kann aus den vorhandenen Schnittstellen ausgewählt werden (z.B. Zugriff auf Camera durch `android.provider.MediaStore.ACTION_IMAGE_CAPTURE`)

Meine erste App – Beispiel Text senden

Schritt 2: Klick auf Button

```
public static final String EXTRA_MSG =
"com.example.verenakaefer.my_unique_message_name";

public void sendText(View view){
    Intent myIntent
            = new Intent(this, TextReceiver.class);
    EditText textView
            = (EditText)   findViewById(R.id.editText1);
    String msg = textView.getText().toStrino ;
    myIntent.putExtra(EXTRA_MSG,msg);
    startActivity(myIntent);

}
```

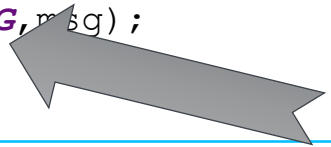


Finden der Instanz des Eingabefeldes über seine ID

Meine erste App – Beispiel Text senden

Schritt 2: Klick auf Button

```
public static final String EXTRA_MSG =
"com.example.verenakaefer.my_unique_message_name";

public void sendText(View view){
    Intent myIntent
            = new Intent(this, TextReceiver.class);
    EditText textView
            = (EditText)   findViewById(R.id.editText1);
    String msg = textView.getText().toString();
    myIntent.putExtra(EXTRA_MSG,msg);
    startActivity(myIntent);

}
```



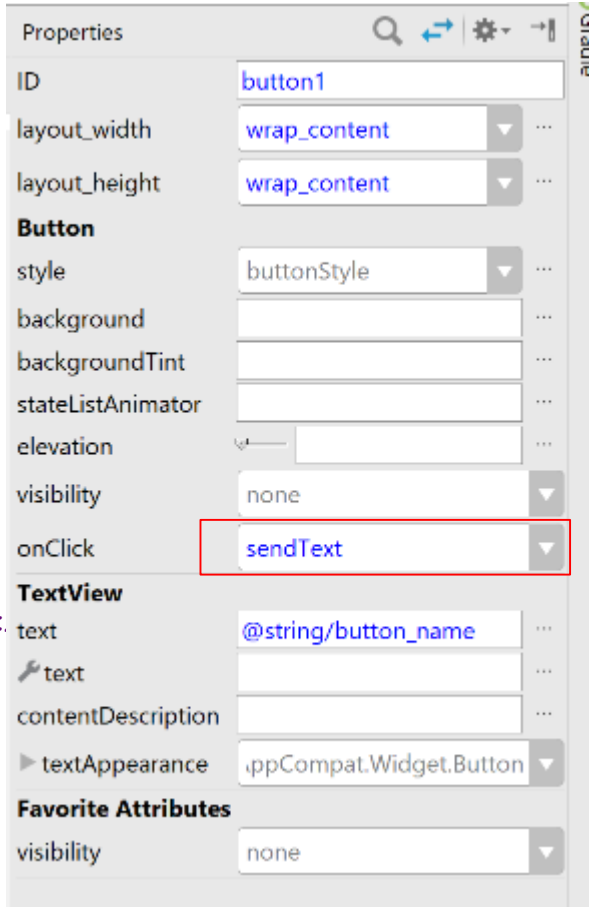
Senden der Eingabe über den Standardmechanismus.
Erster Parameter gibt den Namen an (inklusive Package-Prefix). Zweiter Parameter sind die zu transportierenden Daten (hier String).

Meine erste App – Beispiel Text senden

Schritt 2: Klick auf Button

```
public static final String EXTRA_MSG =
"com.example.verenakaefer.my_unique_message_name";

public void sendText(View view){
    Intent myIntent
            = new Intent(this, TextReceiver.class);
    EditText textView
            = (EditText)   findViewById(R.id.editText
    String msg = textView.getText().toString();
    myIntent.putExtra(EXTRA_MSG,msg);
    startActivity(myIntent);

}
```

Implementierung mittels Listener direkt im Code auch möglich



Meine erste App – Beispiel Text senden

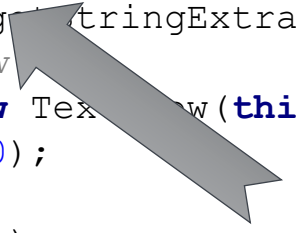
Schritt 3: Neue Activity zur Textanzeige

```
class TextReceiver  extends Activity {

    @Override
    public void onCreate(Bundle savedInstaceState) {
        super.onCreate(savedInstaceState)

        Intent intent = getIntent();
        String msg =   intent.getStringExtra(MainActivity.EXTRA_MSG);
        // Create the text view
        TextView textView = new TextView(this);
        textView.setTextSize(40);
        textView.setText(msg);
        setContentView(textView);
    }

}
```

Meine erste App – Beispiel Text senden

Schritt 3: Neue Activity zur Textanzeige

```
class TextReceiver  extends Activity {

    @Override
    public void onCreate(Bundle savedInstaceState) {
        super.onCreate(savedInstaceState)

        Intent intent = getIntent();
        String msg =   intent.getStringExtra(MainActivity.EXTRA_MSG);
        // Create the text view
        TextView textView = new TextView(this);
        textView.setTextSize(40);
        textView.setText(msg);
        setContentView(textView);
    }

}
```

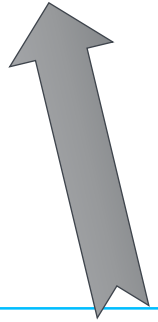


Wir holen uns die Instanz des Intent, der uns gestartet hat.

Meine erste App – Beispiel Text senden

Schritt 3: Neue Activity zur Textanzeige

```
class TextReceiver  extends Activity {

    @Override
    public void onCreate(Bundle savedInstaceState) {
        super.onCreate(savedInstaceState)

        Intent intent = getIntent();
        String msg =   intent.getStringExtra(MainActivity.EXTRA_MSG);
        // Create the text view
        TextView textView = new TextView(this);
        textView.setTextSize(40);
        textView.setText(msg);
        setContentView(textView);
    }

}
```

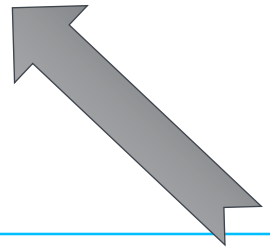


Wir wissen, dass der Intent etwas „extra“ für uns dabei hat, nämlich einen Nachricht in Form eines Strings.

Meine erste App – Beispiel Text senden

Schritt 3: Neue Activity zur Textanzeige

```
class TextReceiver  extends Activity {

    @Override
    public void onCreate(Bundle savedInstaceState) {
        super.onCreate(savedInstaceState)

        Intent intent = getIntent();
        String msg =   intent.getStringExtra(MainActivity.EXTRA_MSG);
        // Create the text view
        TextView textView = new TextView(this);
        textView.setTextSize(40);
        textView.setText(msg);
        setContentView(textView);
    }

}
```

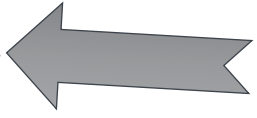


Wir erzeugen uns eine TextView mit der wir die Nachricht zeigen können.

Meine erste App – Beispiel Text senden

Schritt 3: Neue Activity zur Textanzeige

```
class TextReceiver  extends Activity {

    @Override
    public void onCreate(Bundle savedInstaceState) {
        super.onCreate(savedInstaceState)

        Intent intent = getIntent();
        String msg =   intent.getStringExtra(MainActivity.EXTRA_MSG);
        // Create the text view
        TextView textView = new TextView(this);
        textView.setTextSize(40);
        textView.setText(msg);
        setContentView(textView);
    }

}
```



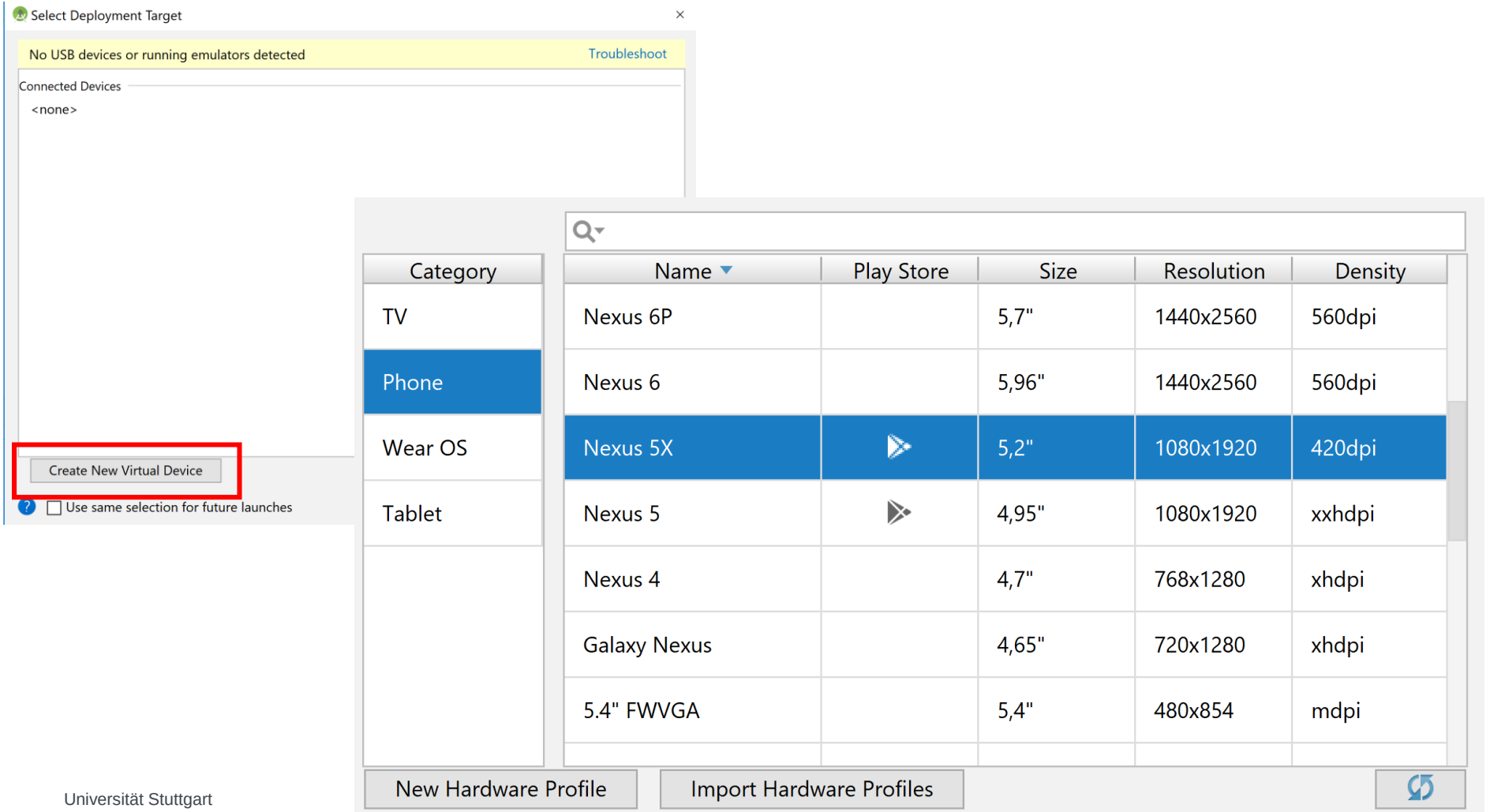
Und setzen sie hier als Element der aktuellen App.

Aufgabe (5min)

Fassen Sie nochmals für sich selbst zusammen

- Was ist eine Activity?
- Wie wird eine Activity aufgerufen?
- Wie kann hierbei ein Text übergeben werden?

Emulator


Select Deployment Target
No USB devices or running emulators detected
Troubleshoot
Connected Devices
<none>
Create New Virtual Device
Use same selection for future launches
Category
TV
Phone
Wear OS
Tablet
Name
Play Store
Size
Resolution
Density
Nexus 6P
5,7"
1440x2560
560dpi
Nexus 6
5,96"
1440x2560
560dpi
Nexus 5X
5,2"
1080x1920
420dpi
Nexus 5
4,95"
1080x1920
xxhdpi
Nexus 4
4,7"
768x1280
xhdpi
Galaxy Nexus
4,65"
720x1280
xhdpi
5.4" FWVGA
5,4"
480x854
mdpi
New Hardware Profile
Import Hardware Profiles

Nützliche Oberflächenbausteine

ListActivity

- Häufig müssen Listen von gleichförmigen Daten dargestellt werden. Typisches Beispiel: Kontaktliste
- Die ListActivity bzw das ListView stellen eine vordefinierte Darstellung zur Verfügung.
- Die Anbindung der Daten geschieht über einen Adaptor.
- Die ListActivity besitz eine implizite ListView als Wurzel ihrer Bildschirmseite. Daher ist kein eigenes Layout nötig, aber möglich.

ListActivity – Basp ArrayAdaptor

```
public class ListDemoAct extends ListActivity {
    private String[] entries = {"eins", "zwei", "drei", "vier", "fünf"};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ArrayAdapter<String> stringAdapter = new
ArrayAdapter<String>(this/*context*/,
                android.R.layout.simple_list_item_1 /*Darstellung*/,
                entries /*Inhalt*/);
        setListAdapter(stringAdapter);
    }
}
```

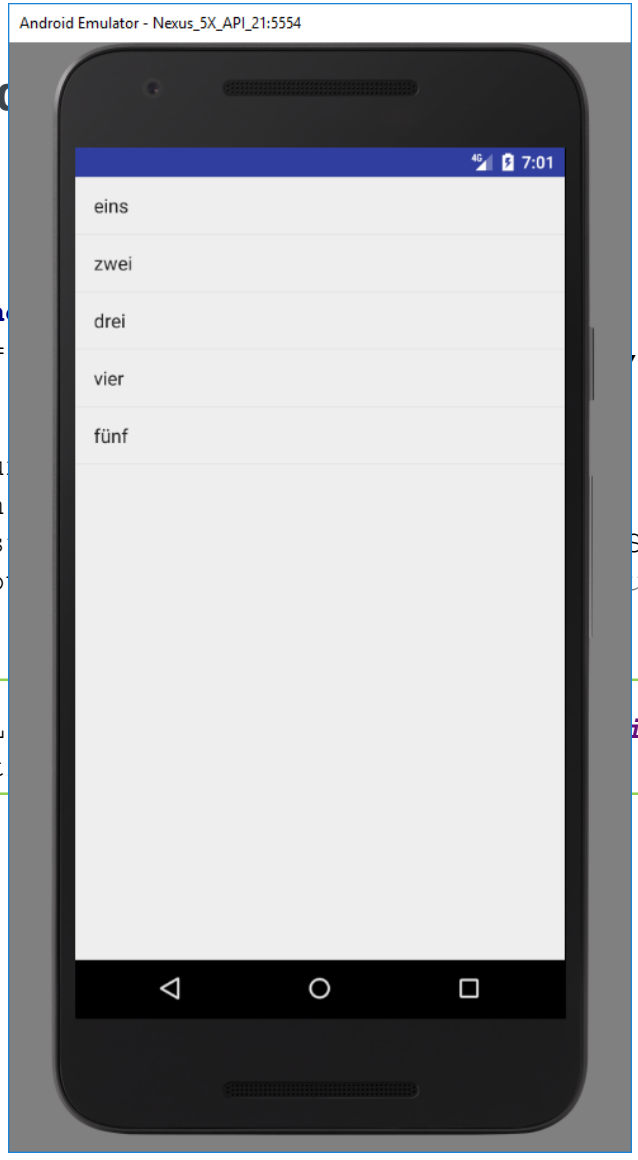
Geht natürlich auch mit „normaler“ Activity und explizitem ListView

ListActivity – Basp ArrayAdaptor

```
public class ListViewDemo extends Activity {
    private String[] entries = {"eins", "zwei", "drei", "vier", "fünf"};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ArrayAdapter<String> stringAdapter = new ArrayAdapter<String>(this/*context*/,
                android.R.layout.simple_list_item_1 /*Darstellung*/,
                entries);

        ListView listview = (ListView) findViewById(R.layout.listView);
        listview.setAdapter(stringAdapter);
    }
}
```

ListActivity – Basp ArrayAd

```
public class ListDemoAct exten
    private String[] entries =                                    , "fünf"};

    @Override
    protected void onCreate(Bu
        super.onCreate(savedIn
        ArrayAdapter<String> s                                    String>(this/*context*/,
                android.R.layo                                    ung*/,
                entries);

        ListView listview = (L                                    iew);
        listview.setAdapter(st
    }
}
```



Das Preference Fragment

- Für jetzt: Fragmente = losgelöste Oberflächenteile die von einer Activity gestartet werden.

- Häufig gibt es Einstellungen für eine Anwendung. Diese werden Mittels XML verwaltet und in einem Preference Fragment dargestellt.

- Das PreferenceFragment übernimmt dabei eigenständig das Speichern der Einstellungen.

Preferences XML

Über den Key kann der Wert später in den Einstellungen wiedergefunden werden.

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">
    <PreferenceCategory android:title="@string/cat1">
        <CheckBoxPreference
            android:key="checkbox_1"
            android:summary="@string/summary_cb1"
            android:title="@string/title_cb1"/>

        <CheckBoxPreference
            android:key="checkbox_2"
            android:summary="@string/summary_cb2"
            android:title="@string/title_cb2"/>
    </PreferenceCategory>

    <PreferenceCategory android:title="@string/cat2">
        <EditTextPreference
            android:dialogTitle="@string/dialog_title_et1"
            android:key="edittext_1"
            android:summary="@string/summary_et1"
            android:title="@string/title_et1"/>
    </PreferenceCategory>
</PreferenceScreen>
```

Definiert eine Checkbox deren Status über den Key manipulierbar ist

Definiert ein Wert, der über einen Dialog veränderbar ist

Das Preference Fragment

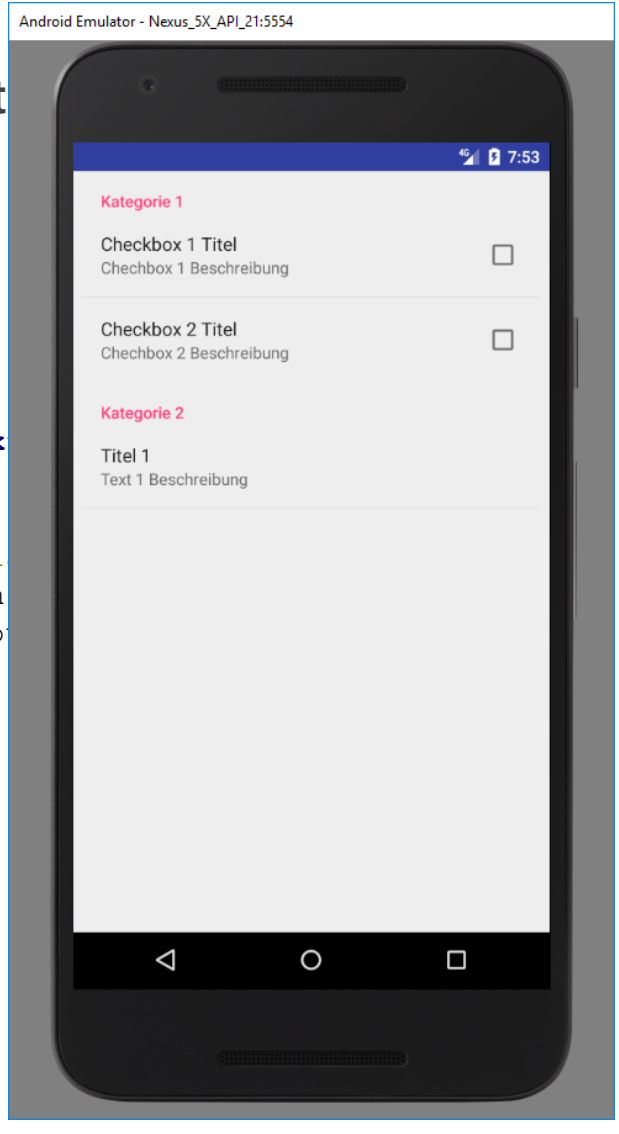
```
public class PreferenceDemo extends PreferenceFragment {

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.preferences);
    }
}
```

Das Preference Fragment anzeigen

```
public class MyPreferencesActivity extends PreferenceActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getFragmentManager().beginTransaction().replace(android.R.id.content,
          new MyPreferenceFragment()).commit();
    }
}
```

Das Preference Fragment

```
public class PreferenceDemo ex

    @Override
    public void onCreate(@Null                    {
        super.onCreate(savedIn
        addPreferencesFromReso
    }
}
```



Dialoge

- Seit Honeycomb (API Level 11) sind Dialoge als Fragmente implementiert.
- Die Dialoge sind nach dem Baukastenprinzip konfigurierbar.

Dialoge

```
public class AlertFragment extends DialogFragment {

    private DialogInterface.OnClickListener listener;

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        if (context instanceof DialogInterface.OnClickListener){
            listener = (DialogInterface.OnClickListener) context;
        }
    }

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

        builder.setTitle(R.string.app_name);
        builder.setMessage(R.string.message);
        builder.setCancelable(false);
        builder.setPositiveButton(R.string.close, listener);
        return builder.create();
    }
}
```

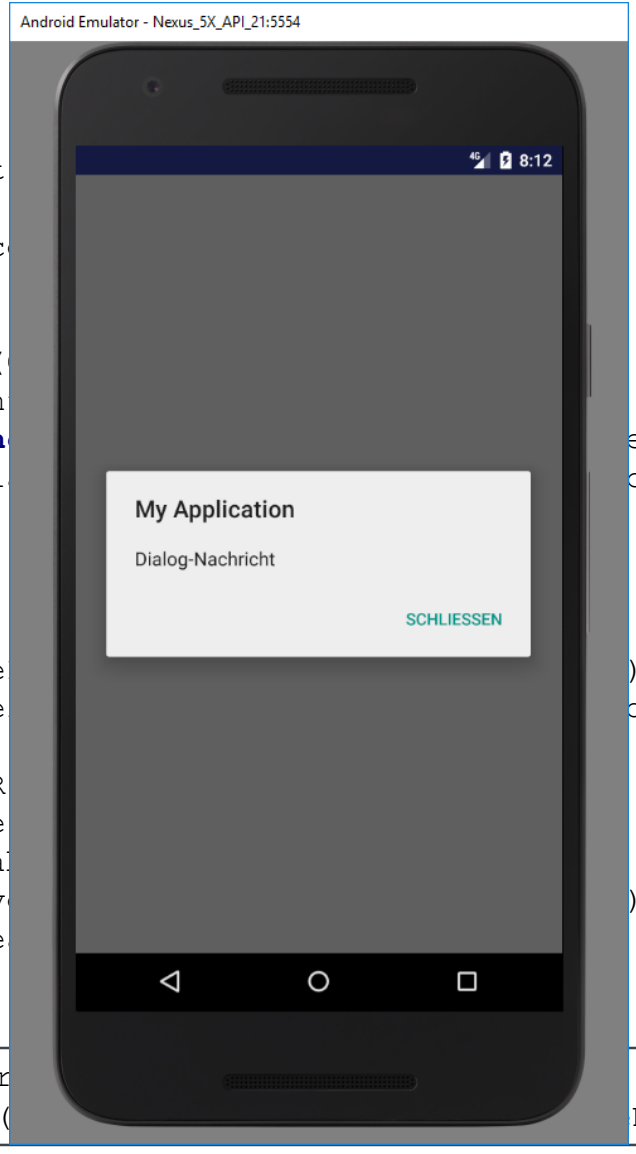
```
AlertFragment af = new AlertFragment();
af.show(getFragmentManager(), AlertFragment.class.getSimpleName());
```

Oder: `getSupportFragmenManager, je nach API-Level`

Dialoge

```
public class AlertFragment

    private DialogInterfac

    @Override
     public void onAttach(
        super.onAttach(con
        if (context instan                    ener){
            listener = (Di                    ontext;
        }
    }

    @Override
    public Dialog onCreate                    ) {
        AlertDialog.Builde                    der(getActivity());

        builder.setTitle(R
        builder.setMessage
        builder.setCancela
        builder.setPositiv                    );
        return builder.cre
    }
}
```

```
AlertFragment af = new Aler
af.show(getFragmentManager(                   Name());
```



Dialoge

```
public class AlertFragment extends DialogFragment {

    private DialogInterface.OnClickListener listener;

    @Override
     public void onAttach(Context context) {
        super.onAttach(context);
        if (context instanceof DialogInterface.OnClickListener){
            listener = (DialogInterface.OnClickListener) context;
        }
    }

    @Override
    public Dialog onCreateDialog(Bundle savedInstanc
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

        builder.setTitle(R.string.app_name);
        builder.setMessage(R.string.message);
        builder.setCancelable(false);
        builder.setPositiveButton(R.string.close, listener);
        return builder.create();
    }
}
```

Das implementierte Interface des Dialogs. Ausführende Activities müssen dieses Interface ebenfalls implementieren.

```
AlertFragment af = new AlertFragment();
af.show(getFragmentManager(), AlertFragment.class.getSimpleName());
```

Dialoge

```
public class AlertFragment extends DialogFragment {

    private DialogInterface.OnClickListener listener;

    @Override
     public void onAttach(Context context) {
        super.onAttach(context);
        if (context instanceof DialogInterface.OnClickListener){
            listener = (DialogInterface.OnClickListener) context;
        }
    }

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

        builder.setTitle(R.string.app_name);
        builder.setMessage(R.string.message);
        builder.setCancelable(false);
        builder.setPositiveButton(R.string.close, listener);
        return builder.create();
    }
}
```

„Baukasten“ für den Dialog.

```
AlertFragment af = new AlertFragment();
af.show(getFragmentManager(), AlertFragment.class.getSimpleName());
```

Dialoge

```
public class AlertFragment extends DialogFragment {

    private DialogInterface.OnClickListener listener;

    @Override
     public void onAttach(Context context) {
        super.onAttach(context);
        if (context instanceof DialogInterface.OnClickListener){
            listener = (DialogInterface.OnClickListener) context;
        }
    }

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

        builder.setTitle(R.string.app_name);
        builder.setMessage(R.string.message);
        builder.setCancelable(false);
        builder.setPositiveButton(R.string.close, listener);
        return builder.create();
    }
}
```

Überschrift des Dialogs

```
AlertFragment af = new AlertFragment();
af.show(getFragmentManager(), AlertFragment.class.getSimpleName());
```

Dialoge

```
public class AlertFragment extends DialogFragment {

    private DialogInterface.OnClickListener listener;

    @Override
     public void onAttach(Context context) {
        super.onAttach(context);
        if (context instanceof DialogInterface.OnClickListener){
            listener = (DialogInterface.OnClickListener) context;
        }
    }

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

        builder.setTitle(R.string.app_name);
        builder.setMessage(R.string.message);
        builder.setCancelable(false);
        builder.setPositiveButton(R.string.close, listener);
        return builder.create();
    }
}
```

Angezeigte Information

```
AlertFragment af = new AlertFragment();
af.show(getFragmentManager(), AlertFragment.class.getSimpleName());
```

Dialoge

```
public class AlertFragment extends DialogFragment {

    private DialogInterface.OnClickListener listener;

    @Override
     public void onAttach(Context context) {
        super.onAttach(context);
        if (context instanceof DialogInterface.OnClickListener){
            listener = (DialogInterface.OnClickListener) context;
        }
    }

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

        builder.setTitle(R.string.app_name);
        builder.setMessage(R.string.message);
        builder.setCancelable(false);
        builder.setPositiveButton(R.string.close, listener);
        return builder.create();
    }
}
```

Dialog kann nicht abgebrochen werden (keine negative Bestätigung)

```
AlertFragment af = new AlertFragment();
af.show(getFragmentManager(), AlertFragment.class.getSimpleName());
```

Dialoge

```
public class AlertFragment extends DialogFragment {

    private DialogInterface.OnClickListener listener;

    @Override
     public void onAttach(Context context) {
        super.onAttach(context);
        if (context instanceof DialogInterface.OnClickListener){
            listener = (DialogInterface.OnClickListener) context;
        }
    }

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

        builder.setTitle(R.string.app_name);
        builder.setMessage(R.string.message);
        builder.setCancelable(false);
        builder.setPositiveButton(R.string.close, listener);
        return builder.create();
    }
}
```

Gibt an was im Falle einer positiven Bestätigung passieren soll.

Dialoge - DatePicker

```
public class DialogDemo extends DialogFragment {

    private DatePickerDialog.OnDateSetListener listener;

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        if (context instanceof DatePickerDialog.OnDateSetListener){
            listener = (DatePickerDialog.OnDateSetListener) context;
        }
    }

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        final Calendar c = Calendar.getInstance();
        int year = c.get(Calendar.YEAR);
        int month = c.get(java.util.Calendar.MONTH);
        int day = c.get(Calendar.DAY_OF_MONTH);

        return new DatePickerDialog(getActivity(), listener, year, month, day);
    }
}
```

Dialoge aufrufen

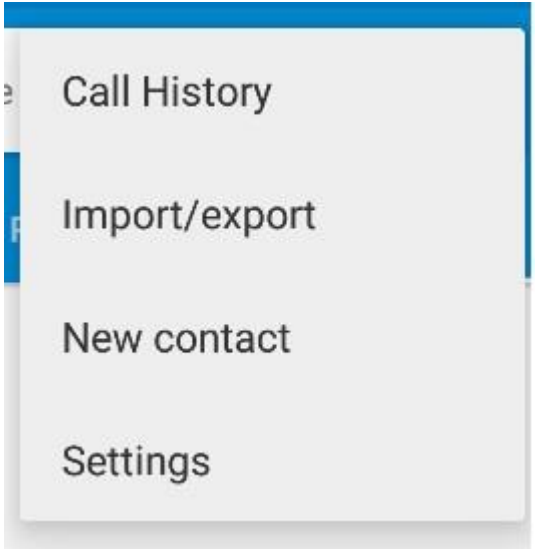
```
public class blankActivity extends Activity implements DialogInterface,
DatePickerDialog.OnDateSetListener, DialogInterface.OnClickListener{

    @Override
    public void onCreate(Bundle saved
        super.onCreate(savedInstaceSt

        //AlertFragment af = new AlertFragment();
        //af.show(getFragmentManager(),
         AlertFragment.class.getSimpleName());

        final DialogDemo dd = new DialogDemo();
        dd.show(getFragmentManager(), DialogDemo.class.getSimpleName());

    }
```

TAG ist ein String, der die Klasse das Fragments eindeutig identifiziert. Hier z.B.: `String TAG = DialogDemo.class.getSimpleName();`

Menüs

- Menüs in Android sind mit klassischen Menüs vergleichbar.

Klassisches Menü



Android Menü

Sollte nur weniger Einträge haben. Üblich: Settings und about.

Menüs

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id = "@+id/item1"
        android:title="@string/action_setting" />
    <item android:id = "@+id/item2"
        android:title="@string/action_setting2" />
</menu>
```

Menüeinträge als Items. Über die „id“ identifizierbar.

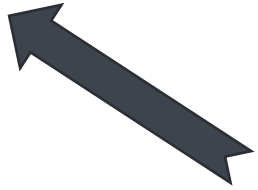
Menüs

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.my_firstmenu, menu);
    return super.onCreateOptionsMenu(menu);

}
```

Legt das generelle Aussehen des Menüs fest bzw. erstellt generell das Menü.

Menüs

```
@Override
public boolean onPrepareOptionsMenu(Menu menu) {
    EditText et = (EditText) findViewById(R.id.editText1);
    et.setText(getString(R.string.app_name));
    return super.onPrepareOptionsMenu(menu);
}
```



Wird aufgerufen, wenn auf den Menü-Button gedrückt wird.

Menüs

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    EditText et = (EditText) findViewById(R.id.editText1);
    switch (item.getItemId()){
        case R.id.item1:
            et.setText(item.getTitle());
            return true;
        case R.id.item2:
            et.setText(item.getTitle());
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Wird verwendet um auf die Auswahl eines Menü-Eintrages zu reagieren.

Preferences – Auf Einstellungen zugreifen

- Ähnlich wie unter dem JDK gibt es die Möglichkeit, Einstellungen einer App in einem dedizierten Bereich zu speichern.

```
SharedPreferences prefs =
this.getDefaultSharedPreferences(getBaseContext();
```

Preferences – Auf Einstellungen zugreifen

```
SharedPreferences.Editor editor = prefs.edit();
editor.putBoolean("on", true);
editor.commit();
```

Eine Instanz des Editors generieren.

Werte einfügen (putString, putInt…)

Änderungen übernehmen

Preferences – Auf Einstellungen zugreifen

```
String lastName = prefs.getString("lastName", "none");
```

Übergeben wird der Schlüssel und ein Standardrückgabewert, falls der Schlüssel nicht existiert

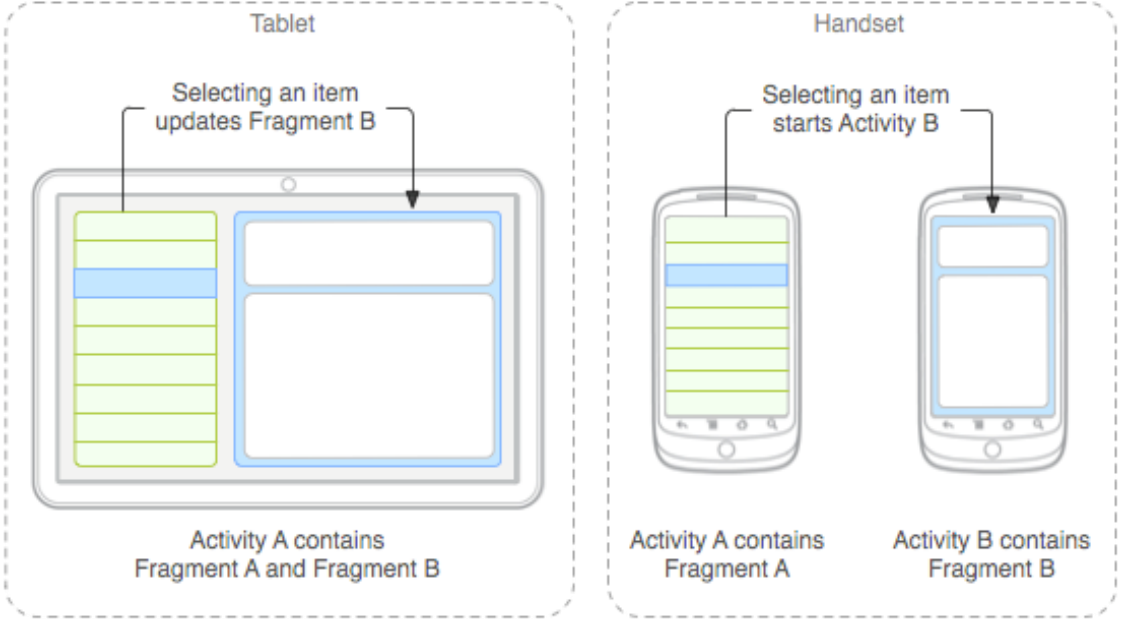
Fragments als erweitertes Oberflächenkonzept

Erweitertes Interface-Konzept

Design Philosophie

- Vielzahl von verschieden Endgeräten
- Unterschiedlichste Displays (Größe, Auflösung)

➢ Aufbrechen der UI in verschiedene Teile



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

Fragments

- Fragments gehören immer zu einer Activity
  - Sie leben und sterben also mit der Activiy.
  - Sie können während der Laufzeit der Activity ausgetauscht werden.

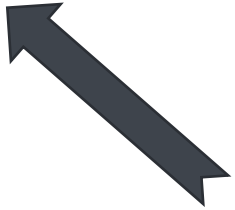
- Daraus folgen 2 unterschiedliche Möglichkeiten ein Fragment einer Activity hinzuzufügen:
  - Per Layout
  - Programmatisch
  - Oder auch eine Kombination von beidem

Fragment Layout Bsp

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

    <fragment
        android:name="com.example.verenakaefer.myapplication.ArticleListFragment"
        android:id="@+id/list"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent"/>

    <fragment
        android:name="com.example.verenakaefer.myapplication.ArticleReaderFragment"
        android:id="@+id/viewer"
        android:layout_weight="2"
        android:layout_width="0dp"
        android:layout_height="match_parent"/>

</LinearLayout>
```

Fragment programmatisch Bsp

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

ArticleListFragment fragment = new ArticleListFragment();
trans.add(R.id.fragment, fragment);

trans.commit();
```



Container i.d.R. Layouts (z.B.LinearLayout)

Arbeiten mit unterschiedlichen Bildschirm-Ausrichtungen

Es besteht die Möglichkeit zusätzliche Layouts speziell für den gedrehten Fall (Land{scape}) anzulegen.

Diese werden unter Layout-Land in Ressourcen-Ordner abgelegt.

Programmatisch kann dann über Layout-Elemente entschieden werden, ob man sich im „normalen“ oder „landscape“ Modus befindet.

Live-Beispiel

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

Dependencies

Dependencies

In der build.gradle(Module:app)

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:28.0.0'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-
core:3.0.2'
}
```