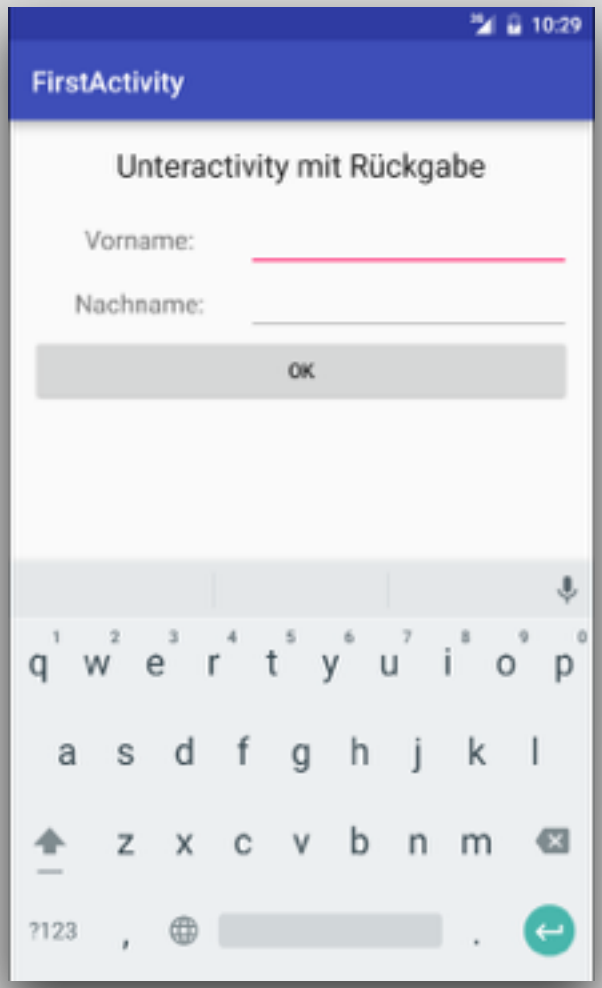
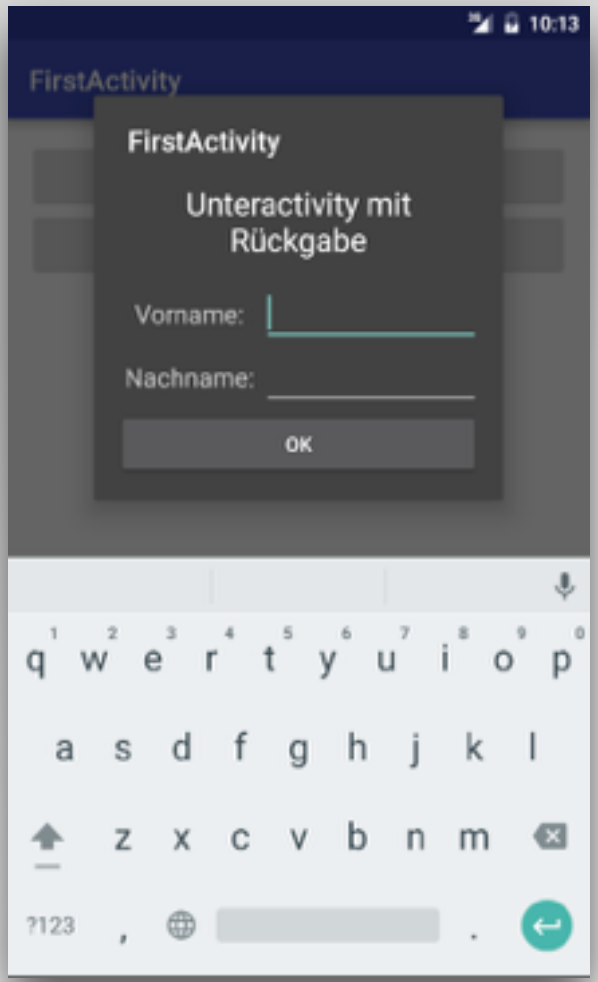
Oberflächen unter Android → Activity





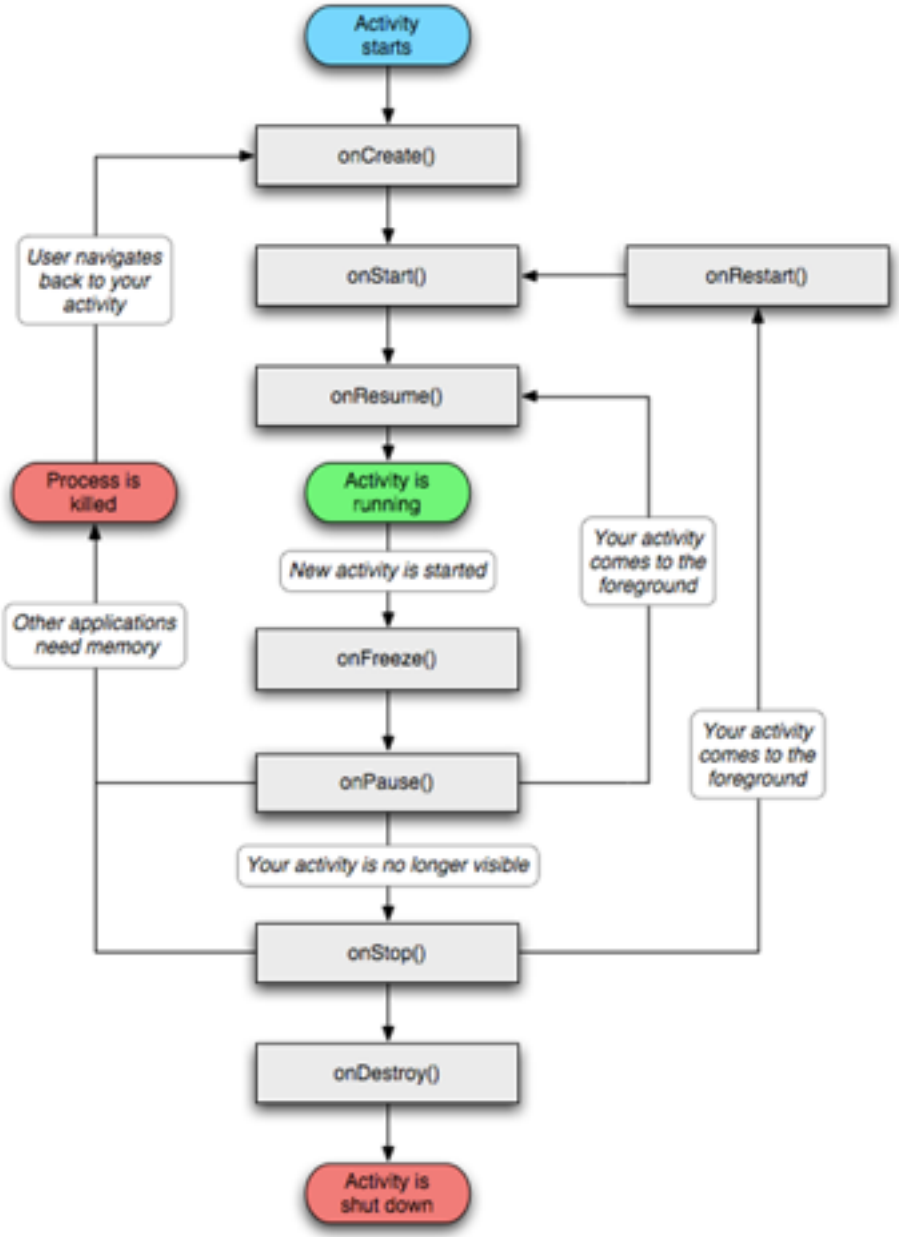
HTL LEONDING

Grundlegendes zu Activities

- Hauptzweck: Verarbeitung der Benutzerinteraktion
- Pro Bildschirmseite wird eine Activity angelegt
- Strikte Trennung Design ⇔ Code
  - Design in res-Ordner ➔ layout
    - Automatische Anpassung an Ausrichtung, Tag/Nacht, …
    - ADT erstellt daraus statische R-Klasse
      - Ressourcencompiler *Android Asset Packaging Tool* (*aapt*)
  - Texte werden ebenfalls in Ressourcen verwaltet
  - uvm ➔ später

Lifecycle einer Activity unter Android

- Activity durchläuft verschiedene Stati
- Ressourcenmangel ➔ Anwendung kann vom System geschlossen werden
  - Eventhandler ermöglichen Sichern und Wiederherstellen von Zuständen
  - Activity muss sich selbst um Zustand seiner Views kümmern
- Prozess kann auch nach Destroy durch Anwender weiterlaufen ➔ schnellerer Start



Richtlinien für Lebenszyklusmethoden

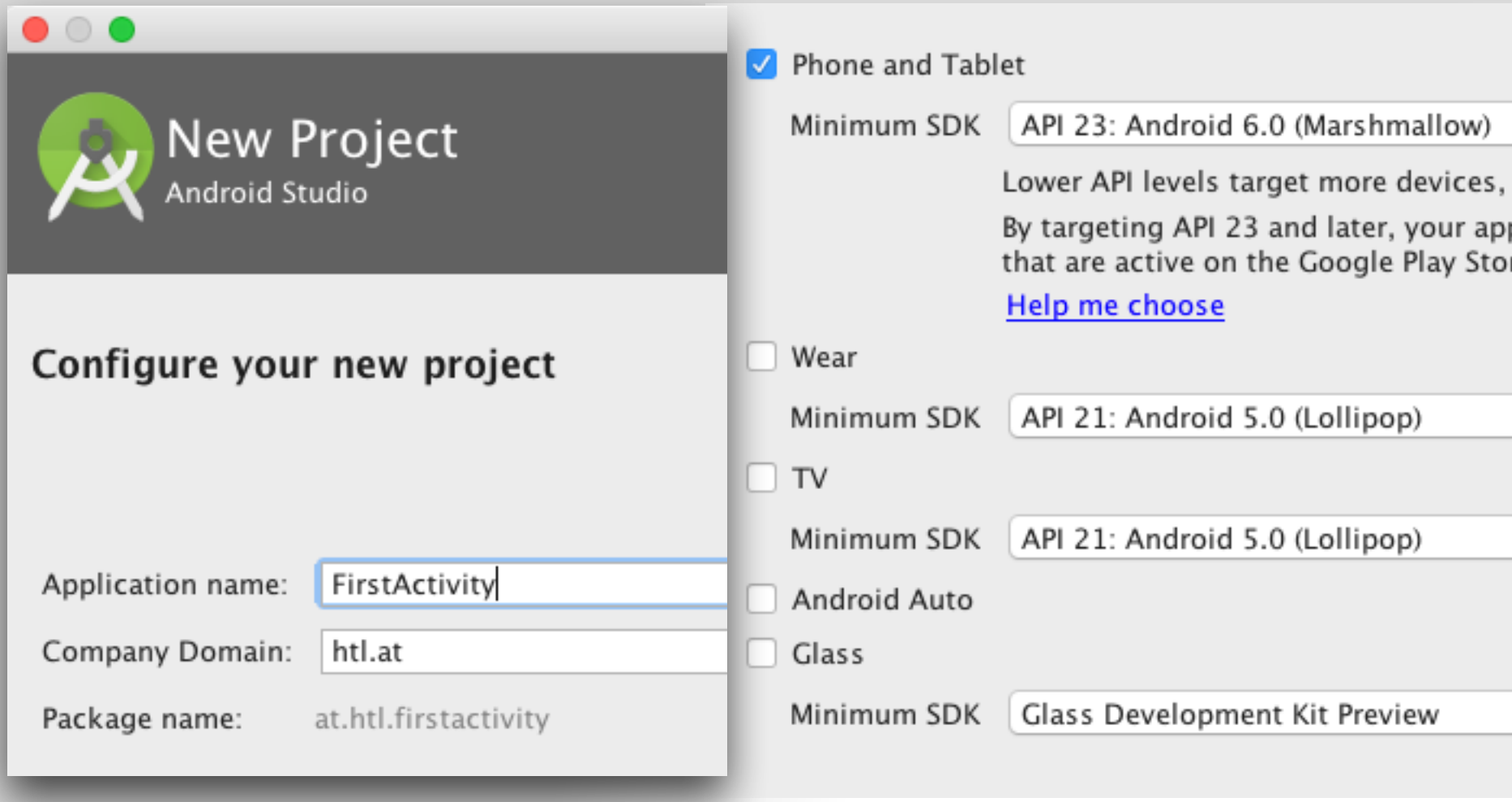
- onCreate() ➔ Objekte instanzieren, die in Activity benötigt werden (z.B. Controls)
- onResume() ➔ starten von Services und Code, der benötigt wird, wenn Activity im Vordergrund ist
- onPause() ➔ stoppen etwaiger Services
- onDestroy() ➔ freigeben von Ressourcen

Erstes einfaches Beispiel als Einstieg

- Hauptactivity
  - Button führt zu Unteractivity
- Unteractivity
  - Zurück-Button führt wieder zu Hauptactivity

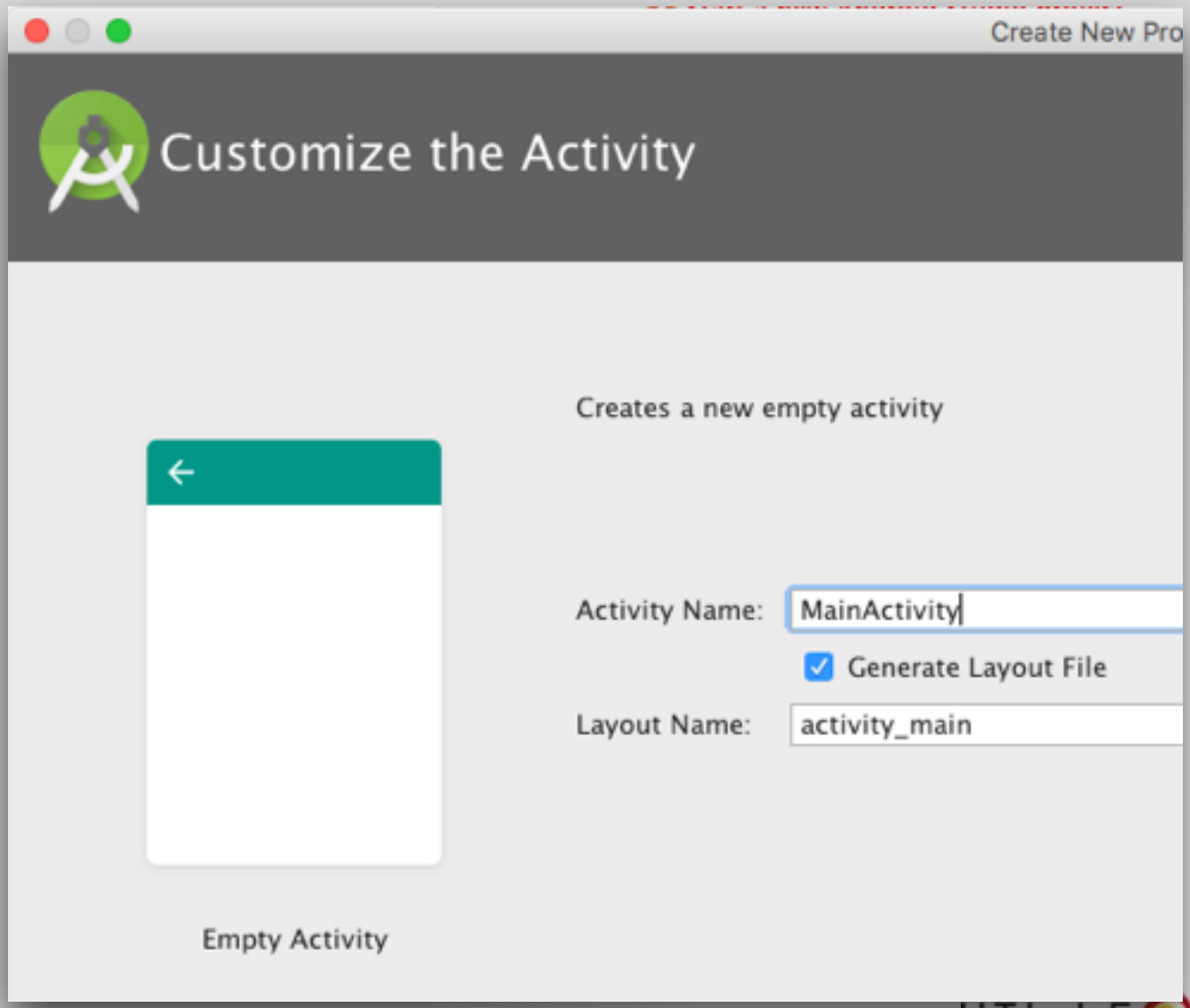
HTL LEONDING

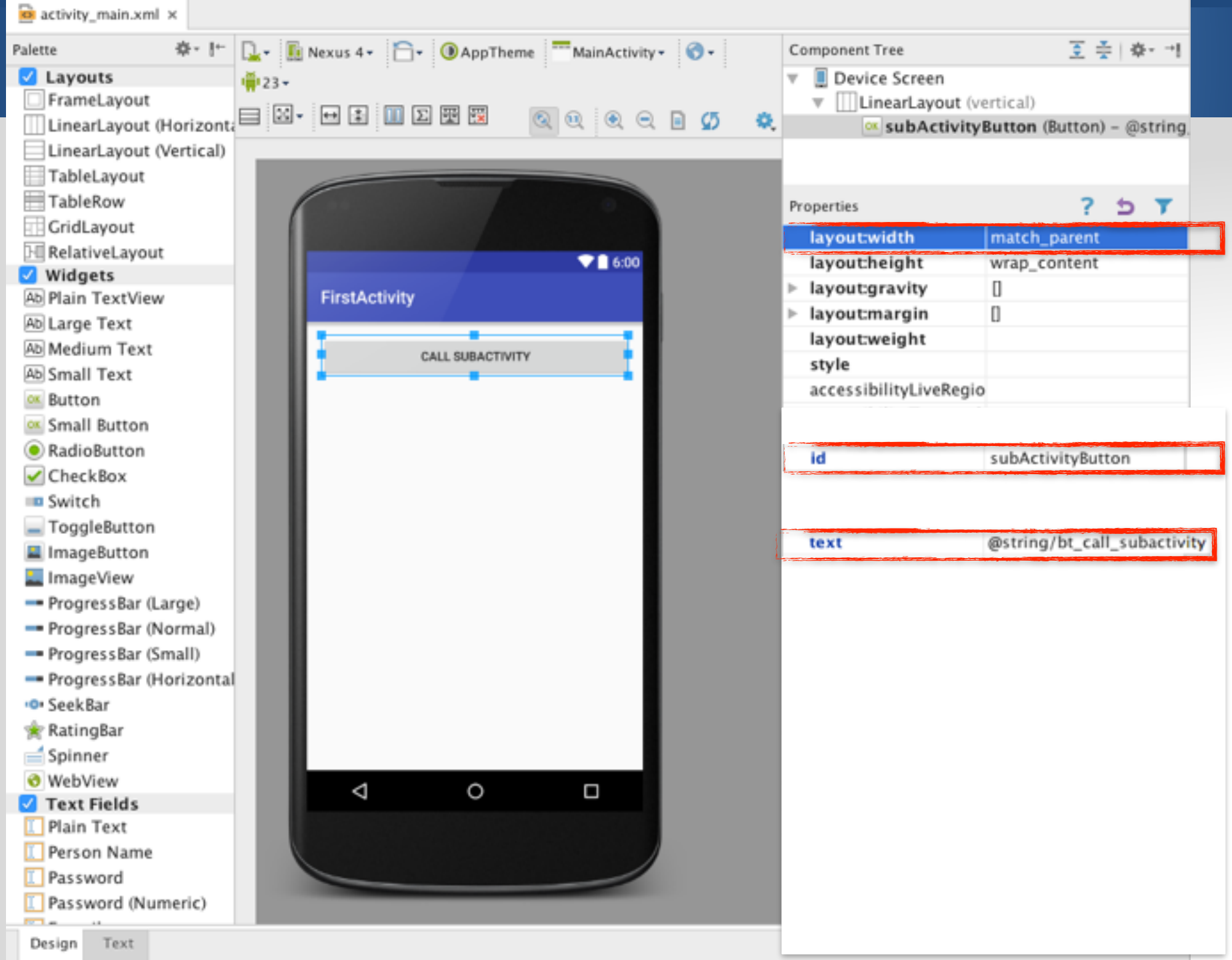
Neues Projekt anlegen 1



HTL LEONDING

Neues Projekt anlegen 2




activity_main.xml
Palette
Layouts
FrameLayout
LinearLayout (Horizonta
LinearLayout (Vertical)
TableLayout
TableRow
GridLayout
RelativeLayout
Widgets
Plain TextView
Large Text
Medium Text
Small Text
Button
Small Button
RadioButton
CheckBox
Switch
ToggleButton
ImageButton
ImageView
ProgressBar (Large)
ProgressBar (Normal)
ProgressBar (Small)
ProgressBar (Horizontal
SeekBar
RatingBar
Spinner
WebView
Text Fields
Plain Text
Person Name
Password
Password (Numeric)
Nexus 4
AppTheme
MainActivity
23
6:00
FirstActivity
CALL SUBACTIVITY
Component Tree
Device Screen
LinearLayout (vertical)
subActivityButton (Button) – @string
Properties
layout:width
match_parent
layout:height
wrap_content
layout:gravity
[]
layout:margin
[]
layout:weight
style
accessibilityLiveRegio
id
subActivityButton
text
@string/bt_call_subactivity
Design
Text

Layout bearbeiten im Design Mode

- Generelles Layout: LinearLayout
  - Vertikale Ausrichtung
  - Padding:16 dp (density-independent pixels) wird von Google empfohlen

orientation	vertical

padding	
all	
left	@dimen/activity_horizontal_margin
top	@dimen/activity_vertical_margin
right	@dimen/activity_horizontal_margin
bottom	@dimen/activity_vertical_margin

- Button
  - Id mit gewöhnungsbedürftiger Syntax

id	subActivityButton

```
android:id="@+id/subActivityButton"
```

  - Beschriftung über XML-Ressourcen

text	@string/bt_call_subactivity

Texte werden in strings.xml verwaltet

- Benennungsschema ist jedenfalls sinnvoll

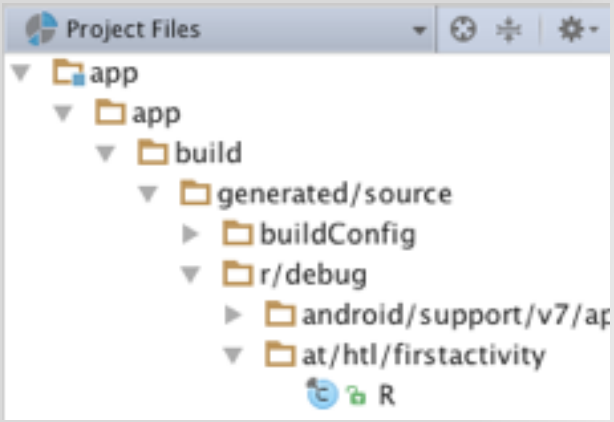
```
<resources>
    <string name="app_name">FirstActivity</string>
    <string name="bt_call_subactivity">Call Subactivity</string>
</resources>
```

Ressourcenklassen werden generiert (aapt)

R.java

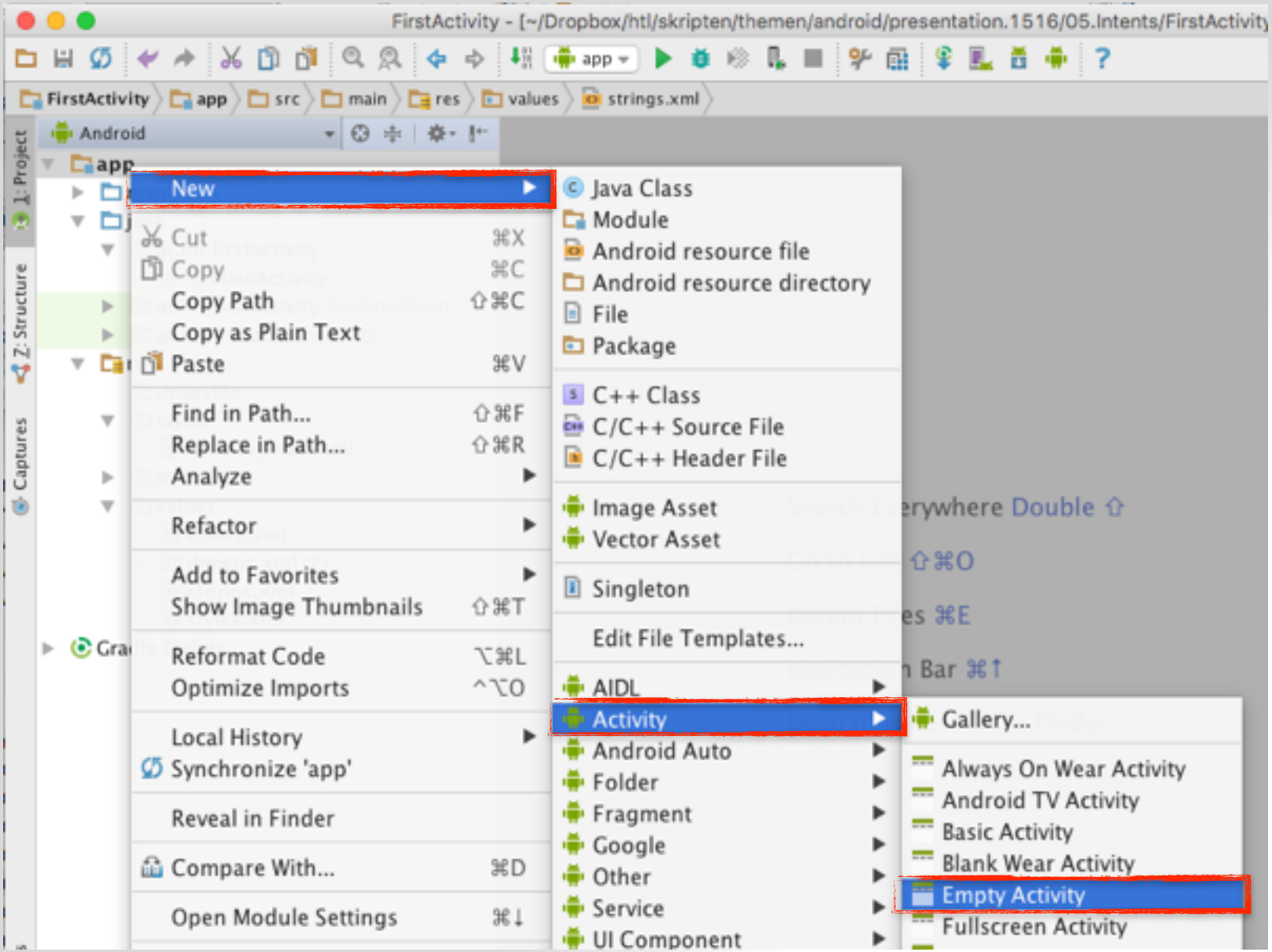
Files under the build folder are generated and should not be edited.

```
1673        public static final int spacer=0x7f0c002c;
1674        public static final int split_action_bar=0x7f0c0007;
1675        public static final int src_atop=0x7f0c0015;
1676        public static final int src_in=0x7f0c0016;
1677        public static final int src_over=0x7f0c0017;
1678        public static final int status_bar_latest_event_content=0x7f0c0053;
1679        public static final int subActivityButton=0x7f0c0050;
1680        public static final int submit_area=0x7f0c004c;
1681        public static final int tabMode=0x7f0c000b;
1682        public static final int text=0x7f0c005b;
1683        public static final int text2=0x7f0c0059;
1684        public static final int textSpacerNoButtons=0x7f0c0034;
1685        public static final int time=0x7f0c0057;
```

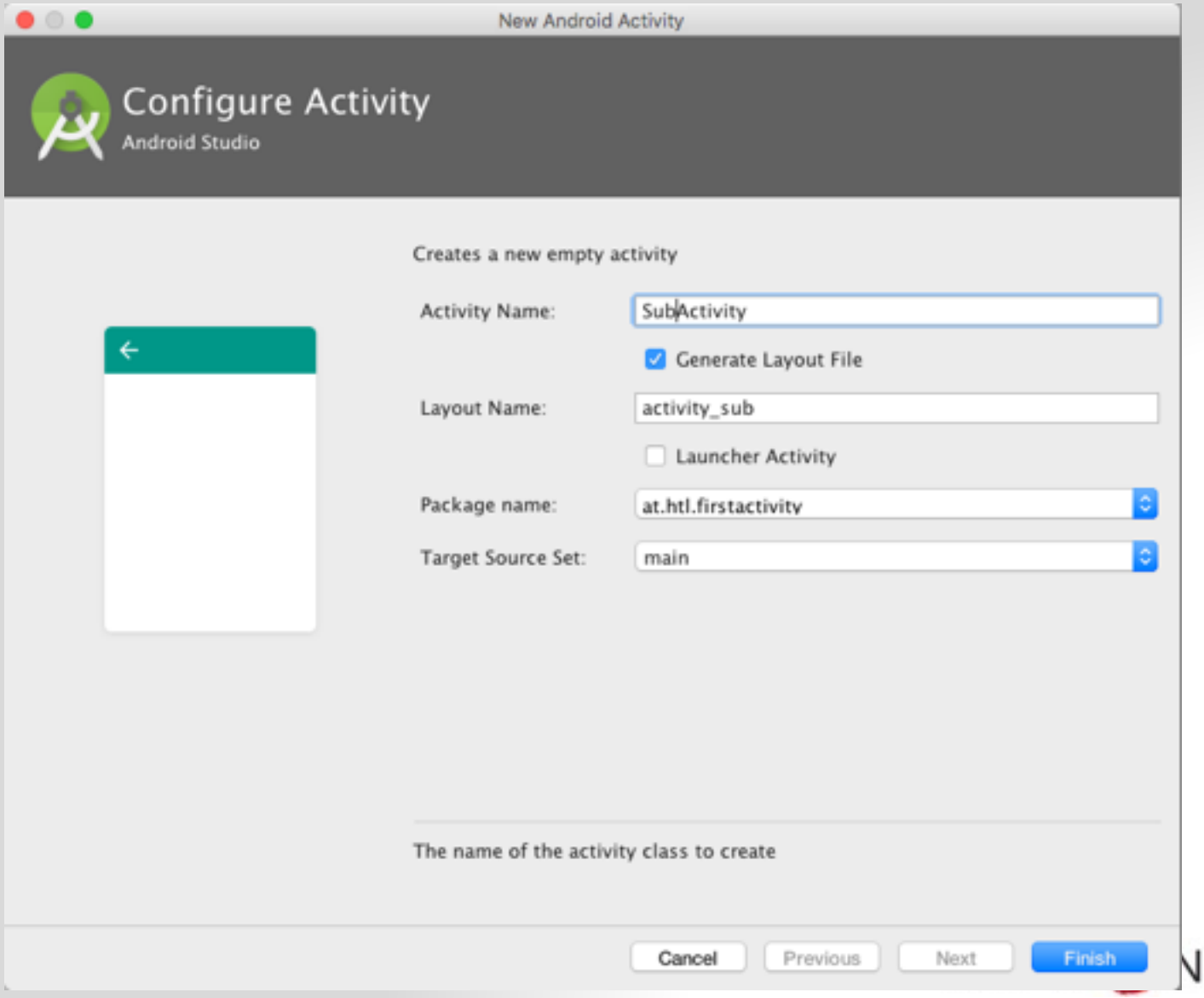


HTL LEONDING

SubActivity anlegen 1

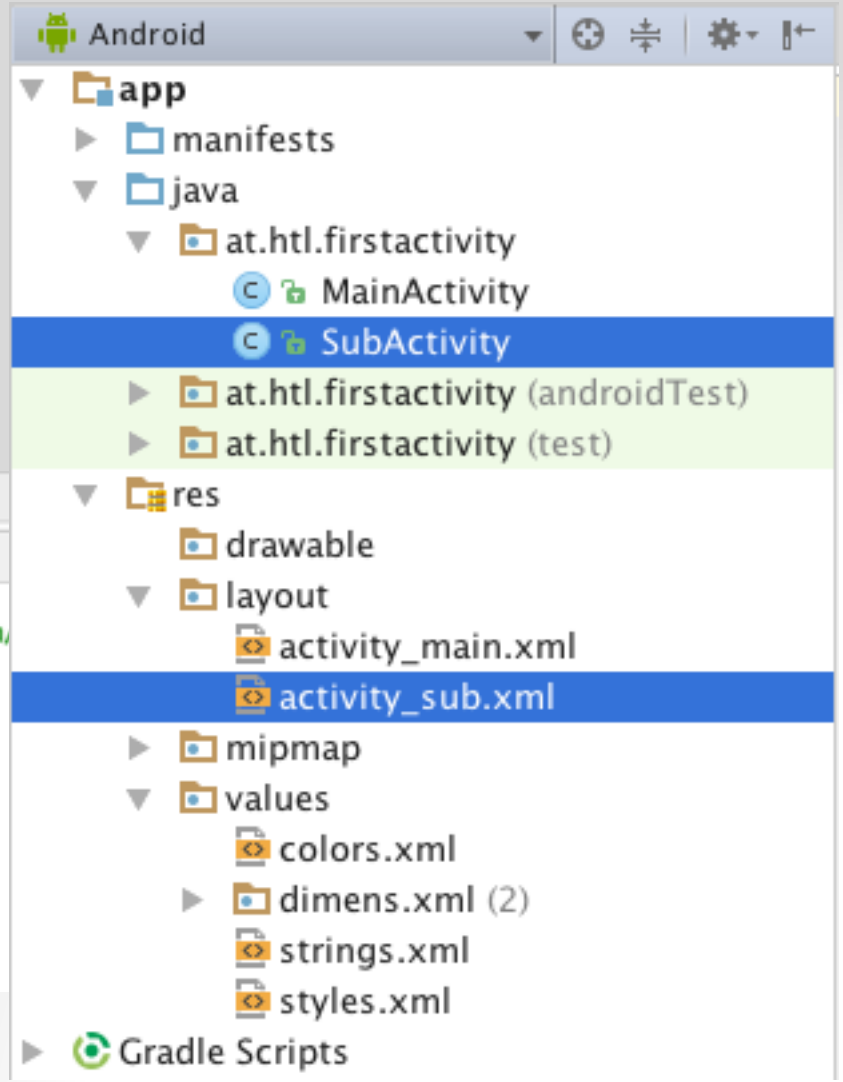


SubActivity anlegen 2



SubActivity anlegen 3

- Es werden eine xml-Layoutdatei sowie die Java-Klasse erstellt
- Im Test-Mode wird ein LinearLayout eingestellt



activity_sub.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/
3      xmlns:tools="http://schemas.android.com/tools"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      android:paddingBottom="16dp"
7      android:paddingLeft="16dp"
8      android:paddingRight="16dp"
9      android:paddingTop="16dp"
10     tools:context="at.htl.firstactivity.SubActivity">
11
12 </LinearLayout>
```

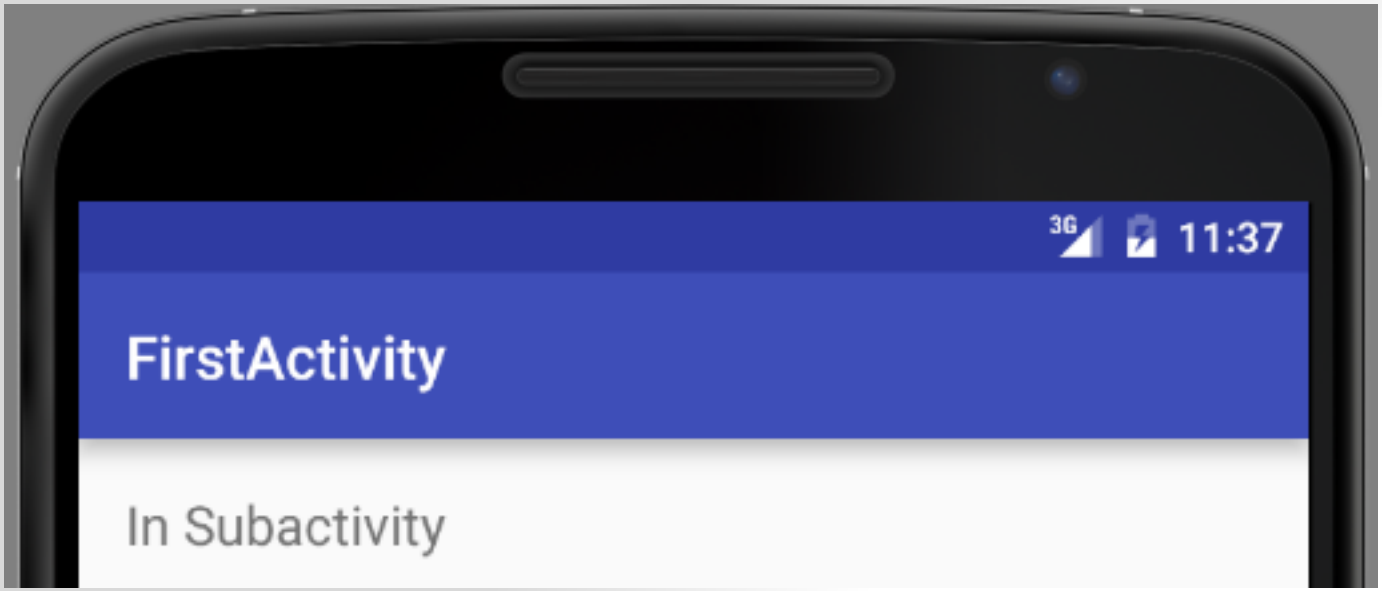
- Im Design-Mode: vertical

orientation	vertical

HTL LEONDING

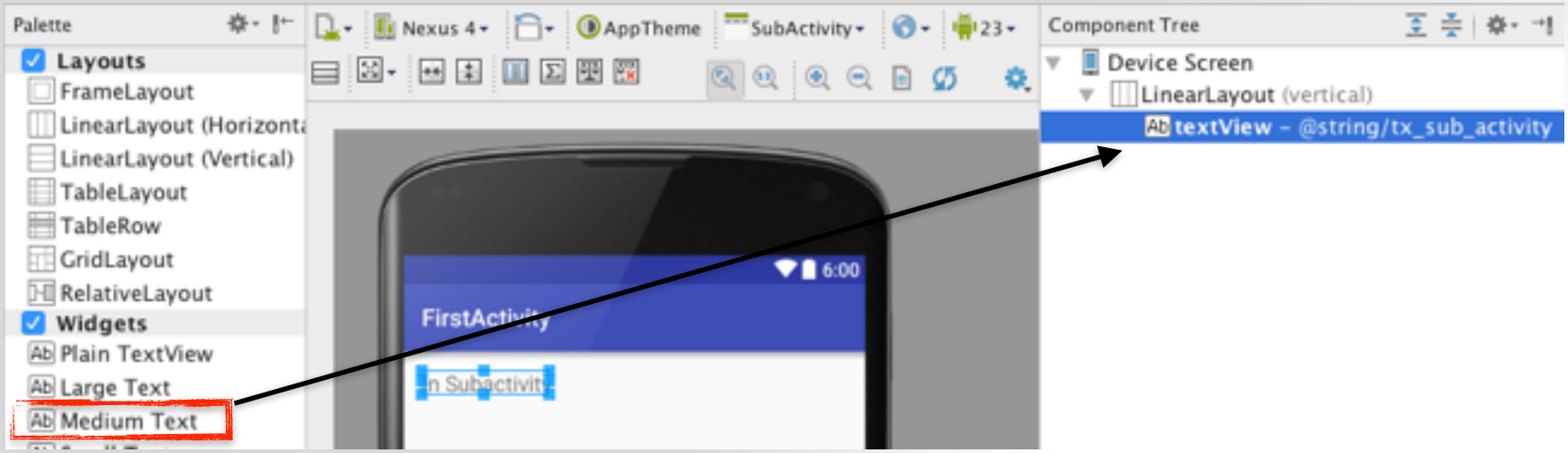
SubActivity: Einfachstes Layout

- Nur einen Text ausgeben



SubActivity: layout wieder in XML

- Id für TextView wird nicht benötigt
  - Kein Zugriff per Code



layout:width	match_parent
layout:height	match_parent

text	@string/tx_sub_activity

HTL LEONDING

SubActivity: Code mit Layout verbinden

SubActivity.java

```
1  package at.htl.firstactivity;
2
3  import ...
5
6  public class SubActivity extends AppCompatActivity {
7
8      @Override
9      protected void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.activity_sub);
12     }
13 }
```

- setContentView() stellt die Verbindung zwischen der Activity und ihrem Layout her

SubActivity in Manifest

- SubActivity wurde automatisch ins Manifest eingetragen

AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="at.htl.firstactivity">
4
5      <application
6          android:allowBackup="true"
7          android:icon="@mipmap/ic_launcher"
8          android:label="FirstActivity"
9          android:supportsRtl="true"
10         android:theme="@style/AppTheme">
11         <activity android:name=".MainActivity">
12             <intent-filter>
13                 <action android:name="android.intent.action.MAIN" />
14
15                 <category android:name="android.intent.category.LAUNCHER" />
16             </intent-filter>
17         </activity>
18         <activity android:name=".SubActivity"></activity>
19     </application>
20
21 </manifest>
```

Click-Funktionalität des Buttons

- Button-Clicks können auf zwei Arten erfasst werden:
  - ```
    Button subActivityButton =
            (Button) findViewById(R.id.subActivityButton);
    ```

    ```
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Call Subactivity"
        android:id="@+id/subActivityButton"
        android:onClick="onClick" />
    ```

    ```
    public void onClick(View view) {
        switch (view.getId()) {
            case R.id.subActivityButton:
                // your code
                break;
            case R.id.subActivityWithReturnButton:
                // your code
                break;
        }
    }
    ```

    Dies funktioniert aber bei Verwendung von Fragments nicht. Daher ist ein onClickListener zu bevorzugen.

  - onClickListener

    ```
    Button subActivityButton = (Button) findViewById(R.id.subActivityButton);
    subActivityButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // Your code
        }
    });
    ```
    ```

Main.java: onClick() realisieren

- Verbindung von Komponenten erfolgt über Intents
- Stellen Funktionalitäten bereit (z.B. Anzeige von Text)

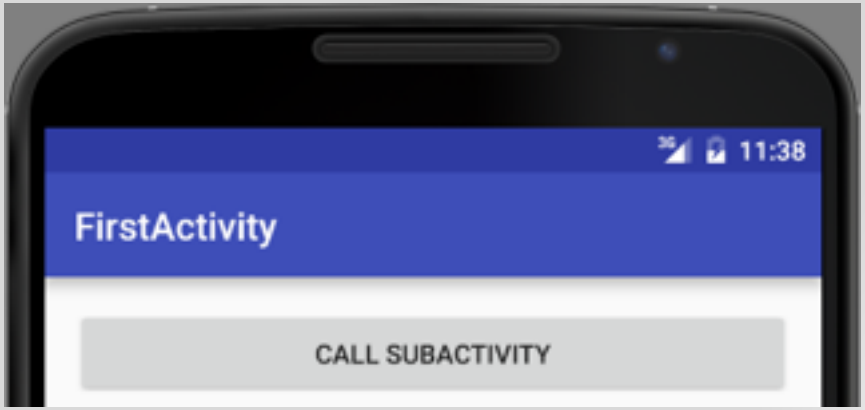
```
public class MainActivity extends AppCompatActivity {

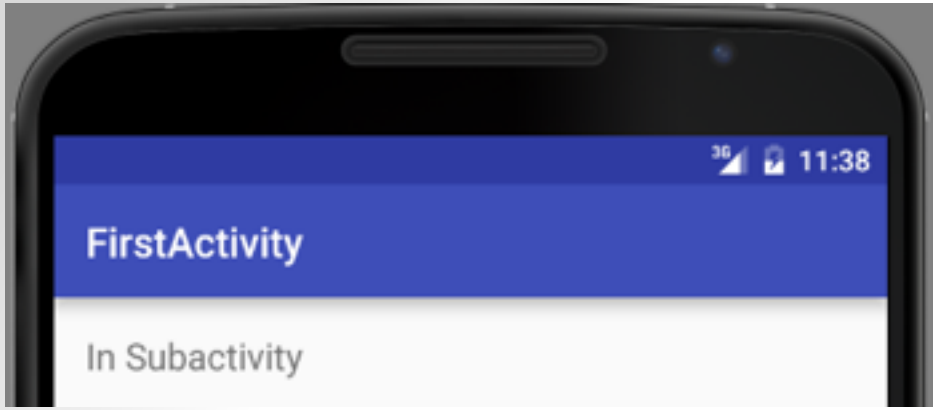
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button subActivityButton = (Button) findViewById(R.id.subActivityButton);
        subActivityButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(v.getContext(), SubActivity.class);
                startActivity(intent);
            }
        });
    }
}
```

Programm im Emulator testen

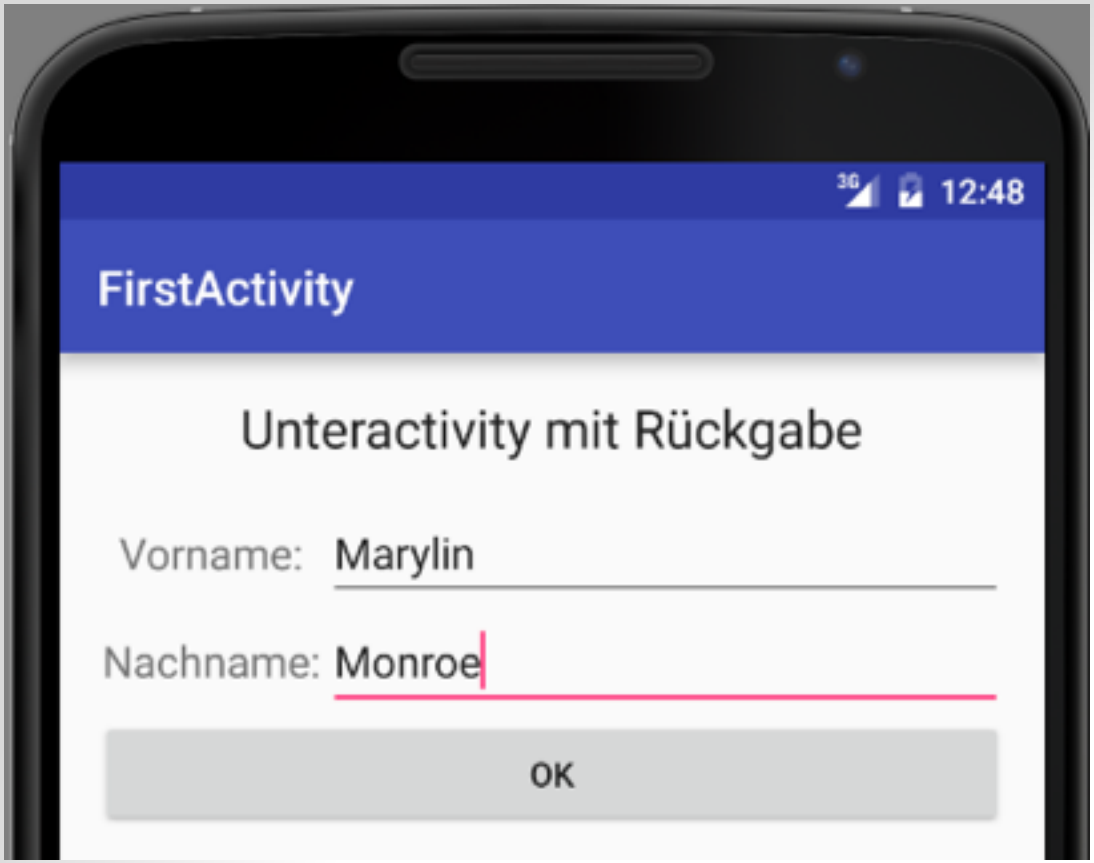
- Aufruf von Unteractivity über Button
  - Wenig überraschend
- Beenden über Zurück-Button





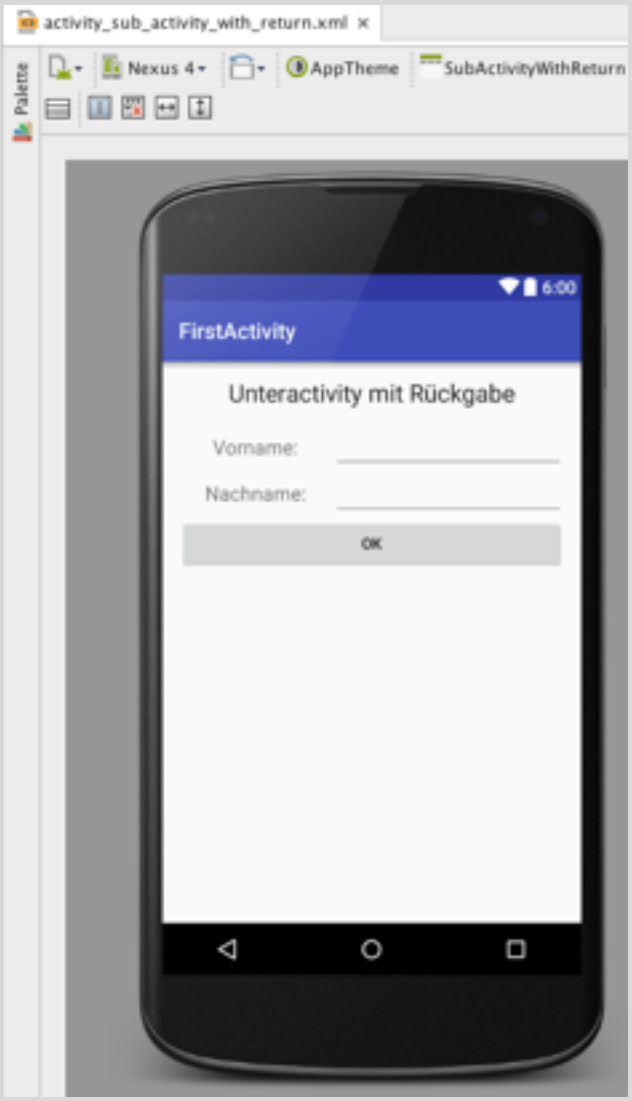
Zweite SubActivity soll Ergebnis rückmelden

- Eingabe von Vor- und Zuname in EditText
- Button zum Bestätigen
- Rückmeldung an Hauptactivity



HTL LEONDING

Layout der SubActivity mit Return 1



Component Tree

- Device Screen
  - LinearLayout (vertical)
    - TextView – @string/tx_subactivity...title
    - LinearLayout (horizontal)
      - TextView – @string/tx_first_name
      - **firstNameEditText** (EditText)
    - LinearLayout (horizontal)
      - TextView – @string/tx_last_name
      - **lastNameEditText** (EditText)
    - **getNameButton** (Button) – @string/bt_ok

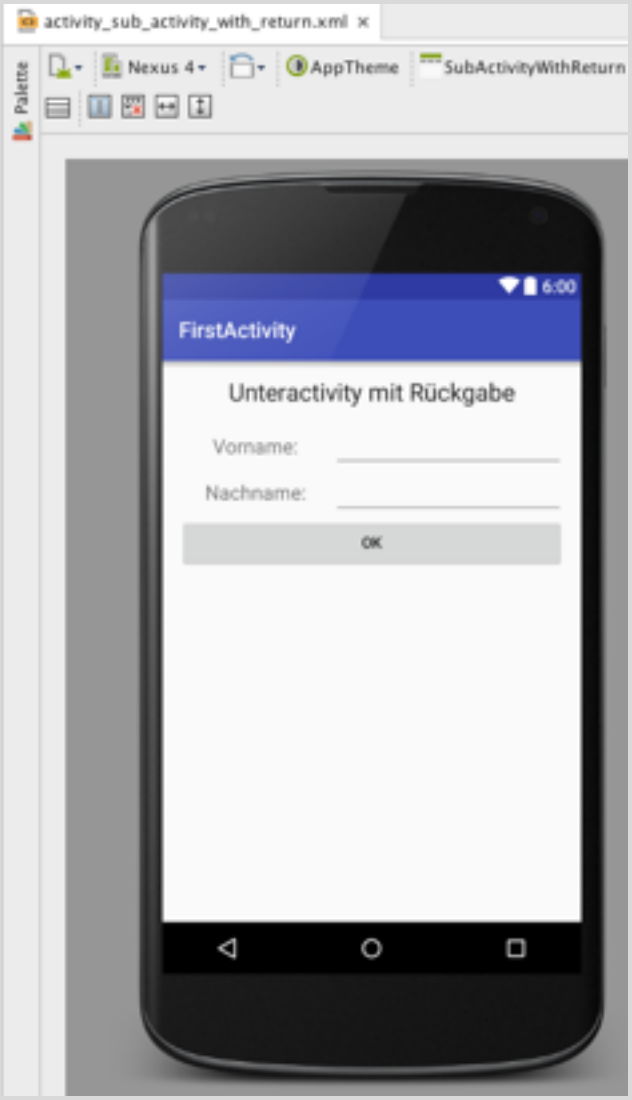
Properties

layout:width	match_parent
layout:height	match_parent
style	
orientation	vertical

▼ padding	[?, @dimen/activity_horizontal_margi
all	
left	@dimen/activity_horizontal_margin
top	@dimen/activity_vertical_margin
right	@dimen/activity_horizontal_margin
bottom	@dimen/activity_vertical_margin

HTL LEONDING

Layout der SubActivity mit Return 2



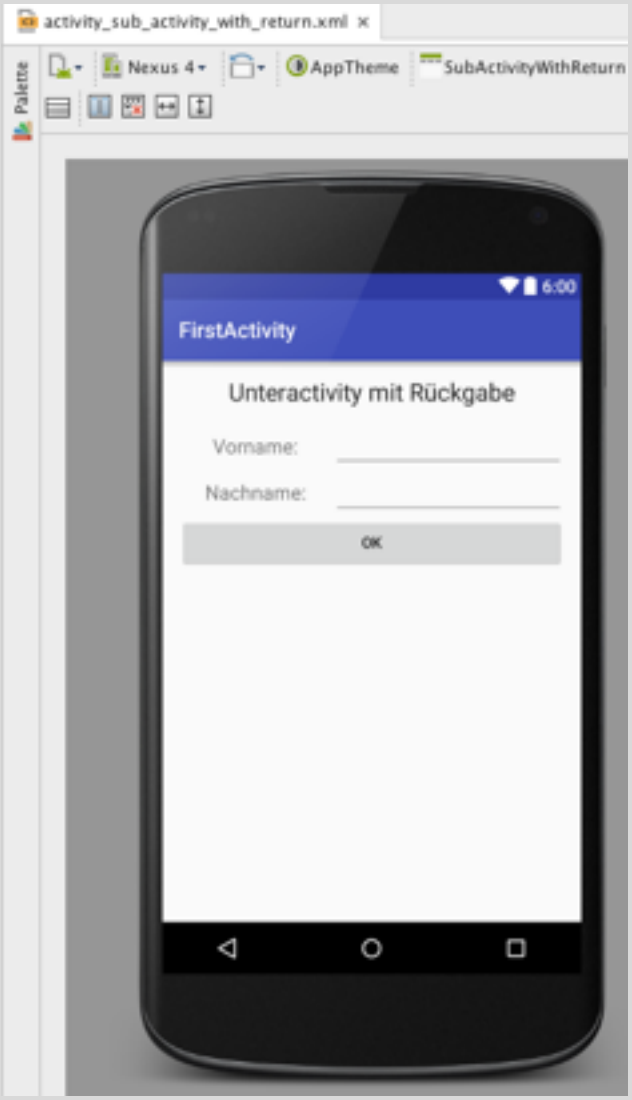
Component Tree

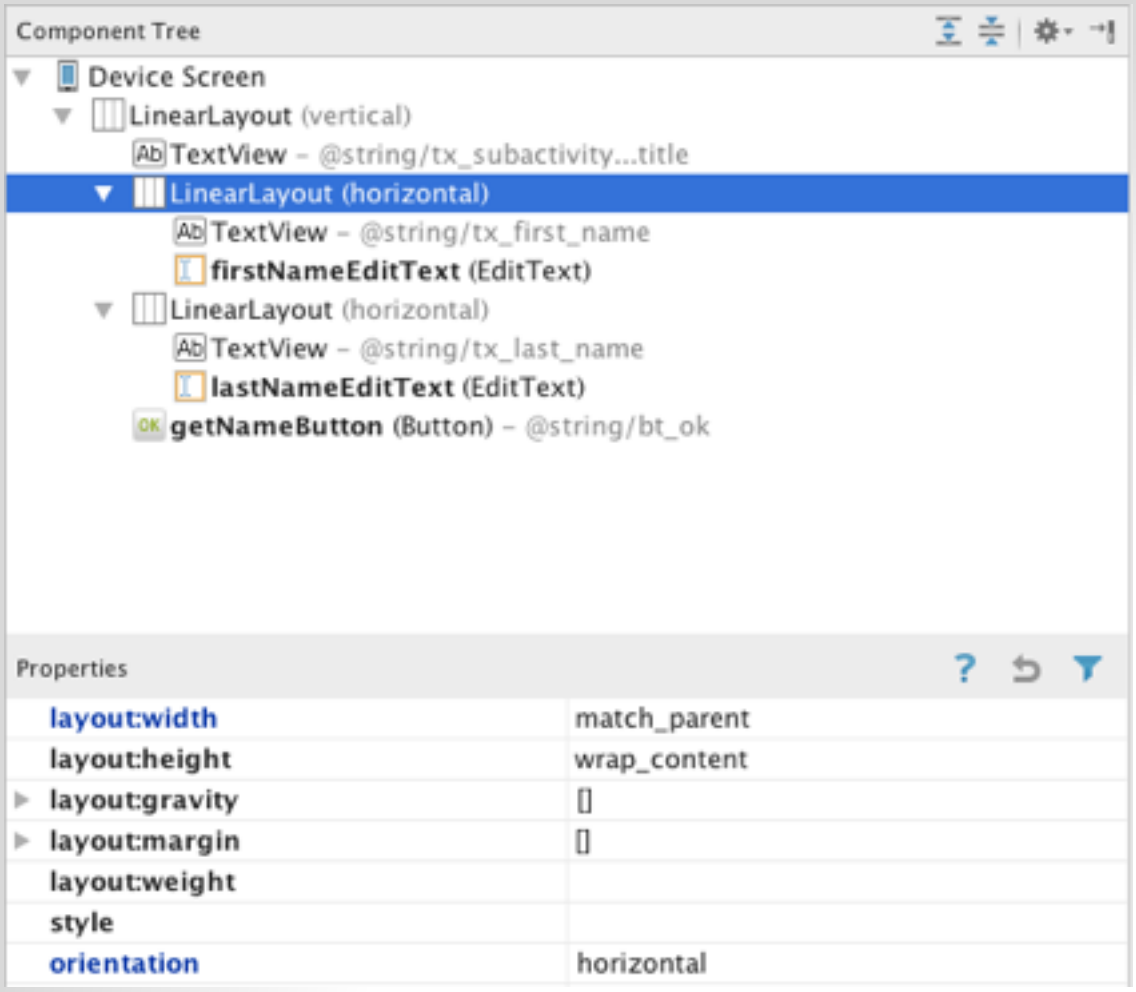
- Device Screen
  - LinearLayout (vertical)
    - TextView – @string/tx_subactivity...title
    - LinearLayout (horizontal)
      - TextView – @string/tx_first_name
      - **firstNameEditText** (EditText)
    - LinearLayout (horizontal)
      - TextView – @string/tx_last_name
      - **lastNameEditText** (EditText)
    - **getNameButton** (Button) – @string/bt_ok

Properties

layout:width	match_parent
layout:height	wrap_content
gravity	[center]
padding	[?, ?, ?, ?, **15dp**]
all	
left	
top	
right	
bottom	**15dp**
text	@string/tx_subactivity_with_return_title
textAlignment	
textAppearance	?android:attr/textAppearanceLarge

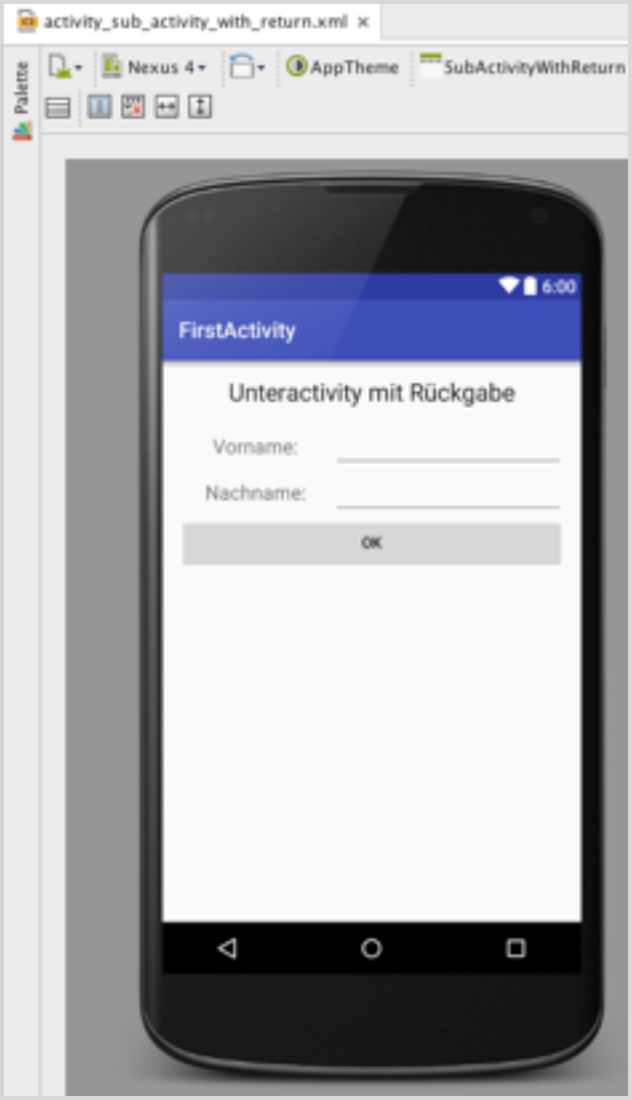
Layout der SubActivity mit Return 3

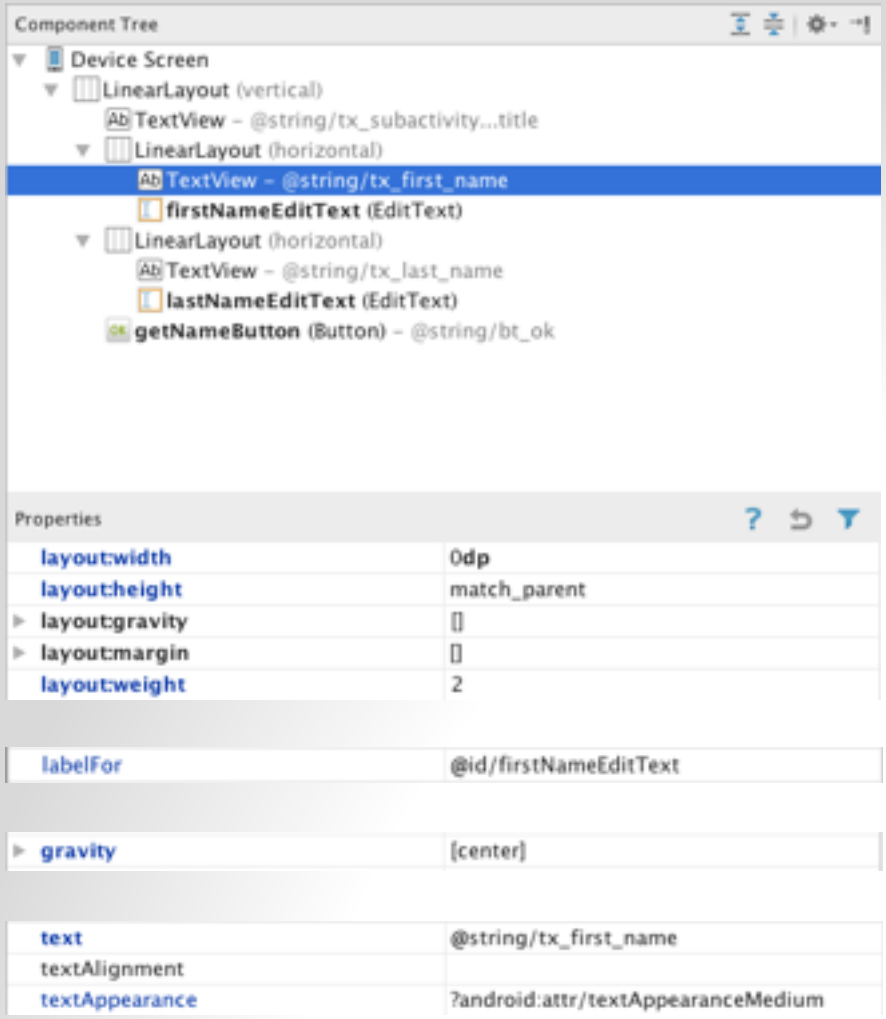




HTL LEONDING

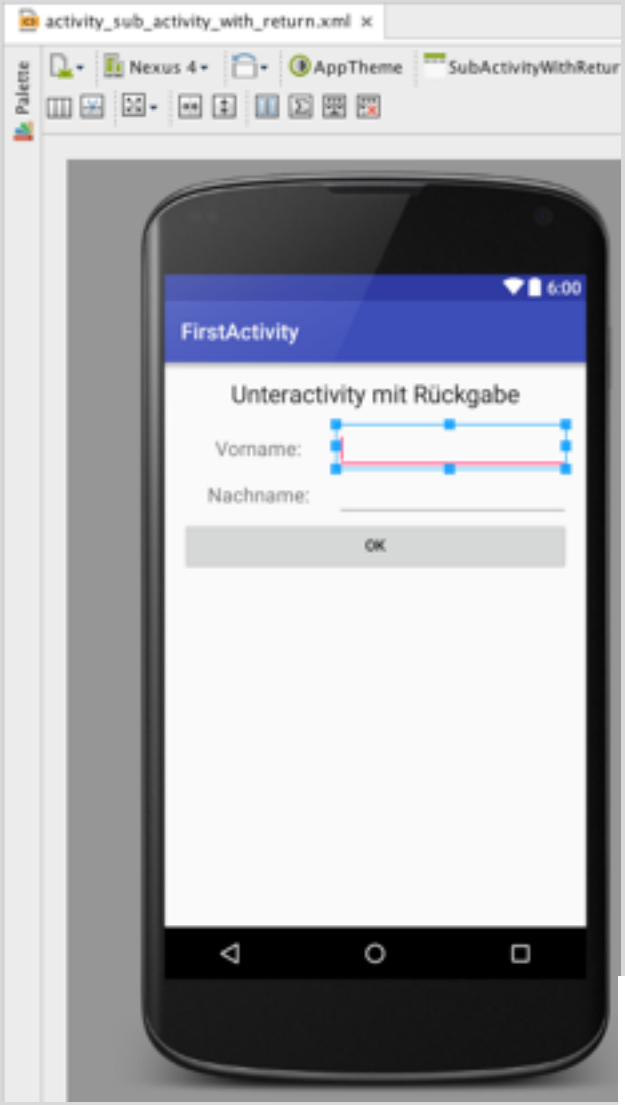
Layout der SubActivity mit Return 4

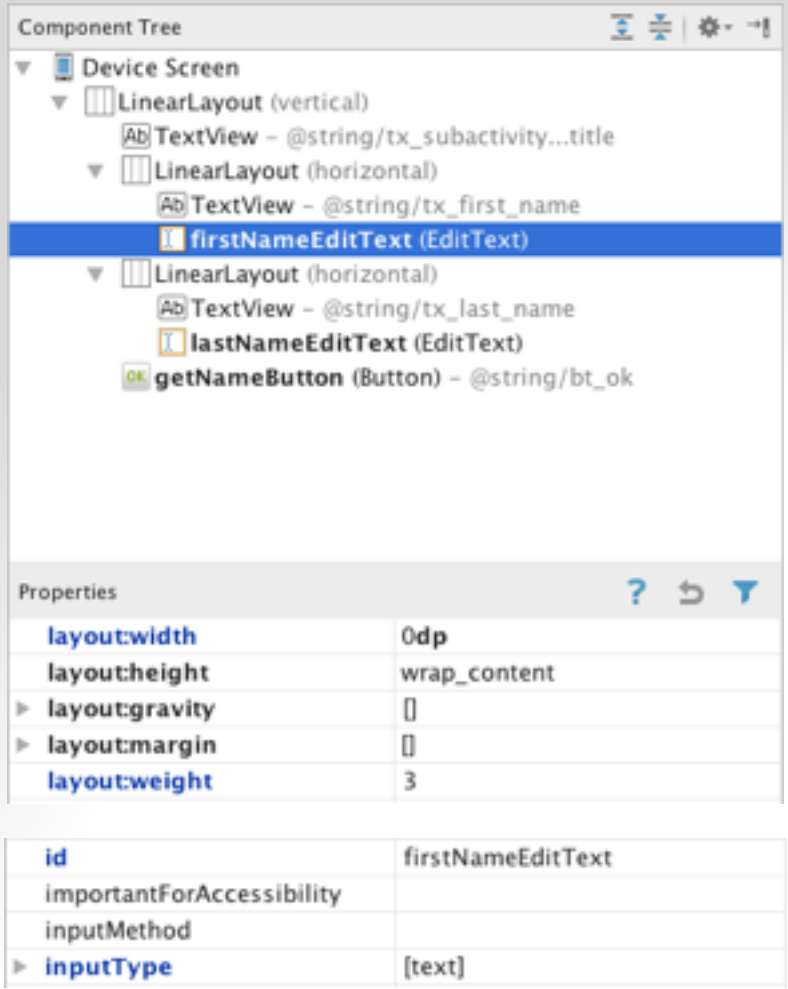




HTL LEONDING

Layout der SubActivity mit Return 5






```
<EditText
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:id="@+id/firstNameEditText"
    android:layout_weight="3"
    android:inputType="text">
    <requestFocus />
</EditText>
```

Focus auf Vorname setzen

Einige neue Texte

strings.xml

resources

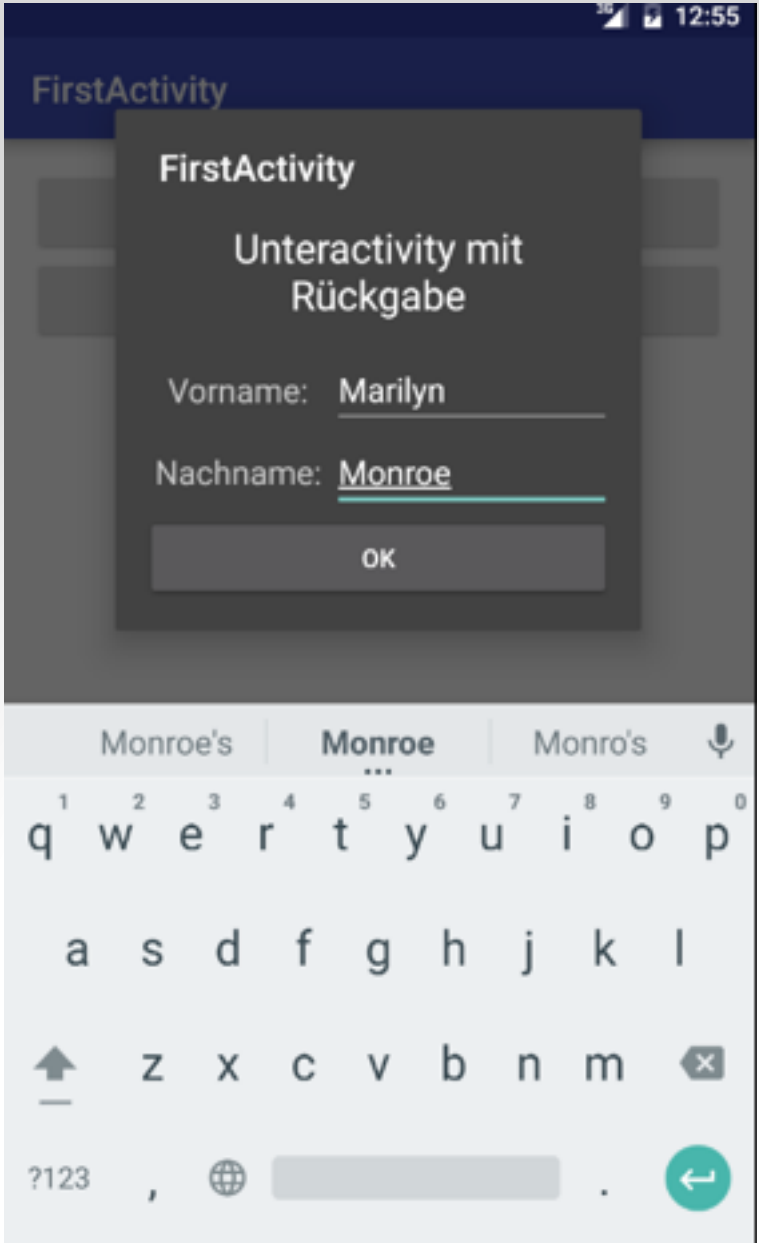
Edit translations for all locales in the translations editor. Open editor

```
1  <resources>
2      <string name="app_name">FirstActivity</string>
3      <string name="bt_call_subactivity">Call Subactivity</string>
4      <string name="tx_sub_activity">In Subactivity</string>
5      <string name="bt_ok">Ok</string>
6      <string name="tx_first_name">Vorname:</string>
7      <string name="tx_last_name">Nachname:</string>
8      <string name="bt_get_name">Namen eingeben</string>
9      <string name="tx_subactivity_with_return_title">Unteractivity mit Rückgabe</string>
10 </resources>
```

Subactivity als Dialog ausführen

- Belegt nur soviel Platz, wie notwendig
- Eintrag im Manifest zur Konfiguration

```
<activity
    android:theme="@style/Base.Theme.AppCompat.Dialog"
    android:name=".SubActivityWithReturn" />
```



Kontrollieren, ob Subactivities in Manifest

AndroidManifest.xml

manifest | application

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="at.htl.firstactivity">
4
5      <application
6          android:allowBackup="true"
7          android:icon="@mipmap/ic_launcher"
8          android:label="FirstActivity"
9          android:supportsRtl="true"
10         android:theme="@style/AppTheme">
11         <activity android:name=".MainActivity">
12             <intent-filter>
13                 <action android:name="android.intent.action.MAIN" />
14
15                 <category android:name="android.intent.category.LAUNCHER" />
16             </intent-filter>
17         </activity>
18         <activity android:name=".SubActivity" />
19         <activity
20             android:name=".SubActivityWithReturn" />
21     </application>
22
23 </manifest>
```

```
<activity
    android:theme="@style/Base.Theme.AppCompat.Dialog"
    android:name=".SubActivityWithReturn" />
```

ev. als Dialog ausführen
(nur durch Setzen eines Styles!)

SubActivity: Ergebnis liefern

- Intent für aufrufende Klasse instanzieren

```
Intent intent = new Intent(v.getContext(), MainActivity.class);
```

- Daten über Dictionary „anhängen“

```
intent.putExtra("firstName", firstNameEditText.getText().toString());
intent.putExtra("lastName", lastNameEditText.getText().toString());
```

- Result entsprechend setzen

```
setResult(RESULT_OK, intent);
```

- finish() beendet Activity und ruft Intent auf

```
finish();
```

SubActivityWithReturn: ButtonListener

```
private OnClickListener getNameButtonOnClickListener = new OnClickListener() {
    @Override
    public void onClick(View v) {
        EditText firstNameEditText = (EditText) findViewById(R.id.firstNameEditText);
        EditText lastNameEditText = (EditText) findViewById(R.id.lastNameEditText);

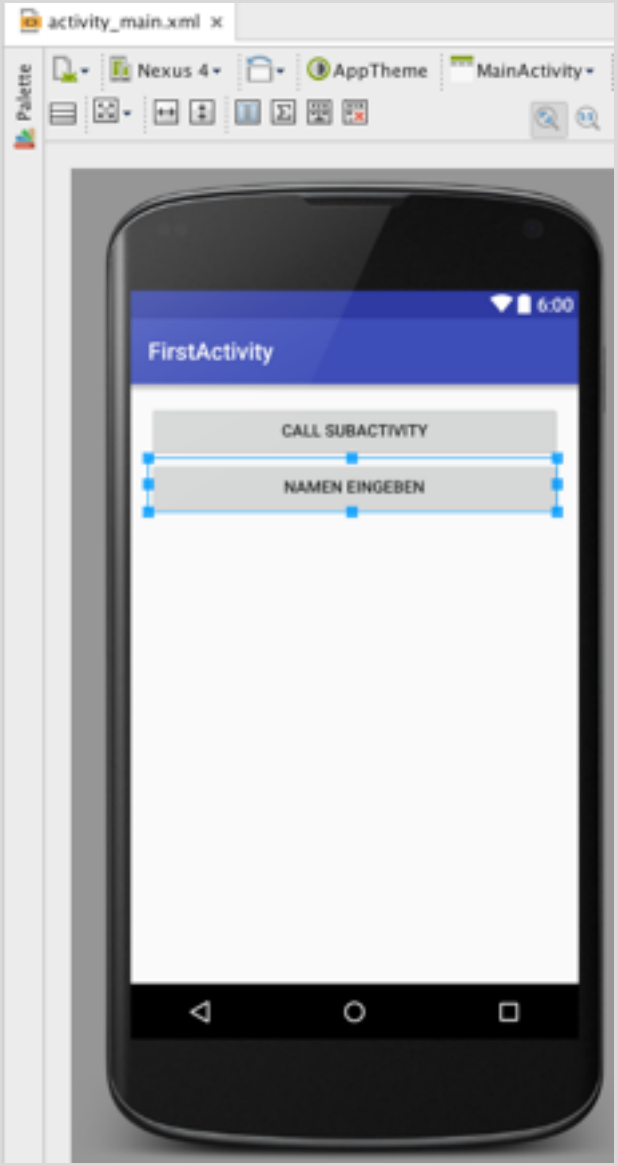
        Log.d(LOG_TAG, firstNameEditText.getText().toString());

        Intent intent = new Intent(v.getContext(), MainActivity.class);  // ruft Hauptklasse auf
        // Info über Dictionary mitgeben
        intent.putExtra("firstName", firstNameEditText.getText().toString());
        intent.putExtra("lastName", lastNameEditText.getText().toString());
        setResult(RESULT_OK, intent);
        finish();
    }
};
```

MainActivity muss angepasst werden

- Für jeden der zwei Buttons wird ein eigener ButtonListener erstellt
- Zweite SubActivity erhält anderen Intent
  - Intent wird mit Ergebnisanforderung gestartet
- Auf Ergebnis der zweiten SubActivity reagieren
  - Namen per Toast ausgeben

activity_main.xml erhält zweiten Button



Component Tree

- Device Screen
  - LinearLayout (vertical)
    - subActivityButton (Button) – @string
    - subActivityWithReturnButton (Butto

Properties

layout:width	match_parent
layout:height	wrap_content
id	subActivityWithReturnButton
text	@string/bt_get_name

HTL LEONDING

Main.java: ButtonListener erstellen

```
public class MainActivity extends AppCompatActivity {

    private final static String LOG_TAG = MainActivity.class.getSimpleName();

    // Zur Absicherung, ob richtige Activity geantwortet hat
    private static final int REQUEST_CODE_SUB_ACTIVITY = 4711;

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

        onClickListener fuer subActivityButton

        //region onClickListener fuer subActivityWithReturnButton
        Button subActivityWithReturnButton = (Button) findViewById(R.id.subActivityWithReturnButton);
        subActivityWithReturnButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intentWithExtras = new Intent(v.getContext(), SubActivityWithReturn.class);
                startActivityForResult(intentWithExtras, REQUEST_CODE_SUB_ACTIVITY);
            }
        });
        //endregion
    }
}
```

MainActivity.java: Ergebnis empfangen und verarbeiten

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode == Activity.RESULT_OK && requestCode == REQUEST_CODE_SUB_ACTIVITY) {
        String text = "Eingegebener Name: " +
                data.getExtras().get("firstName") + " " +
                data.getExtras().get("lastName");
        Toast.makeText(this, text, Toast.LENGTH_SHORT).show();
    }
    super.onActivityResult(requestCode, resultCode, data);
}
```

Ergebnis


11:33
FirstActivity
Unteractivity mit Rückgabe
Vorname:
Marilyn
Nachname:
Monroe
OK


10:15
FirstActivity
CALL SUBACTIVITY
NAMEN EINGEBEN
Eingegebener Name: Marylin Monroe

Noch Fragen?