

HÖHERE TECHNISCHE BUNDESLEHRANSTALT HOLLABRUNN

Abteilung für Elektronik – Technische Informatik

Klasse/ Jahrgang: 5BHEL	Gruppe: 2b	Übungsleiter: Prof. Geischläger
Übungsnummer: III/2	Übungstitel: Android	
Datum der Übung: 16.02.2016	Teilnehmer: Eichhorn, Haschke, Holzmann, Leeb, Weixelbraun	
Datum der Abgabe: 01.03.2016	Schriftführer: Haschke	Unterschrift:

	Beurteilung
Aufgabenstellung	
Dokumentation	
Messschaltungen	
Messtabellen	
Berechnungen	
Programmlistings	
Auswertung	
Diagramme	
Berechnungen	
Simulationen	
Schlußfolgerungen	
Kommentare	
Inventarliste	
Messprotokoll	
Form	
Summe	

Allgemeiner Teil

Titel der Übung: Android
Übungsnummer: III/2
Übungsplatz: T4
Datum der Übung: 16.02.2016
Klasse: 5BHEL
Schriftführer: Haschke
Übungsteilnehmer: Eichhorn
Haschke
Holzmann
Leeb
Weixelbraun

Inhaltsverzeichnis

1. Android.....	3
1.1 Allgemeines.....	3
1.2 Aufgabenstellung	3
1.3 Erstellen eines Projektes	3
1.4 Erstellen eines einfachen User-Interfaces	7
1.5 Eine zweite Activity starten.....	11
1.6 Erstellen der zweiten Activity	12
1.7 Ergebnis.....	15
1.8 Erkenntnis	15
2. Inventarliste	16
3. Messprotokoll	17

1. Android

1.1 Allgemeines

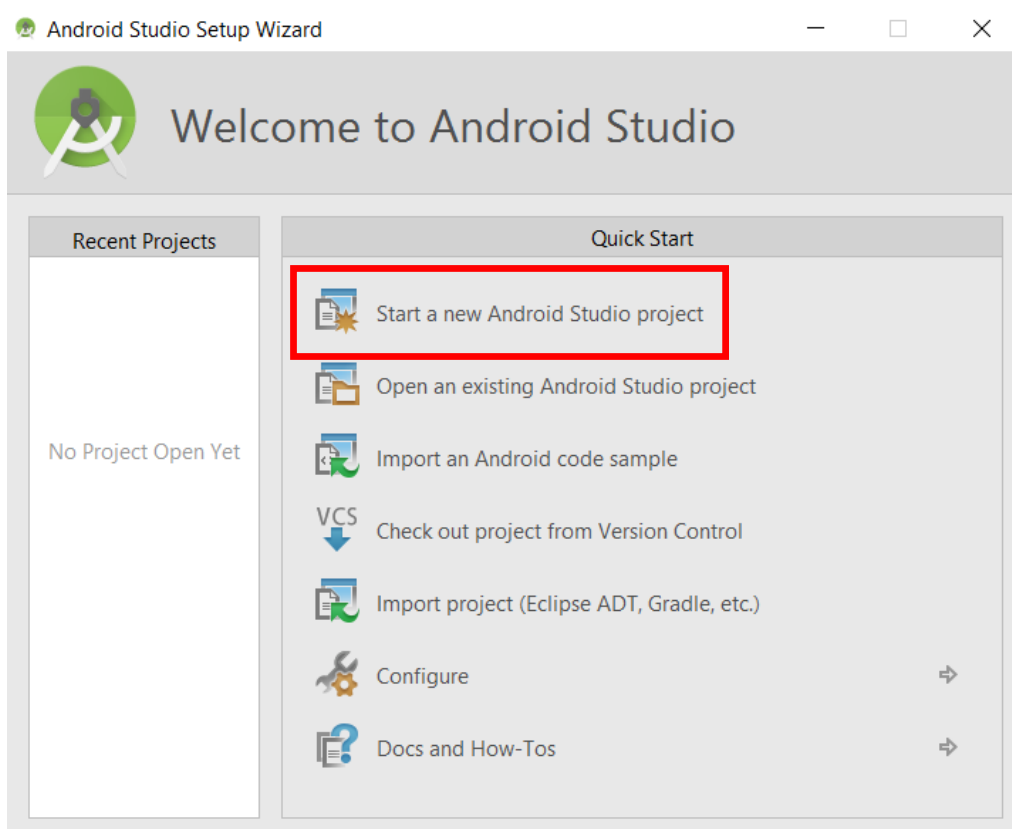
Benutzte Software:	Windows 7 Enterprise Service Pack 1 Android Studio 1.5.1
Benutzte Hardware:	Siehe Inventarliste
Tutorial für Android-Programmierung:	http://developer.android.com/training/index.html

1.2 Aufgabenstellung

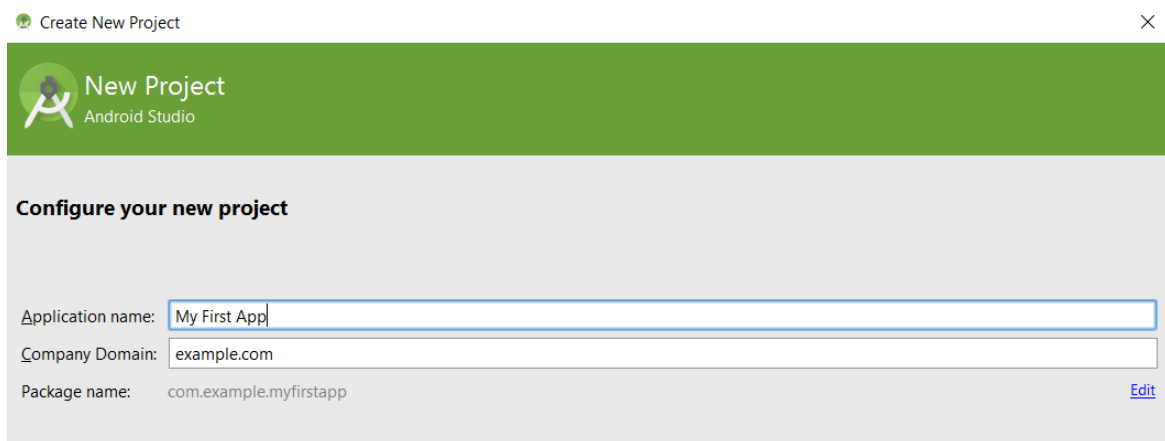
Ziel dieser Übung war es mit Android Studio 1.5.1 anhand des Tutorials von „<http://developer.android.com/training/index.html>“ eine Android App zu erstellen und mit dem internen Simulator von Android Studio zu testen.

1.3 Erstellen eines Projektes

Im ein neues Projekt anzulegen wird im „Welcome“-Screen von Android Studio die Funktion „Start a new Android Studio Projekt“ ausgewählt. Ist bereits ein aktives Projekt geöffnet, kann über „File“ => „New“ => „New Project“ ebenfalls ein neues Projekt geöffnet werden.

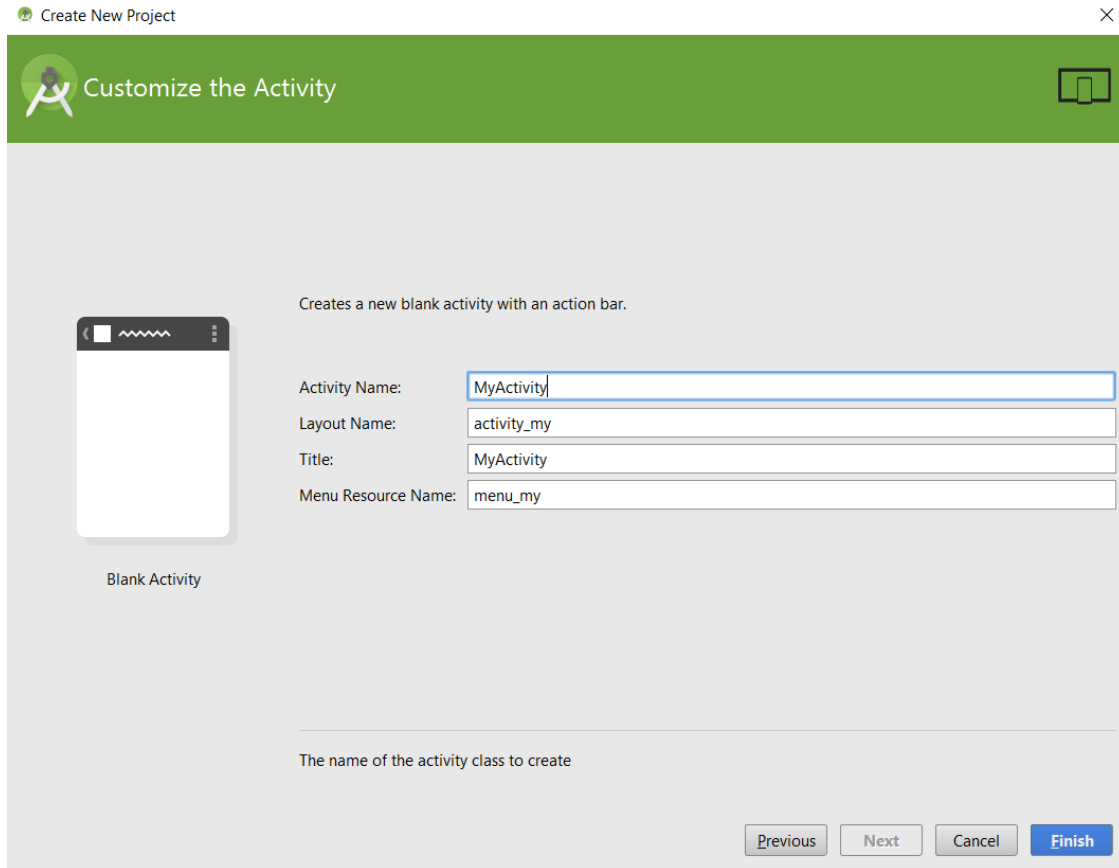


Im nächsten Fenster werden der „Application Name“, die „Company Domain“ und sowie der Speicherort des Projektes eingetragen. Als „Application Name“ wird „My First App“ verwendet um mit dem Tutorial übereinzustimmen. Aus dem „Application Name“ und der „Company Domain“ ergibt sich der „Package Name“, welcher auf einem Android System nur einmal vorkommen darf.



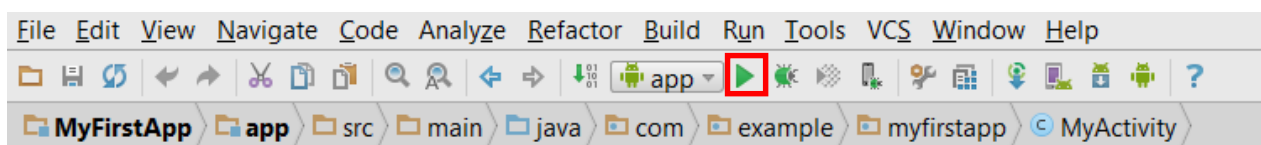
Im nächsten Fenster wird ausgewählt auf welchen Geräten die App später verwendet werden soll. Hier wird „Phone and Tablet“ ausgewählt.
Danach wird im folgenden Fenster eine „Activity“ ausgewählt. In unserem Fall wird hier einfach eine „Blank Activity“ ausgewählt.

Im abschließenden „Create New Project“-Fenster wird die gewählte „Activity“ angepasst. Es wird hier ein Name für die „Activity“ vergeben, werden die restlichen Felder nicht manuell auf etwas anderes geändert, dann werden sie automatisch an den neuen „Activity Name“ angepasst. Im mit dem Tutorial übereinzustimmen wird „MyActivity“ als „Activity Name“ gewählt.

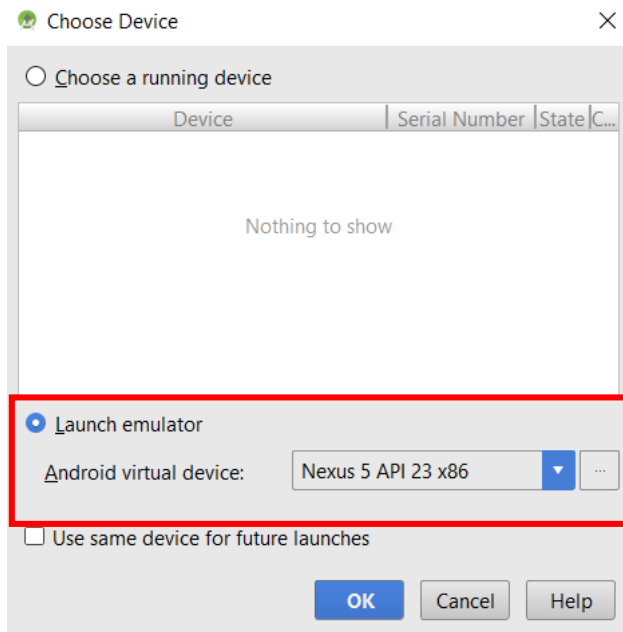


Nach dem Android Studio die App geladen hat, kann diese auch ausgeführt werden. Zu diesem Zeitpunkt zeigt die App jedoch nur den Text „Hello World“ in einer leeren „Activity“ an.

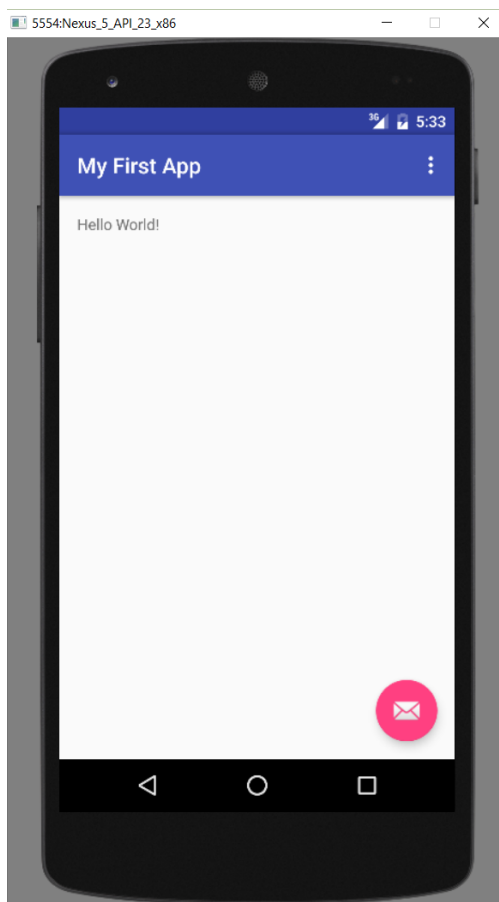
Um die App auszuführen und zu testen muss zuerst der Simulator gestartet werden. Dazu wird einfach der „Run ‘app‘“-Button betätigt.



Im folgenden Fenster wird „Launch emulator“ und das „Android virtual device“ „Nexus 5 API 23 x86“ ausgewählt.

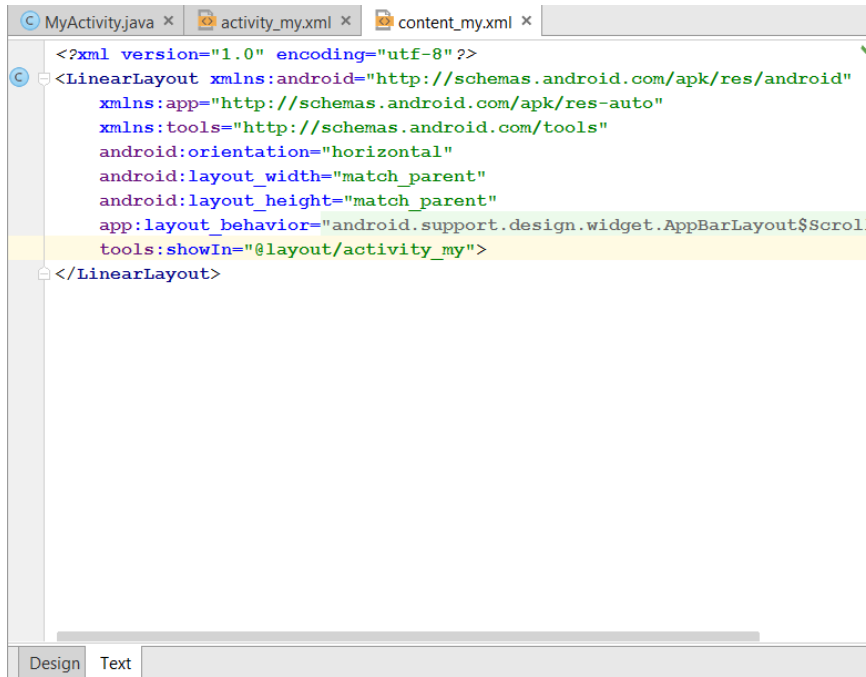


Danach sollte sich der Emulator öffnen, dies kann einige Zeit in Anspruch nehmen. Im Emulator wird nun die App geöffnet. Der Emulator wird nun nicht mehr geschlossen um die erneuten langen Ladezeiten zu vermeiden.



1.4 Erstellen eines einfachen User-Interfaces

Dem Tutorial folgenden wird zuerst das bestehende relative Layout im File „content_my.xml“ in ein lineares Layout umgewandelt. Außerdem wird der bestehende „TextView“, sowie die Zeilen mit „android:padding“ und „tools:context“ gelöscht und die Zeile „android:orientation=„horizontal““ hinzugefügt.



Nun wird ein Textfeld hinzugefügt. Dafür wird in „content_my.xml“ im linear Layout ein „Edit Text“ eingefügt. In dem „Edit Text“ werden folgende „android“ Attribute definiert.

- „android:id=„@+id/edit_message““: In diesem Attribut wird die ID des Views festgelegt. Das „@“ Zeichen wird benötigt wenn ein XML-Objekt referenziert wird. Über das „+“ Zeichen wird nur beim ersten definieren der Ressource benötigt. „id“ definiert um welche Art von XML-Objekt es sich handelt, und nach dem Slash folgt der Name der Ressource.
- „android:layout_height“ und „android:layout_width“: Mit diesen Attributen werden die Breite und Höhe des Views definiert. „wrap_contents“ gibt dabei vor, dass der View nur so groß wird, wie für den Inhalt benötigt wird.
- „android:hint“: Dies ist der Text der Angezeigt wird, wenn das Textfeld leer ist. „@string/edit_message“ => „@“ verweist wieder auf ein XML-Objekt, „string“ definiert, dass es sich bei diesem Objekt um eine Zeichenkette handelt und nach dem Slash folgt der Name der Zeichenkette. Der Inhalt dieser Zeichenkette wird in einem eigenen File, „values/strings.xml“, definiert.

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="android.support.design.widget.AppBarLayout$Scroll
    tools:showIn="layout/activity_my">
    <EditText
        android:id="@+id/edit_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />
</LinearLayout>

```

Nun wird die Zeichenkette im zugehörigen File, „values/string.xml“, eingetragen. Da im Anschluss ein Button erstellt wird, wird der Text für den Button ebenfalls gleich eingetragen. Innerhalb der „resources“ wird ein „string“ mit „name=“edit_message““ bzw. „button_send“ definiert.

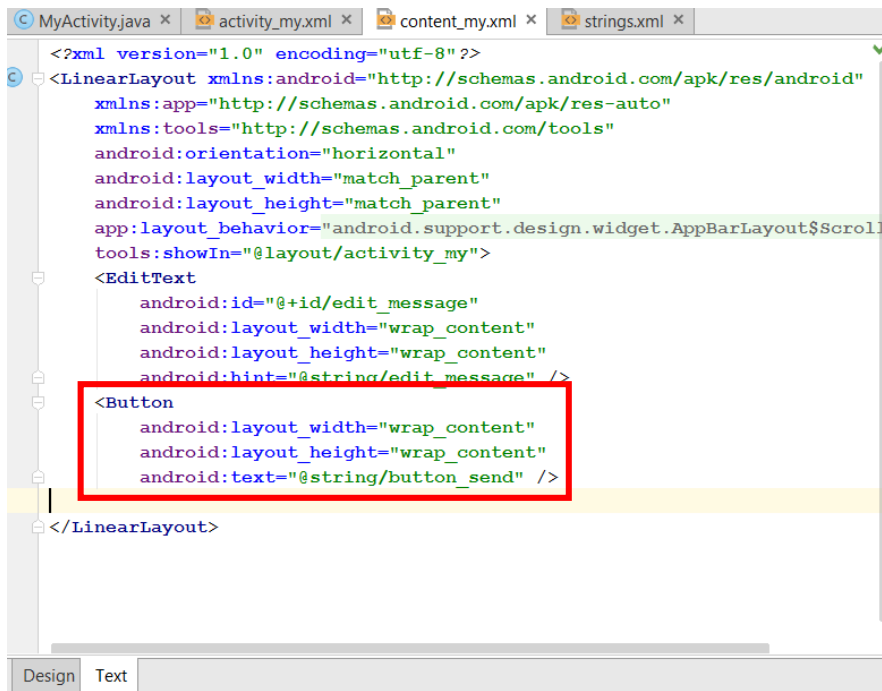
```

<resources>
    <string name="app_name">My First App</string>
    <string name="action_settings">Settings</string>
    <string name="edit_message">Enter a message</string>
    <string name="button_send">Send</string>
</resources>

```

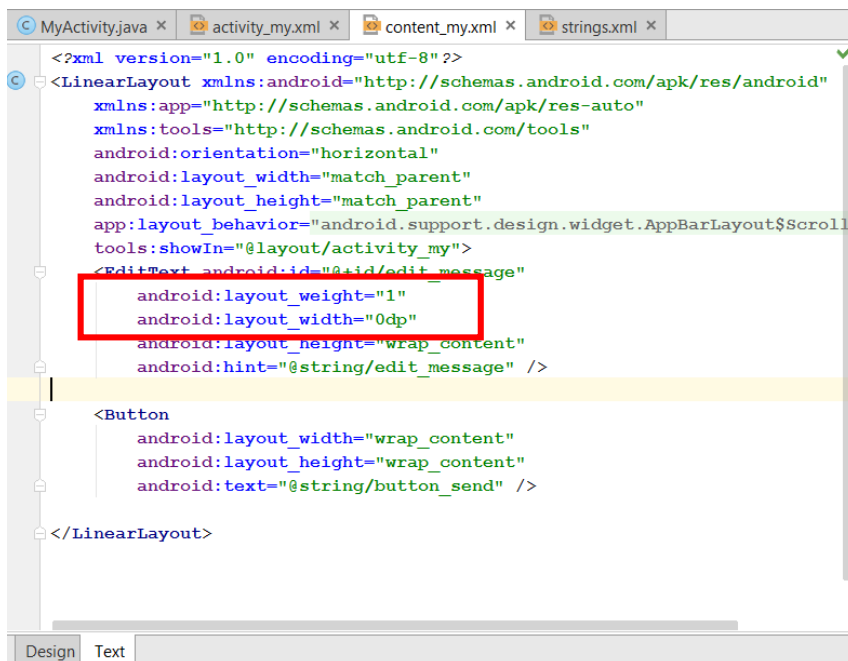
Nun wird der Button im „content_my.xml“ hinzugefügt. Dazu wird im linearen Layout ein „Button“ mit folgenden Attributen eingefügt.

- „android:layout_height“ und „android:layout_width“: Beide werden wieder auf „wrap_content“ gesetzt damit der Button nicht größer als sein Inhalt erfordert wird.
- „android:text“: Dieses Attribut definiert das Label des Buttons, sprich den Text der innerhalb des Buttons steht. Die Zeichenkette dafür wurde bereits im vorherigen Schritt definiert.

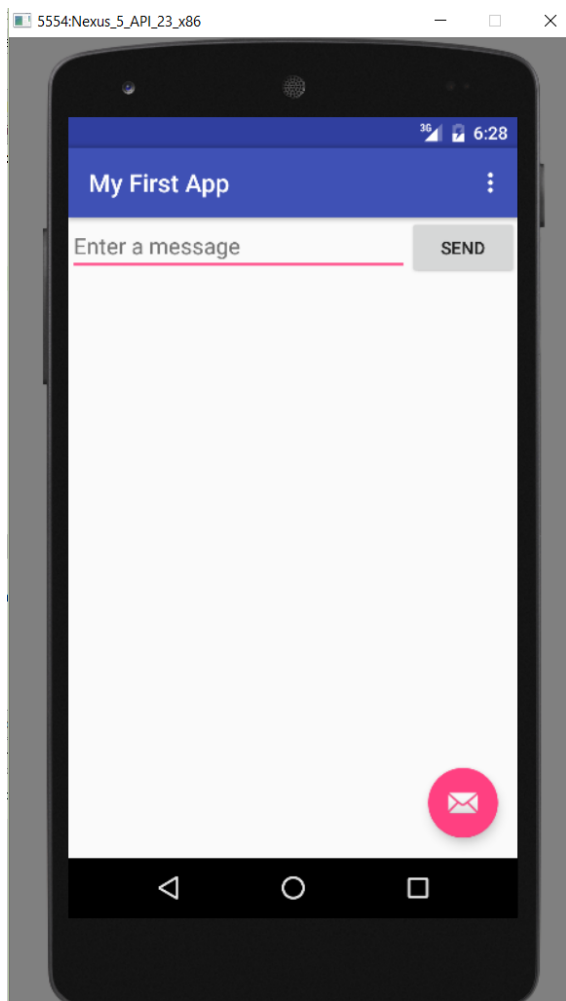


Aufgrund des variablen Inhalts des Textfeldes ist „wrap_contents“ für „layout_width“, also die Breite des Layouts, nicht gut geeignet, stattdessen wird das Textfeld im nächsten Schritt so umgeändert, dass es den Rest des Bildschirms ausfüllt. Dafür werden im „EditText“ folgende Attribute definiert.

- „android:layout_weight“: Dieses Attribut gibt an wie viel Platz dieser „View“ relativ zu allen verwandten „Views“ einnimmt. Definiert man zum Beispiel zwei „Views“ mit jeweils einem „weight“ von 1, würden beide „Views“ jeweils die Hälfte des Platzes bekommen. Da hier nur ein View existiert bekommt dieser den gesamten freien Platz.
- „android:layout_width“: Diese Attribut wird auf 0 gesetzt um das Layout performanter zu machen, da die Breite des „Views“ sowieso über das „layout_weight“ definiert ist.

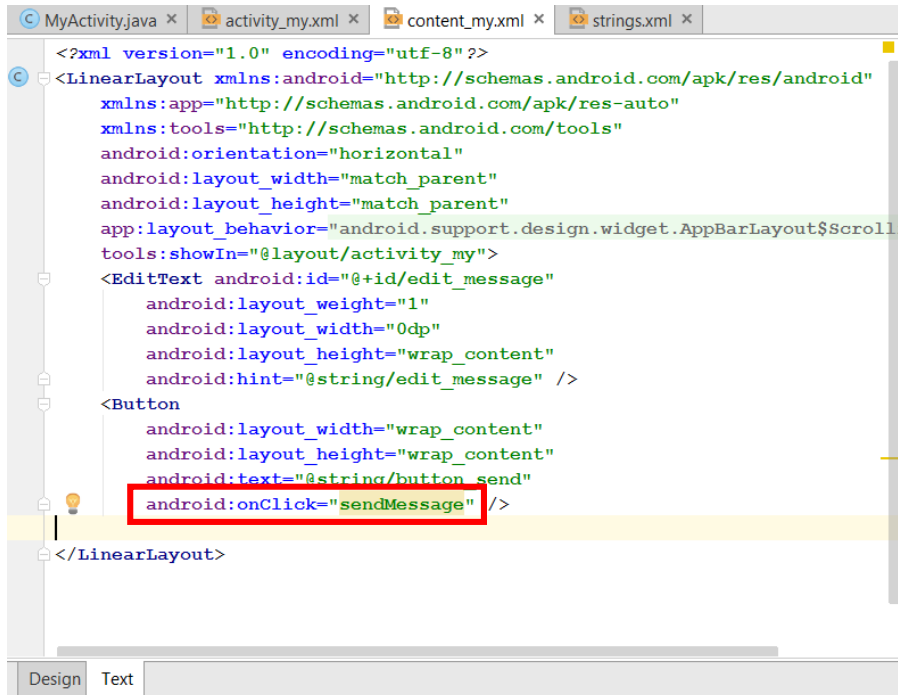


Das Ergebnis:



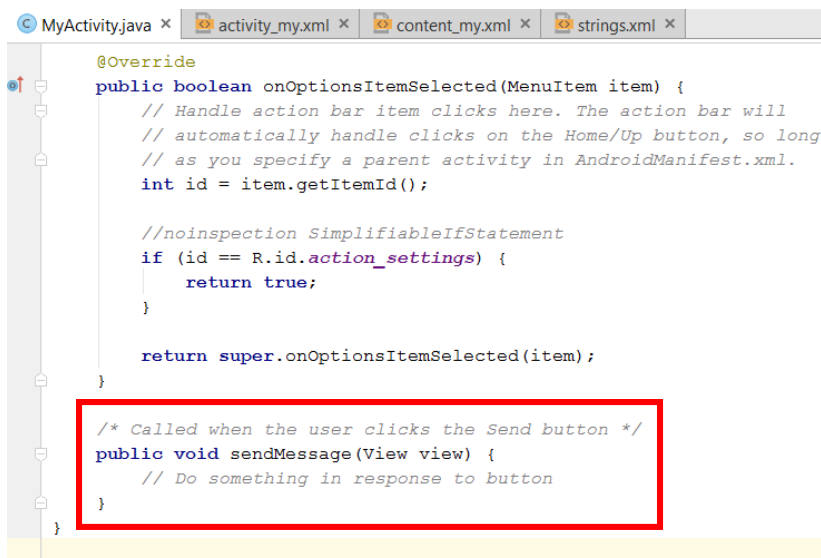
1.5 Eine zweite Activity starten

Zuerst wird im „content_my.xml“ der Button um das Attribut „android:onClick“ erweitert. Dieses wird auf „sendMessage“ gesetzt.



Nun wird im „MyActivity.java“-File die „sendMessage()“-Methode hinzugefügt. Diese muss mit der Methode „sendMessage“ aus dem „content_my.xml“ übereinstimmen. Dafür müssen folgende Kriterien erfüllt sein.

- Die Methode muss „public“ sein.
- Die Methode muss einen „void“-Rückgabewert haben.
- Die Methode muss einen „View“ als einzigen Parameter haben. Dieser wird der „View“ sein welcher die Methode aufruft.



Nun wird die Methode um einen „Intent“ erweitert. Ein „Intent“ ist ein Objekt, welches zwei separate Komponenten zur Laufzeit verbindet. „MyActivity.java“ muss um ein „include“ für „Intent“ erweitert werden. Dies kann direkt über „Alt“ + „Eingabe“ erledigt werden. Die Parameter für das neue „Intent“ sind:

- „this“: Der erste Parameter verweist auf einen „Context“, da die Activity-Klasse eine Subklasse von „Context“ ist, wird „this“ verwendet.
- „DisplayMessageActivity.class“: Der zweite Parameter verweist auf die Klasse des Komponenten, an welchen der „Intent“ vom System übergeben wird. Hier wird auf die zweite Activity „DisplayMessageActivity“ verwiesen.

Über die Funktion „findViewById“ wird das „EditText“ eingelesen. Erneut wird die fehlende „EditText“ Klasse mit „Alt“ + „Eingabe“ hinzugefügt. Der Text aus dem Inputfeld wird einem lokalen String, „message“, zugewiesen. Mit der Funktion „putExtra()“ wird der Text dem „Intent“ hinzugefügt.

Am Beginn der Klasse wird „EXTRA_MESSAGE“ als „public final static String“ definiert. Nun wird am Ende der „sendMessage“ Methode die „startActivity()“ Funktion aufgerufen, welcher der zuvor erstellte „Intent“ übergeben wird.

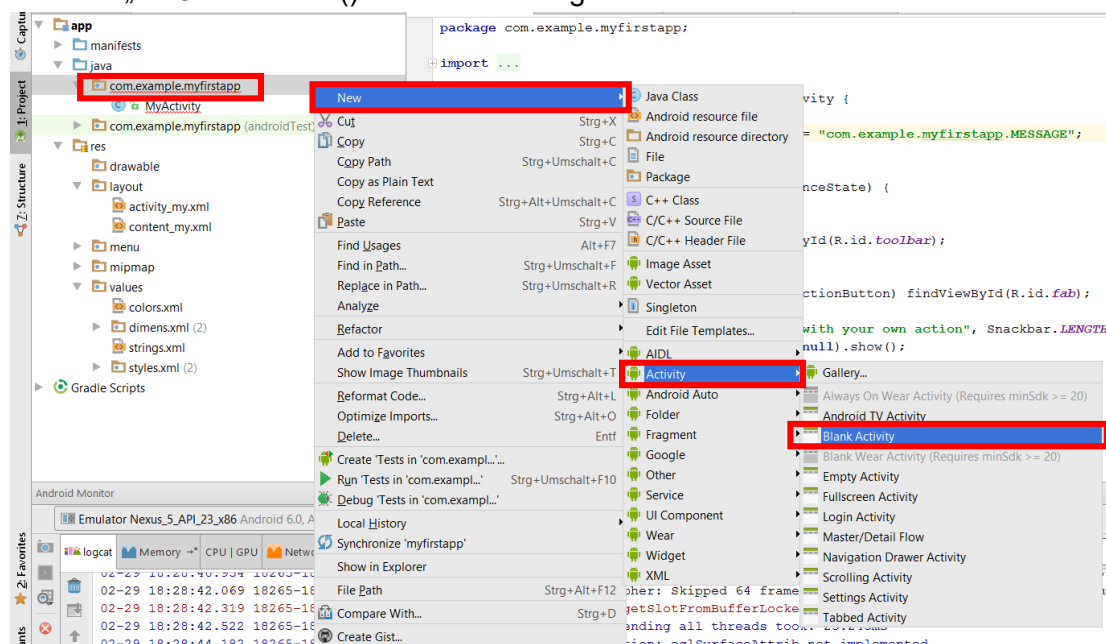
```
public class MyActivity extends AppCompatActivity {

    public final static String EXTRA_MESSAGE = "com.example.myfirstapp.MESSAGE";

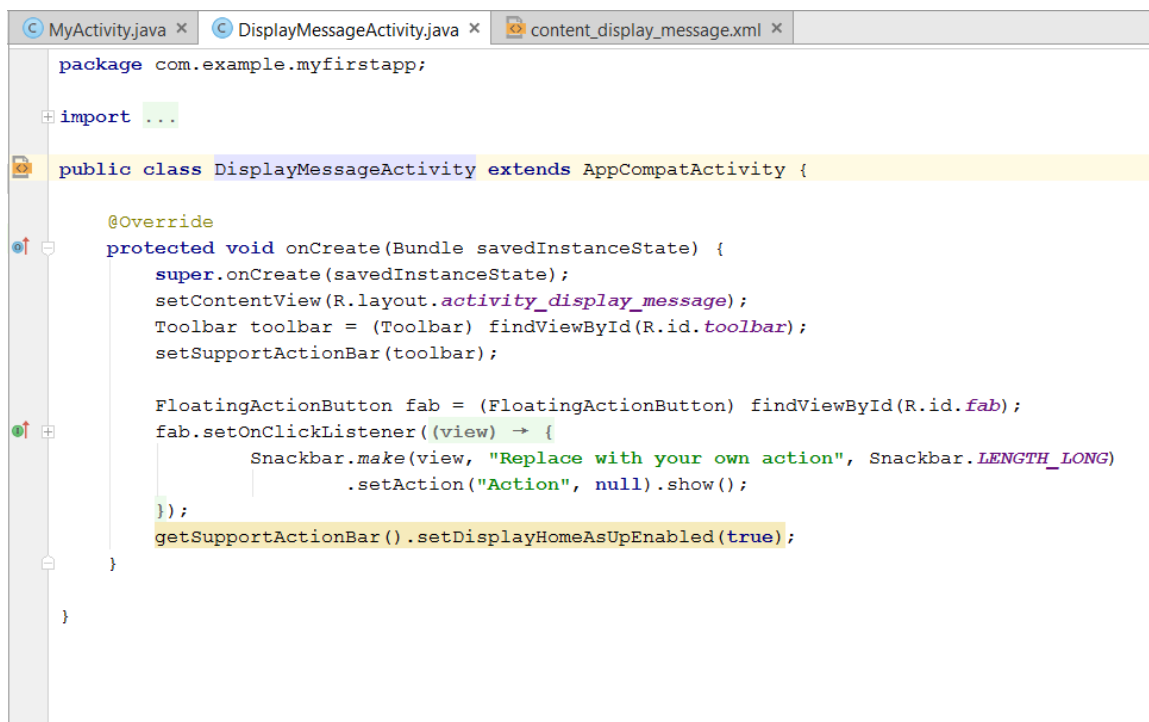
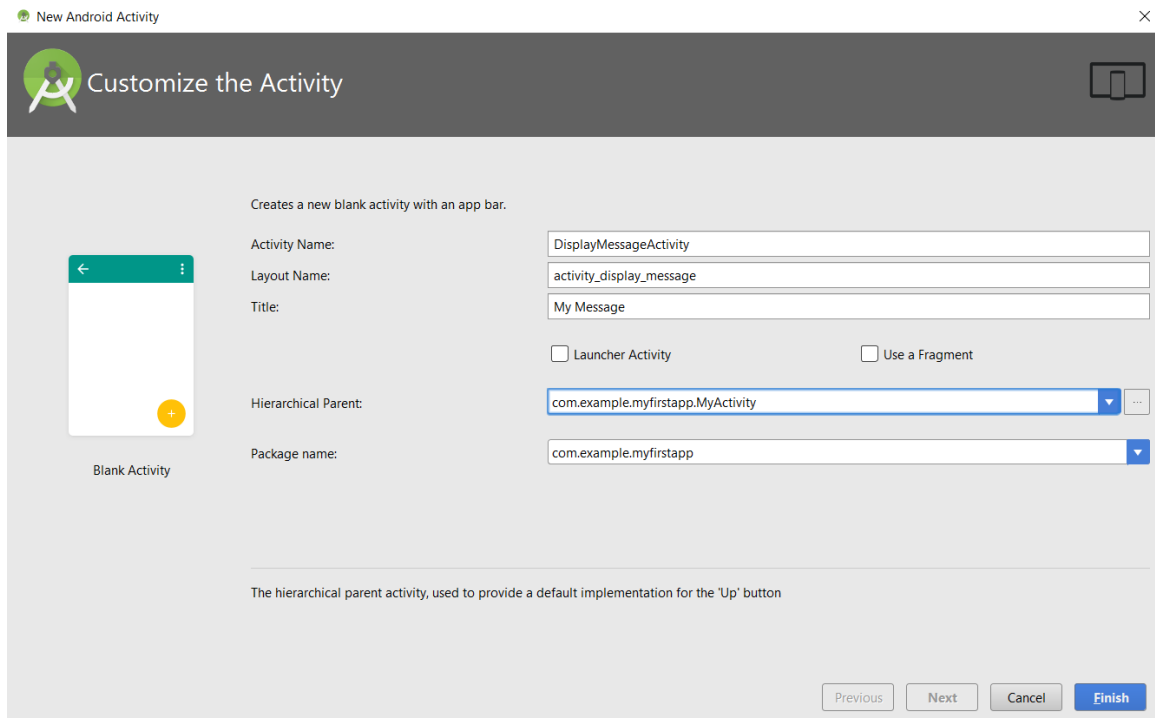
    /* Called when the user clicks the Send button */
    public void sendMessage(View view) {
        Intent intent = new Intent(this, DisplayMessageActivity.class);
        EditText editText = (EditText) findViewById(R.id.edit_message);
        String message = editText.getText().toString();
        intent.putExtra(EXTRA_MESSAGE, message);
        startActivity(intent);
    }
}
```

1.6 Erstellen der zweiten Activity

Alle Subklassen der „Activity“ müssen eine „onCreate()“ Methode implementieren. Weiters muss in dieser „onCreate()“ Methode das Layout der „Activity“ definiert werden. Dies wird über die „setContentView()“ Funktion erledigt.



Über im Java-Ordner den „Package Name“ auswählen, in diesem Fall „com.example.myfirstapp“, dann „New“ => „Activity“ => „Blank Activity“ auswählen. Im folgenden Fenster werden der „Activity Name“, „Layout Name“, „Title“, „Hierarchical Parent“ und „Package Name“ vergeben.



Jede „Activity“ wird von einem „Intent“ aufgerufen. Mit „getIntent()“ können die Daten des „Intents“ der die „Activity“ aufgerufen hat abgerufen werden. Mit „getStringExtra()“ kann das mitgelieferte Extra abgerufen werden.

Nun wird das relative Layout der „DisplayMessageActivity“ im File „content_display_message.xml“ um eine „android:id“ erweitert. Dies wird benötigt um das Layout im Java-Code zu referenzieren.

```
<?xml version="1.0" encoding="utf-8" ?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    app:layout_behavior="android.support.design.widget.AppBarLayout$Scroll
    tools:context="com.example.myfirstapp.DisplayMessageActivity"
    tools:showIn="@layout/activity_display_message"
    android:id="@+id/content">

</RelativeLayout>
```

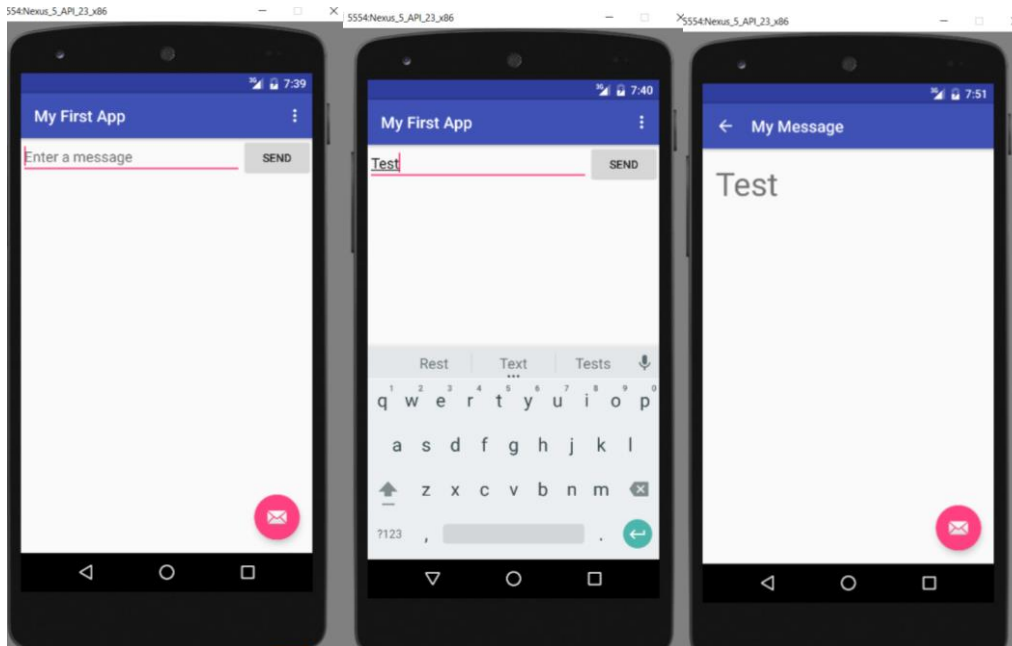
Im „DisplayMessageActivity.java“ wird nun ein „TextView“ Objekt in der „onCreate()“ Methode erstellt. Danach werden die Textgröße und der Inhalt der „TextView“ festgelegt. Die Textgröße wird auf 40 und der Inhalt auf den lokalen String „message“, welcher die mit dem „Intent“ übermittelten Nachricht beinhaltet, gesetzt. Nun wird der erstellte „TextView“ dem Layout hinzugefügt. Die fehlenden „includes“ werden wieder mit „Alt“ + „Eingabe“ hinzugefügt.

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

    FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
    fab.setOnClickListener((view) -> {
        Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
            .setAction("Action", null).show();
    });
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);

    Intent intent = getIntent();
    String message = intent.getStringExtra(MyActivity.EXTRA_MESSAGE);
    TextView textView = new TextView(this);
    textView.setTextSize(40);
    textView.setText(message);
    RelativeLayout layout = (RelativeLayout) findViewById(R.id.content);
    layout.addView(textView);
}
```

1.7 Ergebnis



1.8 Erkenntnis

In dieser Übung wurde der grundsätzliche Umgang mit Android Studio, sowie die Basis der Android-Programmierung gelernt. Außerdem wurden die Grundlagen von Java wiederholt. Weiters wurden einige der Vorteile der Android-Programmierung, zum Beispiel das Unterstützen verschiedener Sprachen anhand des „values/strings.xml“, besprochen.

2. Inventarliste

3. Messprotokoll