Übungen zum Fach ZAPP

1. Activities und Layouts

> Ziel der Übung ist es Ihnen ein Gefühl zu geben, aus welchen Komponenten Android Apps zusammengesetzt sind und Ihnen zu zeigen, wie das Zusammenspiel zwischen XML-Dateien und deren Elementen mit dem Java Code funktioniert.

1.1 Hello World

Legen Sie zuerst ein neues Projekt (`File -> New -> New Project` bzw. auf der Android Studio Startseite `Start a new Android Studio project`) mit beliebigem App Namen und einer Company Domain an.
Als Target Devices `Phone and Tablet` anwählen samt `API Level 21: Android 5.0 (Lolipop)` als `Minimum-SDK`. Auf der nachfolgenden Seite können bereits einige Activity Templates ausgewählt werden - aus Verständnisgründen starten wir aber ohne irgendwelchen Boilerplate Code, wählen Sie daher die Option `Add No Activity` aus.
Nach einer kurzen Gradle Build Phase sollten Sie in der Ansicht *Project (View -> Tool Windows -> Project)* ein Modul `app` sehen, welches sich in die drei "Pakete" `manifests`, `java` and `res` unterteilt.
Klappen Sie das `java` Paket auf und legen Sie mittels Rechtsklick auf den Paketnamen eine neue Activity an (`New -> Activity -> Empty Activity`). Im daraufhin aufgehenden Fenster können Sie noch den Namen der Java Klasse sowie des XML files, in welchem wir gleich das Layout definieren werden, festlegen. Wählen Sie zusätzlich bitte auch die Option `Launch Activity` an und klicken Sie auf .

Folgendes sollte Ihnen in der Projekt-Ansicht auffallen:

1. Im Java Verzeichnis hat Android Studio eine Klasse angelegt welche von `AppCompatActivity` ableitet und bereits eine Methode der Super-Klasse namens `#onCreate(Bundle)` überschreibt. Hier legen wir fest welches XML file inflated respektive angezeigt werden soll. Dies geschieht mittels der `setContentView(View)` Methode.

2. Im res->layout Verzeichnis ist eine XML-Datei hinzugekommen, in welcher bereits das Layout für unsere Activity spezifiziert ist - aktuell ist dieses noch leer.

3. Im Manifests Verzeichnis, näher in der `AndroidManifest.xml` Datei, hat Android Studio unsere neue Activity bereits deklariert und einen `<intent-filters>` hinzugefügt. Dieser signalisiert dem System, welche Activity zum Einstieg verwendet werden soll, sobald diese über den App Drawer gestartet wird.
   Wichtig: Jede Activity einer App muss im Manifest deklariert werden.

Nehmen Sie folgende Änderungen vor:

- Öffnen Sie die XML-Datei Ihrer Einstiegs-Activity und ändern Sie das root-layout von `ConstraintLayout` zu `LinearLayout`. Fügen Sie innerhalb des `<LinearLayout ...>` tags folgendes Attribut hinzu: `android:orientation="vertical"`
- Fügen Sie einen Button hinzu und geben Sie diesem einen Text (`android:text`) und eine Id (`android:id`).

Zum Hinzufügen verwenden Sie entweder den Layout Editor oder direkt die XML-Datei. Es empfiehlt sich die XML-Variante, da hierbei der Lerneffekt größer ist.

Das Resultat könnte in etwa so aussehen:

```
<LinearLayout ... >

    <Button
        android:id="@+id/hello_world_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Say 'Hello World!'" />

</LinearLayout>
```

Um den Button benutzen zu können, muss hierfür in der Activity eine Referenz auf diesen erzeugt werden. Dies geschieht mittels der `findViewById(...)` Methode. Im Anschluss daran wird dem Button ein `OnClickListener` gegeben um beim Klick darauf eine Aktion ausführen zu können.

```
Button button = (Button) findViewById(R.id.activity_main_button);
button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick (View view) {
                Log.d(TAG, "Hello World!");
        }
});
```

1.2 Intents

Fügen Sie ein `EditText` Feld zum Layout hinzu und legen Sie einen Hilfstext (`android:hint`) sowie eine Id (`android:id`) fest.

```
<EditText
        android:id="@+id/activity_main_edit_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter something" />
```

Legen Sie nun eine weitere Activity an und rufen Sie diese, nach dem Klick auf den Button in der ersten Activity auf. Zum starten einer weiteren Activity müssen wir zuerst einen `Intent` erstellen und im Anschluss `startActivity` aufrufen.

```
Intent intent = new Intent(MainActivity.this, SecondActivity.class);
startActivity(intent);
```

Zusätzlich möchten wir auch den eingegebenen Text aus dem `EditText` an unsere zweite Activity übertragen (`EditText#getText()`. Zur Übergabe von Werten an die Ziel-Activity können wir die Methode `Intent#putExtra(name, value)` verwenden. In der aufgerufenen Activity kommen wir an diese Werte wieder mittels `getIntent().getExtras().getCharSequence(name, defaultValue)`.
Hinweis: Übergeben werden können nur primitive Datentypen oder solche die das `Serializable` oder `Parcelable` interface implementieren.

Führen Sie die entsprechenden Schritte durch und zeigen Sie den eingegebenen Text in der zweiten Activity, zum Beispiel innerhalb einer `TextView` an.

Tipp: Referenz auf den EditText erzeugen, Text beim Button Click herausziehen und dem erzeugten Intent mitgeben.