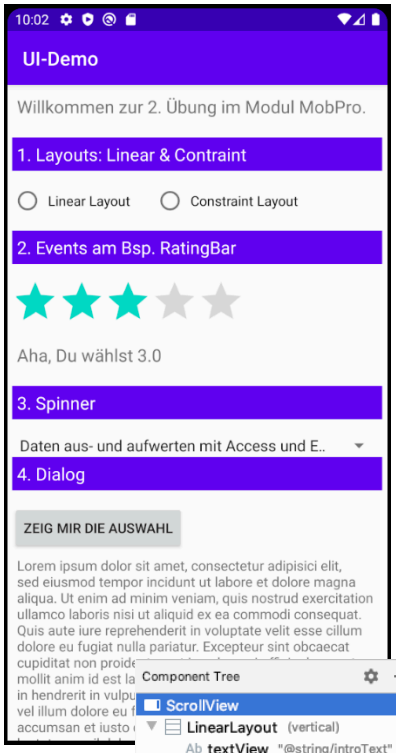
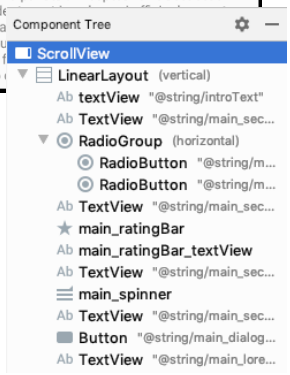


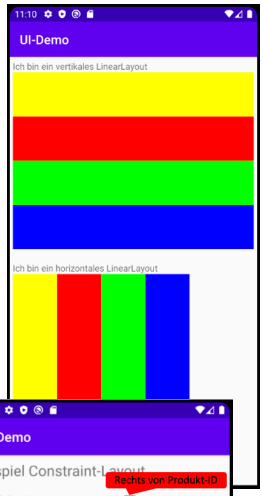
Übung 2: Benutzerschnittstellen & -interaktion

In dieser Übung verwenden wir verschiedene wichtige Klassen und Android-Mechanismen im Zusammenhang mit graphischen Benutzerschnittstellen und Benutzerinterkation. Konkret lernen Sie verschiede Views, Dialoge und Layout-Manager kennen, sowie die layout.xml-Dateien. Daneben kommen weiter Ressourcen wie Texte und Arrays vor. Sie lernen verschiede Möglichkeiten kennen, wie Benutzerereignisse (Events) mit Hilfe von Listeners behandelt werden können. Das Ziel dieser Übung ist es, die wichtigsten Android-Konzepte im Umfeld von Benutzerschnittstellen und –interaktion kennen zu lernen.



0. Neue App: UI-Demo

Erstellen Sie ein neues Android-Applikationsprojekt "UI-Demo" ("File" – "New" – "New Project" - "Phone and Tablet", "Empty Activity"). Diese Übung besteht aus verschiedenen Teilaufgaben, welche in der `MainActivity` von diesem Projekt soweit als möglich resp. sinnvoll dargestellt werden. In der Main-Activity werden, wie in der Vorlesung gezeigt, verschiedene Fragments angezeigt. Das initiale Fragment (Layout-Datei: `fragment_main.xml`) soll am Schluss ungefähr so aussehen wie der Screenshot rechts oben, verwenden Sie dazu eine `ScrollView` mit einem `LinearLayout`, siehe Screenshot vom Component-Tree rechts oben. Setzen Sie dieses initiale Layout in der `MainActivity` in der Methode `onCreate(…)` mit Hilfe des FragmentManagers.



.

1. LinearLayout & ConstraintLayout

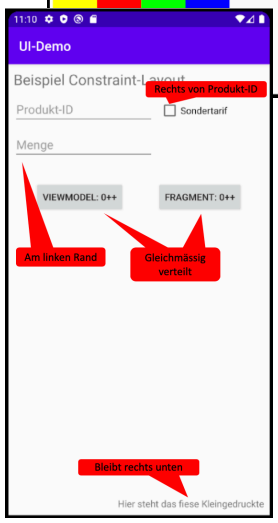
Erstellen Sie mit Hilfe von zwei `RadioButtons` in einer `RadioGroup` eine Auswahl (siehe Screenshot rechts oben), in der das Layout für zwei unterschiedliche Fragments ausgewählt werden kann. Verwenden Sie hier für die RadioButtons das xml-Attribut `android:onClick` zum Registrieren der gewünschten Methode der MainActivity. Implementieren Sie dazu auf der MainActivity die zwei folgenden Methoden:

```
fun layoutLinearSelected(view: View)
fun layoutConstraintSelected(view: View)
```

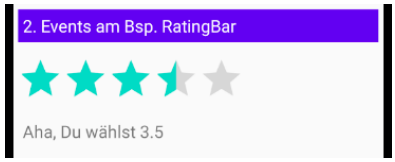
Das gewählte Layout-Fragment soll in der MainActivity wie in der Vorlesung behandelt über den FragmentManager ersetzt werden. Erstellen Sie dazu also zwei Layout-Dateien mit den folgenden Namen:

```
fragment_constraint_layout.xml
fragment_linear_layout.xml
```

In diesen zwei xml-Dateien gestalten Sie entsprechend zwei

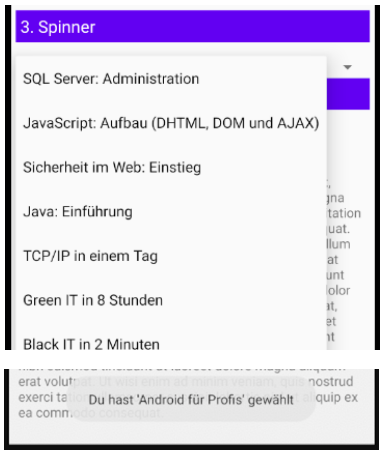


Ansichten mit Linear- und Constraint-Layout gemäss den beiden Screenshots rechts.



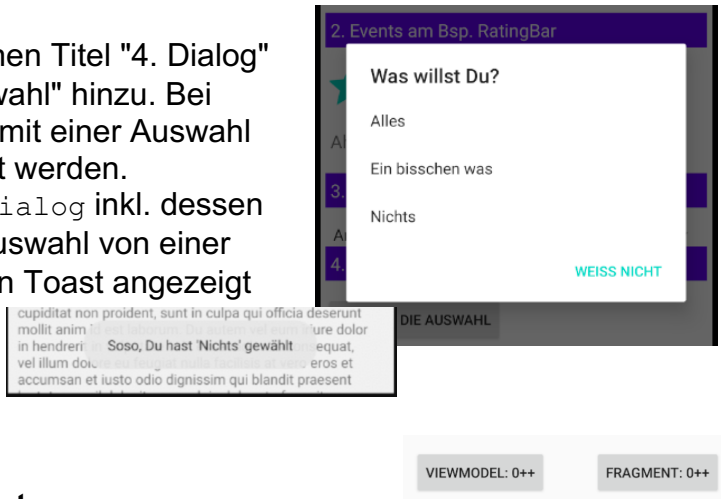
2. Views- und Events-Demo: RatingBar

Fügen Sie auf `fragment_main.xml` eine `RatingBar` inkl. Titel und Textfeld hinzu, siehe Screenshot rechts. Damit soll eine Bewertung von 0 bis 5 erfasst werden können, und die aktuelle Bewertung soll also unten in einem Text angegeben werden. (siehe Screenshot rechts). Bei Bewertung soll in einer `TextView` oben auf dem Bildschirm ein entsprechender Text die neue Bewertung angeben. Registrieren Sie dazu, wie in der Vorlesung gezeigt, einen entsprechenden `Listener` auf der `RatingBar` und reagieren Sie wie beschrieben auf Bewertungs-Events. Implementieren Sie diese Logik in einer Klasse `MainFragment` (als Fragment-Klasse zur Layout-Datei `fragment_main.xml`).



3. Auswahl aus xml-Ressourcen, inkl. Toast

Fügen Sie `fragment_main.xml` einen Titel "3. Spinner" und einen `Spinner` hinzu, siehe Screenshot rechts. Erstellen Sie dazu eine Datei `res/values/arrays.xml` in welcher Sie im XML-Format mögliche Kurse als String-Werte angeben. Bei Auswahl von einen Kurs soll ein einfacher Toast mit dem entsprechenden Kursnamen angezeigt werden, siehe Screenshot rechts unten. Um auf ein Auswahl-Ereignis reagieren zu können, registrieren und implementieren Sie dazu in der entsprechenden Fragment-Klasse einen `AdapterView.OnItemSelectedListener`.



4. Dialog inkl. Toast

Fügen Sie `fragment_main.xml` einen Titel "4. Dialog" und einen Knopf "Zeig mir die Auswahl" hinzu. Bei Drücken vom Knopf soll ein Dialog mit einer Auswahl wie im Screenshot rechts angezeigt werden. Verwenden Sie dazu einen `AlertDialog` inkl. dessen `Builder`-Mechanismus. Und bei Auswahl von einer Option soll diese in einem einfachen Toast angezeigt werden. Implementieren Sie dazu notwendige Logik in der Klasse `MainFragment`.

5. Zustandspeicherung auf Fragment

Legen Sie in der Klasse `ConstraintLayoutFragment` (also der `Fragement`-Klasse hinter `fragment_constraint_layout.xml`) diese beiden Felder an:

```
private var counter = 0
private val counterViewModel: CounterViewModel by activityViewModels()
```

Beim Drücken des Knopfs "Fragment: 0++" auf dem

ConstraintLayout soll dieses Feld `counter` um 1 hochgezählt werden und der neue um 1 erhöhte Wert direkt auf dem Knopf als z.B. "Fragment: 1++" angezeigt werden. Erstellen Sie weiter eine Klasse `CounterViewModel`, welche von `ViewModel` erbt und die folgenden zwei Methoden anbietet: `fun incCounter()` und `fun getCounter(): Int`. Verwenden Sie diese ViewModel-Instanz, um beim Drücken vom Knopf "ViewModel: 0++" den ViewModel-Zähler 1 hochzählt und dies ebenfalls direkt in diesem Knopf als z.B. "Fragment: 1++" anzeigt. Beobachten Sie den Effekt, wenn Sie vom Constraint-Layout-Bildschirm mittels Android-Back-Knopf auf das Main-Fragment zurück wechseln und dann wieder zurück zum Constraint-Layout. Warum behält ein Zähler auf dem Constraint-Fragment sein Zustand und der andere nicht?

6. Übersetzung Englisch

Erstellen Sie parallel zum vorhanden `values`-Ordner einen Ordner `values-en` für Englische Übersetzungen. Kopieren Sie dort die bestehende Datei `strings.xml` hinein und modifizieren Sie deren Texte, um eine englische Version von Ihrer App zu erhalten; siehe Screenshot rechts. Wechseln Sie in ihrem Gerät (oder dem Emulator) die Sprache auf zwischen "Englisch" und "Deutsch" um und testen Sie, ob die Sprachumstellung klappt. Damit haben Sie eine mehrsprachige Android-App erstellt!