

Aufgaben

Hinweis: Zu bearbeiten sind Aufgaben 1 & 2. Aufgabe 3 ist eine Bonusaufgabe

1. SPEISEKARTE - Data Binding

In dieser Aufgabe soll eine App programmiert werden, in der die Bestellungen der Kunden per Klick auf das Symbol aufgenommen werden können. Orientiere dich beim Lösen der Aufgabe an den Beispielen aus der Vorlesung.



- Öffne das Projekt "Speisekarte"
- Wir wollen die Elemente aus dem XML Layout mit dem Code verknüpfen und müssen daher `dataBinding` in der `build.gradle(Module)` Datei aktivieren
 - öffne Gradle Scripts -> `build.gradle(Module:SpeisekarteApp.app)`
 - Schreibe einen `buildFeatures{}` Block und setze innerhalb `dataBinding` auf `true` (siehe Folie 5 der Vorlesung)

Hinweis: synchronisiere das Projekt mit "Sync Now"

- Nun können wir das Layout `activity_main.xml` in ein Data Binding Layout umwandeln (siehe Folie 6 der Vorlesung)
 - Öffne die Context Actions per Rechtsklick auf das umschließende Layout
 - Wähle **Convert to data binding layout** aus ⇒ das Layout File ändert sich und bekommt z.B. einen `<data></data>` Block generiert.
- Erstelle nun in der MainActivity eine `binding` Variable vom Typ `ActivityMainBinding` und setze die `ContentView` (siehe Folie 7 der Vorlesung)
- Nun kannst du mit `binding` und der ID des xml Elements auf alle View Elemente zugreifen, z.B.: `binding.drink1Name`
 - Setze den Text von `drink1Name` bis `drink3Name` auf den Namen, der in der entsprechenden `drink` Variable gespeichert ist
 - Setze den Text von `drink1Price` bis `drink3Price` auf den Preis, der in der entsprechenden `drink` Variable gespeichert ist
 - Setze den Text von `drink1Count` bis `drink3Count` auf die Anzahl, der in der entsprechenden `drink` Variable gespeichert ist

Hinweis:

Bei den Klassenvariablen findest du die `drink` Variablen:

```
private val drink1 = Drink( name: "Kaffee", price: 3.95f)
private val drink2 = Drink( name: "Wein", price: 4.20f)
private val drink3 = Drink( name: "Cocktail", price: 6.90f)
```

- Programmiere in der MainActivity nun drei `onClickListener` für die drei Drink Buttons, in denen jeweils Folgendes passiert:
 - Die Funktion `addToBill` wird aufgerufen und der entsprechende Preis des Drink-Objektes als Parameter übergeben
 - Das `count` Attribut des Drink-Objektes wird um eins erhöht
 - Der Wert des `count` Attributs wird in den Text des entsprechenden `drinkCount` Objekts geschrieben

- Der Wert der `bill` Variable wird in das `text` Attribut der `totalPrice` TextView geschrieben
- Führe die App aus und probiere, ob sie funktioniert
- BONUS: Füge der App eine "ZURÜCKSETZEN" Funktion hinzu, der die Anzahl aller Getränke und die Summe auf 0 setzt und die entsprechenden Layout Elemente anpasst
 - Hinweis: es existiert ein `resetButton`, die über die binding Variable angesprochen werden kann.
- BONUS: Formatiere die Preise, sodass z.B. 4.20€ statt 4.2€ angezeigt wird. Dafür kannst du die Funktion `<String>.format(num)` verwenden

Beispiel: Zahlendarstellung mit 3 Dezimalstellen.

```
"%.3f".format(num)
```

```
3.987654 -> 3.987
```

Beispiel: Zahlendarstellung mit mindestens 3 Stellen

```
"%03d".format(num)
```

-> **3** gibt die mindest Anzahl der Stellen an


-> num Variable in dem Falle vom Typ Integer (**d**)

-> **num** Variable weniger Stellen werden vorne **0** eingefügt

z.B:

```
7 -> 007
```

```
42 -> 042
```

Viel Erfolg! 

2. SPEISEKARTE - Exception Handling

In dieser Aufgabe wollen wir mithilfe von Exceptions verhindern, dass mehr Getränke von einer Sorte bestellt werden, als vorhanden sind.

- Packe dafür deine Anweisungen in jedem onClickListener für `drink1`, `drink2` und `drink3` in einen `try` block
- Falls das Erhöhen der Anzahl die Grenze überschreitet, wirft die Klasse Drink automatisch eine Exception. Diese wird nun im `catch` Block gefangen und Folgendes ausgeführt
 - Lasse dir die Nachricht der Exception im `catch` Block anzeigen, indem du sie mit `Log` in der Logcat ausgibst
 - Deaktiviere den Button für den entsprechenden Drink, damit keine weiteren Exceptions geworfen werden
- Ändere den Resetbutton ab, sodass er nun auch alle Buttons wieder aktiviert
- BONUS: Erstelle eine neue Klassenvariable `customerMoney`, in der festgelegt wird, wie viel Geld der Kunde dabei hat. Falls der Gesamtbetrag darüber hinaus steigt, soll Exception mit einer passenden Nachricht geworfen werden. Diese wird bereits in den clickListenern der Buttons gefangen.

Viel Erfolg!

