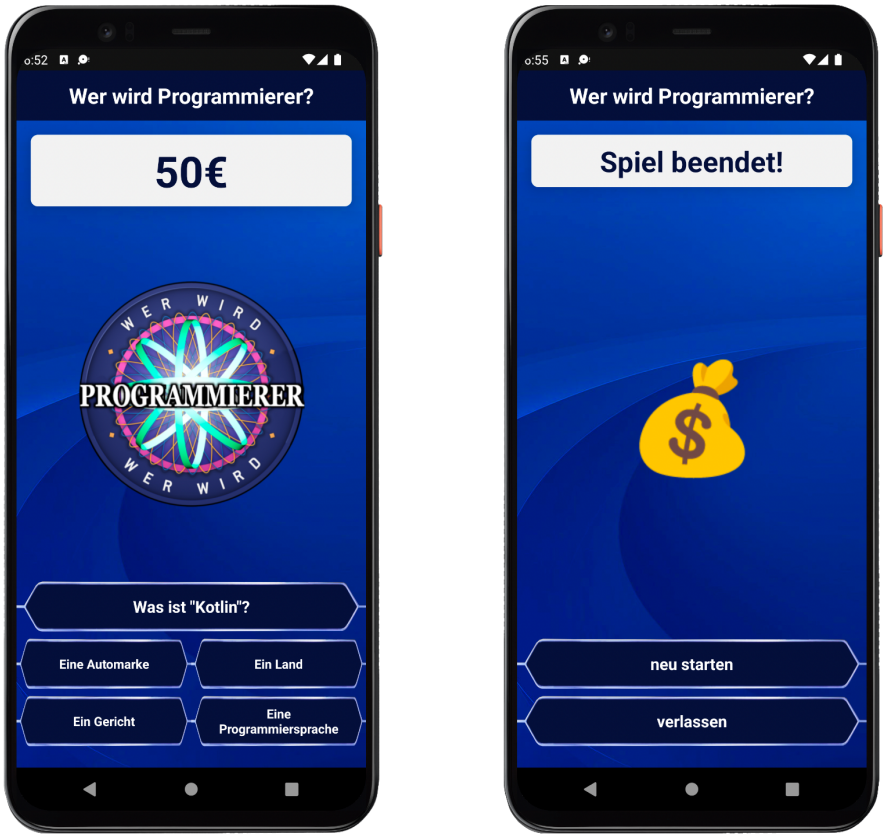
Kotlin 3.6 - Live Data

Aufgaben

Hinweis: Zu bearbeiten ist Aufgabe 1.

1. WER WIRD PROGRAMMIERER? - Live Data

In dieser Aufgabe vereinfachen wir die bekannte App “Wer wird Programmierer?” mithilfe von Live Data.



- Öffne das Projekt “Wer wird Programmierer”
- Schau dir die App an. Statt einem End-Dialog wird bei Spielende mithilfe eines Navgraph ein neues Fragment angezeigt.
- Wir starten in der Klasse `QuizRepository`. Die Funktion `loadQuestions()` ist `private` und nach außen hin nicht sichtbar. Die Liste soll hier in einer von außen beobachtbaren Variable gespeichert werden. Erstelle dafür eine LiveData Variable `questions` mithilfe der bekannten Verschachtelung. In der Variable und verpackt durch LiveData soll die Liste gespeichert werden, die durch die Funktion `loadQuestions()` zurückgegeben wird (siehe Folien Seite 7).

- Weiter geht es mit der Logik, wie gehabt in der Klasse `QuizViewModel`.
    - Erstelle eine Variable `repository`, in der ein Objekt der Klasse `QuizRepository` gespeichert ist.
    - Greife nun über `repository` auf die LiveData Variable `questions` zu und speichere die Liste in einer Variablen `questionsList`
    - Erstelle zudem eine Variable `questionIndex` und weise ihr zu Beginn den Wert `0` zu
    - Erstelle nun drei verschachtelte LiveData Variablen
        - Eine LiveData Variable `currentQuestion`, in der das aktuelle `Question` Objekt gespeichert wird (zu Beginn: das Erste in der Liste)
        - Eine LiveData Variable `currentPrice`, in der die aktuelle Preisstufe gespeichert wird (zu Beginn: der Preis des ersten `Question` Objekts in der Liste)
        - Eine LiveData Variable `gameOver`, in der gespeichert wird, ob das Spiel vorbei ist oder nicht (zu Beginn: `false`)

      Hinweis: Richte dich bei der Verschachtelung nach diesem Beispiel:

      ```
      private val _number = MutableLiveData<Int>(0)
      val number: LiveData<Int>
         get() = _number
      ```

    - Schreibe nun den Inhalt der Funktion `restartGame()`. Diese Funktion wird aufgerufen, wenn das Spiel neu gestartet werden soll. Sie macht nichts anderes, als den Wert aller oben definierten Variablen auf den Wert zurückzusetzen, den die Variablen zu Beginn hatten.

      Hinweis:
      spricht den Inhalt einer LiveData Variablen mit `.value` an, z.B.: `_number.value`
    - Programmiere schließlich noch die Funktion `checkAnswer` mit der eigentlichen Spiellogik. Die Funktion bekommt den Index der Antwort übergeben, die der User ausgewählt hat und Folgendes erledigen:
        - Falls der Index der Antwort dem Index der im `Question` Objekt gespeicherten richtigen Antwort entspricht und es sich nicht um die letzte Frage handelt, erhöhe den Frage-Index um eins und hole das neue `Question` Objekt aus der Liste und die neue Preisstufe aus dem `Question` Objekt.
        - Falls nicht, setze die `_gameOver` Variable auf `true` und führe die Funktion `resetGame()` aus.
- Die Klasse `QuizFragment` ist nun dank LiveData sehr kurz geworden, da wir den Inhalt z.B. der TextViews hier nicht mehr explizit setzen müssen

    - Setze vier `onClickListener` auf die Views mit den IDs `tvAnswerA` bis `tvAnswerD`. Innerhalb der `onClickListener` soll die Funktion `checkAnswer` aus dem ViewModel mit dem entsprechenden Index aufgerufen werden (`1` für `tvAnswerA`, `2` für `tvAnswerB`, usw.)
    - Beobachte mit `observe()` die Variable `gameOver` aus dem ViewModel. Falls sich die Variable zu `true` ändert, soll mithilfe des NavController zum `ResultFragment` navigiert werden (siehe Folien Seite 9).
- Schlussendlich können wir alle Informationen jetzt dank LiveData und Databinding direkt ins Layout laden. Öffne die Layout-Datei `fragment_quiz.xml`
    - Erstelle zwischen den `<data>` Tags eine neue `<variable>` namens “viewmodel” vom Typ “de.syntax_institut.funappsvorlage.ui.quiz.QuizViewModel”. Mit dieser Variablen kannst du auf die Elemente des ViewModel zugreifen.
    - Weise den `text` Attributen der TextViews `tvPrice`, `tvQuestion`, `tvAnswerA`, `tvAnswerB`, `tvAnswerC`, `tvAnswerD` die entsprechende Variable aus dem `Question` Objekt der aktuellen Frage im ViewModel zu. Konvertiere jede Variable mit `toString()`, da die `text` Attribute einen String brauchen.

      Hinweis:
      Nutze folgende Notation, um auf Variablen im ViewModel zugreifen zu können:
      `android:text=“@{viewmodel.dieRichtigeVariable}”`

      Hinweis:
      Du kannst zusätzlich zur Variablen noch weitere Zeichen hinzufügen (z.B. “€”):
      `“@{... + `€`}”`
- Führe die App aus und teste das Spiel. Schaffst du es bis zur Million?

Viel Erfolg!