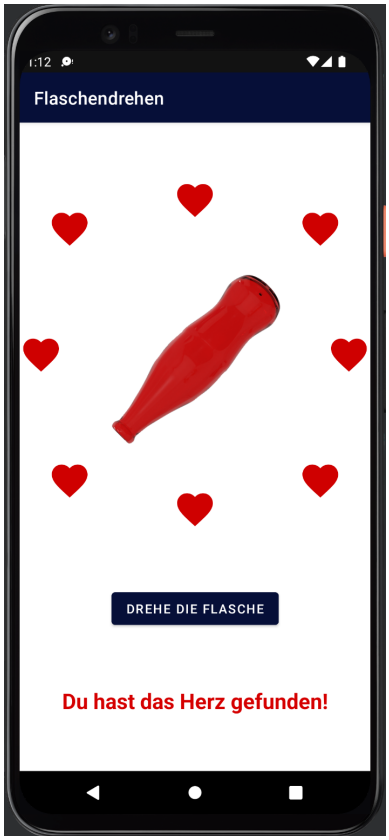
Kotlin 1.3 - Kotlin Basics II

Aufgaben

Hinweis: Zu bearbeiten ist Aufgabe 1 und Aufgabe 2. Aufgabe 3 ist eine Bonusaufgabe

1. FLASCHENDREHEN

In dieser Aufgabe wird eine Flaschendrehen-App programmiert.



- Öffne das Projekt “Flaschendrehen” und die Quelltextdatei “MainActivity”
- Hole als Erstes die drei Views, die wir im Code brauchen und speichere sie in Variablen ab
    - Die ImageView der Flasche mit der ID `flasche`
    - Das Textfeld mit der ID `herzGefunden`
    - Den Button mit der ID `button_drehen`
- Gib dem Button nun einen onClickListener, mit `.setOnClickListener{}`
    - Erstelle hier zuerst eine Variable, die eine zufällige Nummer zwischen 0 und 23 enthält

      Hinweis: hierfür kannst du eine Range anlegen und `.random()` darauf anwenden
    - Ziehe mithilfe der Zufallszahl einen zufälligen Winkel aus der Winkelliste

- setze nun die `rotation` der Flasche auf den Winkel

  XML-Attribut des ImageView's; rotation ist zu erreichen über
  [Name des ImageView][Punkt]

- prüfe im OnClickListener, ob Flanschenwinkel gleich `heartAngle` (oben im Code festgelegt)
  - Wenn das der Fall ist:
    - erstelle eine String Variable, gib ihr den Text "Du hast das Herz gefunden!"
    - Weise diese Variable nun dem TextFeld `herzGefunden` zu
    - Durchlaufe mit einer `for` Schleife nun noch alle Herzen in der Liste `heart` und setze für jedes Herz die `visibility` auf sichtbar

- Führe die App aus und probiere sie aus!

Viel Erfolg! ❤️

2. FINDE DEN FEHLER - Fehlersuche

Bei der Programmierung dieser APP ist etwas schiefgelaufen, begib dich auf die Fehlersuche und versuche das Problem zu lösen

- Öffne das Projekt “Finde den Fehler”
- Versuche die APP auszuführen und auf dem Emulator zu starten (Die APP wird nicht starten)
- Versuche den Fehler zu finden
- Wenn der Fehler behoben ist, führe die APP erneut auf dem Emulator aus und überprüfe, ob die APP funktioniert

Viel Erfolg! 

3. MEMORY - Schleifen (Advanced)

In dieser Aufgabe wird die Funktionalität einer Memory App programmiert.

- Öffne das Projekt “Memory” und die Quelltextdatei “MemoryLogik”
- Die erste Funktion `hideAllCards` ist dafür verantwortlich, alle Karten zu verdecken
    - gehe mithilfe einer `for` Schleife alle Karten (ImageButtons) in der übergebenen Liste durch und verdecke jede Karte, indem du das übergebene Lambda `hideCard()` auf sie anwendest
- Die Funktion `distributeCards` geht jede Zeile und Spalte des Spielfelds durch und platziert nacheinander eine Karte auf jedem Feld
    - die Anzahl an Spalten und Zeilen wird übergeben, nutze zwei geschachtelte `for` Schleifen, um für jede Spalte jeweils alle Zeilen durchzugehen
    - für jedes Feld wird das Lambda `placeCard(spalte, zeile)` ausgeführt
    - zudem soll das platzieren auf dem Feld um 0,05 Sekunden verzögert werden

      Hinweis: in diesem Kontext kann man einfach die Funktion “delay()” aufrufen. Diese akzeptiert als Parameter die Zeit in Millisekunden
- Wenn die App jetzt ausgeführt wird, sollten alle Karten auf dem Spielfeld erscheinen

- Die Funktion `cardClicked` soll bestimmten, was passiert, wenn man auf eine Memory Karte klickt
    - stelle sicher, dass es sich bei der angeklickten Karte nicht um die gleiche, zuvor angeklickte, Karte handelt. Vergleiche dafür den Index der aktuellen Karte mit dem letzten Index
    - Falls es sich um unterschiedliche Karten handelt, soll
        1. das Bild geändert werden, rufe dafür `setImage` mit dem aktuellen Index auf
        2. die Karte mit der letzten verglichen werden, rufe dafür `compareImages` mit dem letzten und dem aktuellen Index auf
- Implementiere die Funktion `assignImages`, sie füllt eine neue Liste zufällig mit BilderIDs aus der BilderIDquelle, sodass jedes Bild aus der BilderIDquelle genau zweimal in der neuen Liste vorkommt
    - die übergebene Liste enthält genau halb so viele Bild-Ids wie die Größe der zurückzugebenden Liste
    - führe für jede Karte eine `while` Schleife aus, die so oft durchläuft, bis die zufällig herausgezogene ID in die neue Liste hinzugefügt werden konnte

      Hinweis: ein zufälliger Index kann durch (a..b).random() generiert werden, wobei a für die niedrigste Zufallszahl und b für die höchste Zufallszahl steht
- Die Funktion `compareCardImages` vergleicht die Bilder zweier Karten
    - Falls beide übergebenen IDs gleich sind, soll das Lambda `makeBothCardsVisible` ausgeführt werden, um beide Karten unsichtbar zu machen
    - vorher soll der Ablauf jedoch um 0,5 Sekunden verzögert werden, damit das übereinstimmende Kartenpaar noch kurz offen daliegt, bevor es verschwindet
- Die Funktion `checkIfFoundAll` prüft alle übergebenen Karten auf ihre Sichtbarkeit und stellt fest, ob alle Karten aufgedeckt wurden
    - Nutze hier wieder eine `for` Schleife, um alle Karten in der übergebenen Liste durchzugehen und überprüfe jede Karte auf ihre `visibility`
    - falls es noch sichtbare Karten in der Liste gibt, soll die Funktion `false` zurückliefern, sonst `true`
- Führe die App aus und spiele eine Runde um die Funktionalität zu testen
    - Herausforderung: der Rekord liegt bei 25 Versuchen 😉

Viel Erfolg!