MOBILFEJLESZTÉS

5. forduló

autsoft

A kategória támogatója: AutSoft Zrt.

Ismertető a feladathoz

Felhasznált idő: 00:00/20:00 **Elért pontszám:** 0/13

1. feladat 0/1 pont

Az alábbiak közül melyik **NEM** animációs lehetőség Androidon?

Válasz

- (●) UIViewPropertyAnimator
  Ez a válasz helyes, de nem jelölted meg.
- ( ) ViewPropertyAnimator
- ( ) ValueAnimator
- ( ) ObjectAnimator

Magyarázat

A UIViewPropertyAnimator valóban egy animációs lehetőség azonban iOS platformra.

2. feladat 0/2 pont

Mi a fő különbség az **Iterable<T>** és a **Sequence<T>** között Kotlinban?

Válasz

- ( ) Iterable csak immutable collection-ön működik, a Sequence mutable-ökön is
- (●) Sequence lazy, Iterable eager kiértékelésű
  Ez a válasz helyes, de nem jelölted meg.
- ( ) Sequence szekvenciálisan, az Iterable párhuzamosan kerül kiértékelésre
- ( ) Kotlin szempontjából semmi különbség köztük

Magyarázat

A Collection-ök mellett a Kotlin standard lib egy másik konténertípust is tartalmaz – szekvenciákat (Sequence<T>). A Sequence-ek ugyanazokat a funkciókat kínálják, mint az Iterable, de máshogy valósítják meg a többlépcsős Collection feldolgozást. Ha egy Iterable feldolgozása több lépésből áll, melyeket eager módon hajt végre: minden feldolgozási lépés befejeződik, és visszaadja az eredményét – egy köztes Collection-t. A következő lépés ezen a Collection-ön hajtódik végre. A Sequence -ek többlépcsős feldolgozása viszont lehetőség szerint lazy módon történik: a tényleges számítás csak akkor történik meg, ha a teljes feldolgozási lánc eredményére szükség van.

3. feladat 0/4 pont

Mit ír ki?

Válasz

- (●) Le sem fordul
  Ez a válasz helyes, de nem jelölted meg.
- ( ) Semmit
- ( ) Number 1 Number 2 Number 3
- ( ) Number 1 Number 2

Magyarázat

A flowOf-ot egy CoroutineScope-ban kell meghívni például egy runBlocking { } segítségével.

4. feladat 0/2 pont

Ha [A] Activityből átváltunk [B] Activityre, milyen sorrendben hívódnak meg az életciklus függvények?

Válasz

- ( ) [A] Activity onPause(), onStop() - [B] Activity onCreate(), onStart() és onResume()
- (●) [A] Activity onPause() - [B] Activity onCreate(), onStart() és onResume() - [A] Activity onStop()
  Ez a válasz helyes, de nem jelölted meg.
- ( ) [B] Activity onCreate() - [A] Activity onPause - [B] Activity onStart(), onResume() - [A] Activity onStop()
- ( ) [B] Activity onCreate() - [A] Activity onPause - [B] Activity onStart() - [A] Activity onStop() - [B] Activity onResume()

Magyarázat

Először meghívodik az [A] Activity onPause() függvénye és csak utána tud létrejönni az új Activity. Miután elindításra került az új, a régi már nem látszik, ezért hívódik meg az onStop() függvénye.

5. feladat 0/4 pont

Mit ír ki a következő SwiftUI kódrészlet, amikor a Slidert hajszálpontosan a 6.0-ról a 7.0-ra húzzuk?

```
@State private var blurAmount = 0.0 {
     didSet {
          print("New value is \(blurAmount)")
    }
}

var body: some View {
     Slider(value: $blurAmount)
}
```

Válasz

- New value is 6.0
- New value is 7.0
- New value is 6.1

  New value is 6.2

  ...

  New value is 7.0

  (UI frissítéstől függően 6.0 és 7.0 közötti értékeket ír ki, az utolsó érték 7.0)
- Semmit sem ír ki

  Ez a válasz helyes, de nem jelölted meg.

Magyarázat

A State<Double> struct nem változik, csak a wrapped property-je a háttérben a Slider által, így a didSet nem hívódik.