

# MOBILFEJLESZTÉS

2. forduló



A kategória támogatója: AutSoft Zrt.

## Ismertető a feladathoz

### Útmutató:

- A **radio button-os kérdésekre** egy helyes válasz van.
- **Ha lejár a feladatlap ideje, a rendszer AUTOMATIKUSAN** beküldi azt az addig megjelölt válaszokkal.
- Az **adatbekérős feladatokra NEM jár részpontszám**, csak a feleletválasztósakra.
- **Badge-eket** a 4.forduló után kapsz majd először.
- Az **adatbekérős kérdéseknél** igyekeztünk minden variációt megadni (kisbetű, nagybetű, szóköz), de ha mégis eltérést tapasztalsz a megoldásokban, kérjük, jelezd felénk!

+1: Azért szólunk, hogy senkit ne a végén érjen meglepetés: a játék nem tipp-mix és csapatkategória sincs! Természetesen akinek nem inge...

Jó versenyzést kívánunk!

Felhasznált idő: 00:00/15:00

Elért pontszám: 0/12

## 1. feladat 0/1 pont

Hogy hívják az egyik leginkább elterjedt, Swift/Objective-C nyelvhez készített függőségkezelőt?

### Válasz

☐ PapayaPods

☒ CocoaPods  
Ez a válasz helyes, de nem jelölted meg.

☐ BananaPods

☐ Keychain

☐ Gradle

### Magyarázat

A PapayaPods és a BananaPods nem létező dolgok. Az iCloud Keychain segítségével folyamatosan frissíthető a jelszavak és az egyéb biztonsági információk az eszközön. A Gradle egy build automatizálási eszköz, amely a szoftverkészítés rugalmasságáról

ismert.

## 2. feladat 0/4 pont

Az alábbiak közül melyik SwiftUI kódrészlet eredményez egy zöld háttérű, lekerekített sarkú labelt?

### Válasz



```
Text("Hello world")  
    .padding()  
    .background(.green)  
    .cornerRadius(20.0)
```

Ez a válasz helyes, de nem jelölted meg.



```
Text("Hello world")  
    .background(.green)  
    .padding()  
    .cornerRadius(20.0)
```



```
Text("Hello world")  
    .padding()  
    .cornerRadius(20.0)  
    .background(.green)
```

### Magyarázat

Az első esetben az elvárt módon jelenik meg, a másodikban a színezés még a padding előtt történik, így a színezetlen, átlátszó rész lesz lekerekítve. A harmadik esetben pedig a színezés későn történik, így a színezett részből nem lesz levágva a sarok.

## 3. feladat 0/2 pont

Mi történik, ha egy SwiftUI Stack-en állítunk be .font modifiert?

### Válasz



A subview-k fontját állítja be és ezt felül lehet írni  
Ez a válasz helyes, de nem jelölted meg.



A subview-k fontját állítja be és ezt nem lehet felülírni



A subview-k magasságát tudjuk megadni vele



Semmi, a Stack-en nem tudunk megadni ilyet

### Magyarázat

Ha egy stacken állítunk be modifiert, az magába foglalt nézeteken is beállítja azt, de ezt felül lehet írni, ha az adott pl.: Text-re

#### 4. feladat 0/2 pont

Milyen típusú információkat tartalmaz az Android Manifest?

##### Válaszok

- ☐ Dependency-k leírása
- ☒ Az alkalmazás összetevői, például Activity-k, Fragment-ek és Service-ek  
Ez a válasz helyes, de nem jelölted meg.
- ☐ BuildConfig titkos kulcsok
- ☒ Az alkalmazásnak szükséges engedélyek  
Ez a válasz helyes, de nem jelölted meg.

##### Magyarázat

Az Android Manifest összetevői: Alkalmazás komponensek listája, szükséges minimális Android verzió, szükséges hardware konfiguráció, illetve az alkalmazásnak szükséges engedélyek. A Dependency-eket és a BuildConfig változókat a build.gradle fájlban írjuk le.

#### 5. feladat 0/2 pont

Hogyan nevezzük a Swift azon feature-ét, ami lehetővé teszi, hogy leírjuk a "some View" visszatérési értéket?

(A választ angolul, lowercase - csupa kisbetűvel várjuk. Ha több szóból álló kifejezésről lenne szó, akkor a szavak között pontosan 1 space legyen, a kifejezés előtt/után pedig semmilyen más karakter. NEM kell az angol többesszámot jelző "s" a szó/kifejezés végére.)

##### Válaszok

##### A helyes válasz:

opaque return type  
opaque type  
opaque return types

##### Magyarázat

Az *opaque return type*-ok lehetővé teszik a várható típus leírását anélkül, hogy azt konkretizálnánk. Ehhez a *some* kulcsszót használjuk.

#### 6. feladat 0/1 pont

Hello, ~~World~~!

## Válasz



```
val styledString: SpannableString = SpannableString( source: "Hello, World!")
styledString.setSpan(UnderlineSpan(), start: 0, end: 5, flags: 0)
styledString.setSpan(StrikethroughSpan(), start: 7, end: 12, flags: 0)
styledString.setSpan(UnderlineSpan(), start: 7, end: 12, flags: 0)
```



```
val styledString: SpannableString = SpannableString( source: "Hello, World!")
styledString.setSpan(StrikethroughSpan(), start: 0, end: 5, flags: 0)
styledString.setSpan(StrikethroughSpan(), start: 7, end: 13, flags: 0)
```



```
val styledString: SpannableString = SpannableString( source: "Hello, World!")
styledString.setSpan(UnderlineSpan(), start: 0, end: 5, flags: 0)
styledString.setSpan(StrikethroughSpan(), start: 7, end: 12, flags: 0)
```

Ez a válasz helyes, de nem jelölted meg.



```
val styledString: SpannableString = SpannableString( source: "Hello, World!")
styledString.setSpan(UnderlineSpan(), start: 0, end: 3, flags: 0)
styledString.setSpan(StrikethroughSpan(), start: 7, end: 12, flags: 0)
styledString.setSpan(UnderlineSpan(), start: 4, end: 5, flags: 0)
```

## Magyarázat

Android SpannableString egy String, amire karakterenként vagy karakter intervallumokra adhatunk meg formázást. UnderlineSpan() aláhúzást, míg a StrikethroughSpan() áthúzást biztosít, illetve a setSpan() függvényben megadható, hogy mettől meddig legyen érvényben az adott stílus.

