iOS – elméleti kérdéssor

1. iOS, macOS

1. **Nevezze meg az iOS architektúra részeit!**
   - Cocoa Touch
   - Media
   - Core Services
   - Core OS

2. **Ismertesse a Core OS réteg feladatait!**
   - OS X kernel
   - Energiagazdálkodás
   - Tanúsítványok
   - Fájlrendszer
   - Biztonság
   - Socket-ek

3. **Ismertesse a Core Services réteg feladatait!**
   - Alacsony szintű szolgáltatások helye
   - Helyzetmeghatározás alapjai
   - Telefonkönyv
   - Hálózati szolgáltatások
   - Beállítások
   - Szálkezelés
   - Fájlhozzáférés
   - URL eszközök

4. **Ismertesse a Media réteg feladatait!**
   - Alapvető média szolgáltatások helye
   - Audio lejátszás, felvétel
   - Videó lejátszás
   - Animáció
   - JPEG, PNG, TIFF

5. **Ismertesse a Cocoa Touch réteg feladatait!**
   - Magasszintű szolgáltatások
   - Multi-touch
   - Térkép
   - Kamera
   - Képkiválasztó
   - Vezérlők
   - Figyelmeztetések
   - Helyzetmeghatározás
   - View Hierarchia

6. **Milyen programtervezési mintán alapulnak az iOS alkalmazások?**
    - Swift és Objective-C

7. **Mi az iOS keretrendszerek elnevezése?**
    - Frameworks
    - Objektumok gyűjteménye
    - UIKit és Foundation

8. **Nevezze meg a leggyakrabban használt keretrendszereket!**
    - UIKit és Foundation

9. **Nevezze meg a macOS operációs rendszer felhasználói felület részeit!**
    - Menu bar
    - Desktop
    - Dock

2. Xcode fejlesztőkörnyezet

10. **Xcode fejlesztőkörnyezettel milyen alkalmazások fejlesztése lehetséges?**
    - iOS
    - macOS
    - watchOS
    - tvOS
    - 3D játékok fejlesztésére is alkalmas

11. **Xcode projekt létrehozásakor a Bundle Identifier miből jön létre?**
    - Alkalmazás nevéből és organization identifier-ből generált egyedi azonosító

12. **iOS alkalmazások fejlesztéséhez milyen programozási nyelveket támogat az Xcode?**
    - Swift és Objective-C nyelveket

1. **Nevezze meg az Xcode felület részeit!**
    - Navigator Pane
    - Editor
    - Utilities Pane
    - Debug Pane

2. **Ismertesse az Xcode Navigator Pane feladatát!**
    - Projekt tartalmát megjelenítő ablak

3. **Ismertesse az Xcode Navigator Pane részeit!**
    - Project navigator
        - Projektfájlok közötti navigáció
    - Source Control navigator
        - Verziókezelés Xcode-ból

- Symbol navigator
    - Osztályok vagy függvények közötti navigáció
- Find navigator
- Issue navigator
- Test navigator
- Debug navigator
- Breakpoint navigator
- Report navigator

4. **Ismertesse az Xcode Utilities Pane feladatát!**
    - Információt biztosít a kiválasztott elemről vagy annak beállításairól
    - Dokumentáció itt jelenik meg
    - Attól függően milyen elem lett kiválasztva további funkciók érhetőek el a panelon belül

5. **Ismertesse az Xcode Editor feladatát!**
    - Kódszerkesztő és felület designer szerepet tölti be, de a beállítások is itt jelennek meg
    - Egyszerre több állományon is dolgozhatunk
    - Megjegyzi az előzményeket

3. Swift 1

6. **Hogyan hozunk létre konstanst és változót?**
    - Konstans létrehozása: **let**
    - Változó létrehozása: **var**

1. **Ismertesse a String-ek és változók összefűzését!**
    - Backslash (\) használatával String-en belül változók értékeit be tudjuk helyettesíteni

2. **Melyik függvénnyel írhatunk ki az Output panelra?**
    - print
    - println

3. **Melyik típus vehet fel nil értéket?**
    - Optional típus vehet fel **nil** értéket

4. **Ismertesse az Optional típus kicsomagolásának lehetőségeit!**
    - ? – használhatjuk az Optional kicsomagolására
    - ! – ha biztosan tudjuk, hogy tartalmaz értéket

5. **Milyen típusú intervallumokat lehet létrehozni?**
    - Closed range
        - Zárt intervallum
    - Half-closed range
        - Félig zárt intervallum

6. **Függvényeket hogyan hozunk létre?**
    - **func** kulcsszót használatával
    - akár név nélkül is létrehozhatóak

7. **Függvények visszatérési értékét hogyan adjuk meg?**
    - visszatérési érték esetén **(->)** használunk
    - több visszatérési értéke is lehet
    - visszatérési érték és a függvény törzse **in** kulcsszóval választható el

8. **Mit jelent, hogy a függvények bemenő paramétereinek létezik külső és belső elnevezése? (????)**
    - függvények akár egymásba is ágyazhatóak, ahol a beágyazott függvény hozzáfér a külső változóihoz

9. **Mivel lehet függvények külső paraméternevét helyettesíteni? (????)**
    - sorszámmal is hivatkozhatunk rájuk
    - **(.)** ponttal

10. **Hogyan hozunk létre egy objektumot egy osztályból?**
    - objektum létrehozása az osztály nevével és **()** zárójelekkel hozható létre

11. **Osztály „konstruktorát" melyik függvénnyel definiáljuk?**
    - **init** függvénnyel

4. View

1. **Ismertesse a View-t!**
    - Egy olyan elem a felhasználói felületen, amely képes önmagát megjeleníteni egy téglalap alapú területre
    - View-nak tekintjük azokat az objektumokat, amelyek az **UIView** osztály leszármazottjai
    - Képes a felhasználóval interakciókba lépni, reagál az érintésre és gesztusokra

2. **View-k között kialakítható hierarchia (szülő-gyerek kapcsolat) a felületen?**
    - Igen kialakítható, létrehozhatunk szülő-gyermek kapcsolatokat
    - ha egy szülő View-t törlünk a felületről a gyerekei is eltűnnek

3. **Melyik vezérlő valósítja meg az egy oszlopos listát?**
    - Egy oszlopos lista: Table View

4. **Alkalmazással érkező képi erőforrásoknak hány különböző felbontásban kell rendelkezésre állnia?**
    - 3 különböző felbontásban kell
    - @1x (10 x 10 px)
    - Nagyobb felbontású képernyőn
    - @2x (20 x 20 px) kétszeres
    - @3x (30 x 30 px) háromszoros

5. **Milyen mértékegységet használunk a vezérlők méretezésénél?**
    - Az iOS **point** mértékegységet használja

6. **Ismertesse az Outlet-et!**
    - A felületre (Storyboard) elhelyezett vezérlőkre nem kapunk referenciát
    - Outlet segítségével tudunk a Storyboard-on elhelyezett elemre hivatkozni a swift kódban
    - Maga a kapcsolatért felelős

7. **Ismertesse az Outlet Collection-t!**
    - Vezérlők referenciáit tömbként is kezelhetjük, erre szolgál
    - Felületen tudjuk kezelni, hogy melyik vezérlők tartozzanak bele
    - Létrehozhatunk több vezérlőre közös eseménykezelőt is

8. **Ismertesse az Action-t!**
    - Vezérlőkhöz tudunk eseményeket rendelni
    - Nyomógombhoz hozzáérnek és ennek hatására egy függvényt szeretnénk futtatni

9. **iOS-ben milyen elrendezési módszert használunk alapértelmezetten?**
    - Auto Layout

10. **Hogyan hozhatunk létre szabályt Auto Layout esetében?**
    - szabályok = auto layout constraints
    - vezérlők kiválasztása után új szabályt hozhatunk létre
    - a vezérlőt kiválasztva a Control-t lenyomva tartva odahúzzuk egy másik elemhez a felületen
    - alapból kézzel állítjuk be őket, de Xcode-ban van lehetőség automatikusan is beállítani

11. **A létrehozott szabályokat (constraint) hol ellenőrizhetjük Xcode-ban?**
    - Document outline-ban jobb felül
    - Utilities panelon további hasznos infók vannak

5. Table View

1. **Hogyan nevezzük a Table View egy sorát?**
    - Egy sor: **cell**

2. **Lehetséges a sorok csoportosítása Table View-ban?**
    - **Section**: cellák csoportosítása

3. **Nevezze meg a Table View két alapvető Outlet-jét!**
    - Swipe gesture
    - drag&drop

4. **Table View adatforrás beállításához milyen protokollt kell implementálni?**
    - dataSource

5. **UITableViewDataSource protokoll mely két függvényét kötelező implementálni?**
    - numberOfRowsInSection
    - cellForRowAt

6. **Table View interakciók eléréséhez milyen protokollt kell implementálni?**
    - Delegate

7. **UITableViewDelegate melyik függvénye reagál a sor kiválasztására?**
    - didSelectRowAt

8. **Többoldalas alkalmazásoknál mely két alapvető navigációs módszer létezik?**
    - Navigation bar:
        - A képernyő felső részén megjelenik egy mező, amivel visszanavigálhatunk az előző oldalra, kiírhatjuk melyik oldalon vagyunk
    - Tab bar
        - Felület alsó részén van egy navigációs sáv, amivel az egyes scene-ket kiválaszthatja a felhasználó
    - Kettő kombinációja is lehetséges

9. **Ismertesse a Segue-t!**
    - navigáció megvalósító objektum
    - meg tudjuk határozni, hogy a másik scene hogyan jelenjen meg

6. MVC, Stack View

1. **MVC: Ismertesse a Modell feladatát!**
    - reprezentálja az adatokat, amivel a felhasználó dolgozik
    - lehetnek View Model-ek, melyek az adatok átadását segítik a View és a Controller között
    - továbbá Domain Model-ek, amik az üzleti logikában valamilyen módosítást hajtanak végre az adatokon

2. **MVC: Ismertesse a View feladatát!**
    - UI egy része

3. **MVC: Ismertesse a Controller feladatát!**
    - feldolgozza a bejövő kéréseket, műveleteket hajt végre a Model-en
    - a megfelelő nézetet (View-t) megmutatja a felhasználónak

4. **Mire használjuk a Stack View?**
    - vezérlőelemek vízszintes és függőleges elrendezését valósítja meg

10. **Ismertesse a Stack View tulajdonságait!**
    - Axis
    - Alignment
    - Distribution
    - Spacing

7. Swift 2

11. **Swift-ben a tulajdonságok milyen tervezési mintát valósítják meg?**
    - Megfigyelő tervezési mintát valósítanak meg

12. **Swift-ben a tulajdonságokat milyen három alapvető esetben használhatjuk?**
    - Tárolt értékek osztályon, struktúrán vagy enum-on belül
    - Tárolt tulajdonságok
    - Lazy tulajdonságok
    - Számolt tulajdonságok

1. **Swift tulajdonság megfigyelő létrehozásának lehetőségeit ismertesse!**
    - Property Observe
        - Megfigyeli és reagál a tulajdonság értékének változására
        - Mindig meghívásra kerül, amikor módosul a tulajdonság értéke
    - Örökölt tulajdonságokat is megfigyelhetünk
    - Két lehetőségünk van létrehozni
        - willSet
        - didSet

2. **Swift tulajdonság willSet blokkja mire használható?**
    - érték beállítása előtt kerül meghívásra
    - új értéket konstansként kapjuk meg
    - willSet implementációban a konstans nevét beállíthatjuk, ha nem állítjuk be a newValue-val érjük el

3. **Swift tulajdonság didSet blokkja mire használható?**
    - érték beállítása után azonnal lefut
    - régi érték itt is egy konstansként érhető el
    - konstans nevét beállíthatjuk vagy az oldValue-val érhetjük el

4. **Mire használjuk a Swift mutating kulcsszavát?**
    - **mutating** kulcsszóval az érték típusból egy új példányt hozunk létre
    - self tulajdonságokhoz is rendelhetjük

5. **Mit valósít meg a Swift subscript nyelvi kifejezése?**
    - osztályok, struktúrák és enum-ok definiálhatnak subscript-eket annak érdekében, hogy a bennük található kollekciók, listák könnyebben elérhetőek legyenek
    - Index alapján olvashatjuk és írhatjuk a belső kollekciók, listák értékeit
    - egy típushoz több subscript-et is létrehozhatunk (**subscript overloading**)

6. **Swift típusellenőrzését mivel valósíthatjuk meg?**
    - Type Casting
        - **is**, **as** operátorok léteznek ellenőrzésre és átalakításra
            - **is**: igaz-hamis értékkel tér vissza

7. **Swift típusok közötti átalakítását mivel valósíthatjuk meg? (????)**
    - öröklődéssel
    - ősosztály függvényeit és tulajdonságait elérjük, és felülírhatjuk őket

8. **Swift-ben mit tárolhat az AnyObject típus?**
    - minden osztályból készült példányt tárolhat

9. **Swift-ben mit tárolhat az Any típus?**
    - minden létező típust képes tárolni, még függvényeket is

8. Multitouch, View Controller lifecycle, Swift 3

10. **Milyen típushoz tudunk gesture-t rendelni?**
    - UIView-hoz tudunk gesture-t rendelni

11. **Sorolja fel a tanult gesture eseményeket! (????)**
    - UIGestureRecognizer
    - UIPinchGestureRecognizer
    - UIRotationGestureRecognizer
    - UISwipeGestureRecognizer
    - UITapGestureRecognizer
    - UILongPressRecognizer

12. **UIGestureRecognizer state tulajdonsága milyen értékeket vehet fel?**
    - .possible
    - .began, .changed, .ended
    - .recognized
    - .failed
    - .cancelled

13. **Milyen View Controller lifecycle események léteznek?**
    - viewDidLoad
    - viewWillAppear
    - viewDidAppear
    - viewDidDisappear

14. **Mire szolgál a guard Swift nyelvi kifejezés?**
    - Optional típusok kibontására és további feltételek (where closure) definiálására
    - **if let** kifejezések egymásba ágyazását válthatjuk ki vele
    - lehetővé tesz **where** feltételek definiálását is, így egy szabályrendszert is megalkothatunk értékellenőrzésre

15. **Swift-ben hibatípusok létrehozásához milyen típust használunk leggyakrabban?**
    - Error típust

16. **Mi a különbség a throw és throws Swift nyelvi elemek között?**
    - throw
        - a hibát tovább dobjuk
    - throws
        - tovább dobja a hibát, amit majd a hívó fél fog lekezelni

1. **Mire használjuk a do-catch blokkot?**
    - hibákat kezeljük le, hibatípusonként különbözőképpen reagálhatunk
    - ha hiba keletkezik a try-al megjelölt hívásnál azonnal a megfelelő catch ágra kerülünk
    - catch-nek nem kell az összes létező hibát kezelnie, ellenőrizhetjük az általunk definiált hibatípusra is, így csak arra fog reagálni

17. **Mire használjuk a try –t?**
    - ha egy függvény hibát dobhat fel kell készülni azok lekezelésére
    - függvény hívása előtt kell használni

18. **Mire használjuk a try? –t?**
    - a hibát úgy kezeli, hogy egy Optional típussá konvertálja
    - ha hiba keletkezik egy **nil** értékkel tér vissza

1. **Mire használjuk a try! –t?**
    - ha tudjuk egy throws kulcsszóval ellátott függvényről, hogy sosem keletkezik benne hiba és nincs szükség annak lekezelésére a try! kulcsszó fog segíteni
    - amennyiben mégis hiba keletkezik a meghívott függvényben futási hiba keletkezik

9. REST

2. **Az iOS a többszálú végrehajtást milyen adatszerkezettel valósítja meg?**
    - FIFO queue segítségével

3. **Milyen Queue típusok léteznek szálkezelésre?**
    - Single
        - egyszerre csak egy feladat végrehajtása történik
    - Concurrent
        - több feladat végrehajtása párhuzamosan

4. **Mi jellemzi a Main Queue-t?**
    - Single queue-k közé tartozik
    - minden felhasználói felülethez tartozó feladatot csak ezen a speciális queue-n lehet végrehajtani
    - UI független feladatokat sosem itt hajtunk végre mivel a felhasználónak biztosítani kell, hogy a felület folyamatosan aktív legyen

5. **Milyen Global Queue típusokat ismer?**
    - shared
    - global

6. **Hogyan érjük el a Main Queue-t?**
    - ha nincsen más által megnyitva vagy használatban

7. **Mire szolgál a Data típus?**
    - byte tömb pufferben tárolhatunk vele a memóriában adatokat
    - Indexelhető
    - képes URL alapján tartalom betöltésére

8. **Mire szolgál az URLSession típus?**
    - HTTP és HTTPS alapú kérések létrehozásért és végrehajtásásért felelős osztály
    - tartalom küldésére és fogadására egyaránt alkalmas

9. **Ismertesse az URLSessionDataTask feladatát!**
    - HTTP GET kérés
    - szervertől adatot kérünk el és a memóriába tároljuk a kapott választ

10. **Ismertesse az URLSessionUploadTask feladatát!**
    - http POST vagy PUT
    - fájlt töltünk fel a szerverre a háttértárról
    - létrehoz egy HTTP kérést, amely a megadott URL-re feltölti az adott tartalmat

11. **Ismertesse az URLSessionDownloadTask feladatát!**
    - fájl letöltése a szerverről és tárolása az eszköz ideiglenes tárhelyén
    - lehetőségünk van szüneteltetni és folytatni a végrehajtást, továbbá felfüggeszteni is
    - létrehoz egy letöltési feladatot és menti a letöltött fájlt

12. **JSON formátum feldolgozásához milyen típus létezik a Foundation keretrendszerben?**
    - Dictionary (Object)
    - Array
    - String
    - Bool
    - Number

10. Perzisztens adattárolás

1. **iOS-ben melyik a legegyszerűbb perzisztens adattárolási módszer?**
    - UserDefaults

2. **Mit tárolhatunk a UserDefaults-ban?**
    - Kulcs-érték párokat
    - Alkalmazás beállításait
    - Csak Property List típusú adatot fogad

3. **Property List milyen típusok kombinációját jelenti?**
    - Array
    - Dictionary
    - String
    - Date
    - Data
    - és számtípusok valamilyen kombinációja

4. **Mely két módszert használhatjuk objektum szerializációra?**
    - NSCoding/NSCoder
    - Codable

5. **NSCoder vagy a Codable szerializációt egyszerűbb megvalósítani?**
    - Codable szerializációt

6. **Codable kimeneti JSON konfigurációja hogyan érhető el?**
    - ha egy típusunk már Codable át tudjuk alakítani pl. JSON vagy Property List típusokká
    - decode függvénnyel

7. **Sorolja fel a Sandbox könyvtárakat!**
    - Application
        - minden, ami az app futásához szükséges
        - csak olvasható könyvtár
    - Documents
        - permanens adatok tárolása
        - látható a felhasználó számára
    - Application Support
        - permanens tárhely
        - nem látható a felhasználó számára
    - Cache
        - ideiglenes fájlok helye

8. **Fájl elérés útját milyen típussal érjük el?**
    - FileManager segítségével kereshetjük ki az URL-t

9. **Sorolja fel a FileManager műveleteket!**
    - másolás
    - mozgatás
    - törlés
    - fájl létezik-e az adott URL-en