UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Okosóra, Okostelefon és OkosTV - Apple Swift alapú alkalmazás fejlesztés

Dr. Bilicki Vilmos

Szoftverfejlesztés Tanszék

Swift programozás alapjai I.

- Objektumorientált programozás
- Swift: változók, konstansok, adattípusok kezelése, optional
- Memóriakezelés
- ARC- strong/weak/unowned
- Strong Reference Cycle

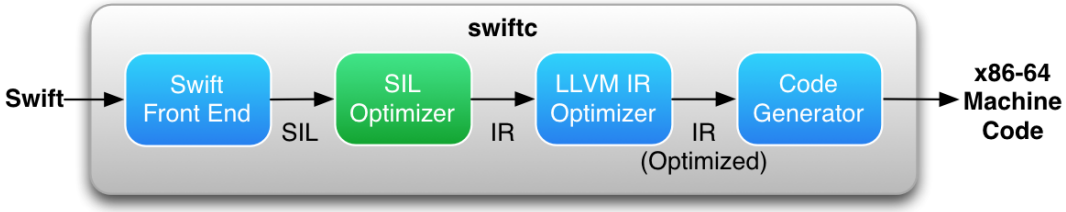
SWIFT

- Nyílt forrású nyelv (swift.org)
  - 3.0
- 2010-ben kezdték el fejleszteni, sok nyelvből merítettek ötleteket (pl.: Objective-C, Rust, Haskell, Ruby, Python, C#, CLU)
- 2015 a legszeretetebb nyelv (Stack Overflow Developer Survey 2015)

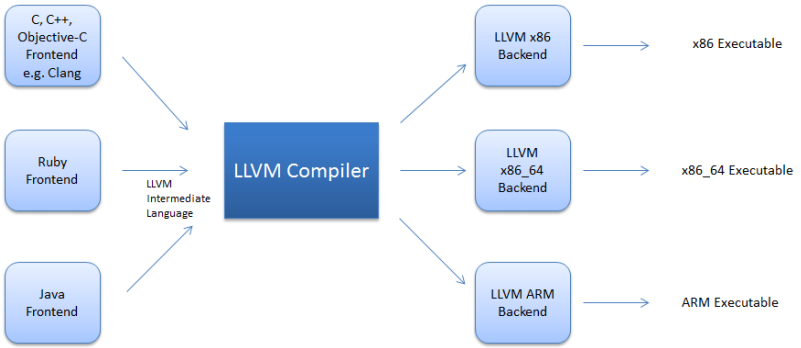
SWIFT elemek

- SWIFT fordító (OSX, IOS, Linux)
- Szabványos progarmkódkönyvtár
  - NSString, NSArray, …
- Alap könyvtárak (hálózat, JSON, …)
- LLDB hibakereső (REPL)
- Csomag kezelő
- Gyakorlás:
  - IBM Sandbox
  - Xcode Playground

SWIFT runtime



- Memória menedzsment
  - Manual Retain-Release (- 2011)
  - Automatic Reference Counting (2011 -)
    - LLVM



Objektumorientált programozás

- Minden objektum (metódus, closure,…)
- Erősen típusos
  - Típus ellenőrzés
  - Inferencia
  - Protokollok
  - Típus helyettesítő név (Alias)
- Dinamikus
  - Rendezett n-es véges lista (Tuple)
  - Bővítés (Extension)

Swift: változók, konstansok, adattípusok kezelése, optional

- Konstansok deklarálása
- Változók deklarálása
- Típusok megadása, inferencia
- Elnevezések, helyettesítő nevek
- Pontosvessző

```
let meaningOfLife = 42
```

```
let cat = "🐱"; print(cat)
```

```
typealias AudioSample = UInt16
var maxAmplitudeFound = AudioSample.min
var red, green, blue: Double
```

```
let π = 3.14159
let 你好 = "你好世界"
let 🐶🐮 = "dogcow"
```

Optional

- Tony Hoare (1965 - Algol)
  - Null References: The Billion Dollar Mistake
- NIL – Objective C, SWIFT
- NULL Pointer láncolás (Optional chaining)
  - NIL értéket felvehető metódusok, adattagok, indexek meghívása

```
if(x.getLocation() != null)
    if(x.getLocation().getBuilding() != null)
        if(x.getLocation().getBuilding().getSolidFuelInd() != null)
            pol.setWood_heat_ind(x.getLocation().getBuilding().getSolidFuelInd()
```

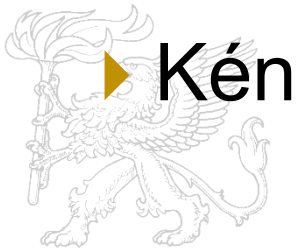
Optional

- Egy enum
  - Vagy NIL
  - Vagy van értéke
- Jelölése <TÍPUS>?

```
var serverResponseCode: Int? = 404
```

- Érték vizsgálat

```
if convertedNumber != nil {
    print("convertedNumber contains some
          integer value.")
}
```



- Kényszerített kibontás (forced unwrapping)

```
if convertedNumber != nil {
    print("convertedNumber has an
          integer value of \
          (convertedNumber!).")
}
```

Opcionális kötés

- Amennyiben a változó tartalmaz értéket akkor adjuk azt át

```
if let constantName = someOptional {
    statements
}
```

```
if let actualNumber =
        Int(possibleNumber) {
    print("\"\(possibleNumber)\" has an
        integer value of \
        (actualNumber)")
} else {
    print("\"\(possibleNumber)\" could
        not be converted to an
        integer")
}
```

Implicit kibontás

```
let possibleString: String? = "An
        optional string."
let forcedString: String =
        possibleString! // requires
        an exclamation mark


let assumedString: String! = "An
        implicitly unwrapped optional
        string."
let implicitString: String =
        assumedString // no need for
        an exclamation mark
```

Optional Chaining

- A kényszerített kibontás alternatívája
- Tetszőleges mélységben használható
- Alapérték megadható

```
let roomCount =
        john.residence!.numberOfRooms
```

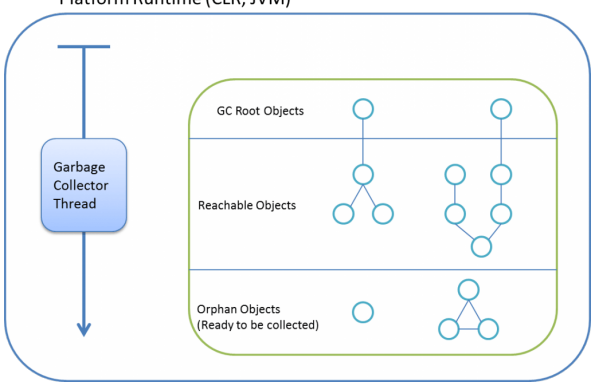
```
var unwrappedValue = optionalValue ?? defaultValue
```



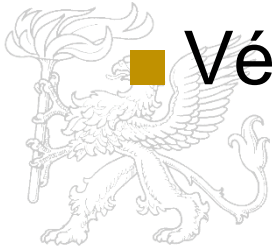
```
if let roomCount =
        john.residence?.numberOfRooms
        {
    print("John's residence has \
        (roomCount) room(s).")
} else {
    print("Unable to retrieve the number
        of rooms.")
}
```

```
if john.residence?.printNumberOfRooms()
        != nil {
    print("It was possible to print the
        number of rooms.")
} else {
    print("It was not possible to print
        the number of rooms.")
}
```

Memóriakezelés

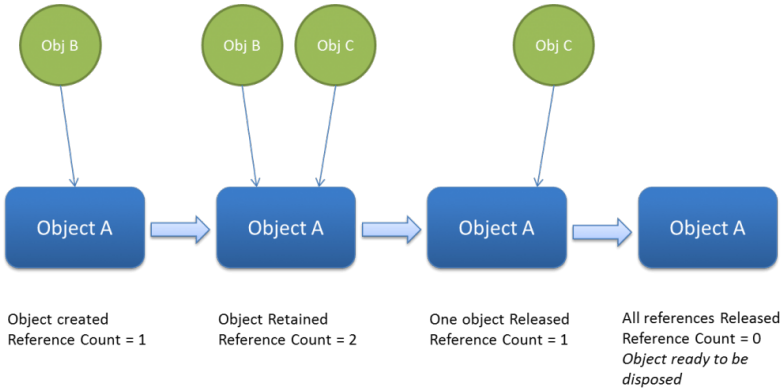


- Klasszikus megoldás (JAVA)
  - Szemétgyűjtő
  - Az infrastruktúra része (JVM)
  - Nem detreminisztikus
  - Befolyásolya az alkalmazás teljesítményét
  - Egyész objektum gráfokat tud kiüríteni
  - Védettebb a memória szivárgással szemben



ARC

- Nem a futtató környezet része
- A fordító a megfelelő helyekre beszúrja a referncia követést és a memória felszabadító kód részleteket (amit egyébként a fejlesztő dolga lenne)
- Determinisztikus: amikor az objektum kikerül a szkópból törötlhető
- Mivel nincs háttér tevékenység ezért kevesebb CPU-t memóriát igényel
- Nem tudja kezelni a körkörös hivatkozásokat
- Védtelenebb a memória szivárgásokkal szemben

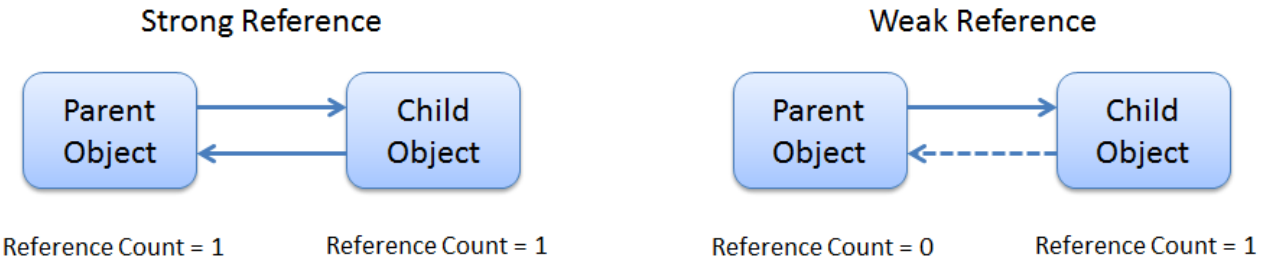


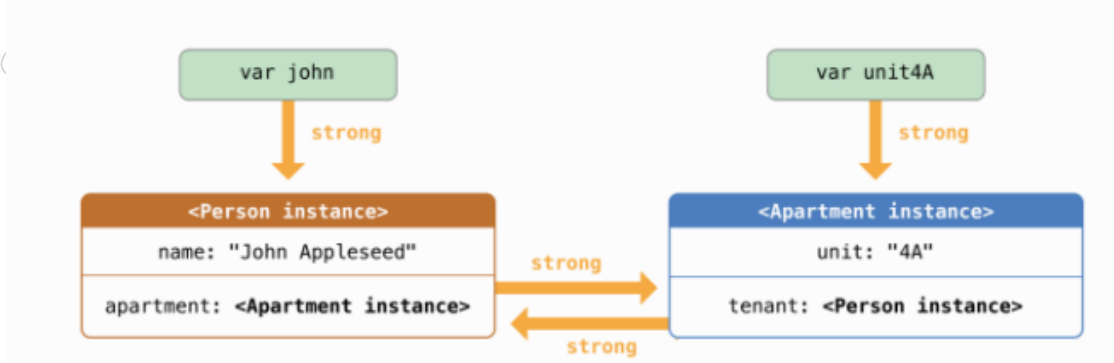
Visszatartó hurok(Retain Circle)

- Két vagy több objektum körkörösen hivatkozik egymásra
  - Pl.: Gyermek – Szülő
- Az ARC nem tudja kitalálni mikor engedhetőek el az objektumok mivel nem foglalkozik a veremmel (szigeteket így kezelhetjük)
- Csak a hivatkozás számot figyeli

ARC- strong/weak/unowned

- Megoldás
  - A fejlesztő segítsen az ilyen hurokt feltörni
  - Adott hivatkozást törölhetőnek jelöl






```
john = nil

unit4A = nil
```

Gyenge, Gazdátlan referencia

- Nem erős referencia:
  - Ha csak ez tartja vissza akkor törölhető
- Típusai
  - Gyenge(weak) - Optional
    - Akkor használjuk, ha előforulhat, hogy nil értékű legyen
    - Ha csak ilyen referencia mutat akkor nil lesz
    - var-t kell használnunk
  - Gazdátlan(unowned), nem Optional
    - Akkor használjuk ha sohasem lehet nil értékű

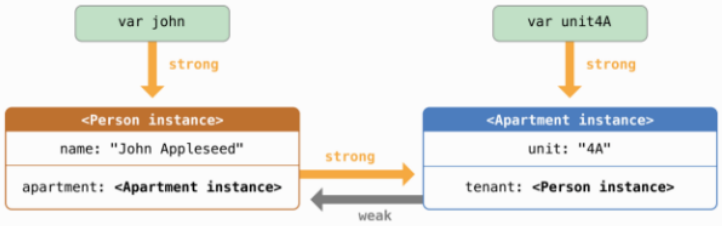
```
class Person {
    let name: String
    init(name: String) { self.name = name }
    var apartment: Apartment?
    deinit { print("\(name) is being deinitialized") }
}

class Apartment {
    let unit: String
    init(unit: String) { self.unit = unit }
    weak var tenant: Person?
    deinit { print("Apartment \(unit) is being deinitialized") }
}
```

```
var john: Person?
var unit4A: Apartment?

john = Person(name: "John Appleseed")
unit4A = Apartment(unit: "4A")

john!.apartment = unit4A
unit4A!.tenant = john
```



Gazdátlan referencia

```
class Customer {
    let name: String
    var card: CreditCard?
    init(name: String) {
        self.name = name
    }
    deinit { print("\(name) is being deinitialized") }
}

class CreditCard {
    let number: UInt64
    unowned let customer: Customer
    init(number: UInt64, customer: Customer) {
        self.number = number
        self.customer = customer
    }
    deinit { print("Card #\(number) is being deinitialized") }
}
```

Gazdátlan referencia

- Mindkét oldalnak értékkel kell bírnia

```
class Country {
    let name: String
    var capitalCity: City!
    init(name: String, capitalName: String) {
        self.name = name
        self.capitalCity = City(name: capitalName, country: self)
    }
}

class City {
    let name: String
    unowned let country: Country
    init(name: String, country: Country) {
        self.name = name
        self.country = country
    }
}
```

Összefoglaló

- Objektumorientált programozás
- Swift: változók, konstansok, adattípusok kezelése, optional
- Memóriakezelés
- ARC- strong/weak/unowned
- Strong Reference Cycle