UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Mobil alkalmazásfejlesztés - IOS

Dr. Bilicki Vilmos

Szoftverfejlesztés Tanszék

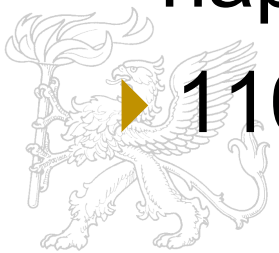
A fóliákhoz felhasznált anyagok:
Google: Android Developer Fundamentals (Version 2), https://developer.android.com/courses/fundamentals-training/overview-v2

Összefoglaló

- Objektumorientált programozás
- Memóriakezelés
- ARC-strong/weak/unowned
- Strong Reference Cycle
- Operátorok
- Függvény, closure, tuple
- Osztályok, objektumok
- Inicializálás
- Enum
- Struct
- Beágyazott típusok
- Extension
- Protokoll
- Delegate
- Hibakezelés

Android vs IOS

- Fejlesztési spebesség: 3x (Android fragmentáltság, eszközök)
- Új operndszer verzió követése (80% vs 10%)
- Alkalmazás kibocsájtás (IOS 7 nap vs 1-2 nap Android)
- 110.000-150.000 USD/év



Cél eszközök/Platformok

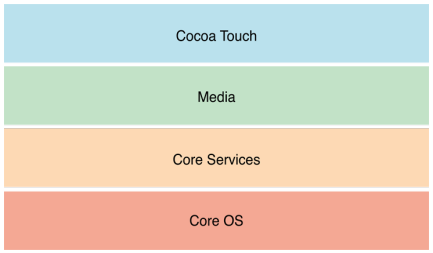
- iPhone
- iPad
- iWatch
- AppleTV
- MacOS
- Linux

Futtató környezetek

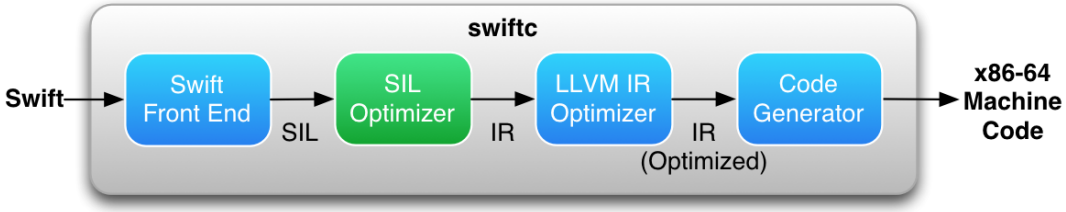
- OSX (MacOS) és IOS BSD alapú
  - 1980
    - NeXT (Steve Jobs) (MACH + BSD)
    - Cocoa keretrendszer (Cocoa – Cocoa Touch)
  - 1996 Apple megvette
  - 2000 OS X
    - Darwin (FreeBSD alapú)
  - 2004 – 2016
    - Jordan Hubbard – FreeBSD alapító

Elemek

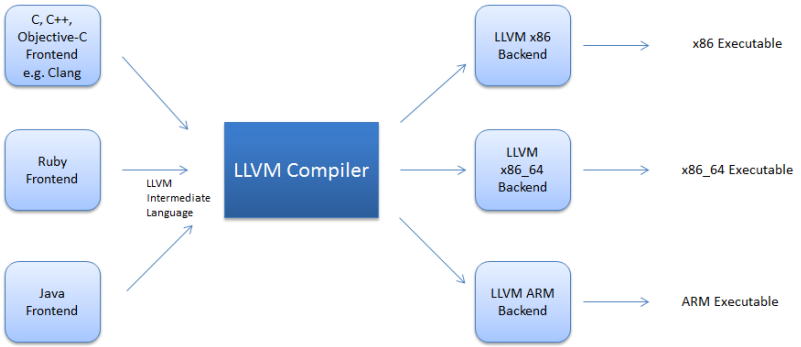
- SWIFT programozási nyelv (+ Objective C)
- Apple API készlet adott platformon
  - IOS
  - IOS keretrendszerek (50+)
    - HealthKit
    - HomeKit
    - …
- Fejlesztő környezet
  - XCode
- Apple felhő szolgáltatások (30+)
  - App Store
  - iCloud x
  - Push



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()
```

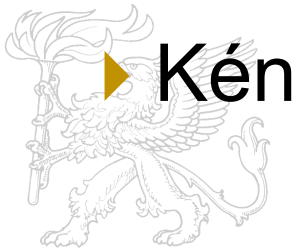
Optional

- 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
```

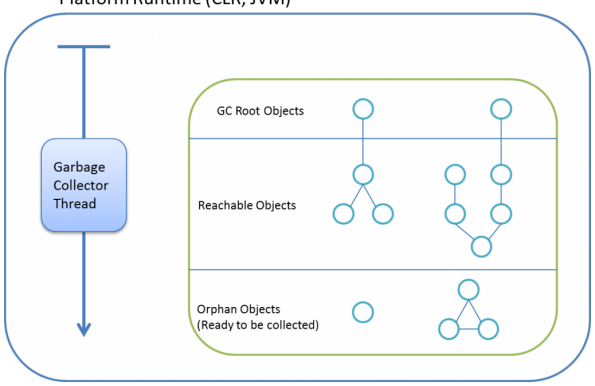
```
var unwrappedValue = optionalValue ?? defaultValue
```



```
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.")
}
```

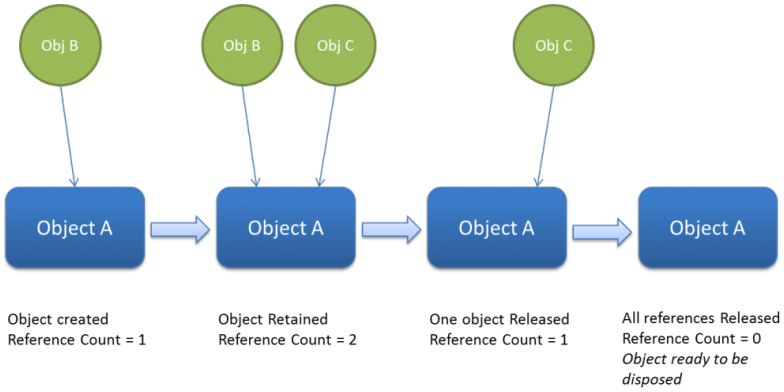
Memóriakezelés



- 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

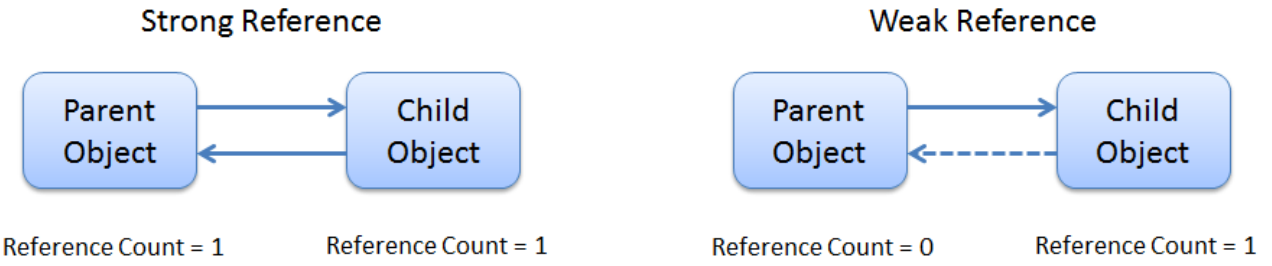


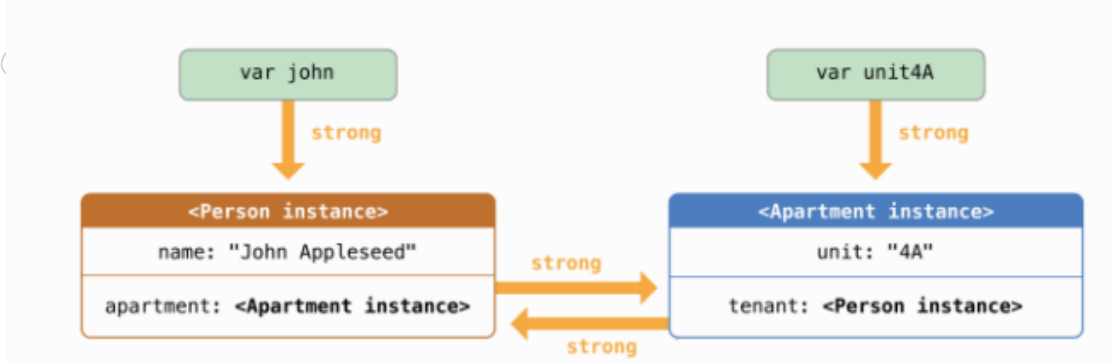
Visszatartó hurok(Retain Circle)

- 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





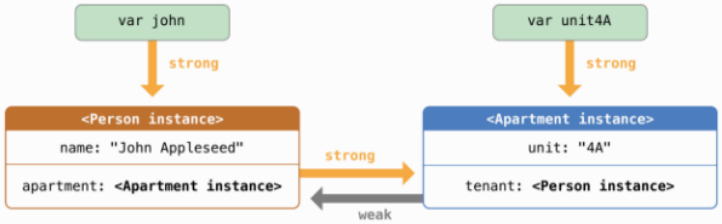

```
john = nil

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
    }
}
```

```
var mathFunction: (Int, Int) -> Int = addTwoInts
```

Függvények

- Első osztályú elemek – típusként kezelhetőek
- Címkézett paraméterek, alapértelmezett értékkel
- Változó hosszúság paraméterlista
- In-out kezelése
- Több visszatérési érték (tuple)

```
func someFunction(parameterWithoutDefault: Int,
        parameterWithDefault: Int = 12) {
    // In the function body, if no arguments are passed
        to the function
    // call, the value of parameterWithDefault is 12.
}
someFunction(parameterWithoutDefault: 3,
        parameterWithDefault: 6) //
        parameterWithDefault is 6
someFunction(parameterWithoutDefault: 4) //
        parameterWithDefault is 12
```

```
func arithmeticMean(_ numbers: Double...) -> Double {
    var total: Double = 0
    for number in numbers {
        total += number
    }
    return total / Double(numbers.count)
}
arithmeticMean(1, 2, 3, 4, 5)
// returns 3.0, which is the arithmetic mean of these
        five numbers
arithmeticMean(3, 8.25, 18.75)
// returns 10.0, which is the arithmetic mean of these
        three numbers
```

```
func swapTwoInts(_ a: inout Int, _ b: inout Int) {
    let temporaryA = a
    a = b
    b = temporaryA
}
```

Closure – Zárvány (Lambda kif.)

- Első osztályú objektum
- Eléri a környezetében definiált változókat, konstansokat, ezekre erős hivatkozást készít
- Általánosítva
  - Függvény
  - Beágyazott függvény
  - Closure

Closure példa

```
func backward(_ s1: String, _ s2: String) ->
          Bool {
    return s1 > s2
}
var reversedNames =
          names.sorted(isOrderedBefore:
          backward)
```

```
reversedNames = names.sorted(isOrderedBefore: {
          (s1: String, s2: String) -> Bool in
          return s1 > s2 } )
```

```
reversedNames = names.sorted(isOrderedBefore: {
          s1, s2 in return s1 > s2 } )
```

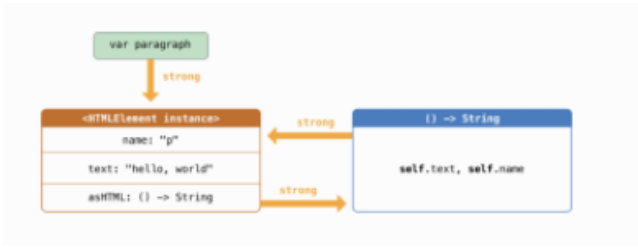
```
reversedNames = names.sorted(isOrderedBefore: {
          s1, s2 in s1 > s2 } )
```

```
reversedNames = names.sorted(isOrderedBefore: {
          $0 > $1 } )
```

```
reversedNames = names.sorted(isOrderedBefore: >)
```

Closure referencia

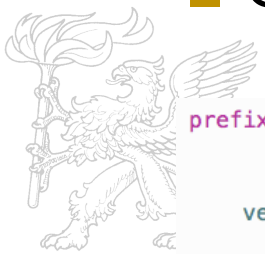
```
let heading = HTMLElement(name: "h1")
let defaultText = "some default text"
heading.asHTML = {
    return "<\(heading.name)>\(heading.text ??
           defaultText)</\(heading.name)>"
}
print(heading.asHTML())
// Prints "<h1>some default text</h1>"
```




```
class HTMLElement {

    let name: String
    let text: String?

    lazy var asHTML: () -> String = {
        [unowned self] in
        if let text = self.text {
            return "<\(self.name)>\(text)</\
           (self.name)>"
        } else {
            return "<\(self.name) />"
        }
    }
```

Operátorok

- Alap, fejlett operátor
- Saját operátor
  - infix (pl.: A+B)
  - prefix (pl.:-A)
  - postfix (pl.: +=)
  - értékadás (pl:=)
- Operátor függvény
  - Operátor kiterjesztése



```
prefix func +++ (vector: inout Vector2D) ->
          Vector2D {
    vector += vector
    return vector
}
```

```
prefix func - (vector: Vector2D) -> Vector2D {
    return Vector2D(x: -vector.x, y: -vector.y)
}
```

```
func += (left: inout Vector2D, right: Vector2D)
          {
    left = left + right
}
```

```
struct Vector2D {
    var x = 0.0, y = 0.0
}
func + (left: Vector2D, right: Vector2D) ->
          Vector2D {
    return Vector2D(x: left.x + right.x, y:
          left.y + right.y)
}
```

```
func == (left: Vector2D, right: Vector2D) ->
          Bool {
    return (left.x == right.x) && (left.y ==
          right.y)
}
func != (left: Vector2D, right: Vector2D) ->
          Bool {
    return !(left == right)
}
```

Tuple (Rendezett n-es)

- Egy értékbe tud több értéket zárni

```
let http404Error = (404, "Not Found")
// http404Error is of type (Int, String), and equals
          (404, "Not Found")
```

```
let (statusCode, statusMessage) = http404Error
print("The status code is \(statusCode)")
// Prints "The status code is 404"
print("The status message is \(statusMessage)")
// Prints "The status message is Not Found"
```

```
let http200Status = (statusCode: 200, description: "OK")
```



```
print("The status code is \(http200Status.statusCode)")
// Prints "The status code is 200"
print("The status message is \
          (http200Status.description)")
// Prints "The status message is OK"
```

Osztályok

- Nem kell osztálynként külön fájl
- Indexelhető
- Inicializáció/Deinicailazáció
- Eseményekezlés
  - willSet
  - didSet
- Referencia szerinti kezelés
- Identitás operátor (referencia)
  - ===
  - !==
- Egyezőség (tartalom szerint)

```
struct Resolution {
    var width = 0
    var height = 0
}
class VideoMode {
    var resolution = Resolution()
    var interlaced = false
    var frameRate = 0.0
    var name: String?
}
```

```
class StepCounter {
    var totalSteps: Int = 0 {
        willSet(newTotalSteps) {
            print("About to set totalSteps to \
          (newTotalSteps)")
        }
        didSet {
            if totalSteps > oldValue  {
                print("Added \(totalSteps - oldValue)
          steps")
            }
        }
    }
}
```

Osztályok

- Tuljadonságok
  - Tárolt
    - laza
    - normál
  - Számított
  - Típus adattag (statikus)

```
class DataImporter {
    /*
     DataImporter is a class to import data from an
          external file.
     The class is assumed to take a non-trivial amount
          of time to initialize.
     */
    var fileName = "data.txt"
    // the DataImporter class would provide data
          importing functionality here
}

class DataManager {
    lazy var importer = DataImporter()
    var data = [String]()
    // the DataManager class would provide data
          management functionality here
}
```

```
struct Point {
    var x = 0.0, y = 0.0
}
struct Size {
    var width = 0.0, height = 0.0
}
struct Rect {
    var origin = Point()
    var size = Size()
    var center: Point {
        get {
            let centerX = origin.x + (size.width / 2)
            let centerY = origin.y + (size.height / 2)
            return Point(x: centerX, y: centerY)
        }
        set(newCenter) {
            origin.x = newCenter.x - (size.width / 2)
            origin.y = newCenter.y - (size.height / 2)
        }
    }
}
var square = Rect(origin: Point(x: 0.0, y: 0.0),
                  size: Size(width: 10.0, height: 10.0))
: initialSquareCenter = square.center
iare.center = Point(x: 15.0, y: 15.0)
int("square.origin is now at (\(square.origin.x), \
        (square.origin.y))")
Prints "square.origin is now at (10.0, 10.0)"
```

```
struct SomeStructure {
    static var storedTypeProperty = "Some value."
    static var computedTypeProperty: Int {
        return 1
    }
}
enum SomeEnumeration {
    static var storedTypeProperty = "Some value."
    static var computedTypeProperty: Int {
        return 6
    }
}
class SomeClass {
    static var storedTypeProperty = "Some value."
    static var computedTypeProperty: Int {
        return 27
    }
    class var overrideableComputedTypeProperty: Int {
        return 107
    }
}
```

Metódusok

- Hasonló mint a függvény
- A self-et használhatja
- Struct/Enum esetén a módosítshoz:
  - mutating


```
struct Point {
    var x = 0.0, y = 0.0
    mutating func moveBy(x deltaX: Double, y deltaY:
            Double) {
        x += deltaX
        y += deltaY
    }
}
var somePoint = Point(x: 1.0, y: 1.0)
somePoint.moveBy(x: 2.0, y: 3.0)
print("The point is now at (\(somePoint.x), \
          (somePoint.y))")
// Prints "The point is now at (3.0, 4.0)"
```

Öröklődés

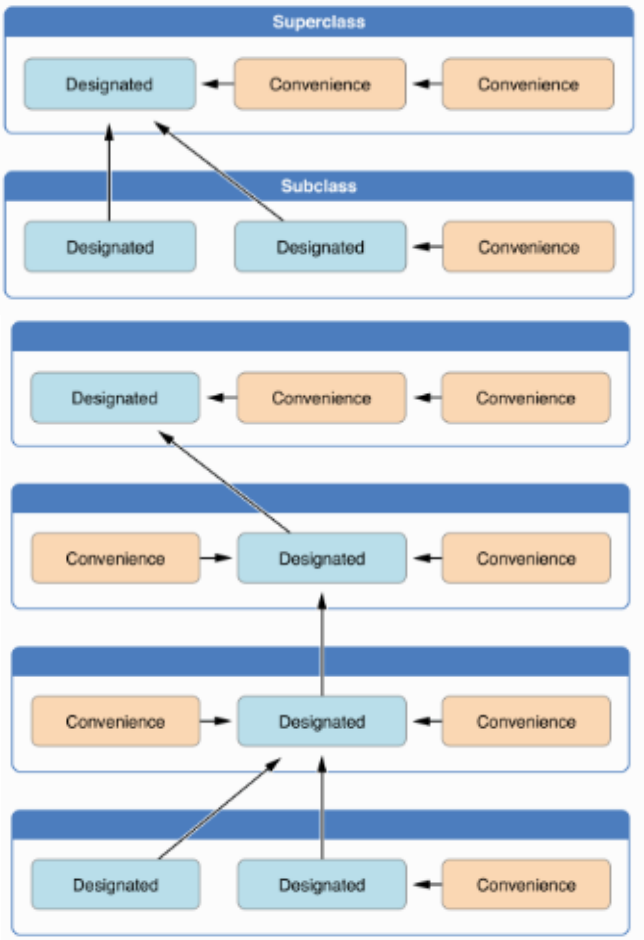
- Osztályokra vonatkozik
- Felülírás
  - override
  - final

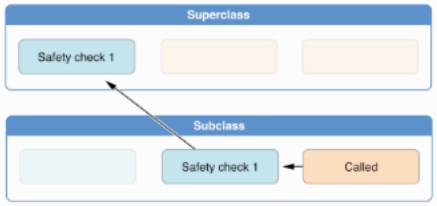
```
class Bicycle: Vehicle {

    var hasBasket = false

}
```

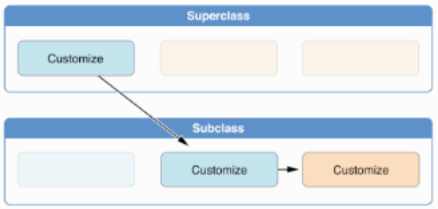
```
class Car: Vehicle {

    var gear = 1

    override var description: String {

        return super.description + " in gear \(gear)"

    }

}
```

Inicializálás

- Az osztály/struktúra/enum felkészítése a használatra
- Nem lehet inicializálatlan adattag
- Testreszabás
  - Paraméterezett
  - Alapértelmezett inicializálók
- Típusai
  - Elsődleges/Kijelölt
  - Kényelmi
- Kétfázisú







Struct

- Ugyanaz mint az osztály csak
  - Érték szerint adódik át
  - Nincs referencia számolás (csak egy referencia lehet)
  - Nincs öröklődés
  - Nincs deinicializálás


```
struct Resolution {
    var width = 0
    var height = 0
}
```

Enum

- Első osztályú elem
- Érték szerint adódik át
- Nyers érték nem csak int lehet

```
enum CompassPoint {
    case north
    case south
    case east
    case west
}
```

```
enum Barcode {
    case upc(Int, Int, Int, Int)
    case qrCode(String)
}
```

```
var productBarcode = Barcode.upc(8, 85909,
                                 51226, 3)
```

```
productBarcode =
                .qrCode("ABCDEFGHIJKLMNOP")
```

```
switch productBarcode {
case .upc(let numberSystem, let
          manufacturer, let product, let
          check):
    print("UPC: \(numberSystem), \
          (manufacturer), \(product), \
          (check).")
case .qrCode(let productCode):
    print("QR code: \(productCode).")
}
```

Extension

- Meglévő típusok bővítése új képességekkel (class/struct/enum/protocol)
- Forráskód nélkül is működik
- Képességek
  - Számított adattagok
  - Példány és típus metódusok
  - Inicializálók
  - Indexek
  - Beágyazott típusok
  - Protokollok

Számított érték példa

```
extension Double {

    var km: Double { return self * 1_000.0 }

    var m: Double { return self }

    var cm: Double { return self / 100.0 }

    var mm: Double { return self / 1_000.0 }

    var ft: Double { return self / 3.28084 }

}

let oneInch = 25.4.mm

print("One inch is \(oneInch) meters")

// Prints "One inch is 0.0254 meters"

let threeFeet = 3.ft

print("Three feet is \(threeFeet) meters")

// Prints "Three feet is 0.914399970739201 meters"
```

Metódusok,…

```
extension Int {
    subscript(digitIndex: Int) -> Int {
        var decimalBase = 1
        for _ in 0..<digitIndex {
            decimalBase *= 10
        }
        return (self / decimalBase) % 10
    }
}
746381295[0]
// returns 5
746381295[1]
// returns 9
746381295[2]
// returns 2
746381295[8]
// returns 7
```

```
extension Int {
    enum Kind {
        case negative, zero, positive
    }
    var kind: Kind {
        switch self {
        case 0:
            return .zero
        case let x where x > 0:
            return .positive
        default:
            return .negative
        }
    }
}
```

```
func printIntegerKinds(_ numbers: [Int]) {
    for number in numbers {
        switch number.kind {
        case .negative:
            print("- ", terminator: "")
        case .zero:
            print("0 ", terminator: "")
        case .positive:
            print("+ ", terminator: "")
        }
    }
    print("")
}
printIntegerKinds([3, 19, -27, 0, -6, 0, 7])
// Prints "+ + - 0 - 0 + "
```

Protokol

- Java – Interface
  - Megadhat adatagot is
- Metódusok, adattagok és egyébb tulajdonságok követelményét rögzíti
- Megvalósíthatja
  - Osztály (Class)
  - Struktúra (Struct)
  - Felsorolás (Enum)
- A típus amely adott protokollnak megfelel az protokoll konform

Tulajdonság követelmények

```
protocol SomeProtocol {

    var mustBeSettable: Int { get set }

    var doesNotNeedToBeSettable: Int { get }

}
```

```
protocol AnotherProtocol {

    static var someTypeProperty: Int { get set }

}
```

Metódus követelmények

- Metódus szintaktika metódus törzs nélkül

```
protocol RandomNumberGenerator {

    func random() -> Double

}
```

```
protocol Togglable {

    mutating func toggle()

}
```

```
enum OnOffSwitch: Togglable {
    case off, on
    mutating func toggle() {
        switch self {
        case off:
            self = on
        case on:
            self = off
        }
    }
}
var lightSwitch = OnOffSwitch.off
lightSwitch.toggle()
// lightSwitch is now equal to .on
```

Protokollok mint típusok

- Nem valósít meg konkrét funkcionalitást
- Teljes értékű típus
  - Visszatérési érték
  - Tulajdonság
  - Belső elem (tömb, …)

```
var d6 = Dice(sides: 6, generator: LinearCongruentialGenerator())
for _ in 1...5 {
    print("Random dice roll is \(d6.roll())")
}
// Random dice roll is 3
// Random dice roll is 5
// Random dice roll is 4
// Random dice roll is 5
// Random dice roll is 4
```

```
class Dice {
    let sides: Int
    let generator: RandomNumberGenerator
    init(sides: Int, generator: RandomNumberGenerator) {
        self.sides = sides
        self.generator = generator
    }
    func roll() -> Int {
        return Int(generator.random() * Double(sides)) + 1
    }
}
```

Protokoll öröklődés

- Egy vagy több protokolltól örökölhet

```
protocol PrettyTextRepresentable: TextRepresentable {

    var prettyTextualDescription: String { get }

}
```

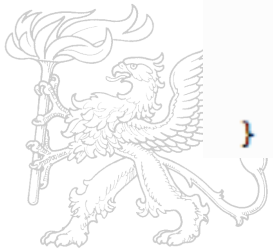
```
extension SnakesAndLadders: PrettyTextRepresentable {

    var prettyTextualDescription: String {

        var output = textualDescription + ":\n"

        for index in 1...finalSquare {

            switch board[index] {

            case let ladder where ladder > 0:

                output += "▲ "

            case let snake where snake < 0:

                output += "▼ "

            default:

                output += "o "

            }

        }

        return output

    }

}
```

Protokoll szűkítés

- class only
- feltételes

```
protocol SomeClassOnlyProtocol: class, SomeInheritedProtocol {

    // class-only protocol definition goes here

}
```

```
@objc protocol CounterDataSource {

    @objc optional func increment(forCount count: Int) -> Int

    @objc optional var fixedIncrement: Int { get }

}
```

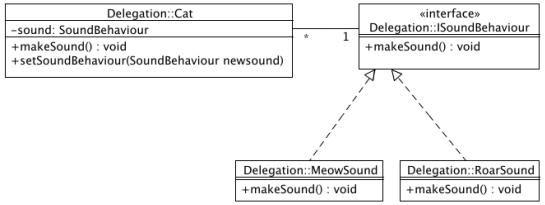


Alap implementáció

```
extension PrettyTextRepresentable  {

    var prettyTextualDescription: String {

        return textualDescription

    }

}
```

Delegate

```
protocol DiceGame {
    var dice: Dice { get }
    func play()
}
protocol DiceGameDelegate {
    func gameDidStart(_ game: DiceGame)
    func game(_ game: DiceGame, didStartNewTurnWithDiceRoll diceRoll: Int)
    func gameDidEnd(_ game: DiceGame)
}
```



- Tervezési minta a funkcionalitás kiszervezésére
- Protokoll segítségével valósítható meg
  - Esemény kezelése
  - Adat lekérése
  - …

```
class SnakesAndLadders: DiceGame {
    let finalSquare = 25
    let dice = Dice(sides: 6, generator: LinearCongruentialGenerator())
    var square = 0
    var board: [Int]
    init() {
        board = Array(repeating: 0, count: finalSquare + 1)
        board[03] = +08; board[06] = +11; board[09] = +09; board[10] = +02
        board[14] = -10; board[19] = -11; board[22] = -02; board[24] = -08
    }
    var delegate: DiceGameDelegate?
    func play() {
        square = 0
        delegate?.gameDidStart(self)
        gameLoop: while square != finalSquare {
            let diceRoll = dice.roll()
            delegate?.game(self, didStartNewTurnWithDiceRoll: diceRoll)
            switch square + diceRoll {
            case finalSquare:
                break gameLoop
            case let newSquare where newSquare > finalSquare:
                continue gameLoop
            default:
                square += diceRoll
                square += board[square]
            }
        }
        delegate?.gameDidEnd(self)
    }
}
```

Generikus adatszerkezet

- Flexibilis, újrahasználható függvény, típus

```
func swapTwoInts(_ a: inout Int, _ b: inout Int) {
    let temporaryA = a
    a = b
    b = temporaryA
}
```

```
func swapTwoValues<T>(_ a: inout T, _ b: inout T) {
    let temporaryA = a
    a = b
    b = temporaryA
}
```

Generikus típusok

- Egyedi osztályok, struktúrák, felsorolás típusok

```
struct Stack<Element> {
    var items = [Element]()
    mutating func push(_ item: Element) {
        items.append(item)
    }
    mutating func pop() -> Element {
        return items.removeLast()
    }
}
```

```
extension Stack {
    var topItem: Element? {
        return items.isEmpty ? nil : items[items.count - 1]
    }
}
```

Típus kényszerek

- Protokoll alapon szűkíthető

```
func findIndex<T: Equatable>(of valueToFind: T, in array:[T]) -
          > Int? {
    for (index, value) in array.enumerated() {
        if value == valueToFind {
            return index
        }
    }
    return nil
}
```

```
func allItemsMatch<
    C1: Container, C2: Container
    where C1.ItemType == C2.ItemType, C1.ItemType: Equatable>
    (_ someContainer: C1, _ anotherContainer: C2) -> Bool {

    // check that both containers contain the same number of
          items
    if someContainer.count != anotherContainer.count {
        return false
    }

    // check each pair of items to see if they are equivalent
    for i in 0..<someContainer.count {
        if someContainer[i] != anotherContainer[i] {
            return false
        }
    }

    // all items match, so return true
    return true

}
```

Hozzáférés vezérlés

- Implementációs részletek elrejtése
- Perferált interfész megadása
- Modul és forrás fájl alapú
  - Public – modulon belül és kívül
  - Internal - modulon belül
  - Private – fájlon belül

```
public class SomePublicClass {          // explicitly public
          class

    public var somePublicProperty = 0    // explicitly public
          class member

    var someInternalProperty = 0         // implicitly internal
          class member

    private func somePrivateMethod() {}  // explicitly private
          class member

}


class SomeInternalClass {               // implicitly internal
          class

    var someInternalProperty = 0         // implicitly internal
          class member

    private func somePrivateMethod() {}  // explicitly private
          class member

}


private class SomePrivateClass {        // explicitly private
          class

    var somePrivateProperty = 0          // implicitly private
          class member

    func somePrivateMethod() {}          // implicitly private
          class member

}
```

Hibakezelés

- Könnyűsúlyú hiba kezelés, return-nal egy súlyú
  - Nincs verem visszapörgetés
- Kivétel dobó metódusok
- Ami nem dob tovább azt belül kell lekezelni
- Kivétel gyakran enum

```
func canThrowErrors() throws -> String


func cannotThrowErrors() -> String
```

```
func vend(itemNamed name: String) throws {
    guard let item = inventory[name] else {
        throw VendingMachineError.invalidSelection
    }

    guard item.count > 0 else {
        throw VendingMachineError.outOfStock
    }

    guard item.price <= coinsDeposited else {
        throw VendingMachineError.insufficientFunds(coinsNeeded:
      item.price - coinsDeposited)
    }
```

Hibakezelés

- Hiba elkapás: do-catch

```
var vendingMachine = VendingMachine()
vendingMachine.coinsDeposited = 8
do {
    try buyFavoriteSnack(person: "Alice", vendingMachine: vendingMachine)
} catch VendingMachineError.invalidSelection {
    print("Invalid Selection.")
} catch VendingMachineError.outOfStock {
    print("Out of Stock.")
} catch VendingMachineError.insufficientFunds(let coinsNeeded) {
    print("Insufficient funds. Please insert an additional \(coinsNeeded)
          coins.")
}
// Prints "Insufficient funds. Please insert an additional 2 coins."
```

Hiba - Optional

```
func fetchData() -> Data? {
    if let data = try? fetchDataFromDisk() { return data }
    if let data = try? fetchDataFromServer() { return data }
    return nil
}
let photo = try! loadImage(atPath: "./Resources/John Appleseed.jpg")
```



```
func processFile(filename: String) throws {
    if exists(filename) {
        let file = open(filename)
        defer {
            close(file)
        }
        while let line = try file.readline() {
            // Work with the file.
        }
        // close(file) is called here, at the end of the scope.
    }
}
```

Összefoglaló

- Objektumorientált programozás
- Memóriakezelés
- ARC-strong/weak/unowned
- Strong Reference Cycle
- Operátorok
- Függvény, closure, tuple
- Osztályok, objektumok
- Inicializálás
- Enum
- Struct
- Beágyazott típusok
- Extension
- Protokoll
- Delegate
- Hibakezelés