Osnove swifta

• Proći appleov tutorial za swift: https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift\_Programming\_Language/TheBasics.html

• Odgovoriti na pitanja:

• Tipovi podataka:

• Nabrojati tipove podataka u swiftu. Opisati tipove te navesti primjer.

O: Int za cjelovite brojeve (2, 4, 5), float/double za decimalne (2.3, 3.14), String za tekstualne podatke (“tekst”), Boolean za točnost (true/false) i

kolekcijski tipovi Array ([1, 1, 2, 3, 4]), Set ([1, 2, 3, 4], razlika je da su svi unikatni i nisu orderani po indexu) i Dictionary ([“kljuc” : vrijednost])

• Objasni konstante i varijable. Napisati primjere.

O: Konstante su vrijednosti koje se ne mogu naknadno mijenjati ( let konstanta = 1 ), varijablama se mogu ( var varijabla = 1; varijabla += 2 )

• Objasni type safety.

O: Prilikom compiliranja compiler validira/provjerava jel se tipovi podudaraju i javlja korisniku ako je nekoj varijabli dodijelio krivi tip, npr int vrijednost u

variable tipa String. Ako korisnik ne navede tip, compiler će pretpostaviti tip prema vrijednosti varijable tako da nije potrebno svaki put deklarirati tip.

• Što su optionalni tipovi? Što oni mogu sadržavati za razliku od ostalih tipova? Kako se označavaju?

O: Tipovi čija vrijednost može sadržavati nil. Takve varijable možda sadržavaju vrijednost ili mogu biti prazne. Označavaju se dodavanjem upitnika prilikom

deklariranja njihovog tipa podatka ( String? )

• Na primjeru objasni unwrap proces te benefite.

O: Unwrapat se moze na tri nacina: if let, guard let i force unwrap. If let i guard let su slični, rade provjeru i ako ne prođe, izvršit će se else dio. Razlika je u tome što se kod guard leta nakon provjere može nastaviti koristiti varijaba. Force unwrap kako samo ime kaže, prisilno pretvara Optional u neki

non-optional tip podatka. Pretpostavka je da korisnik force unwrapa jer zna da može konverirati u non-optional, a Swift će probati konvertirati neovisno

jel to moguće ili ne i ako se ispostavi da nije moguće, kod će se srušiti. Benefit je da ne moramo brinuti o tome jel neka varijabla sadrži vrijednost ili ne i lagano

je odrediti sto da aplikacija ucini u oba scenarija.

• Objasni Dictionary. Navedi primjer.

O: Jedna od Collection tipova podataka, svaka varijabla se sastoji od unikatnog ključa i njene vrijednosti.

var dictionary = [“kljuc” : “vrijednost”, “kljuc2” : “vrijednost”]

U ovom primjeru su ključ i vrijednost tipa String.

• Objasni turples. Navedi benefite i primjer.

O: Tuples su grupirane vrijednosti. Benefit je da u funkcijama možemo returnati više vrijednosti tako što ih grupiramo kao tuple i stavimo u return kao jednu

vrijednost.

let turples : (String, String) = (“Tuples”, “Turples”)

• Objasni type alias.

O: Postojećim tipovima možemo dati sinonime radi boljeg konteksta ili lakšeg razumijevanja. Mozemo koristi za klase, enume, closure itd.

typealias BigNumber = Int64

• Objektno orijentirano programiranje

• Objasni Objasni Klasu.

O: Klasa je blok koda kojoj definiramo propertiese i metoda i tako joj dajemo funkcionalnost.

• Objasni Objekte.

O: Objekt je instanca klase.

• Objasni pointer / pokazivač. Navesti primjer.

O: Pointeri se koriste da bi se referencirala adresa u memoriji.

• Objasni protokole

O: Protokoli su predlošci metoda, propertiesa i drugih requirementsa koje potom možemo implementirati u klase, strukutre i enume.

• Objasni ekstenzije. Navesti benefite.

O: Dodavanje funkcionalnosti postojećim klasama, strukturama, enumima i protokolima. Benefit je što možemo dodati po potrebi funkcionalosti

ukoliko nešto nedostaje.

• Navedi i objasni posebne rezervirane Swift keywords-e

O: Posebne rezervirane keywordse ne možemo koristiti kao identifiere (bez backticka). Ima ih dosta, evo par:

class, struct, for, switch, where, typealias, enum, false, nil, self, default, private, …

• Objasni modifiere-e, posebno modifier-e pristupa.

O: Modifier pristupa ograničava ili omogućava pristup nekom članu (klase, metode itd.) izvan njegovovg scopea. Može biti public, private, internal, fileprivate i open.

Po defaultu, pristup je internal, znači da je dostupno unutar svog modulea (smislene jedinice). File private (pisano zajedno, fileprivate) je dostupnost unutar istog source filea,

private je dostupnost unutar svoje definicije (poput metode unutar klase/strukture) ili unutar definirajućeg source filea ako je top-level. Open i public su dostupni unutar i

i izvan svog modulea, a razlika je što open dopušta subclassanje i overrideanje metoda i klasa.

• Objasni method overloading. Navedi primjer.

O: Overloading je polimorfizam, funkcije istog naziva ali se diferenciraju prema tipu, kvantiteti i/ili redoslijedu parametara.

func sumNumbers(i1: Int, i2: Int) { return int+int2 }

func sumNumbers(i1: Int, i2: Int, i3: Int) { return int+int2+int3}

• Objasni mutabilnost.

O: Mogućnost da se mijenja vrijednost.

• Objasni funkcijsko programiranje.

O: Programiranje gdje je kod gradnje programa naglasak na primjeni funkcija i modularno se povezuju manje funkcije u smislene cjeline.

• Operatori:

• Objasni operator uspoređivanja.

O: Uspoređuje dvije vrijednosti: jednako (==), nejednako (!=), veće od (>), manje od (<), veće ili jednako (>=), manje ili jednako (<=)

• Objasni ternary operator.

O: Operator s tri operanda ( a ? b : c ), ovisno o točnosti “a”, izvršiti će se izraz “b” ili “c”

• Objasni Nil-Coalescing operator

O: Operator ( a ?? b ) koji će unwrap optional “a” ako sadrži vrijednost, inače će dodijeli default vrijednost “b”

• Navedi I objasni logičke operatore.

O: NOT (obrne Booleanovu vrijednost ( true postaje false, false postaje true ), AND (obje vrijednosti moraju biti true da bi izraz bio true) i

OR (barem jedna vrijednost mora biti true da bi izraz bio true)

• Kontrola toka:

• Na Primjeru objasni for each petlju.

O: for i in 1…5 { print i }

For each petljom možemo iterirati kroz neki raspon, niz elemenata (brojeve, znakove, Stringove…) i sl. tako da u bloku koda odredimo što želimo raditi s elementima dok petlja vrti kroz njih. Kad jednom prođe

kroz blok, petlja uzme sljedeći element i ponovi blok.

• Objasni .enumerated() funkciju unutar for each petlje. Navedi načine primjene.

O: Funkcija vraća niz parova gdje jedan predstavlja uzastopni cijeli broj koji započinje s nulom, a drugi predstavlja element niza. Možemo enumerirati kolekcije, brojati elemente

• Na primjeru objasni switch case petlju.

O: Switch je naprednija verzija if grananja, gdje se if koristi za jednostavnije uvjete dok se switch koristi za kompleksnije s više mogućih ishoda.

let i = 2

switch i {

case 1: { print (“broj je 1”}

case 2: { print (“broj je 2”}

default: { print (“nije ni 1 ni 2”) }

}

• Na primjeru objasni while loop.

O: While je petlja koja će se vrtjeti sve dok je uvjet petlje istinit.

let i = 1

while i<10 {

i += 1

print( “i je sada \(i) i ako je tvrdnja istinita, ponovno će se pokrenuti”

}

• Funkcije:

• Objasni funkcije. Navedi primjer deklariranja

O: Funkcije su blokovi koda koji izvršavaju specifičan zadatak. Funkcijama se daju imena koja se koriste da se izvrše ti zadaci na drugim mjestima u programu. Primjer deklariranja:

func primjerFunkcije(parametar: tipPodatka) {

\*ovdje ide kod koji će se izvršiti\*

}

• Objasni argument label.

O: Argument label se može dodati prije imena parametra, a omogućuje ekspresivan način pozivanja funkcije.

• Objasni inout parametre.

O: Ako se doda ključna riječ inout ispred tipa podatka parametra. Inout parametri mogu izaći iz funkcije promijenjenih vrijednosti i da promjene ustraju izvan funkcije.

• Objasni try catch Blok. Navedi razloge korištenja.

O: try catch se koristi za error handlanje, za hvatanje errora umjesto da se samo sruši program. Koristimo kada postoji mogućnost da se nešto neće izvršiti ili da uzrokuje rušenje aplikacije,

kako bi saznali što uzrokuje error…

• Rad sa različitim tipovima podataka:

• Objasni razliku između Any i AnyObject.

O: Any predstavlja instancu bilo kojeg tipa, a AnyObject instancu bilo kojeg tipa klase.

• Objasni razliku između objekta i protokola

O: U objektu je fokus na opis objekta, u protokolu je fokus na što protokol može izvršiti.

• Objasni downcasting.

O: Konvertiranje jednog typea u drugi, iz superklase u podklasu. (as, as? za optional, as! za force unwrap)

• Objasni upcasting.

O: Konvertiranje instance iz jednog typea u drugi, ali obrnuto od downcastinga, iz tipa podklase u tip superklase.

• Analiziraj slijedeću situaciju. Potrebno je napisati array koji će sadržavati više različitih objekata. Objekti koji trebaju biti uneseni u array posjeduje neke zajedničke osobine. Kako bi implementirao ove objekte odnosno array?

O: Ako imaju neke zajedničke osobine, napravio bi klasu koja bi sadržavala te osobine te objekte upcastao u tip nove klase.

• Closures:

• Objasni Closure. Navedi primjer.

O: Blokovi funkcionalnosti koje se mogu prosljeđivati okolo i koristiti u kodu, mogu uhvatiti i zadrzati referencu na bilo koju konstantu ili varijablu unutar konteksta u kojem je definiran. Primjer:

let printClosure = { print(“Jednostavni closure koji printa”) }

printClosure()

• Objasni primjenu closure-a.

O: mogu uhvatiti i zadrzati referencu na bilo koju konstantu ili varijablu unutar konteksta u kojem je definiran

• Objasni razliku između escaping in non escaping closure-a.

O: Escaping closure se poziva nakon što se returna funkcija u kojoj je proslijeđena, a non escaping closure se poziva unutar funkcije u koju je proslijeđena, prije returna te funkcije.

• Objasni odnos između closure-a i funkcije/objekta koja ga poziva.

O: Kada funkcija/objekt poziva closure i closure uhvati referencu na instancu ili njegove članove, stvara se strong reference između closurea i instance.

• Objasni potencijalne memory leakove koji se mogu uzrokovati closuerom.

O: Povezano s prethodnim pitanjem, closure je, kao klasa, reference type. Ako dodijeliš closure jednom propertyu, to je kao da joj dodijeljuješ reference, i onda se ta

Dva strong reference međusobno drže živim iako možda obje ili jedna od njih više nije potrebna.

• Enums:

• Objasni enum-e. Navedi primjer.

O: Enum-i definiraju tip za grupe srodnih vrijednosti. Primjer:

enum Color {

case red

case blue

case green

}

• Navedi benefite korištenja enuma.

O: može se koristi kao switch case, mogu se odrediti vrijednosti koje varijabla može uzeti, mogu se izbjeći greške kod invalid konstanti

• Objasni enum raw value.

O: raw value su unaprijed popunjene defaultne vrijednosti enuma, moraju biti unikatne unutar svog enuma, mogu biti string, znak ili integer/floating-point broj i raw value je uvijek ista za određeni enum slučaj jer

raw value se razlikuje od pridružene vrijednosti.

• Alokacija i memorija:

• Objasni proces kreiranja objekta

O: Kad se kreira objekt, ARC dodijeli dio memorije za pohranu podataka o toj instanci. Ona sadrži informacije

o vrsti instance i vrijednosti svih svojstava povezanih s tom instancom. Kada instanca više nije potrebna, ARC oslobađa memoriju koju koristi ta instanca.

• Objasni ARC

O: Automatic Reference Counting je alat koji Swift koristi za memory management. U većini slučajeva on radi under the hood i korisnik ne mora brinuti o tome. Svaki puta kada instanciramo klasu ARC joj dodijeli dio memorije za spremanje. Kada instanca više nije potrebna ARC automatski čisti memoriju i briše tu instancu.

• Navedi najčešće pogreške koje uzrokuju memory leakove.

O: zadržavanje objekata koje nisu više potrebne i samo drže prostor u memoriji,