30 / 11 / 2021

SWİFT PROGRAMLAMA DİLİ

Hazırlayan

Ümmü Derya Çelik

SWİFT PROGRAMLAMA DİLİ

Swift, Apple tarafından İOS ve macOS işletim sistemlerinde uygulama geliştirmek için oluşturulmuş nesne yönelimli programlama dilidir.

Swift programlama dilinde uygulama geliştirmek için Xcode programı kullanılmaktadır. Xcode programı sadece macOS platformlarında çalıştırılabilir. Elimizde bu imkan yoksa kullanılan platform fark etmeksizin online swift uygulamaları yazabileceğimiz Online Swift Compiler mevcuttur. Xcode kadar işlevsel olmasa da kodun çıktısını görebilme imkanı sunar.

Swift yazımını örnekler üzerinden inceleyelim;

1) Değişkenler

Bir değişken oluşturmak için ilk önce değişken tipini belirlemeliyiz. "var" ve "let" olmak üzere 2 tip değişken vardır.

- Var : "variable" yani değişken kelimesinin kısaltmasından gelmektedir. Uygulama içinde değiştirilebilen değerlerdir.
- Let : Sabit, sürekli aynı değeri tutan değişken tipidir. Let tipinde olan bir değer kod içerisinde değiştirilemez.

```
var userName = "Derya"
```

userName değişkenine Derya string i atandı. Kod devamında;

```
userName = "Piton"
```

Kodunu eklersek Derya olarak verilen değer Piton olarak değiştirilmiş olur.

```
let userName = "Derya"
```

Aynı değer "let" parametresi kullanılarak yazılırsa kod içerisinde artık değiştirilemeyecektir.

Bu kullanım integer, double ve float değerler için de geçerlidir. Örnek kullanımı;

```
var userId = 1 -> Değiştirilebilir

let userId = 1 -> Değiştirilemez
```

2) Döngüler

2.1) While Döngüsü

Bu döngüyü kullanarak myNumber değerini 10'a ulaşana kadar 2'şer arttırarak nasıl ekrana yazdıracağımıza bakalım;

-

```
var myNumber = 0
while myNumber <= 10
{
   print(myNumber)
   myNumber += 2
}
```

2.2) For Döngüsü

For döngüsünün kullanımına birkaç örnekle görelim.

-

```
var myFruitArray = ["Banana","Apple","Orange"]
for fruit in myFruitArray {
   print(fruit)
}
```

myFruitArray adıyla oluşturduğum diziyi fruit adlı değere atayıp bu değeri yazdırdık. Çıktımızda myFruitArray dizisinin elemanlarını görüyor olacağız.

-

```
var number = [50,60,70,10]
for islem in number {
   print(islem / 5)
}
```

-

```
for myInt in 1 ... 5 {
   print(myInt)
}
```

1 den 5 e kadar yazdırır

2.3) İf Döngüsü

Kullanımı aşağıdaki örnekteki gibidir;

```
var myage = 32
if myage < 30 {
   print("30 -")    ->eğer myAge 30'dan küçükse 30- yazdır.
}
else if myage > 30 && myage < 40 {
   print("30s")   ->eğer myAge 30'dan büyük ve 40'tan küçükse 30s
yazdır.
} else {
   print("40s") } ->eğer myAge hiçbir parametreye uymuyorsa 40s
yazdır.
```

3) Fonksiyonlar

Fonksiyonların erişilebilirlik seviyeleri mevcuttur. Bunlar;

- Open: içeriği değiştirilebilir, başka projeler tarafından ulaşılabilir fonksiyonlardır.
- Public: İçeriği değiştirilemeyen ama başka projeler tarafından kullanılabilen, ulaşılabilen fonksiyonlardır.
- Internal: Default tanımlanmış değerdir.
- FilePrivate: Hangi dosya içine tanımlandıysa sadece ona özel olan fonksiyonlardır.
- Private: Hangi sınıfta tanımlandıysa sadece o sınıftan erişilebilen fonksiyonlardır.

Basit örnek üzerinden swift dilinde fonksiyonun yazım şekline bakalım;

-

```
func myFunction() {
   print("Piton")
}                        -> myFunction adında fonksiyon oluşturuldu.
print(myFunction()) /*ya da*/
myFunction()         -> fonksiyon yazdırıldı
```

Fonksiyon kullanarak girilen değerleri toplama işlemi yapan basit bir uygulama yapalım;

```
func sumFunction(x: Int, y: Int) -> Int
{
   return x + y          -> İ şlemin yapılacak olduğu fonksiyon
}
let myExample = sumFunction(x: 10, y: 20)  -> tanımlanan değerlere istenilen sayı girilip myExample değişkenine atadık

print(myExample)     -> myExample değişkenine atanan fonksiyon çalışıp bize sonucu vermektedir.
```

-

```
func logicFunction(a: Int, b: Int) -> Bool {
   if a > b {
      return true  -> a değeri b değerinden büyük ise “true” yazdır.
    } else {
       return false  -> büyük değilse “false” yazdır.
    }  }
print(logicFunction(a: -10, b: 0)) -> “-10” ve “0” değerlerini verdiğimize göre çıktı “false” şeklinde olacaktır.
```

- Sistem hazır fonksiyonları

• Append() = string değerinin sonuna harf ekler. Kullanımı “ userName.append(“z”) ” şeklindedir.

• Lowercased() = değerin tüm harflerini küçük yazdırır ama anlıktır, orijinal değerde değişiklik yapmaz. Kullanımı “ userName.lowercase()” şeklindedir.

• Uppercased() = değerin tüm harflerini büyük yazdırır ama anlıktır, orijinal değerde değişiklik yapmaz. Kullanımı “ userName.uppercase()” şeklindedir.

4) Opsiyoneller

Dışarıdan girilmesi gereken değerin girilmediği veya değerin girileceğinden emin olunan durumlarda uygulama çökmesini önlemek için değişken ya da sabitin alacağı veri tipinin yanına “?” veya “!” koyulur. Özellikle metin kutusu gibi bir yerden değer geliyorsa mutlaka opsiyonel tanımı yapılmalıdır.

? = Herhangi bir değer girilmeme imkanı olduğunu belirtir. Değer girilmemesi ya da belirtilen tip yerine farklı tipte değer verilmesi, integer değer verilmesi gerekirken string girilmesi gibi durumlarda uygulamanın çökmesi önlenmiş olur.

! = kod yazarken hata mesajlarıyla karşılaşmamak adına belirtilen değerin kesinlikle girileceğinden emin olunduğunu gösterir.

Örnek kullanım;

```
var myAge = "apple"
 var myInteger = (Int(myAge) ?? 0) * 5
-> myAge değeri string girildi. işlem yapılabilmesi için İnteger değere
çeviriyoruz. “??” işareti, myAge e girilen değer integer olmazsa "0"
değerini alıp işlem yap anlamına gelir.
 print(myInteger)
```

“??” işareti sayesinde sayı girilmesi gerekirken farklı tip değer girilmesi durumunda uygulamanın çökmesi engellendi.

Eğer random değer vermek yerine, yukarıdaki örnekte 0 değerini vermemiz gibi, farklı bir işlem yaptırma, hata mesajı gönderme gibi, istiyorsak “if let” kalıbı kullanılır.

Örnek kullanımı;

```
 if let myNumber = Int(myAge) {
    print(myNumber * 5)
} else {
    print("Yanlış Girdi")
}
```

5) Dictionary – Sözlük

Dictionary, bir değere anahtar kelime eklemek olarak açıklanabilir. Bir örnekle kullanım şeklini inceleyelim.
Yapılan spor ve o sporun bir saat yapılması durumunda kaç kalori yaktığını gösteren bir anahtarlama örneği yapalım;

```
var myDictionary = ["Run" : 100, "Basketball" : 200, "Swim" : 300]
->kullanımı örnekte görüldüğü gibidir.
print(myDictionary["Run"]) -> Ekran çıktısı koşmanın değeri olan 100 şeklinde olacaktır.
```

Yazılan filmin yönetmenini çıktı olarak veren kodu yazalım;

```
var myFavoriteDirectors = ["Pulp Fiction" : "Tarantino","Lock Stock" : "Guy Ritchie","The Dark Knight" : "Cristopher Nolen"]

print(myFavoriteDirectors["Pulp Fiction"])
print(myFavoriteDirectors["The Dark Knight"])

myFavoriteDirectors["Pulp Fiction"] = "Quentin Tarantino"
->Şeklinde değiştirilme yapılabilir. “Tarantino” olan değer “Quentin Tarantino” şeklinde değiştirildi.

myFavoriteDirectors["Seven Samurai"] = "Akira Kurisowa"
->Şeklinde ekleme yapılır.
```

6) Enum ve İnit

Enum, bir grup bağlantılı değer için ortak bir tür tanımlama olarak düşünülebilir. Örnek üzerinde görelim;

MusiciansType adında bir değer tip oluşturmak istiyoruz. Bu tip içinde leadGuitar, Vocalist, Bassist, Violenist olarak kategorilere ayrılan kod aşağıdaki gibidir;

```
enum MusicianType {
        case Gitarist
        case Vokalist
        case Bas
        case Kemanist }
```

Enum kullanımı, dışarıdan girilen değerlerde istenmeyen veri ve yazım yanlışı yapılmış verilerin girilmesini önlemektedir.
Örneğin, Müzisyen tipine girdi olarak mevcut olmayan bir tip olan piyanistin girilmesini istemiyorsak enum ile bu engellenebilir.

İnit ise parametre almamış fonksiyon gibidir. Bu işlem depolanmış olan her özellik için bir başlangıç değeri ayarlamayı ve kullanıma hazır olmadan önce gerekli işlemlerin yapılmasını sağlar. Böylece gereksiz kod satırlarını ve yazım yanlışları önlenmiş olur.

İnit kullanmadan ve init kullanarak aynı kodun nasıl yazıldığını inceleyelim. Bu örnekte enum örneğinde kullandığım MusicianType tipi de kullanılacaktır.

İnit kullanmadan;

```
var name : String  = " "
var age : Int = 0                    ->  Hata almamak için boş değerler
var Instrument : String = " "            atamamız gerekiyor.
```

Şeklinde değişkenler tanımlandıktan sonra girdi, yani bu örnekte müzisyen eklemekte kolay olmayacaktır;

```
james.age = 50
james.name = "James Hotfield"
james.instrument = "Guitar"
```

Aynı işlemi İnit kullanarak yapalım;

```
var name : String
var age : Int
var Instrument : String
var type : MusiciansType
```

```
    init(nameInit: String,ageInit: Int,instrumentInit: String,typeInit:
    MusicianType) {
        name = nameInit
        age = ageInit
        instrumentInit = instrumentInit
        type = typeInit
    }
```

Şeklinde değişkenler tanımlanıp atanmaktadır. Girdi ekleme işleminin nasıl yapılacağına bakalım;

```
    let james = Musicians(nameInit: "james", ageInit: 21, instrumentInit:
    "guitar", typeInit: .Vocalist)
    let Chris = SuperMusician(nameInit: "james", ageInit: 23, instrumentInit:
    "guitar", typeInit: .leadGuitar)
```

->Görüldüğü gibi bize yazma kolaylığı sağlamaktadır.

Swift Programlama Dili ile Hesap Makinesi

```
import UIkit
class ViewController: UIViewController {
```

```
@IBOutlet weak var firstText: UITextField!
@IBOutlet weak var secondText: UITextField!
@IBOutlet weak var resultLabel: UILabel!
```

```
@IBAction func sumClicked(_ sender: Any) {
if let firstNumber = Int(firstText.text!) {
```

```
        if let secondNumber = Int(secondText.text!) {
                let result = firstNumber + secondNumber
                resultLabel.text = String(result)
                }
        }
}
```

```
@IBAction func minusClicked(_ sender: Any) {
if let firstNumber = Int(firstText.text!) {
        if let secondNumber = Int(secondText.text!) {
                let result = firstNumber - secondNumber
                resultLabel.text = String(result)
                }
        }
}
```

```
@IBAction func multiplyClicked(_ sender: Any) {
if let firstNumber = Int(firstText.text!) {
        if let secondNumber = Int(secondText.text!) {
                let result = firstNumber * secondNumber
                resultLabel.text = String(result)
                }
        }
}
```

```
@IBAction func divideClicked(_ sender: Any) {
```

```
if let firstNumber = Int(firstText.text!) {
        if let secondNumber = Int(secondText.text!) {
                let result = firstNumber / secondNumber
                resultLabel.text = String(result)
                }
        }
}
```