Mobil Uygulama Geliştirme

Öğr. Gör. Zafer SERİN

FONKSİYON NEDİR?

- Fonksiyonlar, belirli bir görevi yerine getirmek için tasarlanmış, yeniden kullanılabilir kod bloklarıdır.
- Neden kullanırız?

1. Yeniden kullanılabilirlik: Aynı kodu tekrar tekrar yazmayı engeller.
2. Modülerlik: Programı daha küçük yönetilebilir parçalara böler.
3. Okunabilirlik: Kodun ne yaptığını anlamayı kolaylaştırır.
4. Bakım Kolaylığı: Bir değişikliği tek bir yerden yapmamızı sağlar.

BİR FONKSİYON NASIL TANIMLANIR?

```
fun selam_ver(){
    println(“Merhaba Kotlin!”)
}
```

- Açıklama:

fun: Fonksiyon tanımını başlatan anahtar kelime

selam_ver: Fonksiyonun adı

(): Parametre listesi(şu an boş)

{}: Fonksiyonun gövdesi(yapacağı işler)

PARAMETRELER VE GERİ DÖNÜŞ TİPLERİ

- Parametre(Argüman): Fonksiyonun çalışması için dışarıdan aldığı veriler.
- Geri Dönüş Tipi(Return Type): Fonksiyonun görevi bittikten sonra geri döndürdüğü sonucun tipi.

OVERLOADİNG

- Fonksiyonlar tamamen aynı isim, parametre ve geri dönüş değerlerine sahip olursa Kotlin bu şekilde bir tanımlamaya izin vermeyecektir. Bunun önüne geçmek için parametrelerin tipi, sırası veya fonksiyonun geri dönüş değeri değiştirilerek overload(aşırı yükleme) işlemi yapılabilir.

EXTENSİON

- Kotlin classlarına doğrudan metot eklemesi yapılabilir. Eğer “infix” anahtar sözcüğü metodun başına eklenirse class member erişim yapısı olan “.” yerine doğrudan isim ile çağrım sağlanır.

ÖRNEKLER

1. Parametre olarak girilen kilometreyi mile dönüştürdükten sonra geri döndüren bir metod yazınız. Mile = Km x 0.621
2. Kenarları parametre olarak girilen ve dikdörtgenin alanını hesaplayan bir metod yazınız.
3. Parametre olarak girilen sayının faktoriyel değerini hesaplayıp geri döndüren metodu yazınız.
4. Parametre olarak girilen kelime içinde kaç adet e harfi olduğunu gösteren bir metod yazınız.

ÖRNEKLER

1. Parametre olarak girilen kenar sayısına göre her bir iç açıyı hesaplayıp sonucu geri gönderen metod yazınız.

- İç açılar toplamı = ((Kenar sayısı - 2) x 180) / Kenar sayısı

2. Parametre olarak girilen gün sayısına göre maaş hesabı yapan ve elde edilen değeri döndüren metod yazınız.

- 1 günde 8 saat çalışılabilir.
- Çalışma saat ücreti : 40 ₺
- Mesai saat ücreti : 80 ₺
- 150 saat üzeri mesai sayılır.

3. Parametre olarak girilen otopark süresine göre otopark ücreti hesaplayarak geri döndüren metodu yazınız.

- 1 saat = 50 ₺
- 1 saat aşımından sonra her 1 saat , 10 ₺'dir.

KOTLİN İLE NESNE YÖNELİMLİ PROGRAMLAMA

- Kotlin ile aşağıdaki şekilde bir class oluşturulabilir:

```
class Ogrenci
{

}
```

KOTLİN İLE NESNE YÖNELİMLİ PROGRAMLAMA

- Kotlin ile aşağıdaki şekilde bir nesne oluşturulabilir:

```
var ogr1 = Ogrenci()
```

KOTLİN İLE NESNE YÖNELİMLİ PROGRAMLAMA

- Kotlin ile aşağıdaki şekilde bir classın yapıcı metodu oluşturulabilir:

```
class Ogrenci {
    constructor(isim: String, yas: Int)
    {
        // yapici metot
    }
}
```

KOTLİN İLE NESNE YÖNELİMLİ PROGRAMLAMA

- Kotlin ile aşağıdaki şekilde bir classın primary yapıcı metodu oluşturulabilir:

```
class Ogrenci(var isim: String, var yas: Int) {


}
```

- init{} bloğu primary constructorın gövdesi gibi düşünülebilir. Birden fazla init bloğu olabilir ve yukarıdan aşağıya doğru çalıştırılır. Önce primary constructor atamaları, sonra init blokları en sonda secondary constructor çalışır!

NULLABLE KONTROLÜ

- Kotlin programlama dilinde tüm tipler(primitif ve referans) null olabilme özgürlüğüne sahiptir. Bunu tip belirtimi sonrası “?” koyarak sağlayabiliriz.

  var isim : String? = null
- Programatik açıdan ise bir değişkenin null olması en büyük problemlerden biridir ve pek çok hataya sebebiyet verebilir. Kotlin’de bunun önüne geçmek için null kontrolü sağlanması gerekir. Burada klasik if ile null kontrolü, ?. ile değişken null ise hata vermek yerine tümden bull verilmesi, !! ile değişkenin null olmayacağını derleyiciye iddia etmek ve sorumluluğu almak ayrıca en güvenlilerden ?.let{} ile kontrol sağlanır.

NULLABLE KONTROLÜ

- Kotlin programlama dilinde primitif olmayan türler için lateinit anahtar sözcüğü kullanılarak bir değişkenin değerinin sonradan atanacağı söylenebilir. Burada ki durum null'dan tamamen farklıdır lateinit ile tanımlanan bir değişkenin değeri null bile olmaz, değeri yoktur.
- lateinit val ile kullanılamaz!

ACCESS MODİFİER

- Kotlin programlama dilinde aşağıdaki access modifiers yapıları kullanılabilir.

public: Her yerden erişilebilir.

private: Sadece tanımlandığı yerden erişilebilir (sınıf veya dosya)

protected: Tanımlandığı sınıf ve alt sınıflarından erişilebilir.

internal: Sadece tanımlandığı modülün içinden erişilebilir.

- Varsayılan erişim belirteci public'tir.

DATA CLASS

- Kotlin programlama dilinde bir classın içindeki metot ve yapılardan ziyade değerleri önemliyse data class olarak tanımlanabilir. Json API'den gelen verilerin karşılanması için çok kullanışlıdır.
- data class Kisi(var isim: String, var yas: Int)

STATİC ÖGELER

- Kotlin programlama dilinde companion object {} yapısı içerisine yazılan yapılar static ögeler haline gelir. Bunlara doğrudan class adı ile erişilebilir.

ENUM CLASSI

- Kotlin programlama dilinde enum classı ile tip güvenliği ve anlam karmaşasının önüne geçilebilir.

COMPOSİTİON

- Composition yapısı ile veritabanı tabloları Kotlin ile karşılanabilir.

Kategoriler Tablosu

kategori_id	kategori_ad
1	Dram
2	Komedi
3	Bilim Kurgu

Yonetmenler Tablosu

yonetmen_id	yonetmen_ad
1	Nuri Bilge Ceylan
2	Quetin Tarantino
3	Christopher Nolan

Filmler Tablosu

film_id	film_ad	film_yil	kategori_id	yonetmen_id
1	Django	2013	1	2
2	Inception	2006	3	3