\*\*\*\*\*Fotoğraflar eklemek faydalı ve daha okunaklı hale getirecektir\*\*\*\*\*

Kaynak dosyaların github linkelri kısmını düzenle bir ders dosyalarından hepsine ulaşım mevcut ek olarak da ders sonrası tekrarlardan spesifik olarak hangisine ulaşabileceklerini belirtmek güzel olur.

Bu yazımda genel olarak Android Studio platformunu üzerinde Kotlin programlama dilini kullanrak mobil uygulama geliştirme adımlarını anlatacağım.

Öncelikle parça parça gitmek adına Kotlin dilinin temelleri ve Anrodis Studio üzerinden proje oluşturmasından başlayacağım sonrasında …

Android studio uygulamasını <https://developer.android.com/studio?hl=tr> sayfasından indirme işlemini yaptıktan sonra kurulum kısmını tamamlıyoruz(gayet vasit bir şekilde ileri-next adımları ile kurulumu yapabiliriz). Sonrasında uygulamayı açtığımızda karşımıza çıkan sayfadan

New project -> Empty views activity ->

Name -> Name of project

Package name -> Package kısmını bir klasör gibi düşünebiliriz. Burada mümkün mertebe küçük harf kullanılmalı

Language -> Kotlin

Minumum Sdk -> Bu kısımda genelde mevcut olarak önerilen hangisi ise o kullanılabilir. Seçtiğimiz sürüm ve sonrasına uygun olacak şekilde programımızın çalışacağını ayarlamak içindir bu kısım. Buradaki yüzdelik ise yaklaşık olarak cihazların kaçına uygun olabileceğini göstermek içindir.

Build configuration language -> Değiştirmeye gerek yoktur.

Bu adımlardan sonra konfigurasyon işlemi yapıldıktan sonra açılan dosyalar üzerinde yapılacak işlemler;

app -> kotlin+java -> proje oluştururken verdiğimiz paket ismine gelip sağ tık ardından New -> kotlin Class/file -> file -> file name(Class ve file isimleri büyük harfle başlar). Bu adımlardan sonra elimizde verdiğimiz dosya ismiyle uyumlu .kt uzantılı bir dosya oluşmuş olması lazım. Buraya kodlarımızı yazıp çalıştırma işlemlerini yapmaya başlayabiliriz.

fun main(){ } bunu yazdıktan sonra (main yazıp tab tuşuna basarak direkt kendisi tamalar) kodlarımızı yazmaya başlayabiliriz

\*\*\*Dosya ismini doğru bir şekilde android studio code üzerinde değiştirmek için dosya isminde sağ tık ardından refactor->rename demek daha doprudur. Direkt rename dediğimizde dediğimizde değiştirirken problemlerin yaşanma ihtimali vardır.

Kotlin değişkenler;

-Kotlinde biz genel olarak String, integer, double ve diğer değişkenlelri tutabilmek için var ifasesini kullanırız(variable) Örneğin ;

var number = 15

var name = “HBA” veya var name : String = “HBA”

Bu şekilde kendisi otomatik olarak hafıza da yer tutma işlemlerini ayarlamış olur. var yerine val ifadesini kullanarak değişken oluşturusak bu sefer bu ifadenin bir sabit olduğunu yani sonrasında değiştirmemize müsade etmediği anlamına gelir. Eğer bir ifade de değişkenlik yoksa val kullanmak hafıza kullanmı açısından çok daha avantajlıdır. Çünkü boyut sabit olarak ayarlanmıştır fakat var ifadelerde ileride bunun boyutu değiştirilebileceği için hafıza da yer ayırma işlemlerinde daha fazla yer tutup hafıza performansı açısından daha işlevsizdir.

Var mutable yani değiştirilebilir değişken tüüryken val bir immutable yani değiştirilemez sadece okumaya izin verilen bir değişken türüdür.

Değişkenlerimizi veya bir ifadeyi terminale vermek için

println() fonksiyonu kullanılır. Burada kendi yazmak istediğimiz ifadeleri çift tırnak içinde yazarız ve bir değişkeni yazdırmak istersek eğer bir string ifade yoksa println içinde sadece değişken ismini vermemiz yeterli olacaktır. Ama bir string ifade ile beraber yazmak içinse şu şekilde bir yol izlenir;

println(“Name of student is : $studentName”) //buradaki $ işaretinden sonra değişken ismini yazabiliriz.

Yorum satırı için hem çift slaş // hemde b,rden fazla satırı yorum satırına alabilmek için /\* ile başlangıç /\* ile bitiş şeklinde yapılabilir.

Tipler arası dönüşüm;

3 çeşit tip dçnüşümü vardır;

1)Number to number;

Burada bir sayıyı başka türde sayıya dönüştürürüz örneğin double bir sayıyı interger dönüştümek gibi;

val doubleNum = 12.3

val intNum = doubleNum.toInt()// intNum şu anda 12 oldu

Burada toInt, toFloat … gibi çeşit çeşit ifade mevcuttur

2)Number to string;

Burada amaç bir sayıyı string ifadeye çevirmektir. Genelde string daha kapsayıcı bir tür olduğu için tercih edilir. Özellikle bir output verileceği zaman kullanılır(sayılar için)

val num = 12.3

val numString = num.toString()

3)String to number

Burada işler biraz karışır çünkü amaç bir stringden sayıya çevirmedir ve bunu yaarken eğer string içinde yanlışlıkla girilmiş bir ifade varsa(sayıya çevrilemeyecek) işlerin karışmaması için direkt olarak .toInt gibi bir ifade kullanmamız programın çökmesine sebebiyet verecektir. Bunu önlemek adına .toIntOrNull() fonksiyonu kullanılır böylece eğer çevrilemeyek bir ifade varsa null ifadesi atanmış olur değişkene.

(1.ders sonu yazısı 23.04.2025/17.34)

Standart prgoramlama yapabilmek için kullandığımız bazı yapılar bulunur bunlardan birincisi koşullarla işlem yapmaktır. Herhangi bir ifadenin değişkenin değerine göre yapacağımız işler değişebilir ve bunun kontrolünü 2 yapı ile kontrol edeibliriz. Bunların ilki;

If yapısı;

if yapısı temelde içerideki ifade doğru olduğu takdirde(true) içerdiği kod bloğuna girer ve orada ne ypaılması isteniyorsa onu yapar yanlış(false) olması durumnda ise koşul bloğuna girmeden devam eder.Birden fazla koşulu yönlendirmek için if-else if-else kullanılır.Burada 1 tane if 1 tane else ifademiz olur arada ise isteiğimiz kadar else if kullanabiliriz. Yukarıdan aşşağı doğru sırayla tüm koşullar denenir ve ilk true ifadesine ulaşılunca o bloktaki kod çalıştırılır ve alt tardındaki diğer koşullar incelenmeye gerek duyulmaz.

metin, ekran görüntüsü, yazılım, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

When yapısı;

When yapısı genellikle menü vb işlemlerin gerektiği durumlarda kullanılan tek bir ifadenin(değişkenin) aldığı değerlere göre ilgili kod parçasının çalışmasına olanak veren bir yapıdır.Çalışması yine if gibidir, eşleşme olduktan sonra alt tarafın bakmaz.

metin, ekran görüntüsü, ekran, görüntüleme, yazılım içeren bir resim

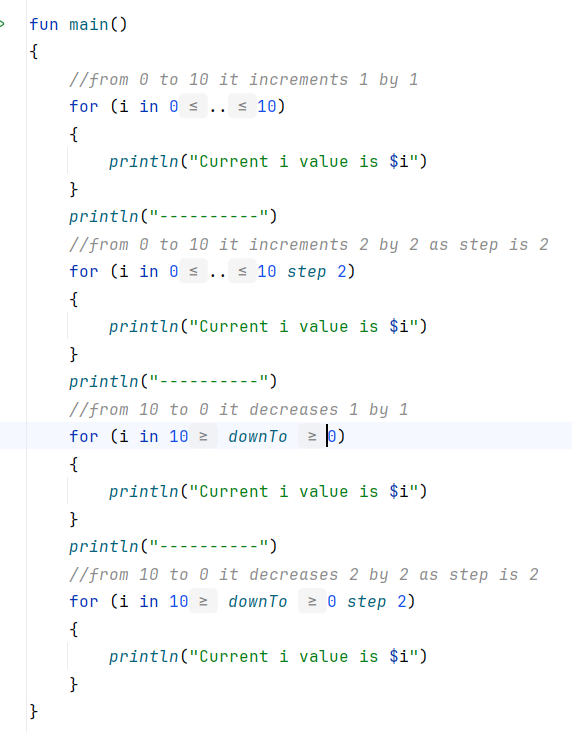
Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Standart programlama kaynak dosyalarını yine github linki üzerinden My Application kısmındaki standard\_programing paketinin aldından bulabilirsiniz

Şimdide standart programlamanın bir diğer önemli yapısı olan döngülere(loops) giriş yapalım. Burada da genel olarak biz 2 tane yapı kullanırız.Bunlardan birincisi ;

For kullanımı;

For yapısı parantez içerisine yazılan bir ifade doğrultusunda ilgili aralıkta ifadenin isteiğimiz şekilde ilerlemesi(artış-azalış) durumuna göre istenilen kodun tekrar tekrar bir döngü halinde çalıştırılmasına olanak sağlar.

metin, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Bir diğer döngü yapımız ise while döngüsüdür.Burada da while(condition) buradaki condition kısmı true oldığı müddetçe döngü sürekli çalışır

metin, ekran görüntüsü, yazılım, bilgisayar simgesi içeren bir resim

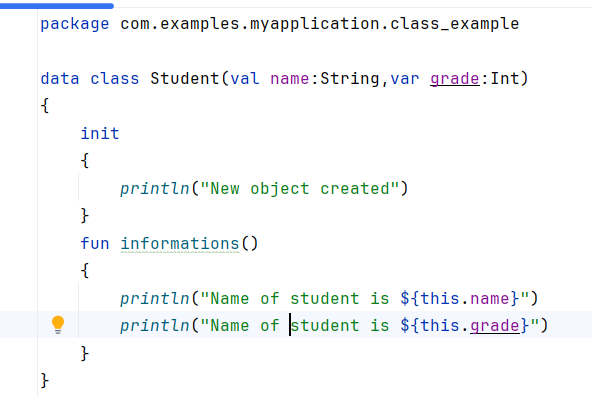
Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

metin, yazı tipi, ekran görüntüsü, beyaz içeren bir resim

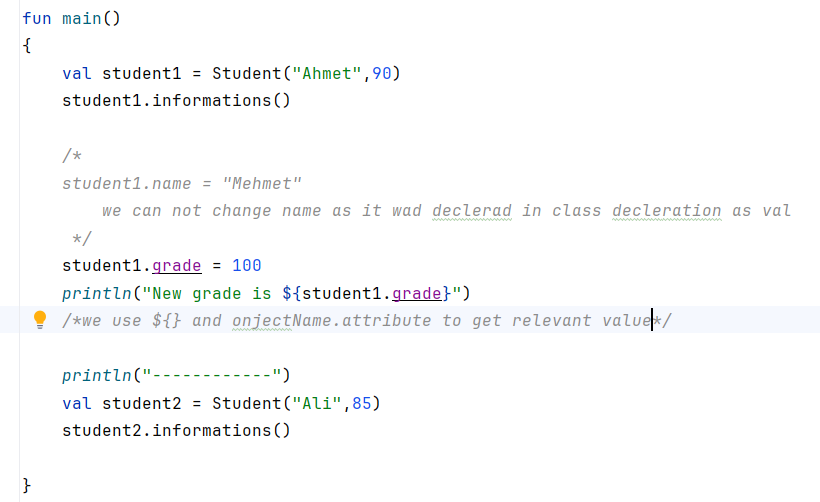
Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Genel olarak bir program yazarken bu yapıları(koşullar ve döngüler) sürekli olarak kullandığmı yapılardır.

Şimdide programlamanın en önemli olaylarından bir tanesine nesneye yönelik programlama(OOP)ya giriş yapalım. OOP kısaca nesneyi baz alarak nesne temelli uygulamalar yaptığımız bir yaklaşım, kod yazma biçimidir. OOP de en önemli unsurlardan biri doğru sınıf(class) oluşturabilmektir. Sınıf yapısını kabaca belli başlı özellikleri(attributes-data field) ve işlevleri(metod-fonksiyon) bulunan bir yapıyı toplu bir şekilde ele alabilmemizi(tutabilmemizi) sağlayan yapı olarak düşünebilriz. Ve biz sınıflardan nesneler oluşturarak onları kullanabiliriz. Bunu en iyi anlamanın yollarından birisi bir film-dizi izleme platformunu düşünelim. Karşımıza herbir film için görsel-isim-süre-puan vb bilgiler geliyor. Biz içinde bu bilgileri saklayan bir class oluşturabiliriz daha sonrasında her bir film için bu sınıftan bir nesne oluşturup onun özelliklerini atayabiliriz(görsel-isim…). Bu tip bir yapı kullanmak bize benzer özellikteki(yapıdaki) nesnelerin bilgilerini daha rahat tutabilmemiz için bir kapalı kutu(sınıf) sağladığı için ve bize herbirine ait bilgiileri atama kısmı kaldığı için işimizi daha düzenli anlaşılır yapmamızı sağlamış olacaktır.Sınıf yapısını biraz da kod üzerinde görelim. Bunun için öncelikle yeni bit Kotlin class/file kısmından bir class olşturamız gerekiyor(\*\*Unutmayınız class isimleri de file isimleri gibi büyük harfle başlamaludur.) Daha sonrasında alttaki gibi bir yapı yapmamız ilk başta yeterli olacaktır.



Buradaki data ifadesini kullanaran hem veritabanları ile kullanım da kolaylık sağlamış oluyor(ileride göreceğiz) hem de otomatik olarak getter-setter oluşturmuş gibi olduk. Kotlin modern bir dil olduğu için diğer dillerdeki(C++,Java…) constructor oluşturmadan burada görüldüğü gibi direkt parametre alarak nesne oluşturmamıza olanak saplıyor aynı zamanda eğer nesne oluşturulur oluşturulmaz(constrcutor) yapılmasını istediğimzi bir şey varsa bunu init (comes from initialize) ile sağlayabiliriz. Sınıf içinde this anahtar sözcüğünü kullanarak bulunduğnuz sınıftaki özelliğe işaret etmiş oluyoruz. Bu classın bir main içinde kullanımı ise şu şekildedir;



Burada görüldüğü gibi direkt class ismi ğzerinden bir nesne oluşturduk ve sonrasında bunun bilgilerini sınıfın içinde bulunana metodlardan birini çağırarak yazdırdık. Burada bizim name parametremiz sınıf tanımlamasında val oldığı için değiştirelmez fakat frade var olduğu için değiştirilebilir. Aynı şekilde nesnelerimizi val yaptığmız için nesneleriöiz değiştirilemez(atama-eşitleme yapılamaz).

Buradan hazır metodları işlemişken fonksiyon(class içinde olursa metod diye adlandırılır) nasıl oluşturulur birazda ona bakalım.

Fonksyionlar bizim bir işlemi(ya da işlem grubunu) tekrar tekrar her istediğimiz yerde kullanmamızı önlemek adına tek bir kere tanımlatıp ardından ihticaımız oldukça sadece fonksiyon çapırması ile o kodu çalıştıraiblmemizi sağlamak adına kullandığımız bir yapıdır. Örneğin bizim bir menümüz olsun biz kod içinde sürekli menüyü alt alta yazmaktansa tek bir yerde (fonksiyon tanımında) bir kere yazıp daja sonrasında menü yazdıracağımız zaman sadece fonksyion adı ile çağırma işlemi yapıp aynı sonıca ulaşabiliriz. Ayrıca fonksiyonların bir diğer avantajı ise bir yerde değişiklik yapmak istediğmizde her bir parçayı değiştirmke yerine sadece tanımlamadaki parçayı dğeiştirmemiz yeterli olacaktır.Unutmayınız fonksyion isimleri küçük harfle başlamalıdır.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir. metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

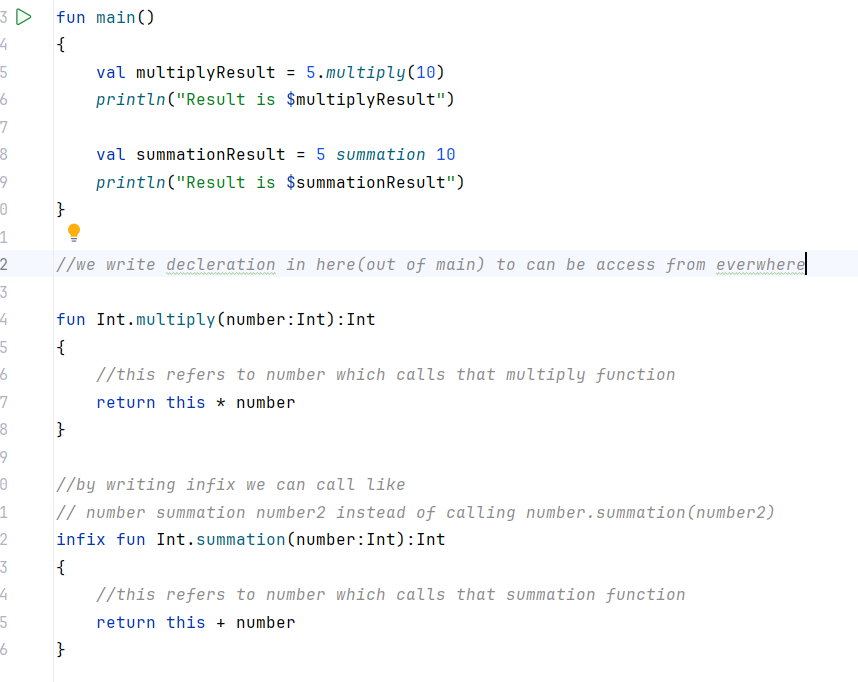
Burada görüldüğü gibi 3 farklı fonksiyon var. Bunların birincisi bir parametre almıyor ve herhangibir değer döndürmüyor. İkincisi ise String tipinde veri döndürüyor(çağrıldığı yerde kullanıcak şekilde). Üçüncüsü ise bir parametre alıyor ve bu parametreyi kullanıyor

İkinci kısımdaki fonksyion ise aynı isimli fakat aldığı paremtre tipi veya sayısı bakımından farklı oldup aynı isimle çağrılıp farklı sayı ve tipte parametre girmemize ve girilenlere göre ilgili fonksyionu çalıştırmamızı sağlayan overloading olayının uygulanmış halidir.

metin, ekran görüntüsü, yazılım, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Fonksiyonalrdaki bir diğer konuda Extension olayıdır. Extension genel olarak bizleremevcut sınıfa yeni metodlar eklememizi sağlayan yapıdır bunun nasıl yapıldığını alt taraktaki kod resminde anlatılmıştır.

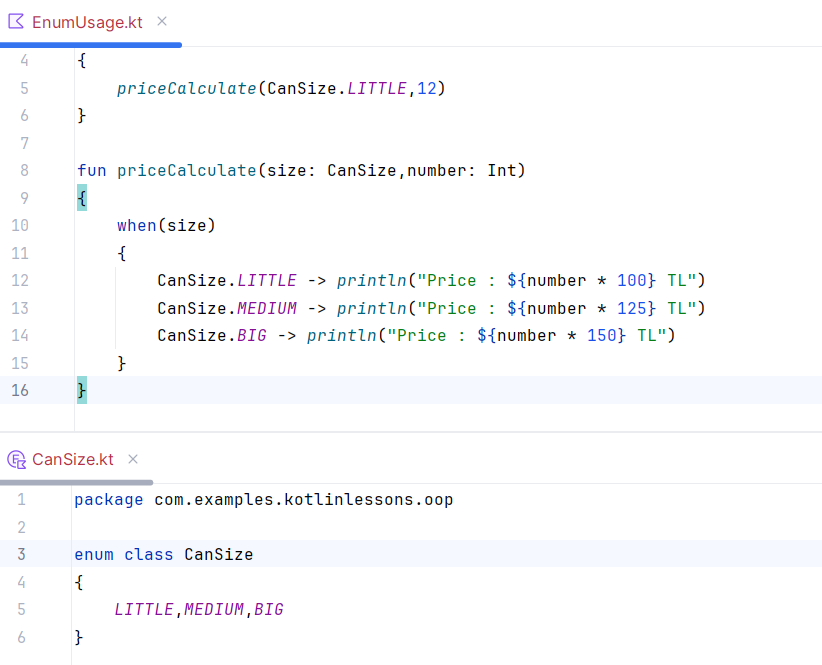


(2.ders sonu yazısı 25.04.2025/17.06)

Kendi oluşturduğumuz veya başkalarının oluşturduğu sınıfları,özellikleri vs kullanabilmek için bunların bulunduğu paketleri import etmemiz lazım. Örneğin bir paket oluşturduk ve burada ilk paketteki bir sınıftan yararlanmak için import java.util.HashMap gibi bir kullanım yapabiliriz

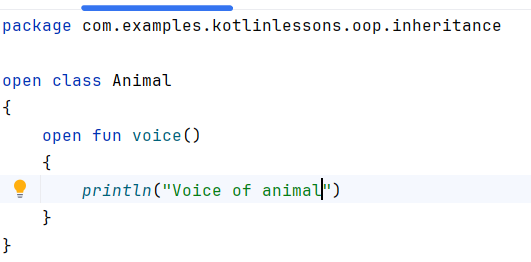
Visibility Modifier-> Oluitırduğumuz sınıflar, metodlar,özellikleri public-private-protected veya internal olarak ayırabiliriz. Burada bir özeeliğin başına hiçbir eşy yazmazsak public(her yerden direkt erişim var), private yazarsak sadece bulunduğu sınıf içinden erişim var, protected yazarsak kendisini miras alan sınıflardan erişim var, internal dersek de(çok kullanılmaz) aynı modül içindeyse erişim olanağı sağlar. Modüller paketlerden farklıdır.Birlikte derlenen ve paketlenen kodların bir setidir örnepin gradle modülü,maven modülü …

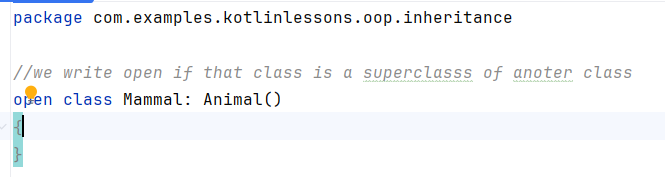
Bunlara ek olarak bazı özellikleri ve metodları statik olarak ayarlayaiblriz. Statik olması onu kullanmak için o sınıftan bir obje oluşturmadan sadece sınıfın adını kullanarak erişim sağlamamızı sağlayan yapıdır. Bunu genelde sınıftan üretilen herbir nesne için farklı bir çıktıya seebiyet vermeyecek metodlar ve her nesne için aynı değeri alacak olan(ya da toplam nesne sayısını tutmak gibi) özelliklerin tnaımlanmasında kullanırız.Statik olmayanlara sınıfın adı ile örneğin ClassA().att ya da ClassA().method() şeklinde erişim sağlanır ve burada sınıf adından sonra parantez kullanılarak her defasında farklı sanal bir nesne oluşturulur ve bunun sayesinde o özelliğe-metoda erişim sağlanır. Fakat static olanlardaysa sadece ClassA.att ya da ClassA.metod() ile erişim sağlanabilri yalnızca.Statik olacakları sınıf içerisinde companion object{} ile parantezler arasında tutarak olultururuz.Diğer dillerde genellikle statick eyword kullanılır.Statik olursa normal erişim olamaz.

Bir diğer önemli konuda enum(enumuration) konusudur.Enumuration genel olarak bizim herhangibir özelliğin alabileceği değerlere göre daha rahat işlem yapmamızı bunları daha rahat akılda kalıcı şekilde tutabilmemizi sağlayan bir yapıdır.Burada enum içerikleri tamamen büyük harflerle yazılır ve birbirinden virgüllerle ayrılır. 

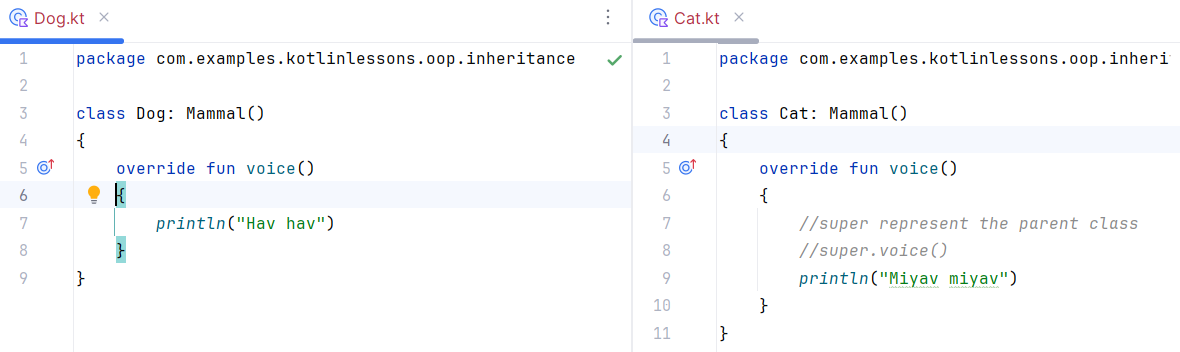
Örneğin burada basit bir şekilde bir kavanozun boyutuna göre fiyatının ne olacağını kolaylıkla hesaplamamızı sağlayan bir enum yapısı var.

Şimdi de oop’nin en önemli yapıalrından biri olan inheritance(kalıtım) konusuna giriş yapalım. İnheritance eğer bir sınıf başka bir sınıfın alt sınıfı oluyorsa(kediler memeli sınıfının altıdır aynı şekilde memeli sınıfı da hayvanlar sınıfının-aleminin bir alt sınıfıdır) alt sınıf üst sınıftan miras alıyor anlamına gelmektedir. Burada özellik aktaran sınıf(üst sınıf) parent class-super class diye adlandırılabilir. Özellik alan sınıf ise(alt sınıf) child class-subclass şeklinde adlandırılabilir. Eğer bir sınıf özellik aktarıyorsa sınıf tanımalması alttaki gibi olmalıdır ve aktardığı özellik ve metodlarda aynı şekilde alttaki gibi yazırlı



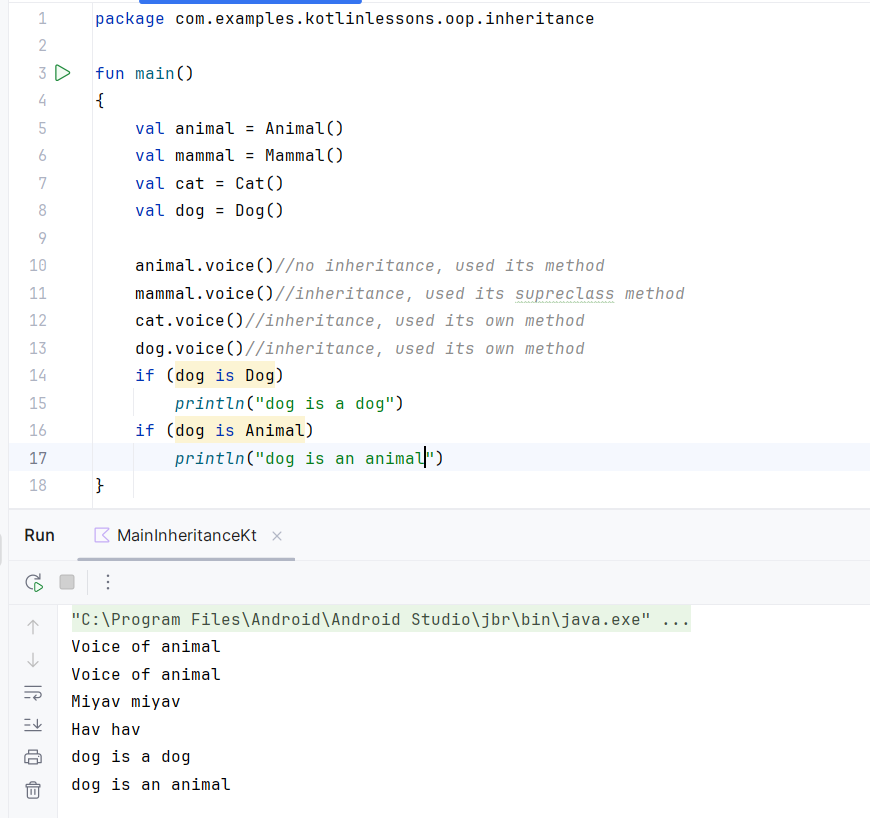
Miras alan sınıf ise tanımlamasında alttaki gibi yamalıdır ve eğer bu örnekte olduğu gibi miras aldığı bir özelliği ya da metodu kendisi override(üstüne yazma) etmezse o zaman miraas aldığı sınıftaki değerleri alır otomatikmen. 

Alt tarafta da miras alan sınıfların miras aldıkları özellikleri nasıl overide ettikleri örneği görülmektedir.Burada sınıf tanımlamasında sınıf isminden sonra parantez kullanılması önemli bir detaydır.Eğer metod üst sınıfın metodunu kullanmak isterse override etmese de olabileceği gibi isterse de override kısmında super.metod() şekliyle miras aldığı sınıftaki o metodu da çağırabilir. Burada super kelimesi parent classı temsil eder.

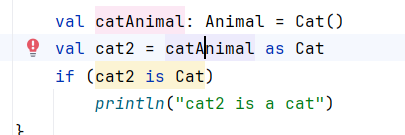


Inheritance ile alakalı çok önemli bir özellik de her sınıf sadece tek bir sınıfı miras alabilir.

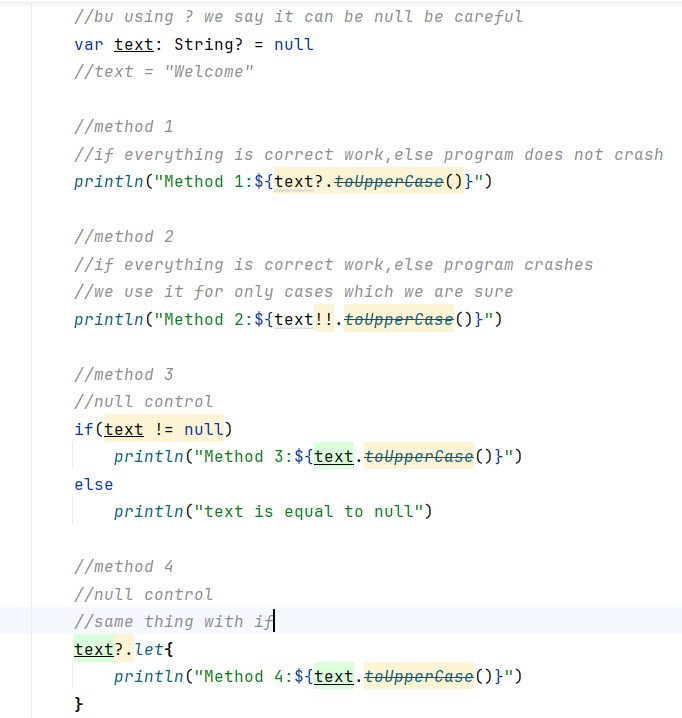
Inheritance ile alakalı bazı durumlar söz konusudur. Bunlar mesela tip kontrolü olayıdır. Tip kontrolü bir objenin bir sınıftan olup olmadığını bulmamıza olanak sağlayan bir kontroldür. Is kelimesi ile yapılır.Diğer dillerin bir çoğunda instanceof yapısı kullanılır.



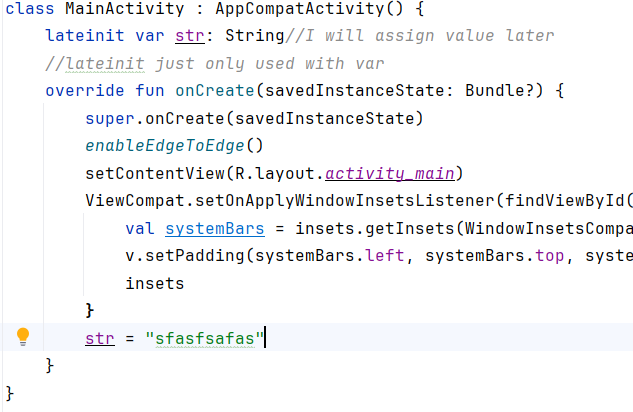
Bir diğer olayda upcasting(.ok kullanılmaz) ve downcasting işlemeridir. Upcasting bir nesneyikullanrak üst sınıfından bir nesne oluşturma işlemidir. Bu genelde problemlere yol açabildiği için(alt sınıfa ait bir özelliğin üst sınıfta mevcut olmaması durumunda yok olması gibi) kullanılmaz. Downcastin ise üst sınıftan alt sınıf nesnesi oluştırulması işelmidir bunu as keyword kullanıalrak ypaılır.



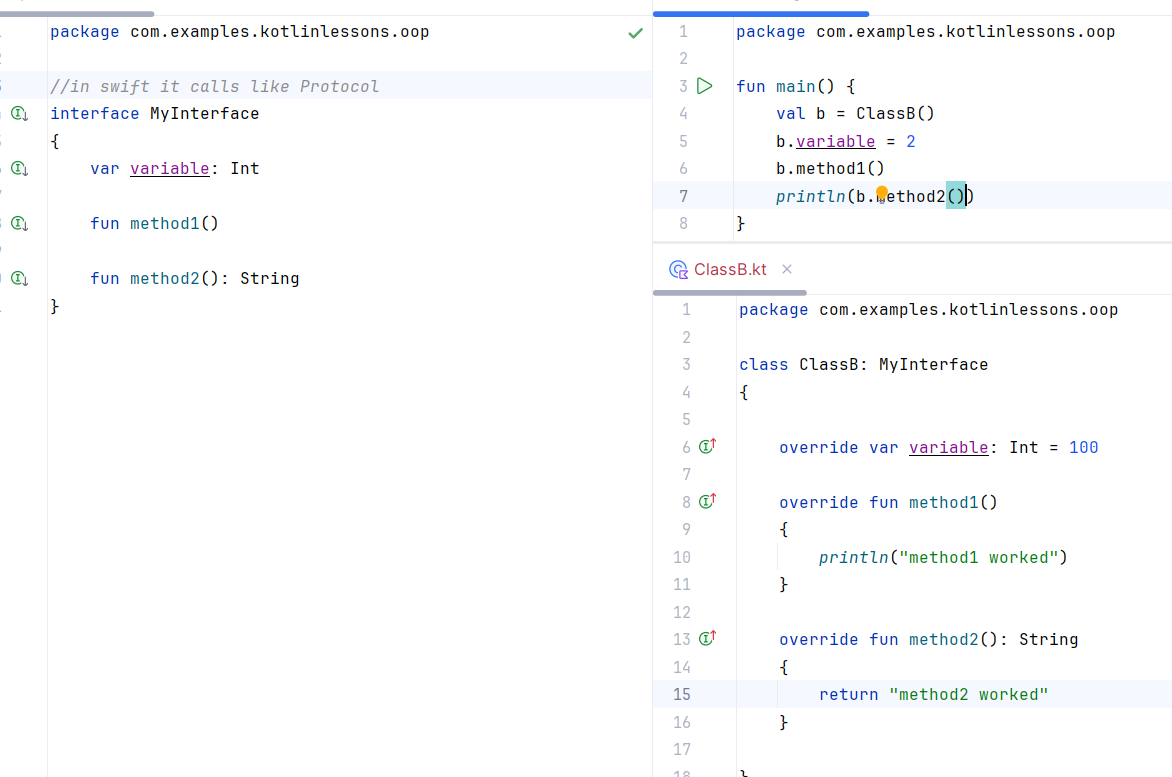
Özellikle mobil uygulama geliştirme dillerinde rastlanılan Nullable/Null/Safety gibi ifadelerle analtılan bir kontrol durumu vardur. Bu bir ifadenin null-NaN-nil olup olmamasının yani boş bir ifade mi yoksa bir içeriği var mı yok mu bunun kontrolünün sağlandığı bir yapıdrı. Genelde programlarımız bu null olayının doğru bir şekilde kontrol edilmemsinden kaynklı olarak çökerler ve biz doğru null kontrolleri ile programlarımızın çökmesinin önüne geçebiliriz. Bunu sağlamanın farklı yolları vardır.



Bir ifadeye null koymadan daha sonra değer atayabilmek için lateinit ifadesinin kullanırız. Bu primitive(Int,Double…) tiplerle kullanılamaz. String gibi tiplerle kullanılabilir. Kullanımı şu şekildedir.

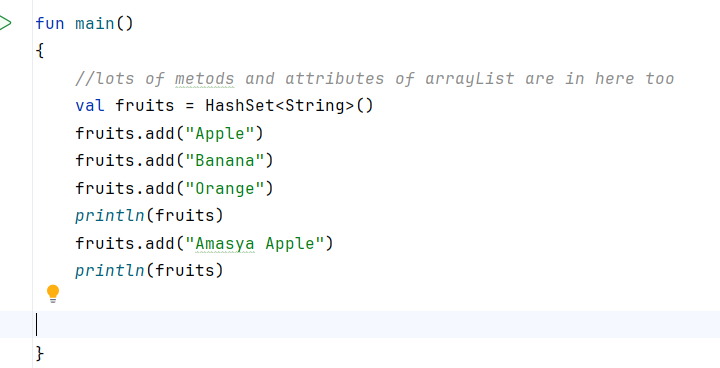


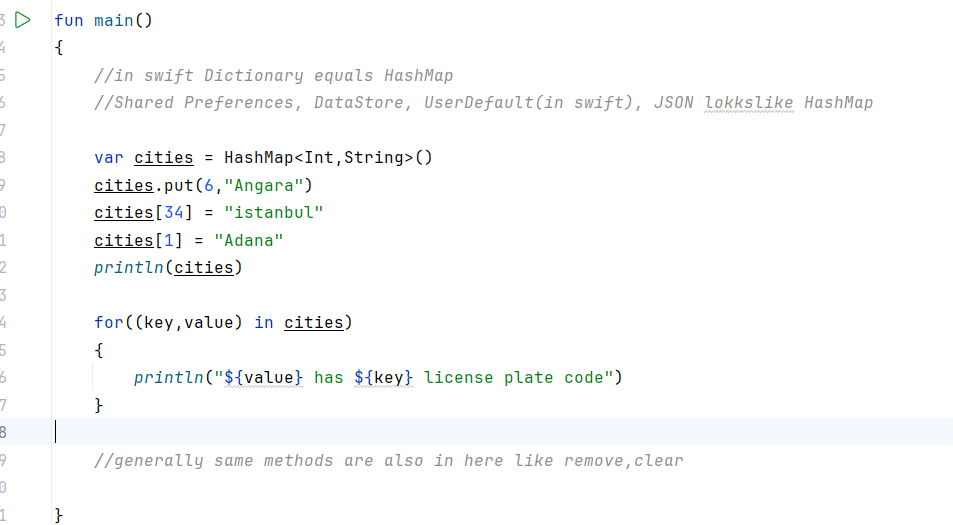
Inheritance benzer ama bazı kritik farklıllıkları olan bir diğer konuda interface konusudr. Interfaceler sınıflar için arayüzler oluştururlar ve içeriğinde sadece metodların ve özelliklerin tanımlamalarını içerirler. Ve bir sınıf bir interfacei implement ettiği takdirde onun tüm metodlarını ve özelliklerini implement etmek zorundadır. Bir sınıf birden fazla interface sahip olabilir.



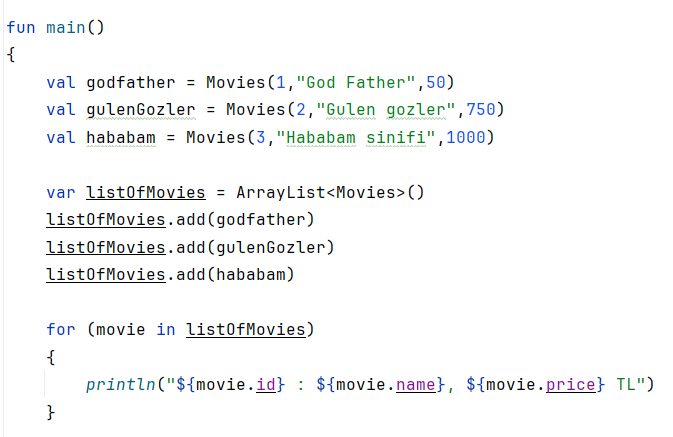
Şimdi de bir diğer önemli konumuz olan veri yapılarına giriş yapalım(Collections da diyeibliriz). Veri yapıları bir gurup değişkeni veriyi (aynı tür)bir arada tutabilmemize olanak sağlayan yapılardır.Bunlardan en çpk kullanılanı ArryaListtir, diğerleri ise HashSet ve HashMaptir.

HashSet belli bir türdeki değişkenleri bir arada tutmamıza olanak sğalayan ve bunalr üzerinde işlemler yapmamıza da izin veren bir veri yapısıdır. Temel özelliği bunları rastgele tutması ve eklediğimiz sırayla tutmamasıdır. Bu yapı genelde müzik listelerinden rastgele oynatma vb işlemler için kullanılabiliecek bir yapıdır. ArrayListin bir çok özelliğine ve metoduna bu da sahiptir. Fakat index olatı ve sıra olayı yoktur ve her bir değerden listede sadce bir tane bulunabilir



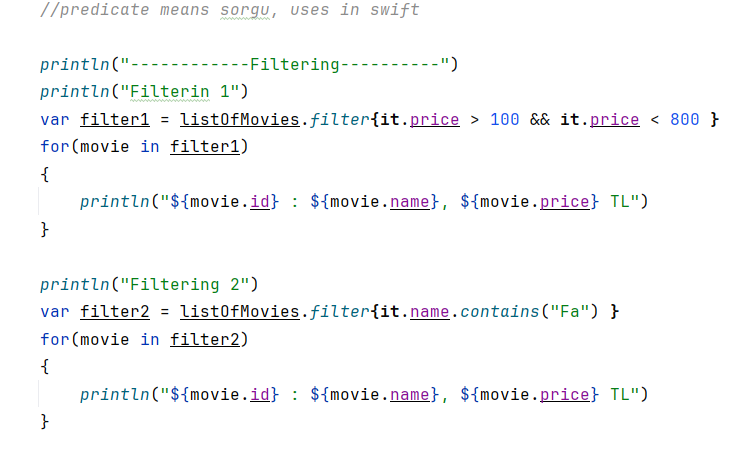
HashMap ise index yerine anahtar-değer(key-value) yapısı ile çalışıp bir anahtar dğeişken ile onun karşılık geldiği değere erişim onun üzerinde değişiklikler vs yapmamıza olanak veren yapıdır. Genellikle internet tabanlı işlerde kullanılır

ArrayList ise index mantığı ile çalışan, içeriğinde aynı ögeden birden fazla bulunmasına olanak sağlayan, eklendiği sırada elemanları tutan bir yapıdır.İçeriğinde bulunanan metodlar ile elemanları üzerinde çok çeşitli oynamalar yapabilmemize olanak sağlayan bir yapırıdr.Sıralama işlemleri yapmamıza(herhangibir özelliğe göre artan ya da azalan şekilde) olanak sağlayan metodları da vardır. Yine aynı şekilde özellik bazlı filtrelemeye olanak veren metodları da içeriğinde barındırır.





Burada özellikle ASC ve DESC terimleri ile it kelimesinin listedeki her bir elemana sırayla referans vermesi kısımları öenmlidir



Any tüm sınıfları kapsayan en üst sınıftır. Java da bunu karşılığı Object sınıfıdır. Ve tüm sınıflar doğrudan ya da dolaylı olarak Any sınıfndan türetilir.

Programlarımızı yaparken bir dieğr önemil konuda exception dediğmiz hata ayıklama işlemidir. Try-catch yapısı ile çalışan bu yapı yaptığımız herhangibir işlemde bir hata alınması durumunda kodun çökmesini engellemek amacıyla yapılan uyarıları veren bir yapıdır.Exceğtionlar runtime error diye geçer bunlara ek olarak compile error denilen compile etmemize olanak sağlamyan hatalrda mevcuttur

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

\*\*\*Buraya kador olan kısım programlamanın temellerini Kotlin dilinin yapısını syntaxını kullanrak gördüğümüz(özelikle de bizim mobil uygulama kısımlarında karşılaştığımız konuları) derslerdi. Bundan sonrasında ise artık Android uygulama geliştirmeye giriş yağıyoruz\*\*\*

Öncelikle projemizdeki bazı temel android kısımlarının neler olduğuyle başlayalım

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Buradaki MainActivity classı uygulamanın çalışma sürecindeki her şeyin yönetildiği kısımdır. Burada onCreate metodu ise uygulama çalışır çalışmaz arkada dönen işlemleri temsil eder.Burada gördüğümüz R harfi res klasörünü temsil etmektedir.Projelerimizde enableEdgeToEdge metodu ile tam ekrana geçiş yapılır bunu kaldırdığımız takdirde tam ekrandan çıkıp yukarıdaki durum çubuğunu da görmemize olanak sağlar.ViewCompat kısmı ise yine aynı şekilde tam ekran görünümlerde ekran boyutlandırılması ile ilgilenen kısımdır.

Tasarım kısımlarında bize faydalı olabilecek sayfalar;

material design: <https://m3.material.io/> ssayfası bizlere tasarım felsefeleri hakkında bilgi vermede faydalıdır.

envato: <https://elements.envato.com/> bu sayfada ise konulara bazlı örnek tasarımlar bulunmaktadır

figma: <https://www.figma.com/> bu sayfa ise tasarımlar yapmamıza olanak veren bir sayfadır

adobe color: <https://color.adobe.com/tr/> bu sayfada ise renklerin kodları birbiri ile uyumlarını görmek açısından önemli sayfalardan biridir.

\*\*Renk kodları hexadecimal şekilde yazılır ve başında # işareti bulunur.8 hane olarak yazılırsa ilk 2 hane rengin şeffaflığını temsil ederken sonraki 6 hane ise 2şerli olarak ayrılarak sırasıyla rgb değeridir(red-green-blue)

Biz tasarımlarımızın ayarlanmasındaki işlemleri res klasörünün altındaki paketleri kullanarak yapıyoruz.

Layout kısmında ekranlarımız karşımıza gelir ve ekrana(sayfaya) eklemeleri(fotoğraf-yazı…) yapmamıza bunları Attributes kısmından istediğimiz şekillerde güncellememize yardımcı olan kısımlar çıakr

Values kısmında ise kullanacağımız renklerin kodlarını, metinlerimizdeki yazıları ayarladığımız ve isimlendirdiğimiz kısım yer almaktadır

(3.ders sonu yazısı 29.04.2025/14.49)