KOTLİN GİRİŞ

[1. DEĞİŞKENLER 2](#_Toc125636687)

[1.1 CONSTANTS - SABİTLER 3](#_Toc125636688)

[1.2 DÖNÜŞÜMLER 3](#_Toc125636689)

[2. KONSOL GİRDİSİ 3](#_Toc125636690)

[3 DÖGÜLER VE KOŞULLAR 3](#_Toc125636691)

[4. HAZIR KOMUTLAR 4](#_Toc125636692)

[5. NESNE YÖNELİK PROGRAMLAMA 4](#_Toc125636693)

[5.1 FONKSİYONLAR 4](#_Toc125636694)

[5.1.1 EXTENSİON FONKSİYONLAR 5](#_Toc125636695)

[5.1.2 CONSTRUCTOR 5](#_Toc125636696)

[5.2 ACSESS MODİFİERS (ERİŞİM) 5](#_Toc125636697)

[5.2.1 STATİC METOTLAR 5](#_Toc125636698)

[5.3 ENUMERATİON 6](#_Toc125636699)

[5.4 COMPOSİTİON 6](#_Toc125636700)

[5.5 KALITIM 6](#_Toc125636701)

[5.5.1 OVERRİDE 6](#_Toc125636702)

[5.5.2 POLİMORFİZM 7](#_Toc125636703)

[5.6 TİP KONTROLÜ 7](#_Toc125636704)

[5.7 UPCASTİNG, DOWNCASTİNG 7](#_Toc125636705)

[5.8 INTERFACE 8](#_Toc125636706)

[6. COLLECTİONS 8](#_Toc125636707)

[6.1 ARRAY LİST 8](#_Toc125636708)

[6.2 SET YAPISI (HASHSET) 9](#_Toc125636709)

[6.3 MAP YAPISI (HASHMAP) 9](#_Toc125636710)

[7.STRING YAPISI 9](#_Toc125636711)

[8.EXCEPTİONS 10](#_Toc125636712)

[9.THREAD 10](#_Toc125636713)

ANDROİD GİRİŞ  
[1. ANDROİD KERNEL YAPISI 11](#_Toc125636714)

[2. ANDROİD STUDİO ANDROİD KURULUMLARI 12](#_Toc125636715)

[2.1 EMÜLATÖR 12](#_Toc125636716)

[3. ANDROİD STUDİO PROJE DİZİNİ 13](#_Toc125636717)

[3.1 MANİFEST DOSYASI 14](#_Toc125636718)

[3.2 RES KLASÖRLERİ 14](#_Toc125636719)

[3.2.1 LAYOUT DOSYASI 15](#_Toc125636720)

[3.2.2 DRAWABLE DOSYASI 15](#_Toc125636721)

[3.2.3 VALUES DOSYASI 16](#_Toc125636722)

[3.2.4 GRADLE DOSYASI 16](#_Toc125636723)

[4. ANDROİD MONİTOR VE LOGCAT YAPISI 17](#_Toc125636724)

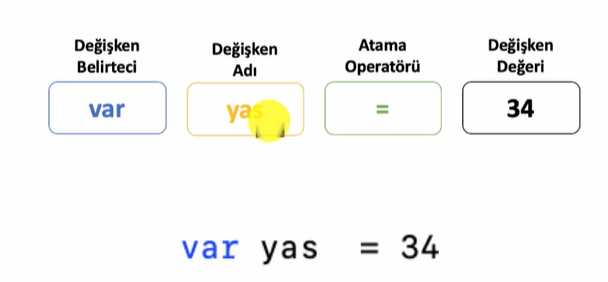
**KOTLİN GİRİŞ**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Kodlarını yazacağımız sayfa kırmızı ile işaretli olan, diğeri de tasarım sayfası.

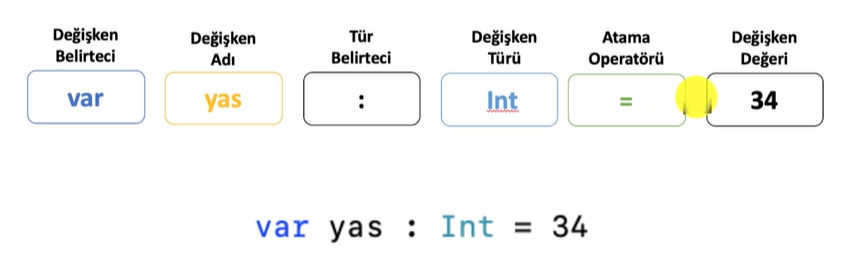
# 1. DEĞİŞKENLER



Şekil 1 Değişken Oluşturma

Var kullanılır ancak türde kullanabiliriz,

|  |  |  |  |
| --- | --- | --- | --- |
| Tam Sayılar | Ondalıklı Sayılar | Metinsel İfadeler | Mantıksal İfadeler |
| Long, Int, Shorti Byte | Double, Float | String, Char | Boolean |



Şekil 2 Tür Tanımlaması

Tür tanımlaması Şekil 3deki gibi olur, Metinler (“Metin” veya ‘Harf’ ), Tam Sayılar(34), Ondalıklı Sayılar(1.78 veya 1.78f) şeklinde tanımlanır. Type Safety özelliği vardır.

Büyük küçük harf farkı var ve @,$,% işaretleri kullanılamaz. En sonda noktalı virgül kullanılmaz !!

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

2 Adet değişken tanımlamaya ait örnek. ${İşlem} şeklinde metin içerisinde işlem yapılabilir.

## 1.1 CONSTANTS - SABİTLER

“Val” kelimesini kullanmak hafızayı rahatlatır, değişmez bir değer olmasını sağlar

(val sayi = 20, sayi = 50) yaparsak hata verir.

## 1.2 DÖNÜŞÜMLER

toIntorNull() int varsa int yap değilse null yap,

sayi?.let{ Kodlar } eğer dönüşümde sorun yoksa kodların içini yapar,

toInt, toDouble, toFloat, toString …

# 2. KONSOL GİRDİSİ

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Konsol girdisi için boşluk bırakarak girilen değerlerin tamamı okunmaz.

Scanner(System.in).nextInt() vb. şeklinde de diğer türlere çevirilebilir.

# 3 DÖGÜLER VE KOŞULLAR

metin içeren bir resim

Açıklama otomatik olarak oluşturulduİf-else her dildeki gibidir,

When, switchin değiştirimiş halidir, yanda örneği bulunmaktadır.

For kaç kere döneceği belli olan, While ise kaç kere döneceği belli olmayan döngü

metin içeren bir resim

Açıklama otomatik olarak oluşturulduNOT : 1..5, 1den 5e kadar olan elemanlar demek eğer index değerini 0 dan başlatıcaksak ***until*** kelimesi kullanılır

Step (deger) kaç kaç artacağını belirtir

(Deger) down to (deger) tersten yazdırma işlemi yapar

Break ve Continue normal yazılım dillerindeki gibidir.

# 4. HAZIR KOMUTLAR

Rastgele sayı oluşturma => Random.nextInt(0,10) // 0 ile 9 arası random sayı oluşturur.

Yukarı yuvarlama işlemi => ceil(deger) // içindeki değeri yukarı yuvarlar

Aşağıya yuvarlama işlemi => floor(deger) // içindeki değeri aşağıya yuvarlar

Karekök alma işlemi => sqrt(deger) // içideki değerin karekökünü alır

Mutlak değer alma işlemi => abs(deger) // içideki değerin mutlak değerini alır

Max ve Min işlemi => max/min(deger,deger) // içideki değerlerden büyük/küçük olanı verir

Üslü sayı alma işlemi => 2.pow(3) // 23 değerini verir

# 5. NESNE YÖNELİK PROGRAMLAMA

metin içeren bir resim

Açıklama otomatik olarak oluşturulduNull safety özelliği bulunmaktadır ve yandaki gibi kullanılır

Lateinit özelliği ise değişkene sonradan değer atanacağını belirtir. (ANCAK PRİMİTİV TİPLERDE(int vb.) KULLANILAMAZ !!!)

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

## 5.1 FONKSİYONLAR

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 3 Geriye değer döndüren fonksiyon Şekil 4 Geriye değer döndürmeyen Fonksiyon

|  |  |
| --- | --- |
| Overloading – Overriding Java dilindeki gibidir ancak Vararg diye bir parametre vardır.  Parametre alırken Vararg ile değişken tanımlarsak birden fazla değer alabiliyoruz. Yani javadaki …int yerine geçer. | metin içeren bir resim  Açıklama otomatik olarak oluşturuldu |

### 5.1.1 EXTENSİON FONKSİYONLAR

Extension fonksiyonlar ile sınıfları genişletebiliriz.

|  |  |
| --- | --- |
|  |  |
| Bu şekilde carpi fonksiyonunu bir değişkenle çarpıp kullanabiliriz, Türü genişletmiş oluruz | Extensionun daha pratik kullanım halidir |
|  |  |

### 5.1.2 CONSTRUCTOR

|  |  |
| --- | --- |
| Constructore yandaki gibi direk class içerisine parametre verilerek tanımlanır.  Aynı zamanda init özelliği ile nesne oluşturulduğu anda yapılmasını istediğimiz kodları çalıştırırız. | metin içeren bir resim  Açıklama otomatik olarak oluşturuldu |

## 5.2 ACSESS MODİFİERS (ERİŞİM)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MODİFİER | PUBLİC | INTERNAL | PRİVATE | PROTECTED |
| TANIM | Sınıf, Metot, Değişken | Metot, Değişken | Metot, Değişken | Metot, Değişken |
| ERİŞİM | Bütün sınıflar ve paketlerden | Yalnızca bulunduğu modül içine yer alan sınıflardan | Yalnızca tanımlandıkları sınıftan | Bulundukları sınıftan veya extend edildiğinde |

Java vb. dillerde private değişkenlere erişmek için kullandığımız getter, setter, hashcode, equals vb. şeyleri Kotlinde classın önüne data ekleyerek yapabiliriz.

### 5.2.1 STATİC METOTLAR

|  |  |
| --- | --- |
| Static bir metot oluşturup nesne oluşturmadan kullanmak istediğimiz zaman “companion object” kullanarak metota ve içideki değişkenlere erişebiliriz. Performansı etkileyebileceğinden dolayı çok sık kullanmamak gerekir !!! |  |

## 5.3 ENUMERATİON

|  |  |
| --- | --- |
| Genellikle verileri eşleştirirken kullanılır. Tanımlanması yandaki gibidir. Erişim için “KonserveBoyut.Kucuk” yazmamız yeterlidir. |  |

## 5.4 COMPOSİTİON

|  |  |
| --- | --- |
| Bir sınıfa başka bir sınıftan değişken yollamak istediğimizde kullanırız.  Genellikle veri tabanı işlemlerinde kullanılır. | metin içeren bir resim  Açıklama otomatik olarak oluşturuldu |

## 5.5 KALITIM

|  |
| --- |
|  |
| Bir class sadece bir classtan türetilebilir. Superclass open ile tanımlanır ve subclass, superclassın constructure özelliklerini almak zorundadır. |

### 5.5.1 OVERRİDE

|  |
| --- |
|  |
| Kalıtım ilişkisindeki alt sınıfların, üst sınıfların metotlarını tekrar kullanmasıdır. Kullanılacak metota da open kelimesi eklenmelidir. |

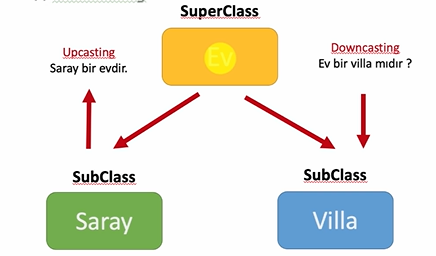
### 5.5.2 POLİMORFİZM

|  |  |
| --- | --- |
| Bazen bazı fonksiyonlar bizden Hayvan türünde değişken ister biz de Köpek türünü Hayvana dönüştürüp yollarız bu amaçlar kullanılır.  Görünümü Hayvan gibi ama içerik olarak Köpek gibi davranır. |  |

## 5.6 TİP KONTROLÜ

|  |
| --- |
|  |
| Bir sınıfın hangi sınıftan türetildiğini veya hangi tip olduğunu kontrol eder, “kontrolEdilecekDeğişken is kontrolEdilecekTür” şeklinde kullanılır |

## 5.7 UPCASTİNG, DOWNCASTİNG



Yukarıdaki örneğe göre upcasting ve downcasting işlemi aşağıdaki gibi yapılır.

|  |  |
| --- | --- |
|  |  |
| Tanımlama yapılırken yukarıdaki gibi yapılırsa upcasting olur. | Tanımlama yapıldıktan sonra yukarıdaki gibi “as” ile kullanılırsa downcasting olur. |

## 5.8 INTERFACE

|  |  |
| --- | --- |
| Bir class birden fazla interface alabilir eğer birden fazla ise interfaceler arasına “,” koyularak tanımlanır,  Interfaceler bir taslak gibidir içindeki metot, değişken vb. her şey implement edildiği sınıfta kullanılmak zorundadır. |  |

# 6. COLLECTİONS

## 6.1 ARRAY LİST

|  |  |
| --- | --- |
| Yandaki gibi tanımlanır |  |

Belirli kodları vardır bunlar;

* ***Liste.add(“Eklenecek string veri”)*** // Eleman ekler,
* ***Liste.get(index) veya liste[index***] // O indexteki elemanı alır,
* ***Liste.isEmpty()*** // ArrayList boş mu?,
* ***Liste.count() ve Liste.size*** // ArrayList boyutunu verir,
* ***Liste.first() ve Liste.last()*** // İlk ve son elemanı getir,
* ***Liste.contains(“Eleman”)*** // O eleman ArrayListte var mı?,
* ***Liste.max() ve Liste.min()*** // Harf sırasına göre en büyük ve en küçük elemanı getir,
* ***Liste.sort()*** // ArrayListi sıralar,
* ***Liste.reverse()*** // ArrayListi tersine sıralar,
* ***Liste.removeAt(index) veya Liste.remove(“Eleman”)*** // ArrayListteki elemanı siler,
* ***Liste.clear()*** // ArrayListi temizler,

Döngüler ile kullanırken;

For(meyve in meyveler) veya for((index,meyve) in meyveler.withIndex()) şeklinde kullanılır

Özel fonksiyonlar;

* ***Liste.sortedWith(compareBy({it.ad}))*** // Ada göre küçükten büyüğe sıralar it listeyi gösterir,
* ***Liste.sortedWith(compareByDescending({it.no}))*** // No’ya göre büyükten küçüğe sıralar,
* ***Liste.filter{(it.ad).contains(“a”)}*** // Listede adı a içeren nesneleri getirir.

## 6.2 SET YAPISI (HASHSET)

|  |
| --- |
|  |
| İçine eklenen veriler sırasız eklenir, içindeki değerden 1 adet olabilir yani değerler uniquedir. ***Mutable*** ile ***HashSet*** üzerinde değişiklik yapılabilen, ***SetOf*** ise sadece okunabilen listedir.  iller.elementAt(1) dersek 1.Indexteki elemanı verir. Diğer parametreleri ArrayListteki gibidir. |

## 6.3 MAP YAPISI (HASHMAP)

|  |
| --- |
|  |
| İçine eklenen veriler key value ilişkisi içerisinde eklenir, ***Mutable*** ile ***HashMap*** üzerinde değişiklik yapılabilen, ***MapOf*** ise sadece okunabilen listedir. |

* ***İller.put(KEY, VALUE)*** şeklinde ekleme yapılır eğer aynı keyde kayıt varsa güncelleme yapar,
* ***İller.get(KEY)*** şeklinde okuma yapılır,
* ***İller.containsKey(KEY) veya iller.containsValue(VALUE)*** şeklinde içerisinde arama yapıp true false döndürür,
* Diğer parametreler ArrayListteki gibidir.

# 7.STRING YAPISI

|  |  |
| --- | --- |
| Yandaki gibi 3 tırnak ile tanımlama yapılırsa içerisine yazılan şekilde konsola yazdırır. trimIndent methodu kırmızı ile gösterilen yerdeki boşlukları kaldırır. |  |

Val string = “Merhaba” şeklinde tanımlanan bir değişken için;

* string.subSequence(0,3) ifadesi kelimenin “Mer” kısmını alır,
* string.Contains(STRİNG) içerisinde bir ifade aratır,
* string.toUpperCase() ya da string.toLowerCase() ifadenin harflerini büyütme/küçültme yapar,
* string.split(“a”) kelimeyi her a harfinden sonra bölüp bir diziye doldurur,
* string.trim() ifadenin önündeki arkasındaki boşlukları alır

# 8.EXCEPTİONS

Derleyici (Compile) Hatası : Derleme öncesi kodlamadaki hatalar.

Hata (Exception) : Çalışma anında gerçekleşen hatalar.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu  
Try Catch Bloğu Runtime Exceptionlar için kullanılır.

# 9.THREAD

Tek bir program akışında birden fazla işlem yapmaya yarar.

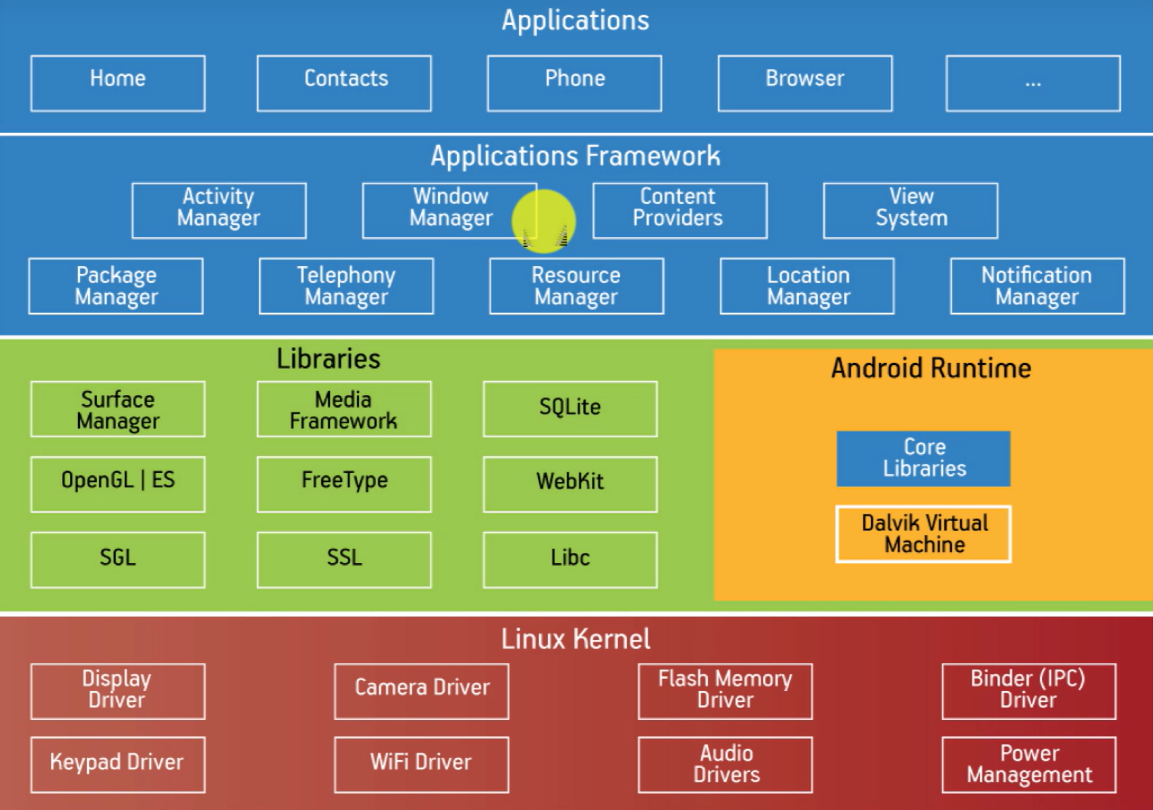
metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

İçerilerinde run metotu vardır onu override ederek içerisine yapılacak işlemleri yazabiliriz.

**ANDROİDE GİRİŞ**

# 1. ANDROİD KERNEL YAPISI



# 2. ANDROİD STUDİO ANDROİD KURULUMLARI

## 2.1 EMÜLATÖR

metin, ekran görüntüsü, ekran, siyah içeren bir resim

Açıklama otomatik olarak oluşturuldu

Eğer bilgisayarınız yavaşsa Size ve Resolution düşük seçmeliyiz. PlayStore olan bir cihaz seçmeliyiz

metin, ekran, ekran görüntüsü, siyah içeren bir resim

Açıklama otomatik olarak oluşturuldu

İşletim sistemi seçme işlemi mümkün olduğunca en güncelini seçmeliyiz.

# 3. ANDROİD STUDİO PROJE DİZİNİ

|  |  |
| --- | --- |
| AndroidManifest uygulama izinlerinin alındığı, Android sayfalarının tanımlandığı dosya |  |
| Projeyi oluştururken isimlendirdiğimiz dosya dizini,  androidTest kısmı arayüzün test edildiği,  test ise kodların test edildiği dizindir.  BuildConfig ise projenin bilgilerinin tutulduğu yer değişiklik yapmıyoruz |  |
| Res dosyası uygulamamızdaki kaynakları temsil eder bunlar;  Drawable uygulama içerisindeki resimleri,  Layout tasarımlarımızı,  Mipmap uygulama ikonlarını,  Values uygulama içerisindeki renk, yazı ve stilleri temsil eder |  |
| Proje içerisine kütüphane eklenecekse yazılacak kodların yazıldığı dosya | metin içeren bir resim  Açıklama otomatik olarak oluşturuldu |

## 3.1 MANİFEST DOSYASI

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. XML ile kodlanmıştır
2. Mavi kutulu kısım uygulamamızın izinlerini gösteriyor.
3. Kırmızı kutulu kısım uygulamamızın sayfalarını ifade eder her sayfa için bir activty açmamız gerekiyor.
4. @ ifadesi dizinlerde bir dosya bulmak için kullanılır.(HTML’deki “../” mantığı)

## 3.2 RES KLASÖRLERİ

|  |  |
| --- | --- |
| Res dosyalarına erişmek için;  Kotlin kodlarındaysan “R.values.styles” şeklinde,  XML dosyasyında isek “@string/app\_name” şeklinde ulaşılır. |  |

## 3.2.1 LAYOUT DOSYASI

metin, ekran görüntüsü, ekran içeren bir resim

Açıklama otomatik olarak oluşturuldu

> Tasarıma ekleyebileceğimiz nesneleri gösteren bir palet,

> Tasarımda var olan nesneler,

> Nesnenin özelliği var ise tıklandığında özelliklerine erişim sağlayabileceğimiz pencere,

> Tasarım ekranımız, Blueprint ekranı ise nesnelerin boyutlarıyla birlikte daha özel bir görünüm.

## 3.2.2 DRAWABLE DOSYASI

|  |  |
| --- | --- |
| metin, ekran, ekran görüntüsü içeren bir resim  Açıklama otomatik olarak oluşturuldu |  |

Drawable dosyasına harici bir dosyadan resimleri kopyalayıp yapıştırabileceğimiz gibi Android Studio içerisinde bulunan resimlerden de yukarıdaki aşamaları uygulayarak yararlanabiliriz.

## 3.2.3 VALUES DOSYASI

1. ***Colors*** uygulamamızın temel renklerini tanımladığımız XML dosyası,
2. ***Styles*** dosyası güncellemeyle birlikte themes dosyasına döndü içerisinde iki adet Light ve Dark temayı bulunduran xml dosyaları bulunmakta içinde yapısı Material Design derslerinde öğrenilecek,
3. ***String*** ise flutterda yaptığımız sabit metinleri tanımladığımız dosya, mesela 50 butonda kaydet yazacaksa bunları bir stringe bağlarız daha sonra değişiklik yapılacaksa ordan değişiklik yaparız.

* Aynı zamanda birden fazla dil kullanılacaksa da kolaylık sağlar,
* Kotlin sınıfından erişmek için R.string.textAdi şeklinde
* Activity içerisinden erişmek için getResources().getString(R.string.textAdi)

## 3.2.4 GRADLE DOSYASI

|  |  |
| --- | --- |
| İçerisinde kütüphaneleri ve onların kodlarını bulunduran dosya | metin içeren bir resim  Açıklama otomatik olarak oluşturuldu |
| metin içeren bir resim  Açıklama otomatik olarak oluşturuldu | metin içeren bir resim  Açıklama otomatik olarak oluşturuldu |
| Kütüphane eklenecekse depencies kısmına implementation yapılır. | Uygulama buradaki içeriğe göre derleniyor |

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

* ***compileSDK*** kısmında api sürümünü görüyoruz,
* ***applicationID*** paket adı Google Play’e yüklerken kontrol edilen ad,
* ***minSDKVersion*** minimum api seviyesi,
* ***targetSDKVersion*** hedeflenen api,
* ***versionCode*** yazılımcılar için versiyon,
* ***versionName*** kullanıcının gördüğü versiyon,
* ***buildTypes*** ***release*** uygulamanın apk veya aab sürümüne dönüştürüldüğü andır,

## 4. ANDROİD MONİTOR VE LOGCAT YAPISI

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

* Logcat yapısı yukarıdaki gibi çalışır hatalarımızı kontrol altında tutmamıza yarar,
* Log.v (verbose) gerekli gereksiz tüm bilgiler,
* Log.d (debug) hata ayıklamaya dönük, metodun ya da sınıfın gidişatına ilişkin bilgiler,
* Log.e (error) hata oluşan yerlere ilişkin bilgiler,

metin, ekran görüntüsü, ekran, iç mekan içeren bir resim

Açıklama otomatik olarak oluşturuldu

Android Monitör ise yukarıdaki gibi Android cihazımızın verilerini takip edebildiğimiz bir kısım.

**TASARIMA GİRİŞ**

Layout uygulamanın her bir ekranında bulunması gereken görsel sayfalardır.

|  |  |
| --- | --- |
| Nesnelerin yükseklik ve genişlik değerleri |  |
| Nesnelerin hizalanması |  |
| Nesnenin iç(padding) ve dış(margin) boşluğu |  |

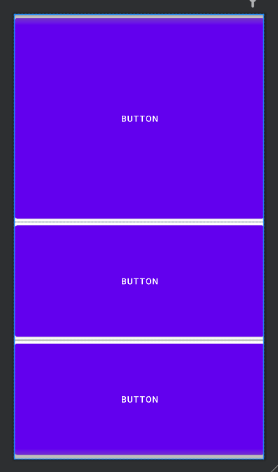
DP – Density Pixels

Uygulamamızın responsive olabilmesi için tasarlanmış boyutlandırma ölçeği, Ekran dokunuşlarında yeterli alanı sağlamak için önerilen parmak genişliği 50dp’dir.

SP – Scale Pixels

Responsive olmayan boyutlandırma ölçeğidir, genellikle font boyutu tercihinde kullanılır.

# 1. LİNEAR LAYOUT

Lineer layout nesneleri dikey veya yatay olarak sıralayabildiğimiz bir yapıdır.

Tasarım altyapısı olarak tanımlanır, Oluşturmak için res>layout klasörüne sağ tıklayıp “new > layout resources file” seçeğini seçmemiz ardından root element kısmına Lineer Layout yazmamız gerekiyor.

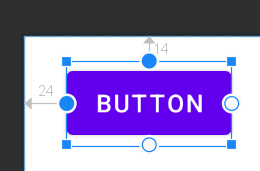
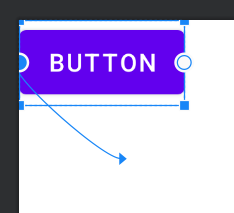
Oluşturduktan sonra Component Treeden Lineer Layoutu seçerek orientation kısmından horizontal (dikey) veya verticle (yatay) seçimi yapabiliriz.

Lineer Layoutun gravitiy özelliği ile nesnelerin merkez noktasını değiştirebiliyoruz.

Yandaki görüntüyü elde ettiğimiz dosya Layouts -> lineer\_layout dosyasıdır.

# 2. RELATİVE LAYOUT

Görsel nesnelerin aralarındaki ilişkiyi baz alarak çalışan bir yapıdır, nesneyi istediğimiz yere yerleştirebiliriz.



Nesneyi koyduktan sonra nereye göre hizalama yapacaksak oraya doğru yuvarlaktan çekerek resim 1 deki gibi kenarlara ya da nesnelere bağlamamız gerekir.

|  |  |
| --- | --- |
|  | Eğer butonları şekildeki gibi birbirlerine bağlarsak birlikte hareket ederler aralarındaki mesafe sürekli eşit olacak şekilde, ilişkilenmiş olurlar. |

|  |  |
| --- | --- |
| Android:layout\_below=”Nesnenin ID’si”  Android:layout\_toRightOf=”ID”  Android:layout\_above=”ID” | Below Altına,  toRightOf sağına,  Above üstüne, Hizalama demektir parantez içerisinde verdiğimiz nesneyi referans alarak hizalar. |
| Android:layout\_alignParentRight=”true”  Android:layout\_alignParentBottom=”true”  Andorid:layout\_alignParentLeft=”true”  Andorid:layout\_alignParentTop=”true” | Sağa, Aşağıya, Sola, Yukarı hizalama komutlarıdır fakat parent nesneyi referans alarak hizalama yapar. |

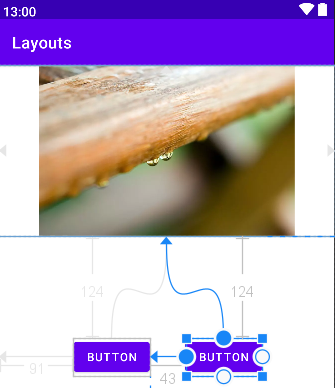
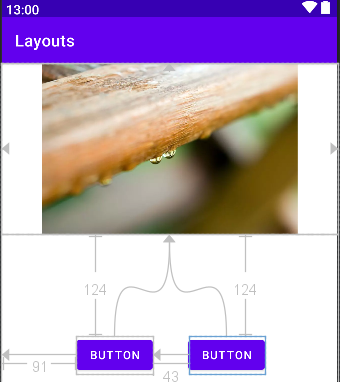
# 3. CONSTRAİNT LAYOUT

En güncel, En çok kullanılan ve Android tarafından önerilen bir yapıdır

## 3.1 SABİTLEME

Bir nesne ekledikten sonra onun orada kalabilmesi için sabitleme yapmamız gerekir, eğer sabitleme yapılmaz ise uygulama çalıştığında kendini otomatik olarak sol üst köşeye hizalar.

Hizalama işlemi 2 Şekilde yapılır, ilki aşağıdaki 2.resimdeki daireleri kullanarak sabitleme yapmamız,



|  |  |
| --- | --- |
| 2. yöntem ise sağdaki resimdeki gibi nesneyi seçtikten sonra attributes kısmındaki bu daireleri kullanmamız |  |

Aynı şekilde kaldırma işlemi de 2. Yöntemdeki çarpıya basılarak yapılabileceği gibi ilk yöntemdeki dairelere veya oklara tıkladıktan sonra delete tuşuna basarak da kaldırabiliriz.

## 3.2 BOYUTLANDIRMA

|  |  |
| --- | --- |
| Boyutlandırma yapmak için nesneyi seçip yanda gösterilen width ve height değerlerinde oynama yapabiliriz veya nesneyi seçtikten sonra köşesinden tutup büyütüp küçültebiliriz |  |

## 3.3 SANAL ÇİZGİLER

|  |  |
| --- | --- |
| Tasarımımızı yaparken bir taslak oluşturduğumak için sanal çizgiler oluşturarak nesneleri o çizgilere sabitleyebiliriz, sanal çizgiler boş sayfaya sağ fare tuşu > Add Helpers > Add … ile eklenebilir. |  |

## 3.4 ZİNCİRLEME

|  |  |
| --- | --- |
|  |  |
| Nesneleri birbirleri arasında eşit boşluklar olacak ve kenarlara eşit mesafeler olacak şekilde ayarlamak için nesnelerin hepsini seçtikten sonra Chains > Create … ile zincirleme işlemi yapabiliriz. | |

## 3.5 LOGİN SCREEN UYGULAMASI

Öncelikle uygulamanın üst barını kaldırmamız gerekiyor bunun için;

Res > values > themes > themes.xml dosyasına girip şu kodları yazmamız gerekiyor;

|  |  |
| --- | --- |
| <item name=”windowTitle”>true</item>  <item name=”windowActionBar”>false</item> | İlk komut başlığı ikinci komut ActionBarı kaldırır. |

Ardından activity\_main.xml yani layout (tasarım) dosyamızın kod kısmına gidiyoruz constraitLayout’un kodlarına şu kodları yazıyoruz;

|  |
| --- |
| Android:background=”Renk” |

Bu kod ile arkaplan rengini ayarlamış oluyoruz.

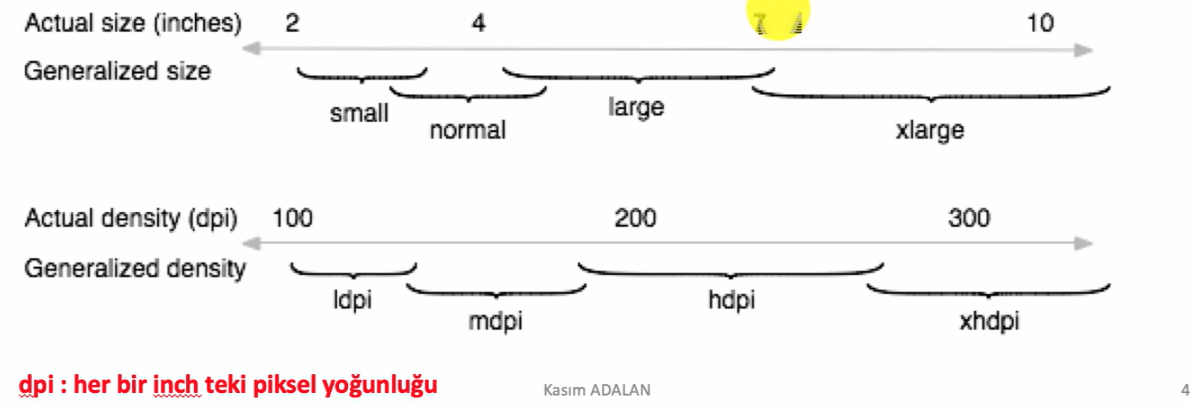
Ardından drawable dosyasına bir vector asset oluşturarak resmimizi seçiyoruz, ardından tasarımımıza bir imageView ekleyerek o resmi seçip sabitleme işlemlerini yapıyoruz.

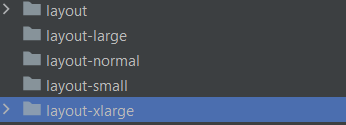
Ardından paletten “Plain Text” seçip ekrana ekliyip sabitleme işlemlerini yapıyoruz.

# 4. ÇOKLU DİL DESTEĞİ

|  |  |
| --- | --- |
| metin içeren bir resim  Açıklama otomatik olarak oluşturuldu | metin içeren bir resim  Açıklama otomatik olarak oluşturuldu |
| Yandaki gibi hangi dili kullanıcaksak o dil ile ilgili bir values dosyası açıp onun içerisine değerleri yukarıdaki gibi tanımlamamız gerekiyor. |  |

# 5. ÇOKLU EKRAN DESTEĞİ (RESPONSİVE)



Çoklu dil desteğinde olduğu gibi layout klasörünü çoğaltıyoruz, yukarıdaki resimde alttaki ölçek çözünürlük değeri üstteki ise inç değeridir, çok farklı çözünürlük olduğu için artık çoğunlukla inçe göre tasarım yapılıyor.

Small klasörü genelde oluşturulmuyor nedeni ise 2inç olduğu için tasarım yapmanın zor olması ve genelde saatlerin arayüzü olarak kullanılır.

Eğer bir cihaz herhangi bir inç değerine uymaz ise normal layout klasöründeki tasarım çalışacaktır.

Bu şekilde layout klasörleri oluşturarak içlerine tasarımları koyup her ekran için farklı uygulama tasarımı yapabiliriz.

# 6. LOGO OLUŞTURMA

|  |  |
| --- | --- |
|  | * Yandaki gibi klasörler içerisinde her çözünürlük için farklı bir logo vardır, en üstteki klasör ise API 26 üzerindeki herhangi bir telefonda çalışacak olan vektör logoyu ifade eder. (Vektör logolar kendi kendilerini çözünürlüğe göre büyültüp küçültürler) * Bu dosyaların içerisindeki boyutlara göre logomuzu kendimiz de oluşturabiliriz veyahut Android studio içerisindeki İmage Assets seçeneğinden de oluşturabiliriz. * Ardından Manifest dosyası içerisindeki AndroidManifest içerisinde de aşağıdaki kodları ekliyoruz |