Activity

Activity sınıfı tarafından oluşturulan nesneler uygulama içinde kullanılan ekranları temsil ederler. Kullanıcı yeni bir ekrana geçtiğinde **Context.startActivity** metoduyla bu ekran başlatılır. Eğer yeni **Activity, AndroidManifest** dosyasında tanımlı değilse uygulamamız çökecektir.

Activity

```
<activity
        android:name=".MainActivity"
        android:label="@string/app_name" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category
    android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
     </activity>
```

Activity

Her uygulamanın **LAUNCHER** olarak tanımlı bir Activity sınıfı olmalıdır. Bu şekilde tanımlanmış bir Activity, uygulama ilk açıldığında otomatik olarak harekete geçer ve kullanıcının karşısına çıkan ilk ekran olur.

Activity

Activity dosyaları ilk çalıştıklarında **onCreate()** metodu devreye girer. Bu metotta genel olarak **setContentView()** metodu çalıştırılarak bir ”layout” dosyasından ekran tasarımı yüklenir.
Eğer ekran ilk oluştuğunda tanımlanması gereken başka değişkenler ve aksiyonlar varsa, onlar da **onCreate()** metodu içinde gerçekleştirilebilirler.

Activity

```
@Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);
  }
```

Activity

onCreate() metodu dışında bir Activity'nin yaşam döngüsü içerisinde başka metotlar da harekete geçirilir. Bu metotları sıralarsak;

- **onCreate() :** Activity ilk oluşturulduğunda çağırılır. Activity herhangi bir nedenden ötürü yok edilmezse (bellek ihtiyacı ya da **finish** metodu) bir daha çalıştırılmaz.
- **onStart() :** onCreate metodu çalıştırılıp görsel öğeler oluştuktan sonra çağırılır. (TextView, EditText, Button, …)

Activity

- **onResume() :** Activity herhangi bir sebepten dolayı durdurulduysa (başka bir ekrana geçiş) tekrar ekrana geldiğinde bu metot harekete geçer. Activity çalıştırılmadan önce harekete geçen son metottur. Bu metottan sonra belirlenen koda göre Activity yaşamına başlar.

Activity

- **onPause() :** Activity bir sebepten dolayı arka plana atılırsa harekete geçer. Bu **Back** tuşuna basılmasından ya da sistemin bellek ihtiyacından kaynaklanabilir. Bu metot içerisinde anlık bilgilerin kaydedilmesi önerilir. **onPause()** işlemi hızlı çalışmak zorundadır, çünkü buradaki işlem bitmeden bir sonraki ”Activity” harekete geçmez.

Activity

- **onStop()** : Activity arka plana atıldığı anda işleme girer. Bu noktada iki seçenek vardır; ya kullanıcı yeni ekrandaki işini bitirip geri gelir ya da Activity tamamen kapatılır. Kullanıcı geri geldiği takdirde **onRestart()** metoduna geçilir ve döngü **onStart()** metodundan tekrar başlar. Eğer kullanıcı geri gelmeyecekse **onDestroy()** metodu çalıştırılır.

Activity

- **onDestroy :** Activity'e ait bütün kaynaklar yok edilir. Activity bu noktada yaşam döngüsünü tamamlamıştır.

Activity

- Bu metotlar bir Activity'nin yaşamını kontrol etmemize yardımcı olur. Özellikle kullanıcı bir ayarı kaydediyorsa ya da oyun oynuyorsa ve bu sırada bir telefon konuşması başlayacaksa anlık durumları kaydetmemiz için bu metotların çalışma esaslarını bilmeliyiz.

Activity

- Herhangi bir ekranın *Activity* sınıfı tarafından oluşturulması yeterlidir. Ancak bunun yanında Android işletim sistemi, sık kullanılan ekranlara özel Activity sınıfları sağlar. Bunlara göz atarsak;

Activity

- **ListActivity :** Bir veritabanındaki veriyi listelemek ya da sunucudan gelen veriyi kullanıcıya göstermek, uygulamalarda oldukça sık rastlanan işlemlerdir. Bunun için Android işletim sistemi bize **ListActivity** sınıfını sunar. Standart bir ”layout” dosyasından oluşturulan bu ekran sayesinde elimizdeki veriyi oldukça rahat bir şekilde görüntüleyebiliriz.

Activity

- **MapActivity** : Harita uygulamaları için sık kullanılan bir ”Activity” tipidir. ”Google Maps” haritaları üzerinde işlem yapacağımız bir uygulamamız varsa bu ”Activity” bize yardımcı olacaktır.

Activity

- **PreferenceActivity** : Android işletim sistemine benzer bir kullanıcı ayar menüsü yapmayı hedefliyorsak, “PreferenceActivity” kullanababiliriz. faydalanabiliriz. “PreferenceActivity” layout dosyasında vereceğimiz yapıya göre sunulan seçenekleri alt kategorilerle ekrana getirir. Aynı zamanda kullanıcının saklayacağı değerleri, her değere özel bir anahtar atayarak (key – value) bellekte otomatik olarak saklar.

Layout

Android uygulamalarında ekran tasarımları **res** klasörü altında bulunan **layout** dosyaları ile belirlenir. Bu dosyalar ”xml” formatında hazırlanan dosyalardır ve Android uygulamalarına özel etiketler kullanarak görsel öğelerin yerleşimlerini ve özelliklerini bildirirler.

Layout

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent">
   <TextView
      android:text="@string/hello_world"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />
</LinearLayout>
```

Layout

- Linear Layout kullanımı Android'de, tüm nesneleri tek bir yönde kullanmamızı sağlar. Linear layout sayesinde nesneleri ”android:orientation” özelliğini kullanarak, tamamen yatay veya dikey olarak konumlandırabiliriz.

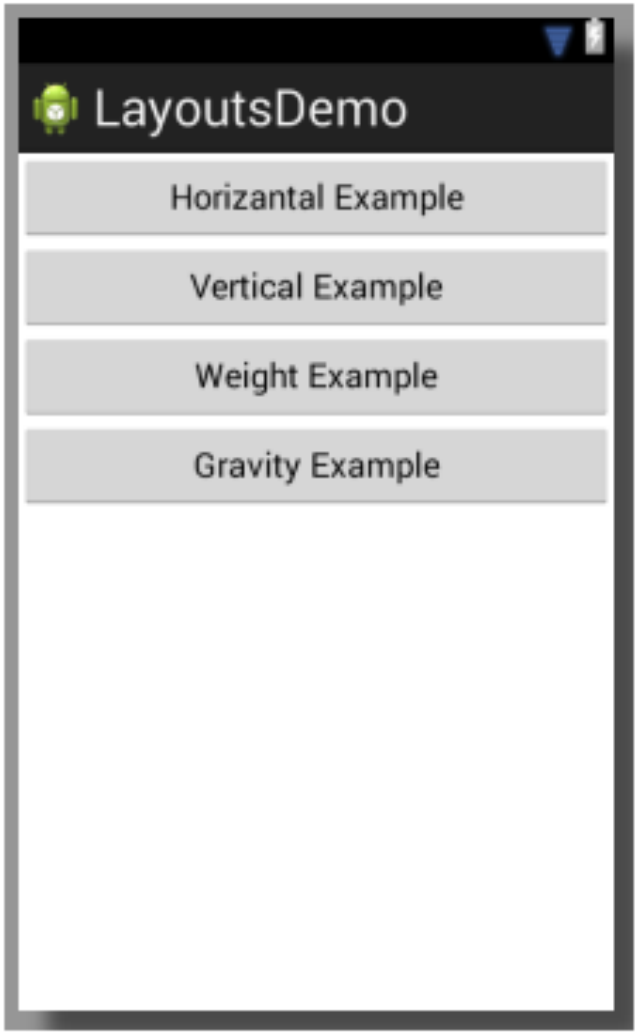
Layout

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   >
   <Button
      android:id="@+id/btn“
      android:text="@string/hello_world"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />
</LinearLayout>
```

Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <Button
        android:id="@+id/btn_show_horizantal_example"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Horizantal Example" />

    <Button
        android:id="@+id/btn_show_vertical_example"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Vertical Example" />

    <Button
        android:id="@+id/btn_show_weight_example"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Weight Example" />

    <Button
        android:id="@+id/btn_show_gravity_example"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Gravity Example" />

</LinearLayout>
```

Layout


LayoutsDemo
Horizantal Example
Vertical Example
Weight Example
Gravity Example

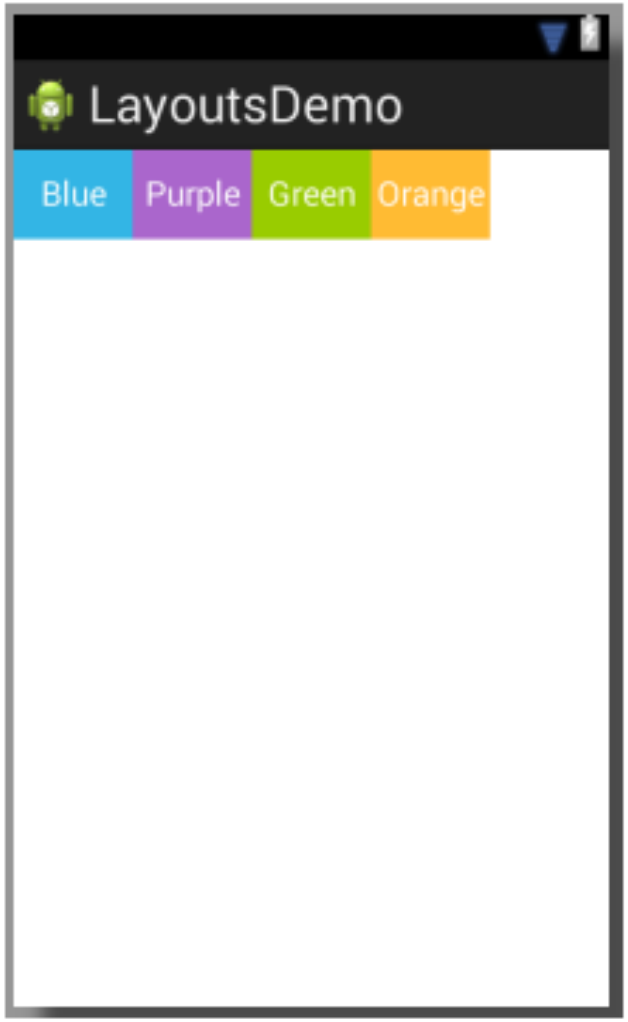
Layout

- Linear Layout'un içine eklediğiniz view'ları yatay bir şekilde göstermek için aslında herhangi bir şey belirtmemize gerek yok. Öntanımlı olarak android:orientation=*"horizontal"* olarak gelmektedir. Burada dikkat edilmesi gereken nokta, eğer yatayda çok fazla *view* eklemek istiyorsak Linear Layout'u bir Scroll View içine koymamızın gerekmesidir. Aksi takdirde son olarak eklediğimiz view'lara yer kalmadığı zaman bunu sistem ekrana doğru düzgün bir şekilde çizemeyecektir.

Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >

    <Button
        android:id="@+id/btn_blue"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@color/blue_color"
        android:text="Blue"
        android:textColor="@color/white_color" />

    <Button
        android:id="@+id/btn_purple"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@color/purple_color"
        android:text="Purple"
        android:textColor="@color/white_color" />

    <Button
        android:id="@+id/btn_green"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@color/green_color"
        android:text="Green"
        android:textColor="@color/white_color" />

    <Button
        android:id="@+id/btn_orange"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@color/orange_color"
        android:text="Orange"
        android:textColor="@color/white_color" />

</LinearLayout>
```

Layout



Layout

Linear Layout'un içine koyduğumuz view'ları nerede konumlanacağını ”gravity” değeri ile verebiliriz. Aşağıda örnekte tam ortaya gelecek şekilde değer verilmiştir:

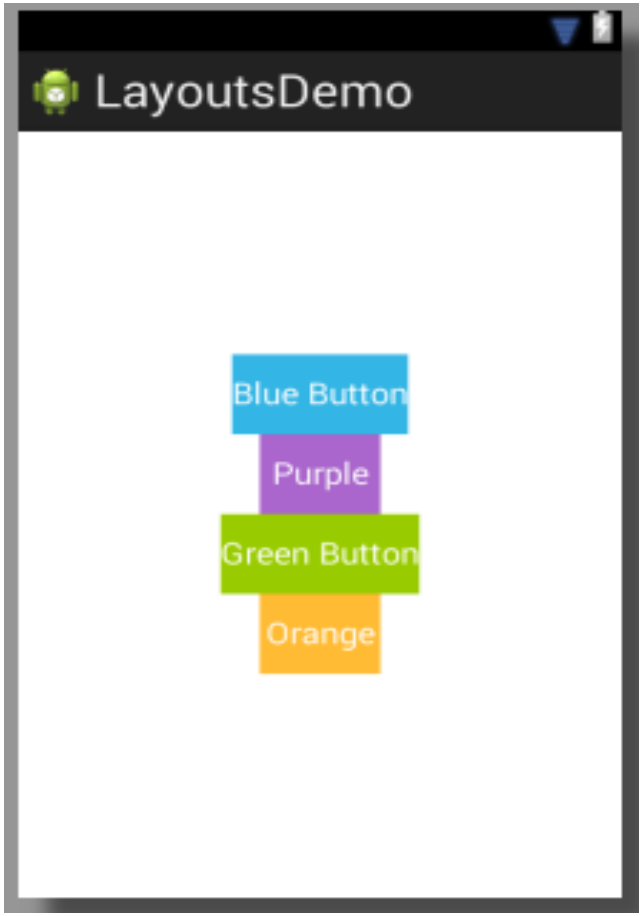
Layout

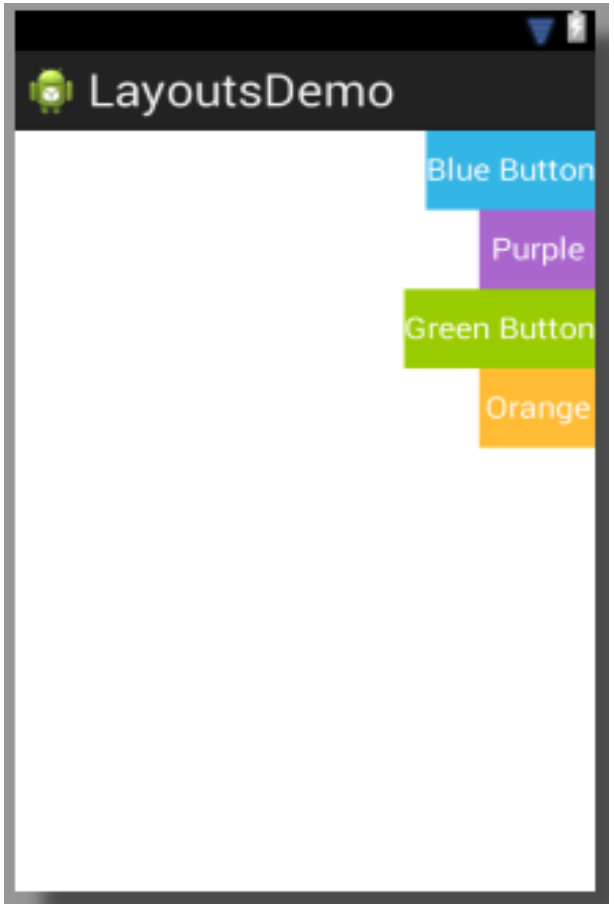
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   android:gravity="center" >
   <!--
      android:gravity="left"
      android:gravity="right"
      android:gravity="center_vertical"
      android:gravity="center_horizontal"
      android:gravity="center"
    -->

   <Button
      android:id="@+id/btn_blue"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:background="@color/blue_color"
      android:text="Blue Button"
      android:textColor="@color/white_color" />

   <Button
      android:id="@+id/btn_purple"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:background="@color/purple_color"
      android:text="Purple"
      android:textColor="@color/white_color" />
```

Layout

```
<Button
    android:id="@+id/btn_green"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@color/green_color"
    android:text="Green Button"
    android:textColor="@color/white_color" />
  <Button
    android:id="@+id/btn_orange"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@color/orange_color"
    android:text="Orange"
    android:textColor="@color/white_color" />
</LinearLayout>
```

Layout


LayoutsDemo
Blue Button
Purple
Green Button
Orange


LayoutsDemo
Blue Button
Purple
Green Button
Orange