DERS İÇERİĞİ:

- **Intent Kavramı ve Kullanım Amaçları**
- **Activity – Intent İlişkisi**
- **Intent Bileşenleri ve Intent Çözümleme**
- **Intent Filter**
- **Broadcast**

INTENT KAVRAMI VE KULLANIM AMAÇLARI

Şu ana kadar öğrendiklerimizde hep tek Activity üzerinde çalıştık. Bundan böyle bakış açımızı genişleterek uygulama içinde birden fazla **aktivite/servis/yayın algılayıcı /içerik sağlayıcı** arasından ve hatta farklı uygulamalar arasında nasıl haberleşme sağlayabileceğimize bakacağız.

Intent yardımıyla başka bir aktiviteyi başlatabilir, bir servisin çalıştırılmasını tetikleyebilir veya gerçekleşen bir olaydan bütün sistemi haberdar edebilirsiniz. Intent'ler uygulamayı kendi içerisinde ve farklı uygulamaları birbirleri ile bağlayan bir köprü görevi görür. En kısa tanımıyla **Intent** bir şeyin yapılmasını isteyen bir mesaj nesnesidir.

ACTIVITY – INTENT İLİŞKİSİ

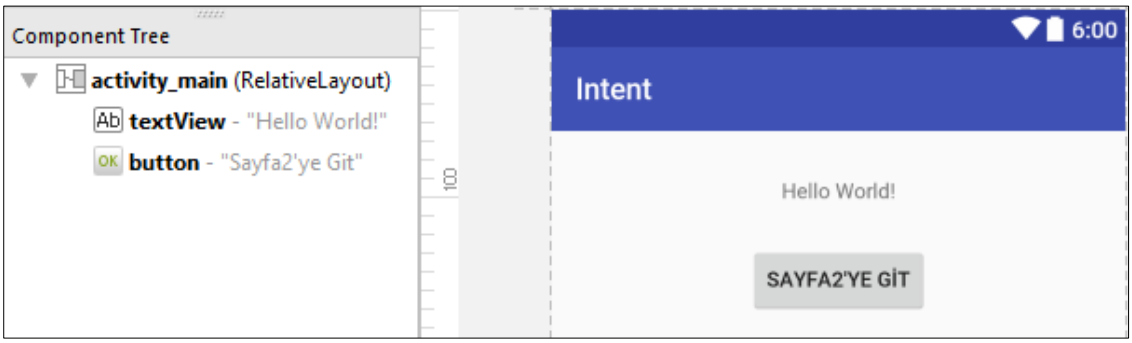
Geliştirdiğiniz uygulama içerisinden Intent nesneleri yardımıyla activity'leriniz (ekranlarınız) arasında geçiş sağlayabilirsiniz.

Uygulama

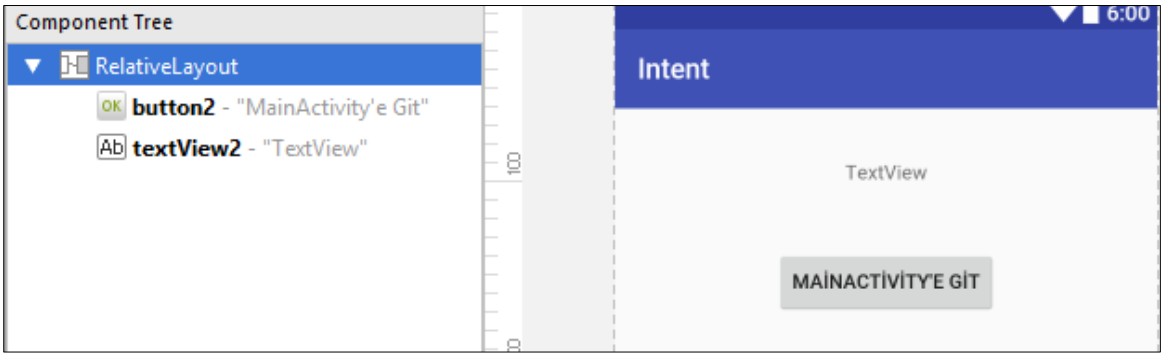
Amaç: **(Bu kısmı uygulamanın sonunda siz dolduracaksınız!!!)**

……
……
……
……
……

1. Start New Android Project, Appliation Name: Intent, Min. SDK: API20, Empty Activity, Finish.
2. Aşağıdaki gibi tasarım oluşturun.



3. **res/layout** klasörünün altında **sayfa2.xml** adında aşağıdaki bir dosya oluşturun.



4. MainActivity'nin olduğu klasörde Sayfa2 adında yeni bir class oluşturun. **setContentView'in** içeriğine dikkat edin.

```
public class Sayfa2 extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.sayfa2);
    }
}
```

5. **MainActivity'nin** buton click koduna aşağıdaki ifadeyi yazın.

```
Intent i = new Intent(MainActivity.this, Sayfa2.class);
startActivity(i);
```

6. **Sayfa2'nin** buton click koduna aşağıdaki ifadeyi yazın.

```
Intent i = new Intent(Sayfa2.this, MainActivity.class);
startActivity(i)
```

7. Sistemin bir Explicit Intent aracılığıyla bir Activity sınıfını başlatabilmesi için önce uygulamanızda böyle bir Activity olduğunu belirtmeniz gerekir. ***Yani her yeni Activity için AndroidManifest.xml dosyasında bir tanım eklemeniz gerekir.*** Eğer proje oluştururken bir default activity oluşturulmasını seçmeseydik, mainactivity için de bu tanımı manuel olarak eklemek zorundaydık. Aşağıdaki ifadeyi Manifest dosyasına ekleyin.

```
<activity
    android:name=".Sayfa2"
    android:label="Sayfa2">
</activity>
```

Intent nesneleri için yapmak istediğimiz iş açık olduğunda bu tür Intent'lere **Explicit(Açık) Intent** denir.

8. Activity'ler arasında basit bir şekilde bilgi gönderebiliriz. Kodları aşağıdaki gibi güncelleyin.

```
Intent intent = new Intent(MainActivity.this, Sayfa2.class);
intent.putExtra("paket","MainActivity'den geldim");
startActivity(intent);
```

```
Intent intent = new Intent(Sayfa2.this, MainActivity.class);
intent.putExtra("paket","Sayda2'den geldim");
startActivity(intent);
```

9. **onCreate** metodunun içine aşağıdaki ifadeyi yazın.

MainActivity için:

```
Bundle gelen=getIntent().getExtras();
if(gelen!=null){
   Toast.makeText(MainActivity.this,gelen.getString("paket"),Toast.LENGTH_LONG).show();
}
```

Sayfa2 için:

```
Bundle gelen=getIntent().getExtras();
if(gelen!=null){
   Toast.makeText(sayfa2.this,gelen.getString("paket"),Toast.LENGTH_LONG).show();
}
```

10. res/values/strings.xml dosyasına aşağıdaki string'i ekleyin.

```
<string name="gonder">Hangi program ile göndermek istiyorsunuz?</string>
```

11. Peki, **Implicit Intent(Üstü Kapalı)** nesnelerini nasıl ve ne zaman oluştururuz? *Yapmak istediğiniz şeyi bilip kimin yapacağına karar vermediğiniz durumda, bu seçimi sistemin yapmasını sağlayabilirsiniz.*

MainAcivity sınıfına bir button daha ekleyin ve kodunu aşağıdaki gibi yazın.

```
Intent i = new Intent();
i.setAction(Intent.ACTION_SEND);
i.putExtra(Intent.EXTRA_TEXT, "Bu benim gönderi metnimdir.");
i.setType("text/plain");
startActivity(Intent.createChooser(i, getResources().getText(R.string.gonder)));
```

12. Yukarıdaki kodun yerine getirdiği işlevi satır satır arkadaşınızla tartışarak aşağıya yazınız.

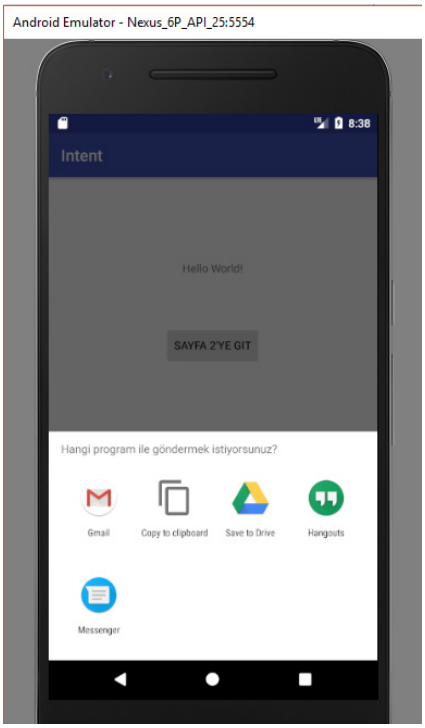
……

……

……

……

13. Aşağıdaki gibi bir görüntü alacaksınız.



INTENT FILTER

Bu güne kadar uygulamalarımızda ihtiyacımıza göre sistemin bir uygulama bulmasından bahsettik. Peki, biz kendi Web Tarayıcı, SMS, Arama veya Harita uygulamamızı geliştirmek ve başka kullanıcıların uygulamalarında bizim uygulamamızı tetikleyebilmelerini isteseydik veya sistemin belli durumlarda bize haber vermesine ihtiyaç duysaydık ne yapmalıydık?

Sorunun cevabı **intent-filter** tanımlamakta yatıyor. Intent-filter; bileşenlerin kabiliyetlerini sisteme bildirmenize yarar. AndroidManifest.xml dosyasını inceleyin ve intent-filter tanımına bakın. Bir **action** ve bir **category** elemanına sahip olduğunu göreceksiniz.

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

- **action** tanımındaki **android.inten.action.MAIN** ifadesi MainActivity'nin bu uygulamayı başlatan aktivite olduğunu ifade eder.
- **category** alanındaki **android.intent.category.LAUNCHER** değeri de, bu uygulamanın Android tarafından başlatılabilecek uygulamalar arasında listelenmesini sağlar.

INTENT BİLEŞENLERİ VE INTENT ÇÖZÜMLEME

- **COMPENENT NAME**
  Hangi komponentin **Intent** nesnesini ele alması gerektiğini belirten özelliktir. Eğer hangi kompenentin ele alınacağı belliyse, buna **Explicit Intent** denir. Eğer **Intent'in** komponent name özelliği belirtilmemişse, buna **Implicit Intent** denir ve hangi hedef uygulamanın başlatılacağına dair sistem bir çözümleme yapar.
- **ACTION**

Ne yapılması istendiğini belirten özelliktir. **ACTION_CALL, ACTION_DIAL, ACTION_VIEW** vs. Bu özellik tek başına hangi hedef uygulamanın başlatılacağına karar vermeye yetmez. Data özelliği ile beraber bir anlam ifade eder.

- **DATA**
  Action özelliğinin ne tür bir veri üzerinde işlem yapacağını belirtir. Örneğin **ACTION_DIAL** için bir telefon numarası setData metoduna argüman olarak verilebilir. Yani sistem ilgili data'yı ele alabilecek uygun action'ı bulmaya çalışır.
- **EXTRA**
  Intent bir mesaj olduğuna göre, bu mesajın içerisinde ekstra bilgi bulundurmak da son derece normaldir. Yani oluşturduğunuz Intent ile birlikte ek bilgi göndermek istiyorsanız bu özelliği kullanabilirsiniz. Örneğin İşletim Sistemi, bildirimlerini Intent'ler aracılığıyla yapar. Telefonunuza kulaklığın takılması ve çıkarılması bir bildirim sebebidir. Bu Intent Extra alanında kulaklığın türüne dair bilgi gönderir.
- **CATEGORY**
  Intent'ler kategorilere sahip olabilirler. Sistem belli durumlarda belli kategorilere sahip Intent'leri arar. Mesela; telefonunuz açılırken LAUNCHER kategorisine sahip Inten'ler bulunur ve listelenir. Ya da DEFAULT kategorisine sahip olmayan Intent'ler, Implicit Intent olarak çağrılamazlar.

BROADCAST

Broadcast Recivers(Yayın Algılayıcılar): Sistem genelindeki olaylardan haberdar olmak veya sistemi bir durumdan haberdar etmek için kullanılır. Örneğin kulaklığın çıkarıldığının bilinmesi müzük ugyulaması için önemlidir. Çünkü uygulama müzik çalmayı durdumak isteyebilir.

Broadcast Gönderme

Yayınlar, Intent'ler halinde bildirilirler. Intent nesnesini ihtiyaç duyulacak olan bilgilerle oluşturup, **sendBroadcast** metodunu çağırmanız yeterlidir. Dikkat etmeniz gereken şey, action değerinin unique(benzersiz) olmasıdır. Önerilen **Package Name'**nize ek olarak gönderiğiniz broadcast mesajını ifade eden bir **isim ekleyerek action** belirlemenizdir. Inten Extras alanına göndermek istediğiniz ek bilgileri koyabilirsiniz.

Uygulama

Amaç: **(Bu kısmı uygulamanın sonunda siz dolduracaksınız!!!)**

……
……
……
……
……

1. Start New Android Project, Appliation Name: Broadcast, Min. SDK: API20, Empty Activity, Finish.

2. Projenize bir button ekleyin ve click olayını aşağıdaki gibi yazın (***İntentin içine kendi paket adınızı yazın***).

```
Intent i = new Intent("com.example.hacer.TSUNAMI_ALARMI");
i.putExtra("MESAJ", "Acil Evinizi Terkedin");
sendBroadcast(i);
```

3. MainAcitvity sınıfıyla aynı pakette olmak üzere TsunamiReceiver adında **`BroadcastReceiver`** sınıfından türeyen yeni bir sınıf oluşturun.

```
public class TsunamiReceiver extends BroadcastReceiver {
@Override
    public void onReceive(Context context, Intent ıntent) {
        String msg = ıntent.getStringExtra("MESAJ");
        Toast.makeText(context, msg, Toast.LENGTH_LONG).show();
    }
}
```

4. Yukarıdaki kodun yaptığı işi arkadaşınızla tartışıp aşağıya yazın.

.....
.....
.....

5. AndroidManifest dosyasında <application> tagları arasına aşağıdaki kodu yazın.

```
<receiver android:name=".TsunamiReceiver" >
    <intent-filter>
        <action android:name="com.example.hacer.TSUNAMI_ALARMI" />
    </intent-filter>
</receiver>
```

6. Yukarıdaki kodun yaptığı iş nedir? Bu kodu eklemeseydik bir değişiklik olurmuydu? Arkadaşınızla tartışıp aşağıya yazın.

.....
.....
.....
.....