

DERS İÇERİĞİ:

- Kullanıcı Arayüzleri
- Layouts
 - Constraint Layout
- Button Click
- Toast

KULLANICI ARAYÜZLERİ

- **View:** Bir ekran arayüz elemanını temsil eder. Örneğin **TextView**, **EditText**, **Button**, **Spinner** vs.
- **ViewGroup / Container:** Arayüz elemanlarını (**View**) içeren kontrollerdir. Amacı içinde birden fazla **View** bulundurmaktır.
- **Layout:** Farklı **View** elemanlarını, **ViewGroup** elemanlarını veya farklı **Layout'ları** içinde bulunduran ve bu kontrollerin nasıl görüntüleceğinin tanımını içeren arayüz elemanlarıdır.

Bütün **View** elemanlarında ortak olan bazı özellikler bulunmaktadır. Bu özellikler ekran kontrollerinin **Layout** içinde nasıl davrandıklarını belirler.

- **layout_width ve layout_height:** Kontrolünüzün içinde bulunduğu Layout'a ne kadar alan kaplayacağını belirleyen özelliklerdir. Değer olarak **fill_parent**, **match_parent** ve **wrap_content** girebileceğiniz gibi, doğrudan **dimension(ölçü)** değeri de girebilirsiniz (px, dp, vs).



Ölçü girdiğiniz zaman farklı ekran boyutundaki cihazlarda tasarımınızın bozulma yaşayabileceğini dikkate alın!

- **fill_parent ve match_parent** aynı anlamlara gelir. View elemanınızın bir üstündeki Layout'tan mevcut bütün alanı istemesini sağlar. Yani yatay ya da dikey olarak container içindeki bütün alanı kaplamalarına yardımcı olur. Android 2.2'ten itibaren fill_parent, match_parent olarak değiştirilmiştir. fill_parent geriye dönük uyumludur.
- **wrap_content** kontrolünüzün ihtiyaç duyduğu kadar alan kaplamasını istediğiniz anlamına gelir.
- **layout_weight:** Bir kontrole içinde bulunduğu Layout'a göre diğer kontroller arasında önem sırası vermek için kullanılır. 0-1 arasında bir değer alır.
- **layout_gravity:** Bir kontrolün içinde bulunduğu Layout'un neresine yerleştireleceğine karar vereceğimiz özelliktir. **top**, **bottom**, **left**, **right**, **center_vertical...** gibi değerler alır.
- **gravity:** **gravity ve layout_gravity** birbirinden farklı kavramlardır. **gravity** kullandığınız View'in kendisi için bir anlam ifade ederken, **layout_gravity** bu View'in içinde bulunduğu Layout'a göre konumlandırılmasında kullanılır.
- **Text:** **text** kontrolünüzün sahip olacağı **Text** değeridir. **String resource** olarak verebileceğiniz gibi doğrudan **Layout** tanımınızda da değer belirleyebilirsiniz.



Doğrudan değer girmeniz tavsiye edilmez!

Constraint Layout : ConstraintLayout, özellikle RelativeLayout ile yaşanan bazı sorunlara çare olabilmesi amacıyla düşünülmüştür. Örneğin yüzdelik değerlerin verilebilmesi ve iç içe (nested) layout kullanımı nedeniyle oluşan karmaşıklığın azaltılması gibi. Ayrıca ConstraintLayout ile yatay ve dikey ekran kolay bir şekilde tasarlanabilir.

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

.....

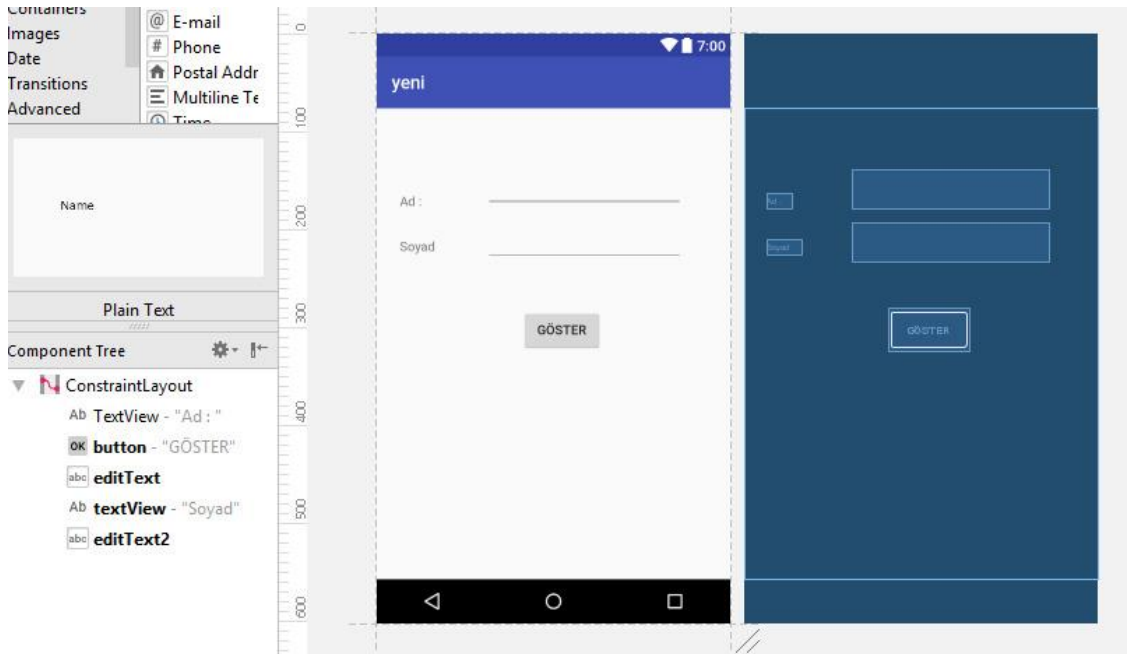
.....

.....


.....

.....

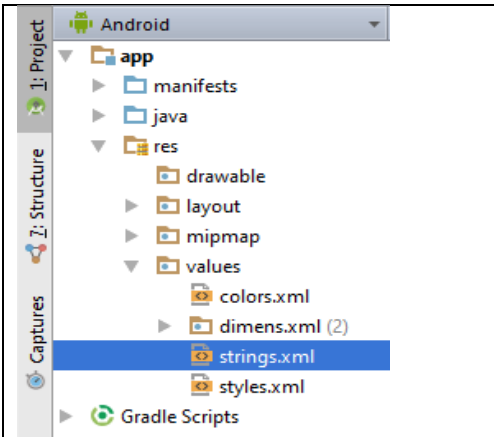
1. Yeni bir proje başlatın, min SDK 20 seçin ve adını **ConstraintLayout** olarak değiştirin.
2. **Component Tree** penceresinde **TextView** kontrolünü silin. Ve aşağıdaki gibi bir tasarım oluşturun.



3. Eklediğiniz **View**'lere birkaç farklı constrain layout uygulayıp değişiklikleri gözlemleyin.

 *Layout tasarımında gerçek hayatta çok fazla sayıda kontrol olabileceği için kontrollerin **"id"** kısmına anlamlı isimler yazmanız önerilir. Örneğin **android:id="@+id/button1"** yerine **android:id="@+id/buttonGoster"** yazılabilir.*

4. Solda gösterilen **string.xml** dosyasını açın ve içeriğine sağdaki değerleri ekleyin.



```
<resources>
  <string name="app_name">ConstraintLayout</string>
  <string name="adTextView">Ad</string>
  <string name="soyadTextView">Soyad</string>
  <string name="adEditText">Ad</string>
  <string name="soyadEditText">Soyad</string>
  <string name="gosterButton">Göster</string>
</resources>
```

5. View'lerin **properties** kısmından **text** özelliklerini **string** değerlerini kullanarak aşağıdaki gibi değiştirin. **id** özelliklerini de yine aynı yerden aşağıdaki gibi güncelleyin.



6. **app→res→layout→activity_main.xml** içindeki **Button** kısmına aşağıdaki ifadeyi ekleyin.

```
android:onClick="adsoyadGoster"
```

7. **app→java→MainActivity.java** içerisine aşağıdaki kodu ekleyin.

```
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
```

8. **app→java→MainActivity.java→MainActivity class** içerisine aşağıdaki kodu ekleyin. Programı çalıştırın.

```
public void adsoyadGoster(View view) {
    EditText ad = (EditText) findViewById(R.id.editTextAd);
    EditText soyad = (EditText) findViewById(R.id.editTextSoyad);
    String adText = ad.getText().toString();
    String soyadText = soyad.getText().toString();
    String mesaj = "Merhaba " + adText + " " + soyadText;
    Toast.makeText(MainActivity.this, mesaj, Toast.LENGTH_LONG).show();
}
```

9. 6. ve 7. Adımlarda yaptığınız değişiklikleri silin ve aşağıdaki kodu **onCreate metodu** içine ekleyin. Programı çalıştırın.

```
final EditText ad = (EditText) findViewById(R.id.editTextAd);
final EditText soyad = (EditText) findViewById(R.id.editTextSoyad);
Button goster = (Button) findViewById(R.id.buttonGoster);

goster.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String adText = ad.getText().toString();
        String soyadText = soyad.getText().toString();
        String mesaj = "Merhaba " + adText + " " + soyadText;
        Toast.makeText(MainActivity.this, mesaj, Toast.LENGTH_LONG).show();
    }
});
```

10. Sizce iki yöntem arasında bir fark var mı? Arkadaşınızla tartışıp aşağıya yazınız.

.....

.....

.....

.....

.....