DERS İÇERİĞİ:

- **Service**
- **Intent Service**
- **Notification**

Service

Eğer yapacağımız işlemler çok uzun ise **Thread** ve **AsyncTask** sınıfları yetersiz kalır. Böyle işlemler için servisler kullanılmaktadır. Servisler, uzun süreli ya da süresini kestiremeyeceğiniz işlemleri yapan, herhangi bir arayüze sahip olmayan uygulama bileşenleridir.

Siz durdurmadığınız sürece veya servis kendini durdurmadıkça (ya da kaynak ihtiyacı olup da sistem tarafından sonlandırılmadıkça) servis çalışmaya devam eder.

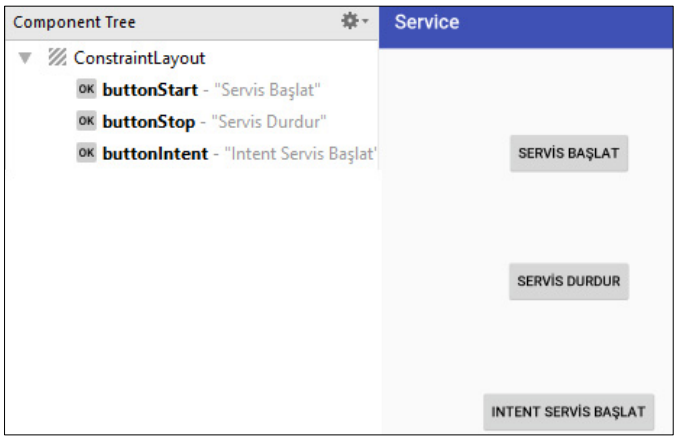
Uygulama

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

……
……
……
……
……

1. Start New Android Project, Appliation Name: Service, Min. SDK: API23, Empty Activity , Finish.

2. Aşağıdaki gibi bir tasarım oluşturun.



3. **HelloService** adında bir sınıf oluşturun ve içeriğini aşağıdaki gibi doldurun.

```java
public class HelloService extends Service {

    private static final String TAG = "HelloService";
    private boolean isRunning  = false;

    @Override
    public void onCreate() {
        Log.i(TAG, "Service onCreate");
        isRunning = true;
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Log.i(TAG, "Service onStartCommand");

        new Thread(new Runnable() {
            @Override
            public void run() {

                for (int i = 0; i < 5; i++) {
                    try {
                        SystemClock.sleep(1000);
                    } catch (Exception e) { }

                    if(isRunning){
                        Log.i(TAG, "Service running " + i);
                    }
                }
             stopSelf();
            }
        }).start();

        return Service.START_STICKY;
    }

    @Override
    public IBinder onBind(Intent arg0) {
        Log.i(TAG, "Service onBind");
        return null;
    }

    @Override
    public void onDestroy() {
        isRunning = false;
        Log.i(TAG, "Service onDestroy");
    }
```

4. Servisinizi **Manifest** dosyasına **<application> </application>** tag'ları arasına aşağıdaki gibi ekleyin.

```xml
<service android:name=".HelloService"></service>
```

5. Start Service ve Stop Service **button** kodlarının içeriğini aşağıdaki gibi yazın.

```java
Intent intent = new Intent(MainActivity.this, HelloService.class);
startService(intent);

Intent intent = new Intent(MainActivity.this, HelloService.class);
stopService(intent);
```

6. Start Service butonuna basın, Logcat'e bakın, onDestroy metodu çalıştı mı? Uygulamayı tekrar başlatın ve Stop Service butonuna basın. onDestroy metodu çalıştı mı?

   ……
   ……
   ……

7. onStartCommand metodu içindeki **i<5** ifadesini **i<500** yapın ve tekrar çalıştırın. Programı arka plana atın. **"Service Running"** ifadesi Logcat'de yazmaya devam ediyor mu? Stop Service butonuna basın.

   ……
   ……
   ……

8. onStartCommand metodu içindeki **i<500** ifadesini **i<5** yapın. Metodun sonundaki **stopSelf()** metodunu aktif edin. Start Service butonuna basın. onDestroy metodu çalıştı mı?

   ……
   ……
   ……

Not: HelloService içinde yukarıda olduğu gibi Thread sınıfını kullanabileceğiniz gibi AsyncTask da kullanabilirsiniz.

Intent Service

Servisiniz çalıştırıldığı anda işlemlerini tamamlayıp, işi bittiğinde de kendiliğinden sonlandırılmasını istediğiniz durumlar olabilir. Bunu yaparken her seferinde yeni bir **thread oluşturmak** ve onu yönetmekle uğraşmak da **istemeyebilirsiniz.**

IntentService sizi thread yönetiminden kurtarmak ve işleri bittiğinde kendi kendilerini durdurmakla sorumlu servislerdir.

Service vs Intent Service

Eğer **sürekli çalışmasını** istediğiniz bir işlem varsa bunun için **Service** kullanmalısınız. Ama **aralıklarla çalışmasını** istediğiniz bir işlem varsa sistem kaynakları açısından **IntentService** kullanmalısınız.

9. HelloIntentService adında bir sınıf oluşturun ve içeriğine aşağıdaki kodları ekleyin.

```
public class HelloIntentService extends IntentService {

    private static final String TAG = "HelloIntentService";

    public HelloIntentService(){
        super("HelloIntentService");
    }

    @Override
    protected void onHandleIntent(Intent intent){
        for (int i = 0; i < 5; i++) {
            SystemClock.sleep(1000);
            Log.i(TAG, "Service running " + i);
        }
    }
}
```

10. Bu servisi de Manifest dosyasına eklemeyi unutmayın.
11. Start Intent Service button kodunun içeriğini aşağıdaki gibi yazın.

```
Intent intent = new Intent(MainActivity.this, HelloIntentService.class);
startService(intent);
```

12. Start Intent Service butonuna bastığınızda Logcat'te nasıl bir çıktı alıyorsunuz?

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

13. Programı arka plana attığınızda nasıl bir çıktı alıyorsunuz?

Notification

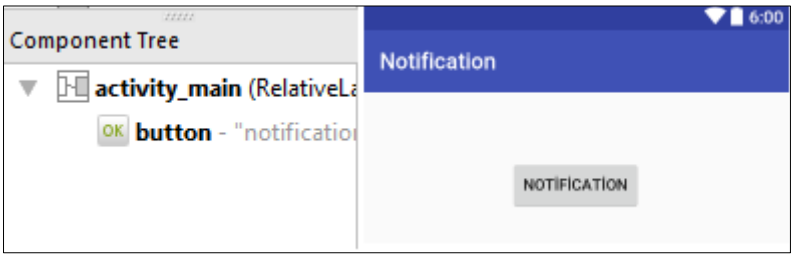
Uyarılar, uygulamaların kullanıcıyı kalıcı olarak bilgilendirmek için kullandığı yapılardır.

Uygulama

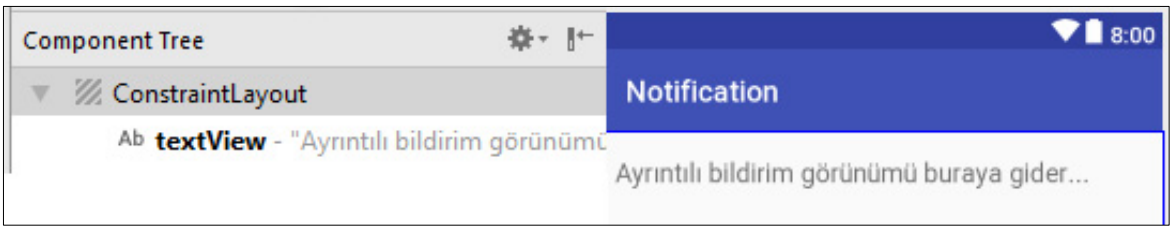
Amaç: **(Bu kısmı uygulamanın sonunda siz dolduracaksınız!!!)**

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

1. Start New Android Project, Appliation Name: Notification, Min. SDK: API23, Empty Activity , Finish.
2. Aşağıdaki gibi bir tasarım oluşturun.



3. Layout kısmında **notification** adında aşağıdaki gibi bir tasarım daha oluşturun.



4. **ResulActivity** adında bir sınıf oluşturun ve içeriğini aşağıdaki gibi doldurun.

```
public class ResultActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.notification);
    }
}
```

5. Manifest dosyasına aşağıdaki ifadeyi ekleyin.

```
<activity android:name=".ResultActivity"
    android:label="Details of notification"
    android:parentActivityName=".MainActivity">
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value=".MainActivity"/>
</activity>
```

6. Bu ayarlama **ResultActivity'**nin **MainActiviy'**nin normal **iş akışının** bir parçası olarak devam etmesini sağlamaktadır.
7. Size verilen resmi drawable kalsörüne kopyalayın.
8. MainActivity'e aşağıdaki kodları ekleyin.

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        findViewById(R.id.button).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
              NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(MainActivity.this)
                                    .setSmallIcon(R.drawable.ic_favorite_border_black_24dp)
                                    .setContentTitle("Benim Bildirimim")
                                    .setAutoCancel(true)
                                    .setContentText("Merhaba!");
              Intent resultIntent = new Intent(MainActivity.this, ResultActivity.class);


              TaskStackBuilder sb = TaskStackBuilder.create(MainActivity.this);
              sb.addParentStack(ResultActivity.class);
              sb.addNextIntent(resultIntent);


              PendingIntent rpi = stackBuilder.getPendingIntent(0, PendingIntent.FLAG_UPDATE_CURRENT);
              nb.setContentIntent(rpi);
              NotificationManager nm = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
              nm.notify(0, nb.build());
            }
        });
    }
}
```

9. Sarı ile işaretlenmiş satırları inceleyin. Bu satırlarda **new** ile oluşturulan ve ardındaki satırlarda özellikleri belirlenen nesnenin ne olduğunu arkadaşınızla tartışıp aşağıya yazınız.

……
……
……
……
……

10. Mavi ile işaretlenmiş satırları inceleyin. Bu satırlarda bir yığın olşturulup içine ilk eleman olarak MainActivitiy eklendiğini, ikinci olarak da Result Activity eklendiğini gözönünde bulundurun. Buna göre burada hangi işlevin yerine getirildiğini arkadaşınızla tartışarak aşağıya yazın.

……
……
……

11. Pembe ile işaretlenmiş satırları inceleyin. Bu satırlarda **PendingIntent** ile MainActivity yetkilerini ResultActivity'e devretmektedir.