G54MDP Lab Session 03 - Services

The aim of this exercise is for you to implement two simple applications that make use of a Service component in two different ways.

Necessarily this will be similar to the example application shown in lecture 7/8, so spend some time reviewing the code provided, however it is important to try and implement a similar application yourself rather than cutting and pasting code, so that you get used to the coding style required.

Intent Service

Create an application that includes an Activity and an Intent Service.

Create a new Activity in the usual way, with a simple View containing a Button and an onClick handler method.

Next create a new IntentService component in the application.

- Create a new class that extends the IntentService class

- As in the example this new class should have a constructor that simply calls the super-class constructor, and a single method that overrides the default onHandleIntent method:

  ```
  public class MyIntentService extends IntentService

  …

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

  …

  protected void onHandleIntent(Intent intent)
  {

  …
  ```

- Add a new element to the manifest file to register the new component

  ```
  <service android:name=".MyIntentService" />
  ```

Next extend your Activity to make use of this service, by sending Intents to the startService method.

```
        Intent msgIntent = new Intent(this,
MyIntentService.class);
        msgIntent.putExtra("some name", someData);
        startService(msgIntent);
```

Add functionality to the Service's worker method to do something that takes some time, and prints a result out to the debug log, for example calculating a factorial.

Queue up several work requests by clicking the button several times, and observe the lifecycle of the IntentService.

Bound Service

Create an application that includes an Activity and a bound Service.

Create a new Activity in the usual way, with a simple View containing a Button and an onClick handler method.

Next create a new Service component in the application.

- As before, create a new class but this time extend the Service class. You must override the onBind method, and include an inner class that extends the Binder class. This defines the interface that your activity will be able to make use of.

```
public class MyBoundService extends Service
{
        private final IBinder binder = new MyBinder();

        …

        public IBinder onBind(Intent intent)
        {
                return binder;
        }

        …

        public class MyBinder extends Binder
        {
                void doSomething()
                {
                        MyBoundService.this.serviceDoSomething();
                }
        }

        …

        public void serviceDoSomething()
        {

        …
```

Next, bind and make use of your new Service from within the Activity. You need a local reference to the Service binder object that the service connection will provide:

```
public class MainActivity extends Activity
{
     private MyBoundService.MyBinder myService = null;
```

Next, you need an instance of the ServiceConnection class to handle the new connection to the service, and that handles callbacks when the service connects and disconnects, providing the object that represents the Binder proxy.

```
     private ServiceConnection serviceConnection = new
ServiceConnection()
     {
          public void onServiceConnected(ComponentName name,
IBinder service)
          {
               myService = (MyBoundService.MyBinder) service;
          }

          public void onServiceDisconnected(ComponentName
name)
          {
               myService = null;
          }
     };
```

Finally, you need to bind the service using these new additions. You should probably do this in the onCreate method of the Activity.

```
     this.bindService(new Intent(this, MyBoundService.class),
serviceConnection, Context.BIND_AUTO_CREATE);
```

Add debug statements to ensure that the service is created and bound as you would expect. Finally, connect the button to call a method on your service binder object, and ensure that the corresponding method is called in the service object.

http://developer.android.com/guide/components/services.html

http://developer.android.com/guide/components/bound-services.html

http://developer.android.com/guide/components/processes-and-threads.html