475/575 In Class Lab
Broadcast Receiver – Service

This lab demonstrates how to create a broadcast receiver and service. Its purpose is to listen for texts that contain a key phrase. For testing you need to send text messages, typically from

Define Receiver

1. Determine what fires your receiver (Custom or System). If custom then you create the broadcast
    a. System- going to be limited here. See https://developer.android.com/develop/background-work/background-tasks/broadcasts/broadcast-exceptions
    b. Custom
        i. Define as in lecture "12_Services and Broadcast Receivers"
2. Determine BR lifespan
    a. Manifest defined (always active until uninstall)
    b. Dynamic defined (active only when app running)
3. Create receiver

```
    a. public class myBroadcastReceiver extends BroadcastReceiver {}
    b. Red squiggles appear under above, hover over and hit alt-enter and
       implement required methods.
    c. Set up a log in the onReceive method to verify its called
  public void onReceive(Context context, Intent intent) {
      //start service here
      Log.e(TAG,"In Broadcast Receiver");
```

```
4. We want it to be always active so define it in manifest inside application tags
<receiver android:name=".MySMSreceiver" >
    <intent-filter android:priority="999" >
        <action android:name="android.provider.Telephony.SMS_RECEIVED" />
    </intent-filter>
</receiver>

5. And we also need some permissions to receive SMS messages
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.READ_SMS" />

6. Ask for permissions:
In MainActivity.java

     //for permissions
    import android.Manifest;
 in Mainactivity Class
private static final String[]
        PERMISSIONS={Manifest.permission.RECEIVE_SMS,
        Manifest.permission.READ_SMS};
private static final int PERMS_REQ_CODE = 200;


in onCreate add a call
```

```
verifyPermissions();
```

Then add the following functions;

```
/**
 * Verify that the specific list of permisions requested have been granted, otherwise 
ask for
 * these permissions. Note this is coarse in that I assumme I need them all
 */
private boolean verifyPermissions() {
    //loop through all permissions seeing if they are ALL granted
    //iff ALL granted then return true
    boolean allGranted = true;
    for (String permission:PERMISSIONS){
        //a single false causes allGranted to be false
        allGranted = allGranted && (ActivityCompat.checkSelfPermission(this, 
permission ) ==
                PackageManager.PERMISSION_GRANTED);
    }
    if (!allGranted) {
        //OH NO!, missing some permissions, offer rationale if needed
        for (String permission : PERMISSIONS) {
            if (ActivityCompat.shouldShowRequestPermissionRationale(this, permission)) 
{
                Snackbar.make(findViewById(android.R.id.content),
                        permission+" WE GOTTA HAVE IT!", Snackbar.LENGTH_LONG).show();
            }
        }
        //Okay now finally ask for them
        requestPermissions(PERMISSIONS, PERMS_REQ_CODE);
    }
    //return whether they are granted or not
    return allGranted;
}

/***
 * callback from requestPermissions
 * @param permsRequestCode user defined code passed to requestpermissions used to 
identify what 
callback is coming in
 * @param permissions list of permissions requested
 * @param grantResults //results of those requests
 */
@Override
public void onRequestPermissionsResult(int permsRequestCode, String[] permissions, 
int[] grantResults) 
{
    super.onRequestPermissionsResult(permsRequestCode, permissions, grantResults);
    boolean allGranted = true;
    switch (permsRequestCode) {
        case PERMS_REQ_CODE:
            for (int result: grantResults){
                allGranted = allGranted&&(result== PackageManager.PERMISSION_GRANTED);
            }
            break;
    }
}
```

Define Service

Create Service

1. `public class MyService extends Service{`
   a. Red squiggles appear under above, hover over and hit alt-enter and implement required methods.
2. Hover over MyService and hit ctrl-O and override onStartCommand

3. Set up a log in the onStartCommand method to verify that its called

```
public int onStartCommand(Intent intent, int flags, int startId) {

   Log.e(TAG,"In BRandServiceandSystemAction");

   //do work here, stop service when done
   stopSelf();                return 0;
}
```

Register Service in manifest **inside application tags**

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

Start the service from the Broadcast Receiver's onReceive method.

```
//start the service
Intent myIntent = new Intent(context, MyService.class);
context.startService(myIntent);
```

Test it,

send it a text from another phone number

Further- Now lets parse the text a bit, we can search on a word withen the text or a particular phone number, Here is a receiver that does that

```
public class myBroadcastReceiver extends BroadcastReceiver {

    private static final String TAG  = "myBroadcastReceiver";
    private static final CharSequence SECRETSTRING = "secret";

    @Override
    public void onReceive(Context context, Intent intent) {
        //start service here
        Log.e(TAG,"In Broadcast Receiver");

        doStuff(context, intent);

        //start the service
        Intent myIntent = new Intent(context, MyService.class);
        context.startService(myIntent);

    }


    void doStuff(Context context, Intent intent){
        //lets see whats inside
        Bundle extras = intent.getExtras();

        if ( extras != null )
    {
        //A PDU is a "protocol description unit", which is the industry format for an
SMS message. because SMSMessage reads/writes them you shouldn't need to dissect them.
        //A large message might be broken into many, which is why it is an array of
objects.
        Object[] smsextras = (Object[]) extras.get( "pdus" );

        for ( int i = 0; i < smsextras.length; i++ )
        {
            SmsMessage smsmsg = SmsMessage.createFromPdu((byte[])smsextras[i]);

            //see whats in the message
            String strMsgBody = smsmsg.getMessageBody().toString();

            //does it contain our string?
            if (strMsgBody.contains(SECRETSTRING)){
                Log.i(TAG, "contains secret string");

                //start the service
                Intent myIntent = new Intent(context, MyService.class);
                context.startService(myIntent);
            }
            else
                Log.i(TAG, "Does Not contain secret string");

            //can also do this by phone number
            //String strMsgSrc = smsmsg.getOriginatingAddress();
        }
    }
}
}
```