Lecture 4

- Intent
- Start a new Activity
- BroadcastReceiver

Android – component based

Android is component based. This means that an application in Android can take advantage of already pre-written components.

Android Components:

- Activity, application.
- Background service.
- Broadcast Receiver: Receiver of messages from another components.
- Content Provider: Providing data accessible for other components.

Activity

- An Activity can start another activities, as well as other applications installed in the device.
- Your application can be also targeted and called by other applications.

Intent

An Intent is a message in Android that can be used for:

- Starting an Activity
- Starting a service.
- Sending a Broadcast Intent that broadcasts a message to other. Other applications receive message.
- Receiving a Broadcast Intent sent from another application.

Intent - data transfer

An Intent can contain data that can be used by the target of the intent.

The method putExtra (key, value) adds a <key, value> pair to the Intent.

The key should be a string and the value can be:

- simple variable (e.g. double)
- String
- Arrays (double [], String[], …)
- Serializable (Object that implements serializable)
- Parcel Variable (Object that implements Parcelable)

E.g:

```
Intent intent = new Intent(…);
intent.putExtra( "name", "Ruth" );
intent.putExtra ( "age", 26 );
```

Intent - data transfer

The target of an intent can use the data stored in the Intent.

The method getIntent() provides a reference to the Intent.

The methods get<X>Extra (key) / get<X>Extra (key, default value) return the value associated to the provided key. The type of this value is X.

E.g:

```
Intent intent = getIntent();
String n = intent.getStringExtra( "name” ); // If name is not found then n will be null.
int age = intent.getIntExtra ( "age", -1 );  // If age is not found then age will be -1.
```

Start an Activity

You can start an Activity from the current Activity. Activities should be focused in one special task (or “activity”), so every time a change of task is needed, you should start it in a new Activity. E.g. taking a picture, sending an email, texting somebody, opening a browser, etc.

Activities are launched from Intents. The content of the Intent determines the Activity to be launched.

- One can explicitly (in plaintext) express the target Activity.

```
Intent intent = new Intent( context, ActivityToStart.class );
startActivity( intent );
```

- The Intent can handle information about the specific action that the target Activity must perform. E.g. handing a phone call.

```
Intent intent = new
Intent(Intent.ACTION_DIAL,Uri.parse("tel:+46708123456"));
startActivity( intent );
```

Explicit Intent

An explicit Intent is used to start an Activity (or other component) in the application. It explicitly declares the Activity that must be launched.

Constructor

```
public Intent (Context packageContext, Class<?> cls)
```

E.g:

```
Intent intent = new Intent( context, ActivityToStart.class );
startActivity( intent );
```

Every Activity in the application must be declared in the app's manifest.

```
<application
    :
    <activity
        android:name=".ActivityToStart"
        android:label="@string/title_activity_to_start" >
    </activity>
    :
</application>
```

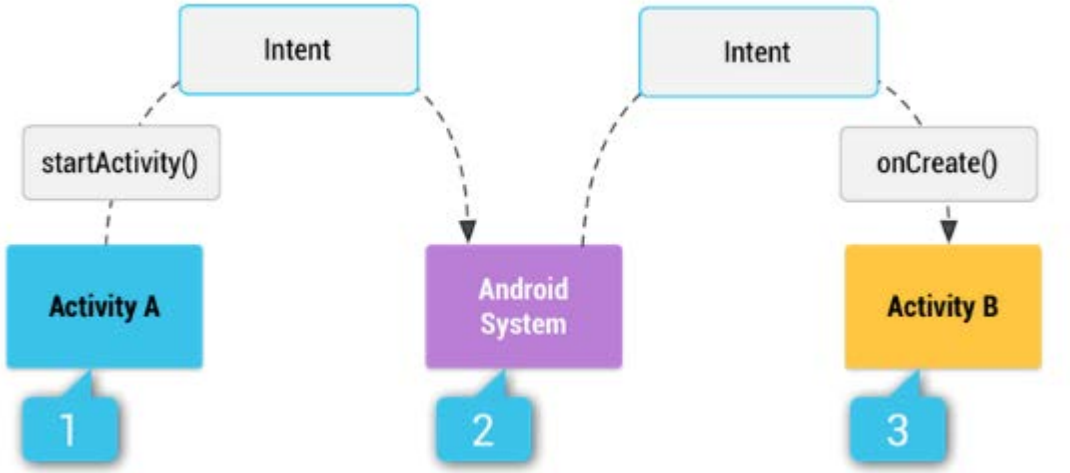
Implicit Intent

An implicit Intent is used to start an Activity outside the application. This activity must be located on the Android device.

It is called an implicit Intent because the Intent does not call for a specific Activity. The intent declares that there is an action that must be performed. It is on the operating system (and the user) to provide the app that would handle that action.

The intent is set to contain the following:

- **Action** The action that must be performed.
- **Data** Data needed to set up the action.
- **Category** Supplementary information.



Intent filter

An intent filter is used to declare the type of action / data / category that the target Activity must handle. Intent filters must be declared in the Manifest file as a part of the Activity tag.

E.g:

```
<activity android:name="ShareActivity">
    <!-- This activity handles "SEND" actions with text data -->
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
    </intent-filter>
</activity>
```

It is possible to add more action-tags / category tags / data tags to the filter. An Activity to be started implicitly must always contain the above DEFAULT category tag

```
<category android:name="android.intent.category.DEFAULT"/>
```

Intent resolution

When the system receives an implicit intent to start an activity, it searches for the best activity for the intent by comparing the intent to intent filters based on three aspects:

1 Action

- The Action in the Intent matches an action tag in a filter.
- A filter without action tags can not matched.
- However, if an Intent does not specify an action, it will pass the test (as long as the filter contains at least one action)

2 Category

- Every category in the Intent must match a category in the filter.
- An intent with no categories should always pass this test, regardless of what categories are declared in the filter.

3 Data

- URI structure and a data type (MIME media type). Following the structure: <scheme>://<host>:<port>/<path>
- E.g: content://com.example.project:200/folder/subfolder/etc

Special types of Intent

Pending Intent

A Pending intent is an Intent to start another Activity. But this will happen sometime in the future, for example, at a Notification. The Pending Intent escapsulates the permissions needed to perform the target Activity.

```
int seconds = 5;

Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse("http://developer.android.com"));

PendingIntent pendingIntent = Pend-
ingIntent.getActivity(MainActivity.this, 0, intent, Pend-
ingIntent.FLAG_ONE_SHOT);

AlarmManager am = (AlarmManager) getSystemService(ALARM_SERVICE);
am.set(AlarmManager.RTC, System.currentTimeMillis() + seconds*1000,
pendingIntent);
```

Broadcast Receiver

A Broadcast Receiver is used primarily to receive a message from the system, e.g. battery charge or whether the device has changed to airplane mode.

See the API documentation of Intent for Standard Broadcast Actions. Additionally, message types can be declared in other classes.

A Broadcast Receiver must inherit BroadcastReceiver and overrides the method onReceive.

```
public class PhoneReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
    }
}
```

onReceive method should end quickly. Time consuming action must be carried out asynchronously.

Broadcast Receiver

Here is an example of a Broadcast Receiver that is listening for incoming phone calls.

```
public class PhoneReceiver extends BroadcastReceiver {
    public void onReceive(Context context, Intent intent) {
        if (intent != null) {
            String state = intent.getStringExtra(TelephonyManager.EXTRA_STATE);
            Log.w("PhoneReceiver", "EXTRA_STATE="+state);
            if (state.equals(TelephonyManager.EXTRA_STATE_RINGING)) {
                String phoneNumber = intent
                        .getStringExtra(TelephonyManager.EXTRA_INCOMING_NUMBER);
                Log.w("PhoneReceiver", "EXTRA_INCOMING_NUMBER="+phoneNumber);
            }
        }
//        Intent startIntent = new Intent(context, MainActivity.class);
//        Intent startIntent = new Intent("se.mah.tsroax.f4startactivity.MainActivity");

//        startIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
//        context.startActivity(startIntent);
    }
}
```

Last four lines give examples on how to start an Activity from a Broadcast Receiver.

Broadcast Receiver

Broadcast Receivers can be registered in the manifest or in the source code at runtime, using registerReceiver.

If registered in the manifest, the system will activate them when needed.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="se.mah.tsroax.f4broadcastreceiver" >

   <uses-permission android:name="android.permission.READ_PHONE_STATE" />

   <application
      :
      <activity
         :
      </activity>

      <receiver android:name=".PhoneReceiver" >
         <intent-filter>
            <action android:name="android.intent.action.PHONE_STATE" />
         </intent-filter>
      </receiver>

   </application>

</manifest>
```

Broadcast Receiver

Broadcast Receivers can be registered in the manifest or in the source code at runtime, using registerReceiver.

Register it in onResume and then unregister it in onPause. This way, the Broadcast Receiver is only active when the Activity is active.

```
public class MainActivity extends Activity {
   private PhoneReceiver phoneReceiver = new PhoneReceiver(this);
   private IntentFilter filter =
                    new IntentFilter(TelephonyManager.ACTION_PHONE_STATE_CHANGED);
   :
 @Override
   protected void onResume() {
      super.onResume();
      registerReceiver(phoneReceiver, filter);
   }

   @Override
   protected void onPause() {
      super.onPause();
      unregisterReceiver(phoneReceiver);
   }
}
```

Broadcast Intent

A BroadcastIntent is the Intent used to deliver a broadcast events. These events are generated by the system and its native apps. E.g:

- Incoming call.
- SMS received.
- Battery status.

```
OutgoingCallReceiver receiver = new OutgoingCallReceiver();
IntentFilter filter = new IntentFilter(Intent.ACTION_NEW_OUTGOING_CALL);
registerReceiver(receiver, filter);
```

You can also define your own Intents, in a way analogous to implicit intents.

```
Intent intent = new Intent("com.example.travellist.NEW_TRAVEL_ADDED");
intent.putExtra(TravelInfo.EXTRA_CITY, info.getCity());
sendBroadcast(intent);
```

Special types of Intent

Sticky Intent

An intent stored in the system that can be later fired by other apps.

An app can launch a sticky intent to broadcast that the current state of the battery is low. This intent gets “stuck” to the system, so that when any other app wants to know the state of the battery, they just recall this intent.

```
Intent intent = new Intent("some.custom.action");
intent.putExtra("some_boolean", true);
sendStickyBroadcast(intent);
```