**Title: Intents**

1. Intents: Intents in Android are used as messaging objects. You will use intents whenever you want to get a task done by another part of the system. These tasks can range from starting another activity to starting a background service to download some data. We haven’t discussed Services and Broadcasts yet so I will restrict these notes to starting new activities.

Intents can be broadly classified into two categories.

a. Explicit Intents: When we specify the exact activity class/component that should handle the intent. We will use this methodology to launch new activities within the app. E.g. in our stocks app, when the user clicks on a company in the list we would like to show a detailed page to user. For this we can create an explicit intent. Example code for that:

Intent i = **new** Intent();

i.setClass(getApplicationContext(), DetailedActivity.**class**);

startActivity(i);

Here in this code we created an intent and set the DetailedActivity class as the class, which should handle the intent. startActivity function is used to pass the intent to Android and start the transition. Instead of using setClass function, you could have used the setComponent, setClassName functions as well. In order to make this work, you would need to create a DetailedActivity class and add that activity to your manifest.xml file.

b. Implicit Intents: Sometimes we don’t care which activity takes care of our intent. We just want some activity to handle the task at hand. E.g. if user clicks on a website link in our app, we would like the link to be opened by a browser. We don’t really care which app is used to open the link as long as it can take care of the task at hand. In these cases we won’t set the class that should handle the intent, instead we tell the task to be done and Android finds all the activities, which can handle it. Here are some attributes, which we can use to describe a task.

i. Action: This string specifies the generic action type. E.g. ACTION\_VIEW or ACTION\_SEND. E.g. if you want to open a link in browser it will be ACTION\_VIEW whereas to send an email you should use ACTION\_SEND.

ii. Data: Data is an Uri type object. It contains the details of the task to be done as well as the [MIME](http://en.wikipedia.org/wiki/MIME) type of the task. E.g to open a website we will set the data Uri to be “http://www.google.com”. Here http:// is the MIME type, which tells Android that activities that can handle http data should be used to complete this task. [www.google.com](http://www.google.com) is the data that tells the target activity which website should be opened.

iii. Category: This contains additional information about the activity to be launched. E.g. CATEGORY\_DEFAULT tells that the activity to be launched should be an activity, which opens in standard full screen mode whereas CATEGORY\_TAB tells that the activity should be launched as a tab in a TabActivity.

iv. Type: This is the MIME type that we have already discussed while talking about data. In case you don’t have any data to be passed you can use the setType function and just set the type of intent.

Here are few examples of implicit Intents:

Link:

Intent i = **new** Intent();

i.setAction(Intent.*ACTION\_VIEW*);

Uri a = Uri.*parse*("http://www.google.com");

i.setData(a);

startActivity(i);

Email:

Intent i = **new** Intent (Intent.*ACTION\_SENDTO*);

i.setData(Uri.*parse*("mailto:ankush@codingblocks.com"));

i.putExtra(Intent.*EXTRA\_SUBJECT*, "subject");

**if** (i.resolveActivity(getPackageManager()) != **null**) {

startActivity(i);

}

In the second example we first checked if the intent can be resolved by any of the applications installed by the user or not. We start the activity only if intent can be resolved, otherwise our app would have crashed.

c. Receiving Intents (Intent Filters): As discussed while using implicit intents you can tell Android the kind of task you want to get done and it will find the activities that can take care of the task. Now, if build an activity which can take care of some kind of tasks, we would like Android to show our application as one of the options to the user. In order to achieve this, we will be using Intent Filters. Intent Filters are added in the Manifest.xml file and from there Android can find out what kind of tasks can be taken care by which activity of our application. E.g. if we were building a browser and BrowserActivity is the class which can take open webpages, we can add following code in the manifest file.

<activity

android:name=*".BrowserActivity"*

android:label=*"Browser"* >

<intent-filter>

<action

android:name=*"android.intent.action.VIEW"* />

<category

android:name=*"android.intent.category.DEFAULT"*/>

<data android:scheme=*"http"*/>

</intent-filter>

</activity>

Here we are telling Android that we have an activity BrowserActivity that can take care of view action on data with MIME type http. Default category is the category used by intents when we don’t mention any other category. We can similarly add intent filters for various actions, MIME types and categories.

2. Passing data between activities:

a. Caller to callee: A lot of times you would like to pass additional data to the target activity and the data can range from the database id of an object or other attributes that can be used by the target activity to show data. Intent class provides a bunch of putExtra functions, which can be used to pass different kind of data. E.g. in our stocks app we might want to pass the company’s ticker symbol to the detailed activity. Example code on caller side:

Intent i = **new** Intent();

i.setClass(getApplicationContext(), DetailedActivity.**class**);

i.putExtra("company\_name", c.name);

startActivity(i);

On the callee side we can use the getExtras function to get an object of class Bundle that contains all the extra data that was passed by the caller. Example code:

Intent i = getIntent();

Bundle b = i.getExtras();

String companyName = b.getString("company\_name");

b. Callee to caller: Not as common as the first case but sometimes we want the target activity to complete a task and return the result to us. E.g. we may want to create a new activity where user can add a new company to list of companies. We might want this activity to return the data that user has entered and save the data in the original activity. For cases like these caller need to use startActivityForResult function instead of the startActivity function. Caller also needs to implement onActivityResult method, which gets called once the target activity finishes. Here are the functions:

startActivityForResult(intent, requestCode);

onActivityResult(**int** requestCode, **int** resultCode, Intent data)

Here the caller sets the requestCode and the callee sets the resultCode. These codes are used to differentiate between various possible intents that could have finished and also to differentiate between different outcomes (like success or failure). E.g. our main activity could have called Activity A in one function and Activity B in some other function and Activity A in third function but this time for some other task. We could use the requestCode to differentiate between these three cases.

On the callee side you need to set the resultCode and data to be sent back. You can add any kind of data to be sent back in intent extras, similar to what we did while sending data from caller to callee.

Intent data = **new** Intent();

Data.putExtras(“extra\_data”, “done”);

setResult(resultCode, data);

You can add this code to finish() method of activity, which is called everytime the activity is about to be finished.

3. Homework

a. Add following features to the todo app that we have been working on:

i. Add a new item to your menu: “Add new”

ii. Take user to an activity where user can enter a new todo item to the list.

iii. Bring data from the new activity back to our main activity and add the new item to the list view.

iv. On click of any item take user to a detailed page, where user can look at further details.

v. In the detailed page add an item to menu: “Delete”. Show an alert dialog asking for user confirmation when user clicks this item, if confirmed finish the activity and pass this data back to the main activity and remove the todo item from list view.

vi. Add a “About us” item to MainActivity menu and on click take them to your website. For now take them to [www.google.com](http://www.google.com)

vii. Add a “Contact us” item to MainActivity menu and on click dial your phone number.

viii. Add a “Email us” item to MainActivity menu and on click open user’s email application with your email id prefilled.