CHAPTER – 4

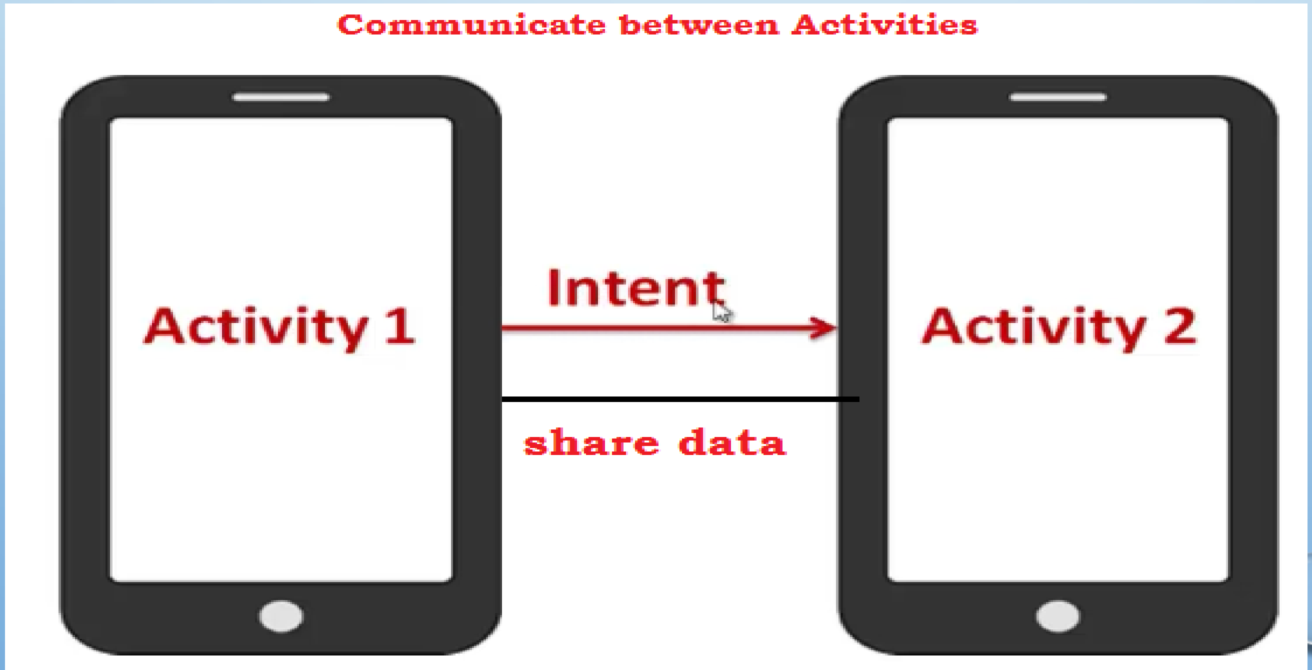
INTENTS

AGENDA

- Implicit intents
- Explicit intents
- Hands on example
  - Explicit event - send a message from one activity to another activity.
  - Implicit event – sending message through e-mail, dial up screen and WhatsApp
- Run apps on real device
- Getting result from the activity

INTRODUCTION

- We’re going to show you how to build apps with multiple activities, and how you can get your apps talking to each other using intents.
- An Intent is a messaging object you can use to request an action from another app component. The only components you have seen so far are Activities, but there are also Services, Broadcast receivers, and Content Providers.
- Intents are multi-purpose communication tools, and the Intent class provides different constructors depending on what you are using the intent to do.



Types of intents

- **Explicit intents** specify the component to start by name (the fully-qualified class name). You'll typically use an explicit intent to start a component in your own app, because you know the class name of the activity or service you want to start. For example, you can start a new activity in response to a user action or start a service to download a file in the background.
- **Implicit intents** do not name a specific component, but instead declare a general action to perform, which allows a component from another app to handle it. For example, if you want to show the user a location on a map, you can use an implicit intent to request that another capable app show a specified location on a map.
- Intents are created using **Intent** class. Following are the major method in Intents class.

setAction()	setData()	putExtra()
setComponent()	setType()	getExtra()

How Intent works

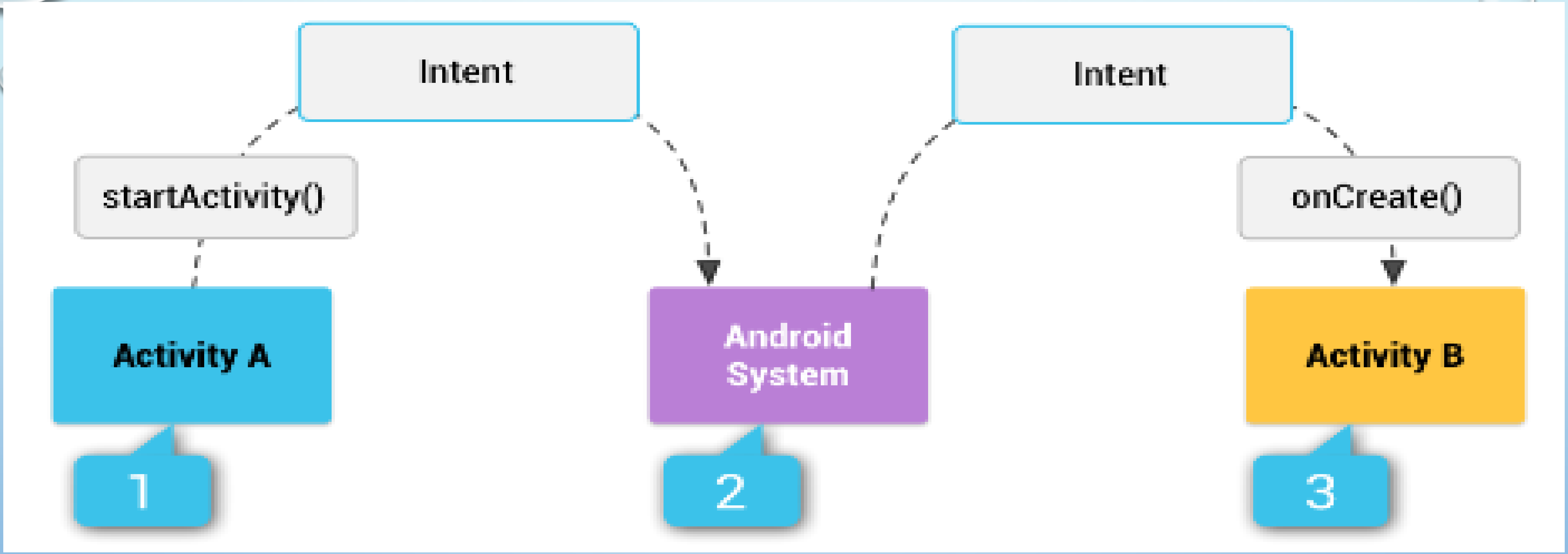
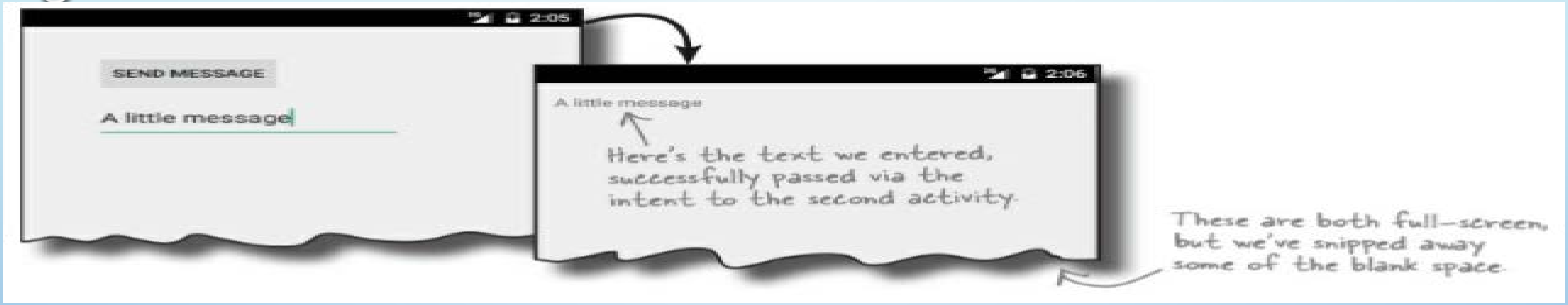
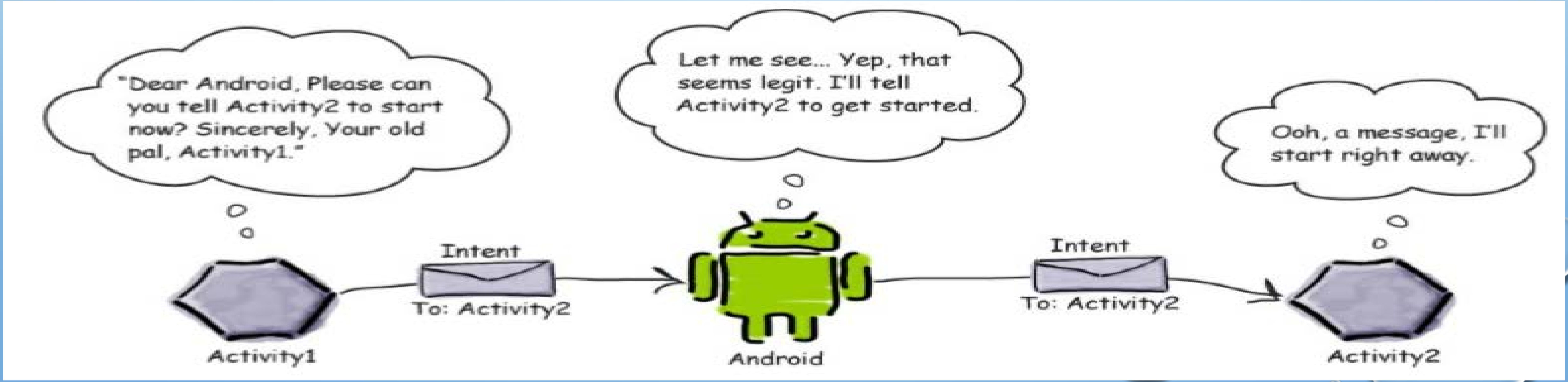


Figure 1. How an implicit intent is delivered through the system to start another activity: [1] activity A creates an intent for Activity B with an action description and passes it to startActivity(). [2] the android system searches all apps for an intent filter that matches the intent. When a match is found, [3]the system starts the matching activity (activity B) by invoking its onCreate() method and passing it the intent.

Hands on Example – Explicit Intent

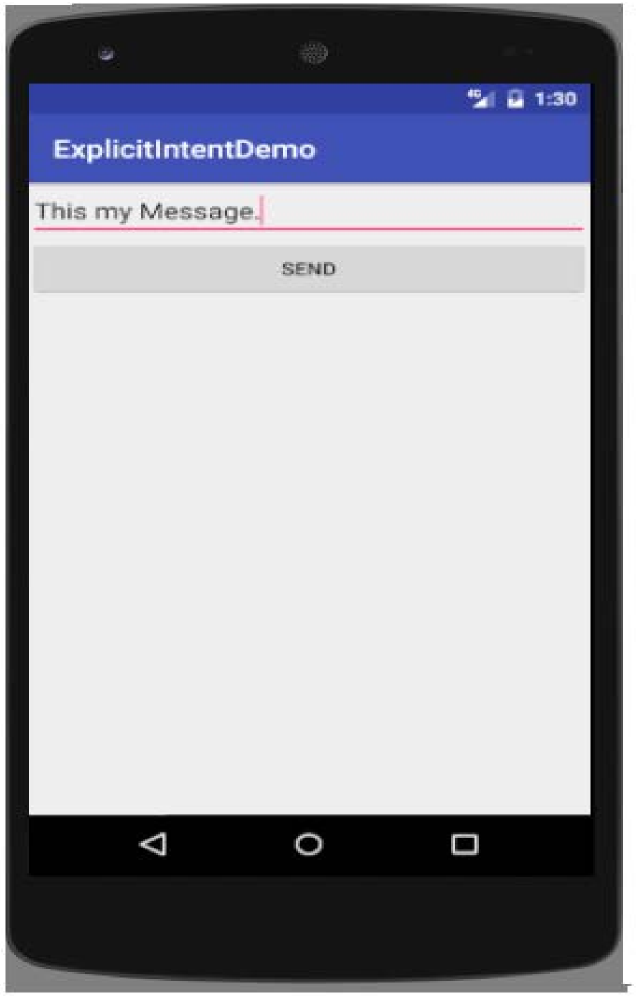
Problem : Send a message from one activity to another activity.


2:05
SEND MESSAGE
A little message
2:06
A little message
Here's the text we entered, successfully passed via the intent to the second activity.
These are both full-screen, but we've snipped away some of the blank space.

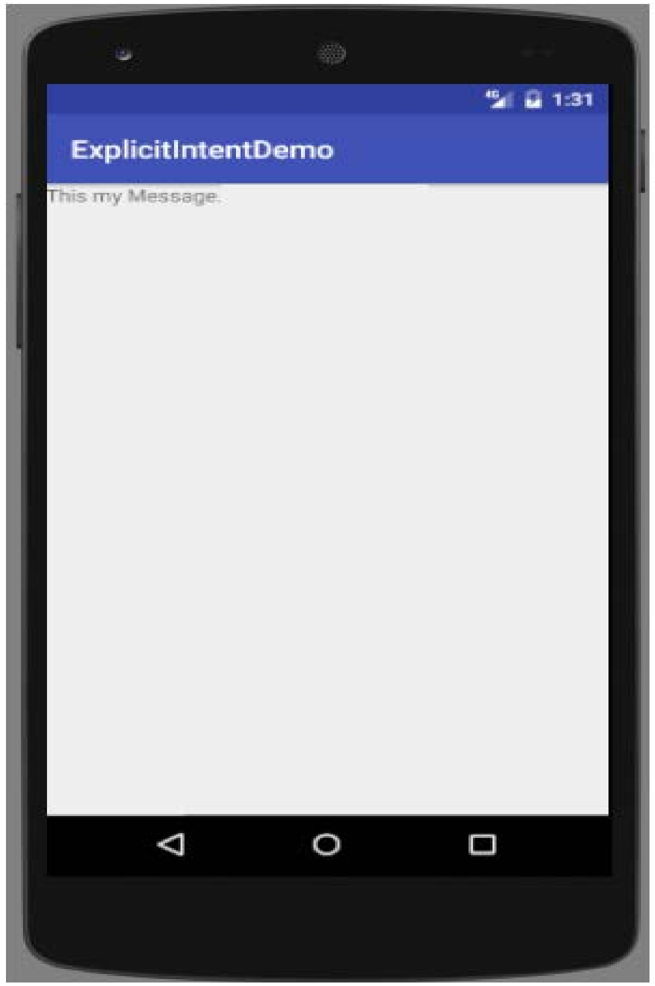

"Dear Android, Please can you tell Activity2 to start now? Sincerely, Your old pal, Activity1."
Let me see... Yep, that seems legit. I'll tell Activity2 to get started.
Ooh, a message, I'll start right away.
Intent
To: Activity2
Intent
To: Activity2
Activity1
Android
Activity2

Problem Requirement

First Activity

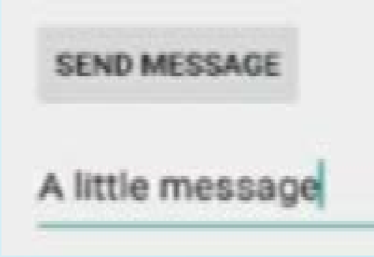


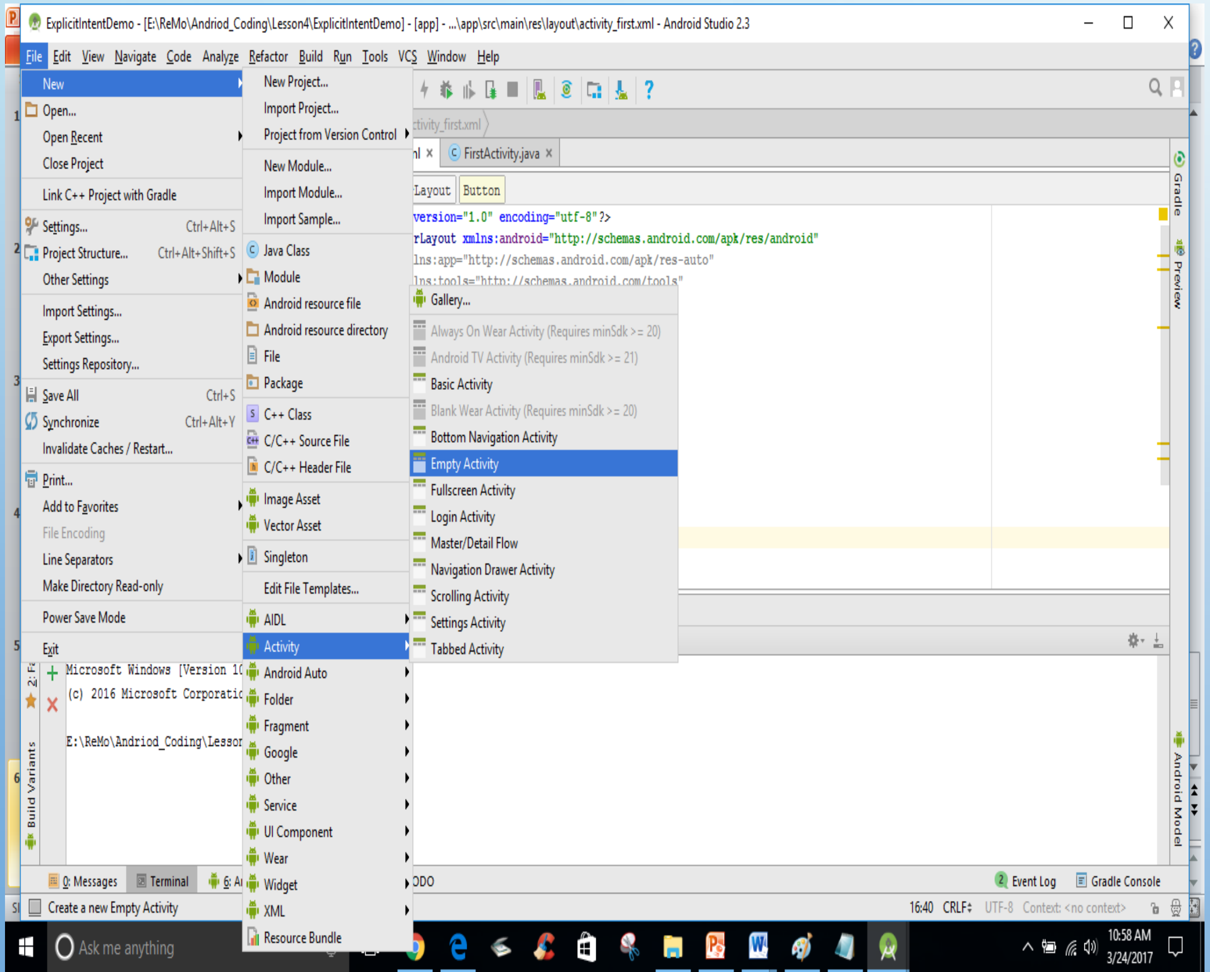
Second Activity



Steps to find the solution

- Step 1: Create your first activity with the controls of one text box and one button.
- Step 2 : Create a second activity with one Text view to receive message from the first activity in the same project by choosing File→ New → Activity and choose the option for Blank Activity

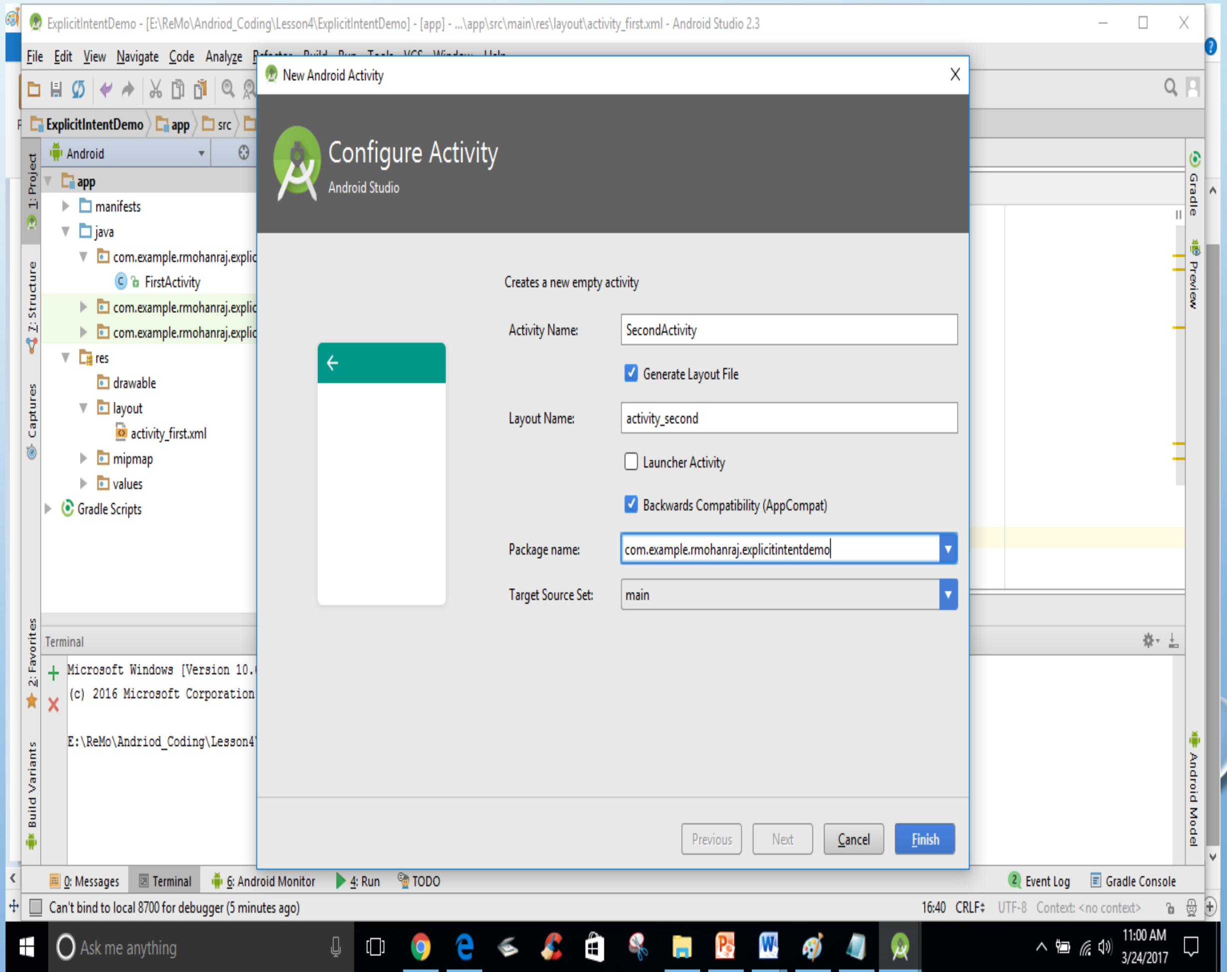




```
version="1.0" encoding="utf-8"?>
rLayout xmlns:android="http://schemas.android.com/apk/res/android"
lns:app="http://schemas.android.com/apk/res-auto"
lns:tools="http://schemas.android.com/tools"
```

```
Microsoft Windows [Version 10
(c) 2016 Microsoft Corporatio

E:\ReMo\Andriod_Coding\Lesson
```



Creates a new empty activity

Activity Name: SecondActivity

[✓] Generate Layout File

Layout Name: activity_second

[ ] Launcher Activity

[✓] Backwards Compatibility (AppCompat)

Package name: com.example.rmohanraj.explicitintentdemo

Target Source Set: main

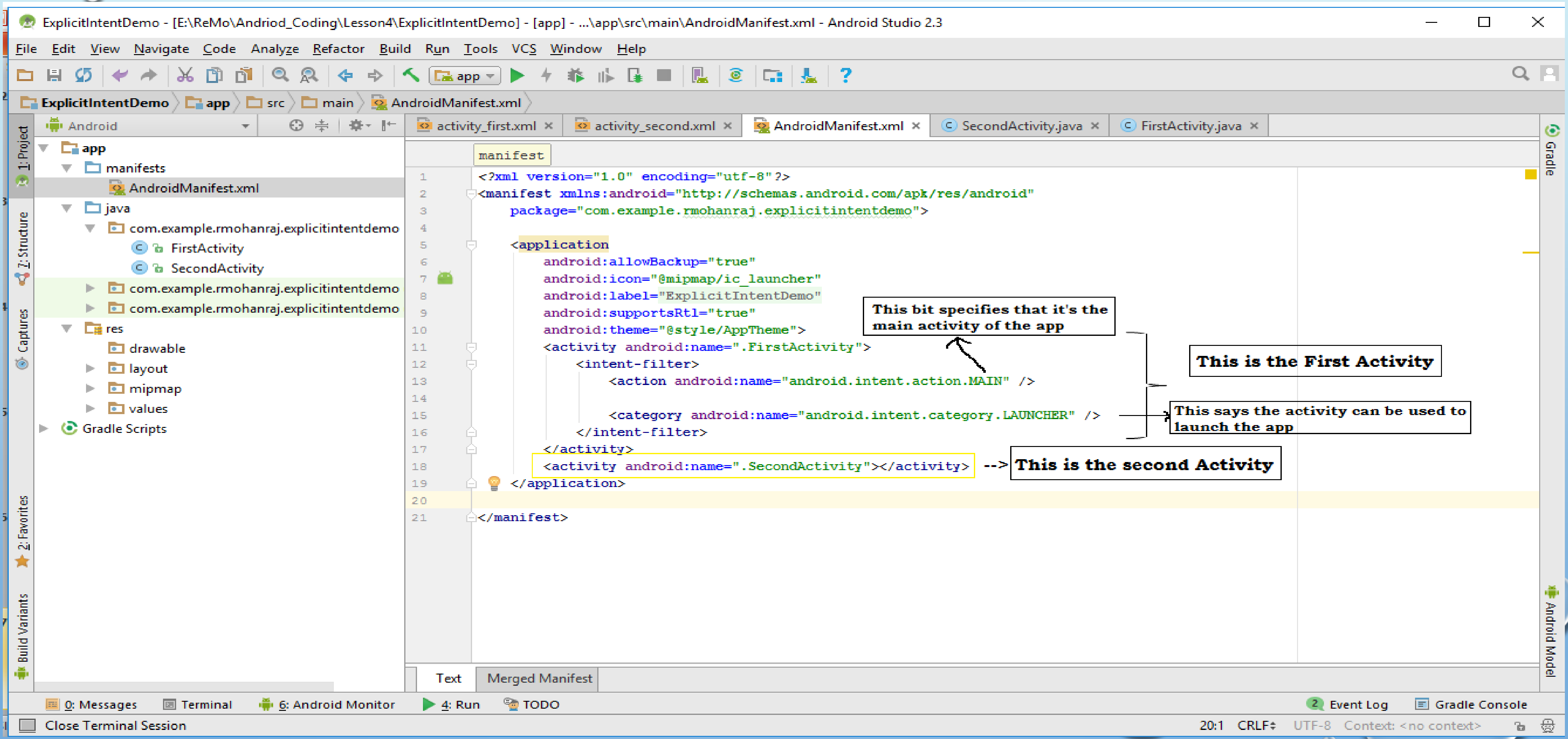
```
Microsoft Windows [Version 10.
(c) 2016 Microsoft Corporation

E:\ReMo\Andriod_Coding\Lesson4
```

Can't bind to local 8700 for debugger (5 minutes ago)

- Step : 3 Configure Second Activity in manifest folder → AndriodManifest.xml and include

  <activity android:name=".SecondActivity"></activity>



ExplicitIntentDemo - [E:\ReMo\Andriod_Coding\Lesson4\ExplicitIntentDemo] - [app] - ...\app\src\main\AndroidManifest.xml - Android Studio 2.3

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.rmohanraj.explicitintentdemo">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="ExplicitIntentDemo"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".FirstActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".SecondActivity"></activity>
    </application>

</manifest>
```

This bit specifies that it's the main activity of the app

This is the First Activity

This says the activity can be used to launch the app

--> This is the second Activity

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ExplicitIntentDemo app src main AndroidManifest.xml
activity_first.xml activity_second.xml AndroidManifest.xml SecondActivity.java FirstActivity.java
Android
app
manifests
AndroidManifest.xml
java
com.example.rmohanraj.explicitintentdemo
FirstActivity
SecondActivity
com.example.rmohanraj.explicitintentdemo
com.example.rmohanraj.explicitintentdemo
res
drawable
layout
mipmap
values
Gradle Scripts
manifest
Text
Merged Manifest
0: Messages
Terminal
6: Android Monitor
4: Run
TODO
Event Log
Gradle Console
Close Terminal Session
20:1 CRLF UTF-8 Context: <no context>

- Step : 4 : Create an Intent and StartActivity

You can create and send an intent using just a couple of lines of code. You start by creating the intent like this:

```
Intent intent = new Intent(this,TargetClassname.class);
```

The first parameter tells Android which object the intent is from, and you can use the word this to refer to the current activity. The second parameter is the class name of the activity that needs to receive the intent. Once you've created the intent object, you pass it to Android like this:

```
startActivity(intent);
```

This tells Android to start the activity specified by the intent. Once Android receives the intent, it checks everything's OK and tells the activity to start. If it can't find the activity, it throws an ActivityNotFoundException.

```
public void onSendMessage(View view){
        Intent intent = new Intent(this,SecondActivity.class);
        startActivity(intent);
  }
```

- Step 5:Pass text to the Second Activity from the First Activity.
  - Put the data into the intent as Key/Value pairs. To do this, you use the putExtra() method

```
    intent.putExtra("message", value);
```

  - Change the onSendMessge() with putExtra().

```
public void onSendMessage(View view){
        et1 = (EditText) findViewById(R.id.smsg);
        String input = et1.getText().toString();
        Intent intent = new Intent(this,SecondActivity.class);
        intent.putExtra("message",input); // Here message is a key to retrieve the input text in the second activity
        startActivity(intent);
    }
```

The putExtra() method is overloaded so value has many possible types. As an example, it can be a primitive such as a boolean or int, an array of primitives, or a String. You can use putExtra() repeatedly to add numerous extra data to the intent. If you do this, make sure you give each one a unique name.

- STEP 6 : Retrieve extra information from an intent in the Second Activity
- There are a couple of useful methods that can help with this. The first of these is **getIntent();**
- getIntent() returns the intent that started the activity, and you can use this to retrieve any extra information that was sent along with it.
- How you do this depends on the type of information that was sent.
- As an example, if you know the intent includes a string value with a name of “message”, you would use the following:

Intent intent = getIntent();

String output = intent.getStringExtra(“message”);

- You’re not just limited to retrieving String values. As an example, you can use

int intNum = intent.getIntExtra("name", default_value);

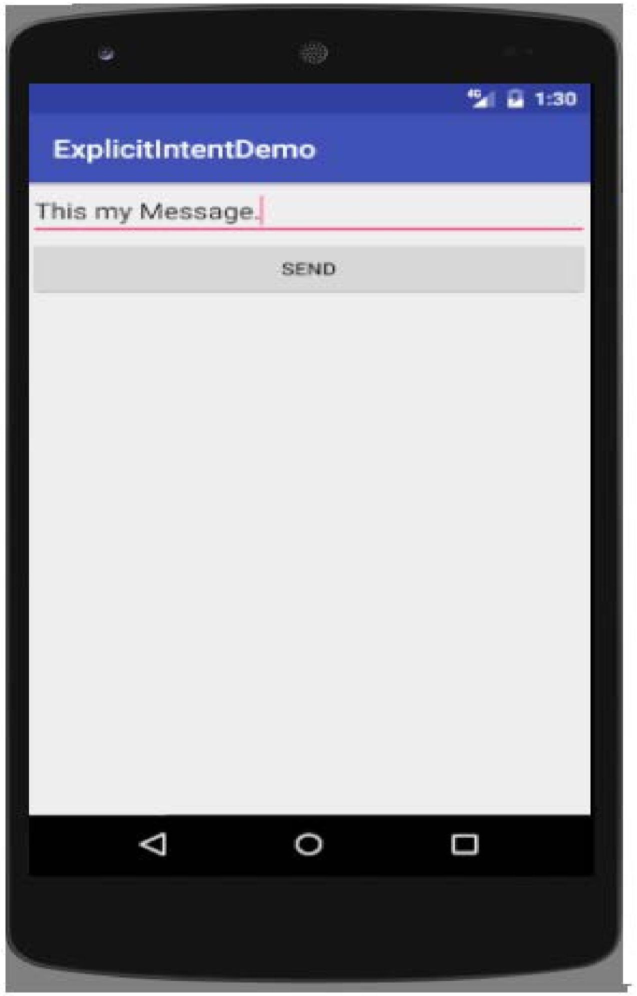
to retrieve an int with a name of name. default_value specifies what int value you should use as a default.

- Write this code in the second activity class.

```
public class SecondActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);
        Intent intent = getIntent();
        String output = intent.getStringExtra("message");
        TextView tv = (TextView) findViewById(R.id.rmsg);
        tv.setText(output);
    }
}
```

- Step 7 : Run your code. After running the app, you will get result as below.

First Activity

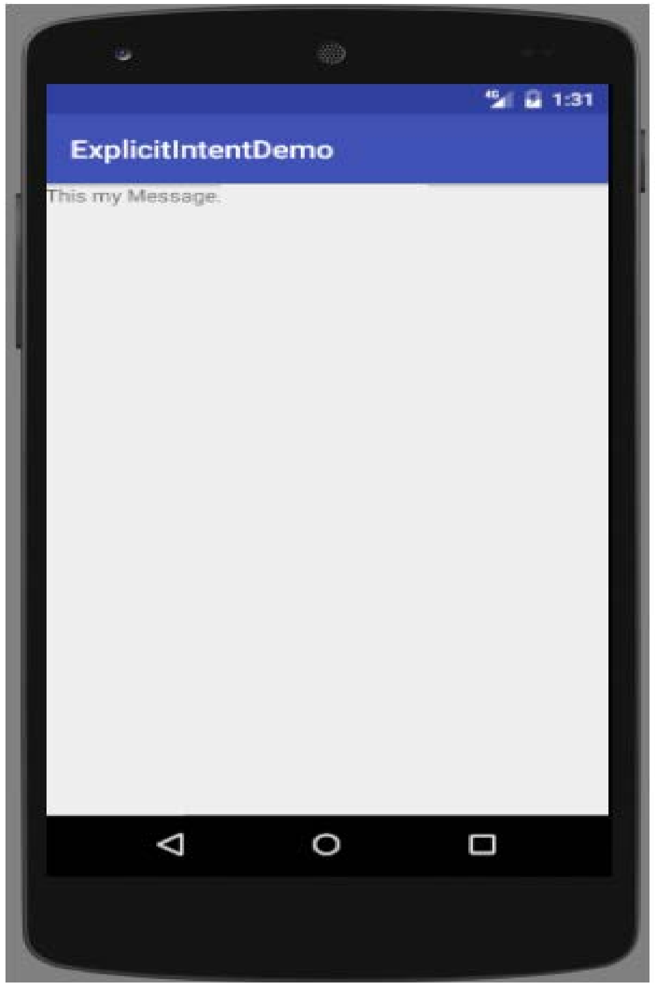


1:30

ExplicitIntentDemo

This my Message.

Second Activity



1:31

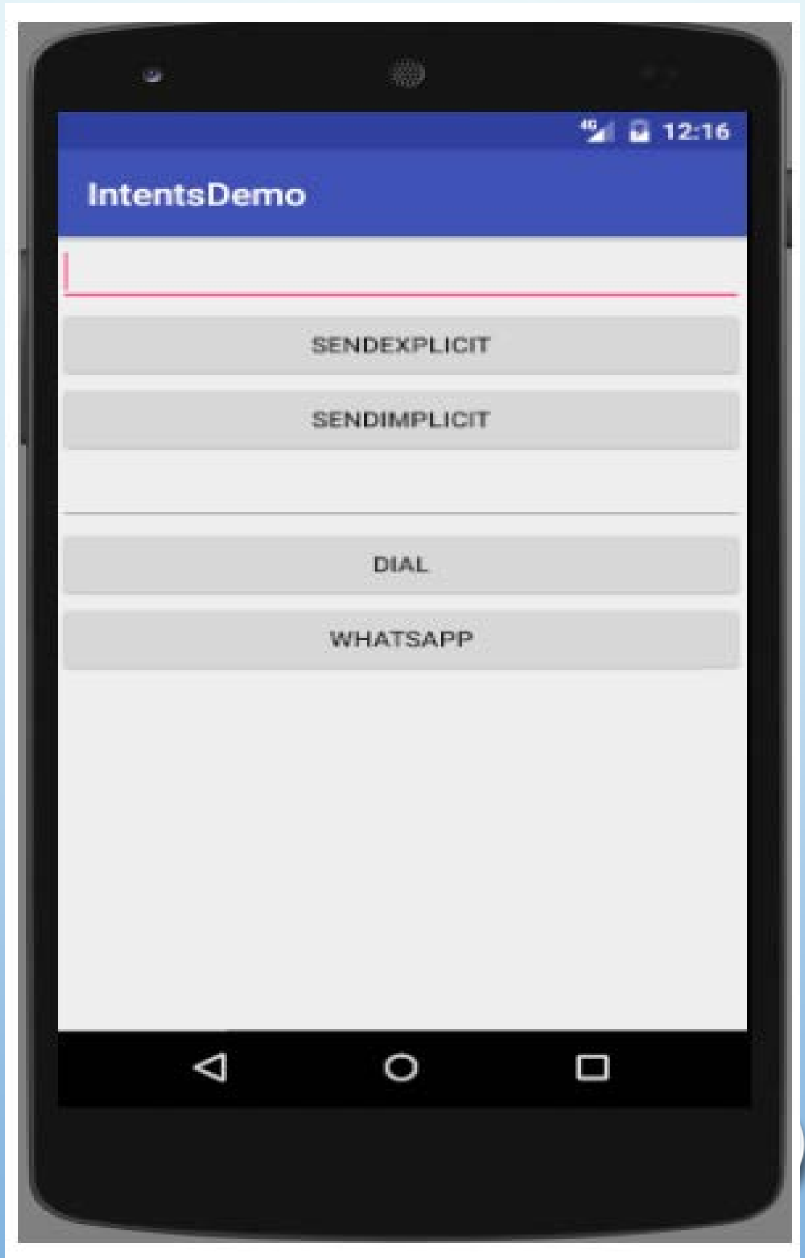
ExplicitIntentDemo

This my Message.

Hands on Example – Implicit Intent

- However, you can also start up activities from the operating system or third-party apps. In those cases, though, you will not have a Java Class object representing the other activity in your project, so you cannot use the Intent constructor that takes a Class.
- Instead, you will use what are referred as the “Implicit” Intent structure.
- Lets enhance the previous example by introducing implicit intents to perform the following.
    - Sending message to the particular person through Email
    - Showing a Dial screen App (**INTENT.ACTION_DIAL)**
    - Showing a WhatsApp App

DESIGN SCREEN



IMPLICIT INTENT TO SEND MESSAGE

XML Code for the Button SendImplicit

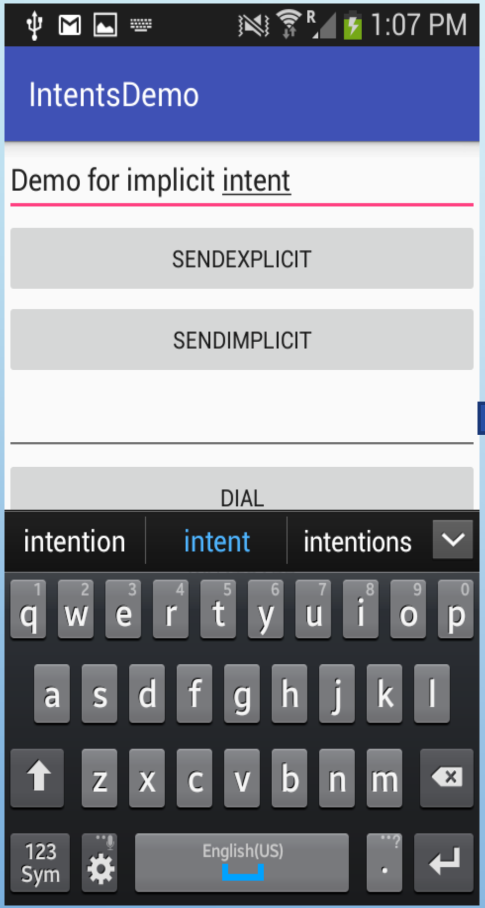
```
<Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/send2"
        android:onClick="onSendMessageImplicit"
        android:text="SendImplicit"/>
```

Java Code for the Button Click Event

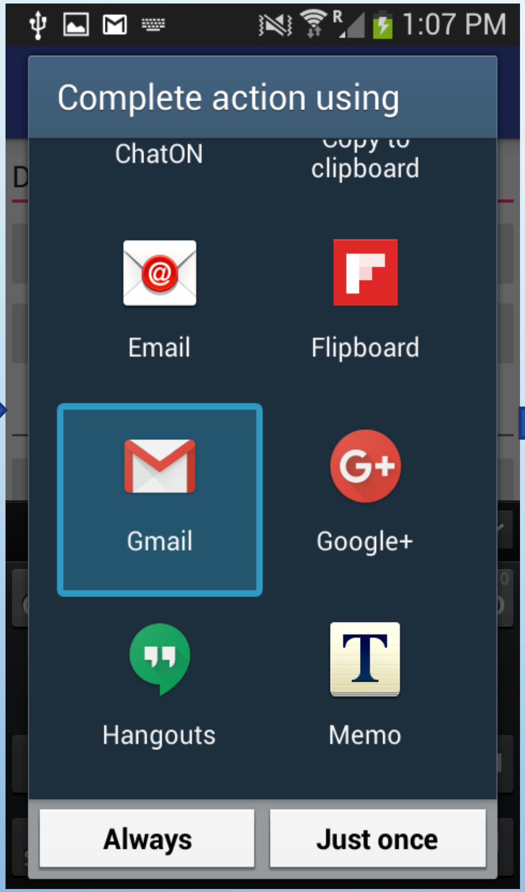
```
public void onSendMessageImplicit(View view){
        et1 = (EditText) findViewById(R.id.smsg);
        String input = et1.getText().toString();
        Intent intent = new Intent();
        intent.setAction(Intent.ACTION_SEND);
        intent.setType("text/plain");
        intent.putExtra(Intent.EXTRA_TEXT,input);
        startActivity(intent);
    }
```

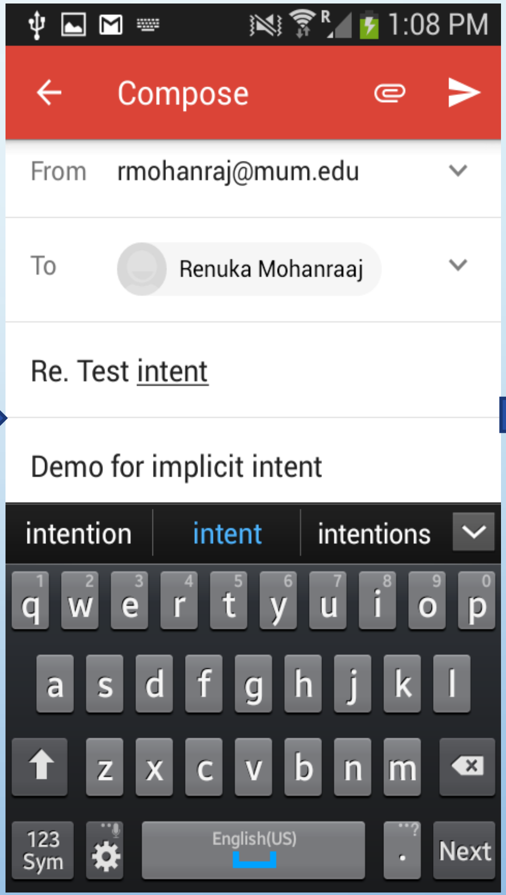
The intent specifies an action of ACTION_SEND, and a MIME type of text/plain.

After clicking **SENDIMPLICIT** button



Demo for implicit intent



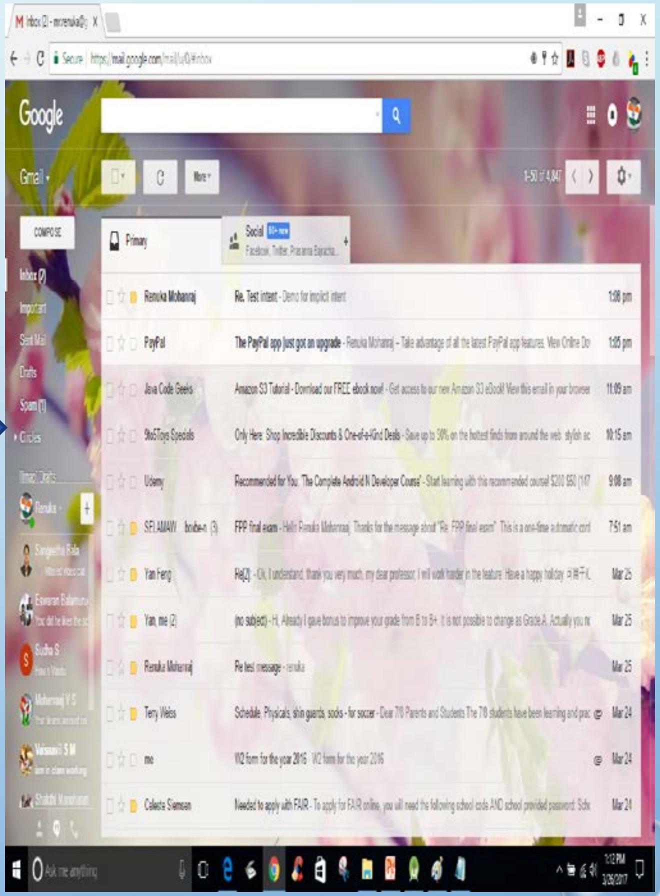


From rmohanraj@mum.edu

To Renuka Mohanraaj

Re. Test intent

Demo for implicit intent



Renuka Mohanraj Re. Test intent - Demo for implicit intent 1:08 pm

PayPal The PayPal app just got an upgrade 1:05 pm

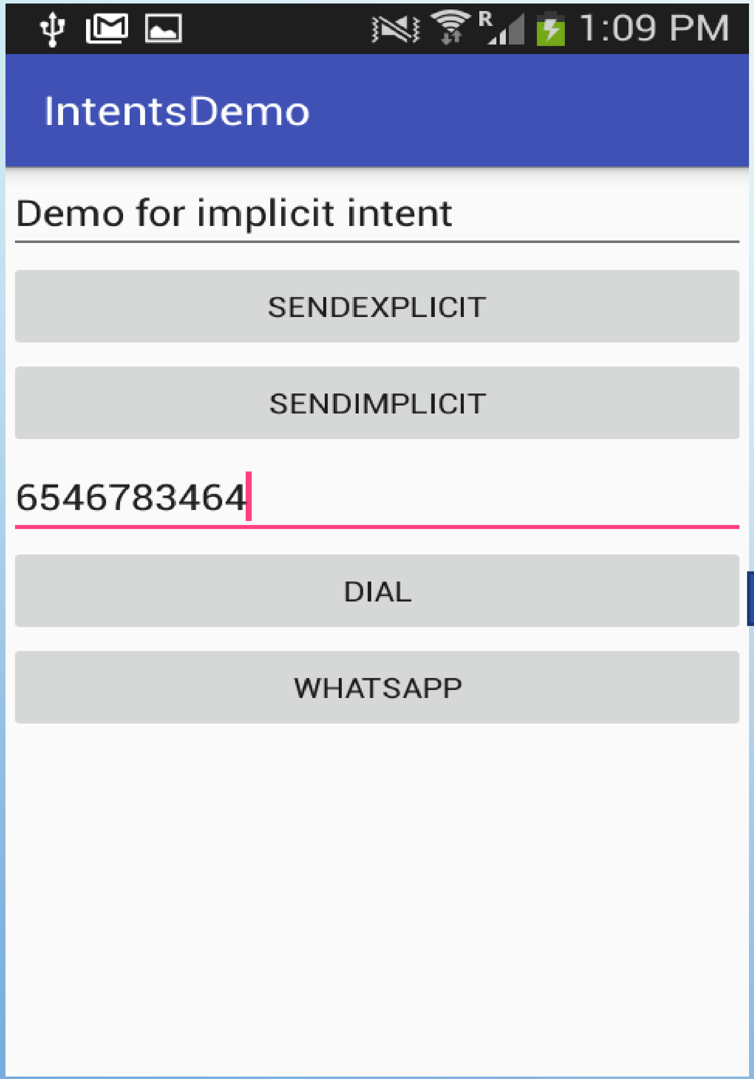
IMPLICIT INTENT – DIAL SCREEN

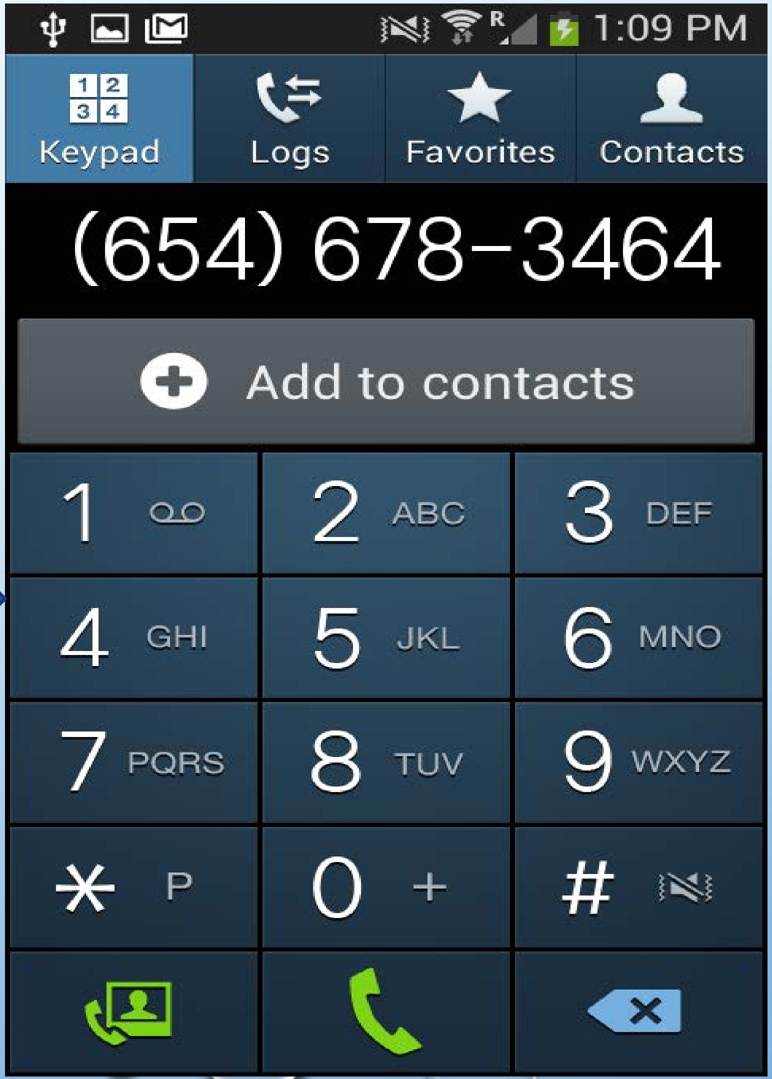
```
<EditText
        android:id="@+id/tel"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
 <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/dl"
        android:onClick="dial"
        android:text="Dial"/>
```

```
public void dial(View view){
        Intent i = new Intent();
        i.setAction(Intent.ACTION_DIAL);
        EditText et2 = (EditText) findViewById(R.id.tel);
        i.setData(Uri.parse("tel:"+et2.getText().toString()));
        startActivity(i);
    }
```

- Whenever the user clicks the DIAL button, calling built-in dial screen by invoking Intent.ACTION_DIAL to call number when the application is launched.
- To send any data to the built in activity use setData(), the parameter for this method is Uri component. To get the Uri component use Uri.parse() method which takes the String input along with what type of data we are sending. Here we are sending telephony type of data which accepts number, #, and *. Ordinary number accepts 0-9.
- tel : A data representing kind of telephone number.

After clicking **DIAL** button

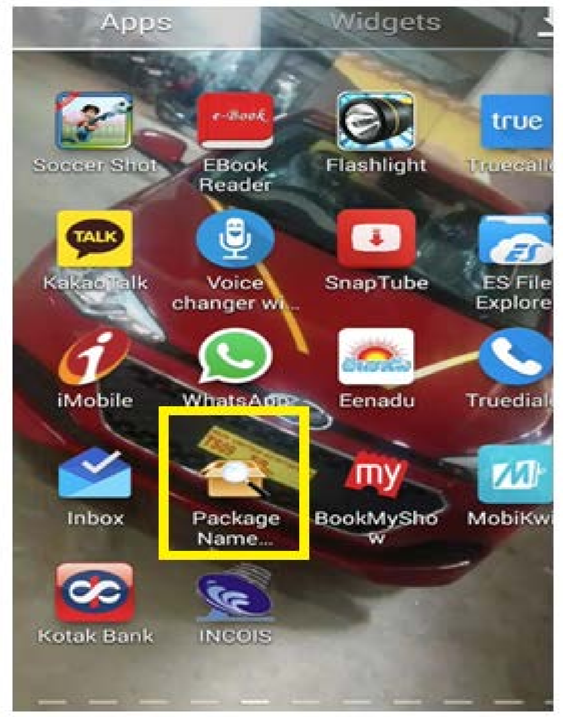


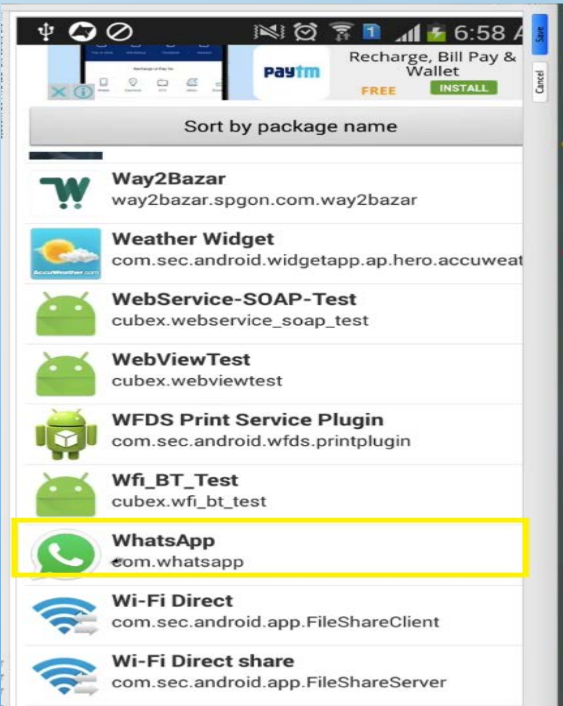


IMPLICIT INTENT-WHATSAPP

```
<Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="whatsapp"
        android:text="WhatsApp"/>
public void whatsapp(View view){
Intent i = getPackageManager().getLaunchIntentForPackage("com.whatsapp");
startActivity(i);
   }
```

If you want to call any third party application in your logic, first you should get the package name from your App called as PackageNameViewer. If you don't have the application install the app from play store. Then see package name and use that in the code.





Sort by package name

Way2Bazar
way2bazar.spgon.com.way2bazar

Weather Widget
com.sec.android.widgetapp.ap.hero.accuweat

WebService-SOAP-Test
cubex.webservice_soap_test

WebViewTest
cubex.webviewtest

WFDS Print Service Plugin
com.sec.android.wfds.printplugin

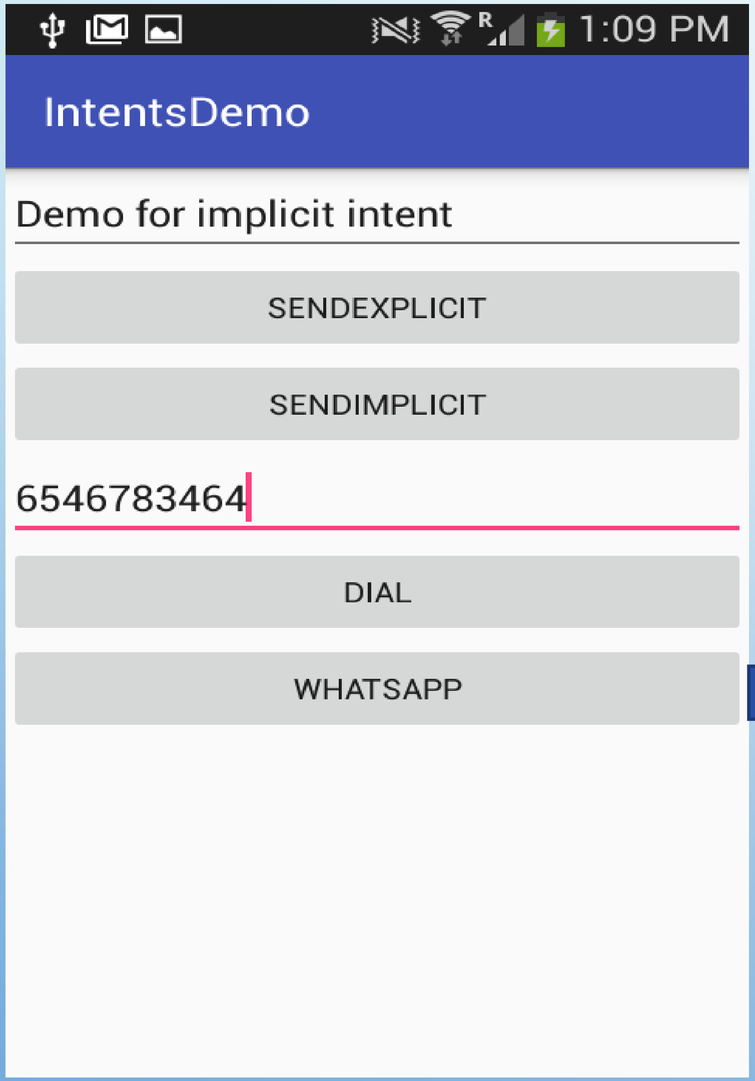
Wfi_BT_Test
cubex.wfi_bt_test

WhatsApp
com.whatsapp

Wi-Fi Direct
com.sec.android.app.FileShareClient

Wi-Fi Direct share
com.sec.android.app.FileShareServer

After clicking **WHATSAPP** Button



1:09 PM

IntentsDemo

Demo for implicit intent

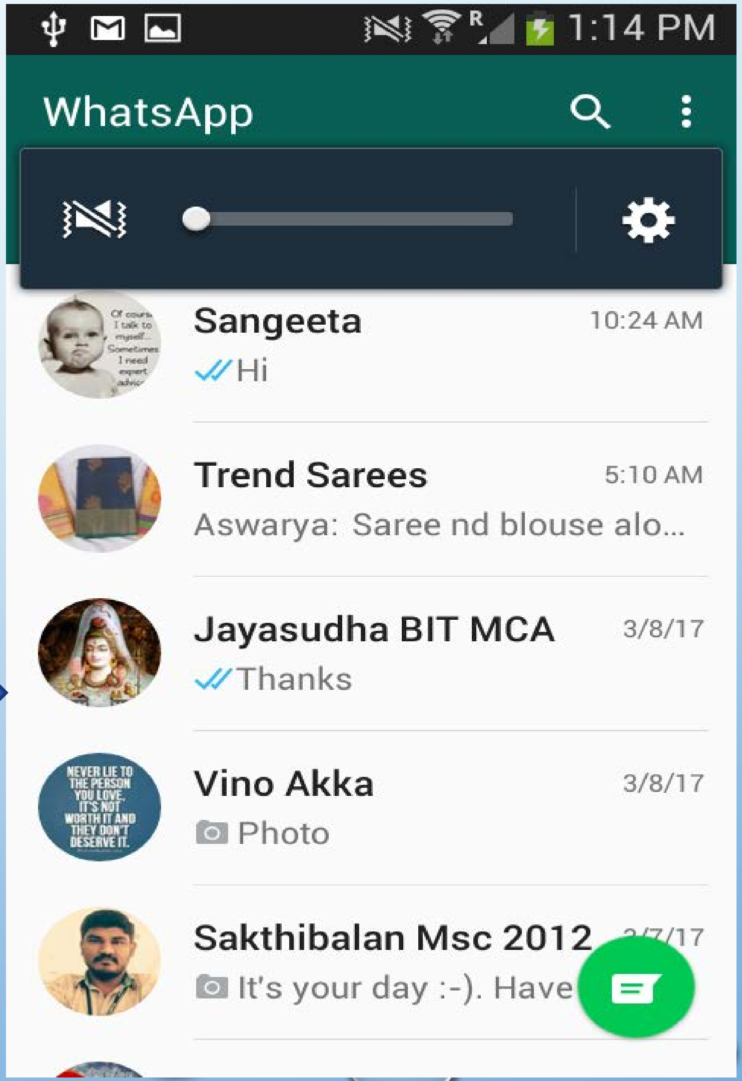
SENDEXPLICIT

SENDIMPLICIT

6546783464

DIAL

WHATSAPP



1:14 PM

WhatsApp

Sangeeta	10:24 AM
Hi	
Trend Sarees	5:10 AM
Aswarya: Saree nd blouse alo...	
Jayasudha BIT MCA	3/8/17
Thanks	
Vino Akka	3/8/17
Photo	
Sakthibalan Msc 2012	3/7/17
It's your day :-). Have	

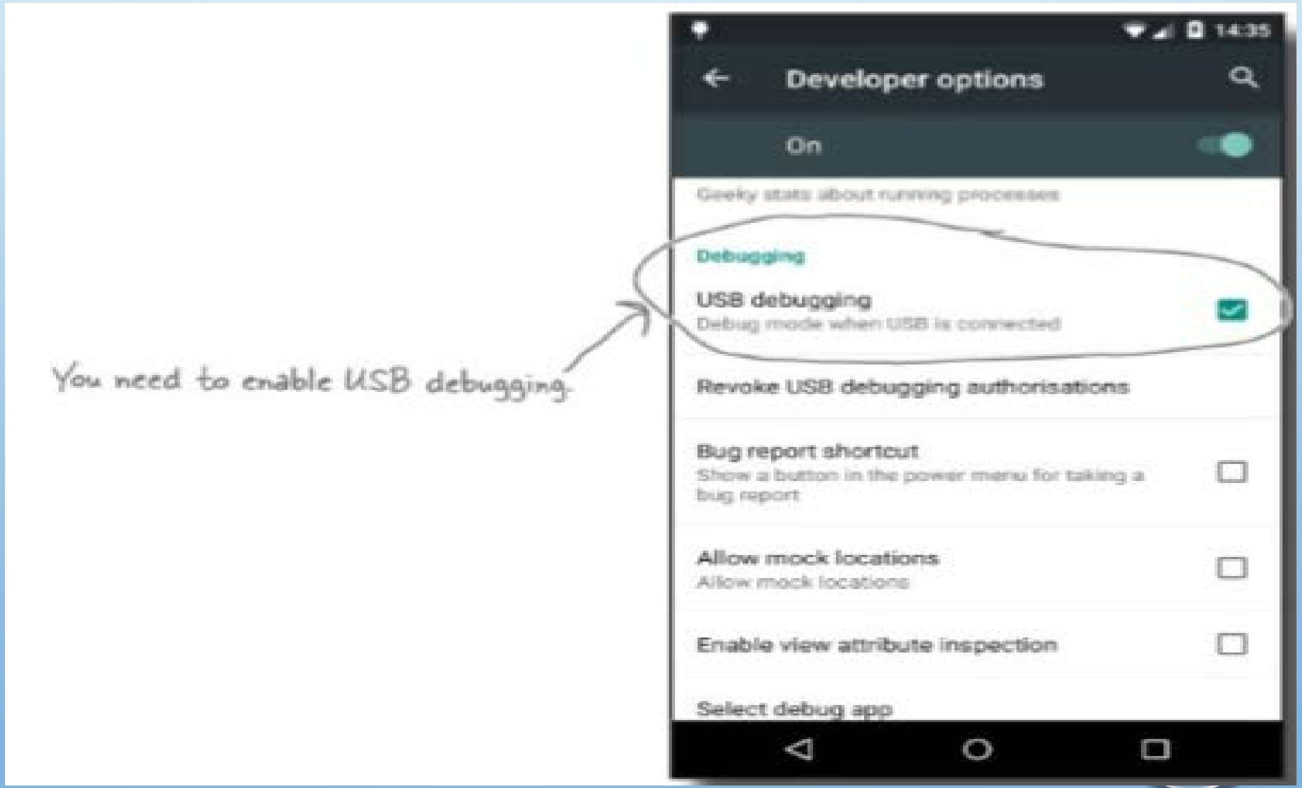
Frequently used Implicit Intents

- Web browser
- Email
- Alarm clock
- Calendar
- Camera
- Contacts/people app
- File storage
- Maps
- Music or video
- Phone
- Settings
- Text messaging

Refer for complete information : **https://developer.android.com/guide/components/intents-common.html**

Run on your Android Mobile device

1. Enable USB debugging on your device
   - On your device, open “Developer options” (in Android 4.0 onward, this is hidden by default). To enable it, go to Settings → About Phone or About Device and tap the build number seven times. When you return to the previous screen, you should be able to see “Developer options.”
   - Within “Developer options,” tick the box to enable USB debugging.


You need to enable USB debugging.
14:35
Developer options
On
Geeky stats about running processes
Debugging
USB debugging
Debug mode when USB is connected
Revoke USB debugging authorisations
Bug report shortcut
Show a button in the power menu for taking a bug report
Allow mock locations
Allow mock locations
Enable view attribute inspection
Select debug app

2. Set up your system to detect your device

- if you're using a mac, you can skip this step.
- If you're using windows, if need to install a USB driver. You can find the latest instructions here:
- http://developer.android.com/tools/extras/oem-usb.html

3. Plug your device into your computer with a USB cable

- Your device may ask you if you want to accept an RSA key that allows USB debugging with your computer. If it does, you can tick the "Always allow from this computer" option and choose OK to enable this.

4. Run your app in Android Studio as normal

- Android Studio will install the app on your device and launch it. You may be asked to choose which device you want to run your app on. If so, select your device from the list available and click OK.

Getting a result back from a child activity

- Starting another activity doesn't have to be one-way. You can also start another activity and receive a result back. To receive a result, call startActivityForResult() (instead of startActivity()).
- For example, your app can start a camera app and receive the captured photo as a result. Or, you might start the People app in order for the user to select a contact and you'll receive the contact details as a result.
- Of course, the activity that responds must be designed to return a result. When it does, it sends the result as another Intent object. Your activity receives it in the onActivityResult() callback.
- When you want to hear back from the child activity, you call the following Activity method:

public void startActivityForResult(Intent intent, int requestCode)

- An Intent that carries the result data.
- A result code specified by the second activity. This is either RESULT_OK if the operation was successful or RESULT_CANCELED if the user backed out or the operation failed for some reason.

Fetching the result

- In the first activity, when the result has been returned, the

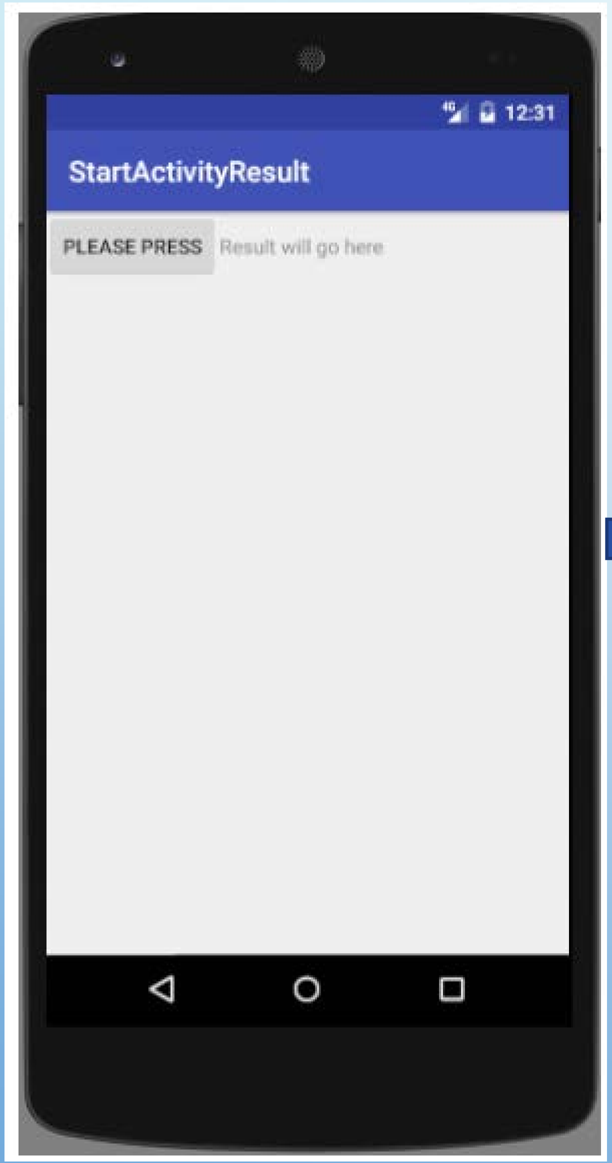
  onActivityResult(int resultCode, int requestCode, Intent intent)

method will be called.

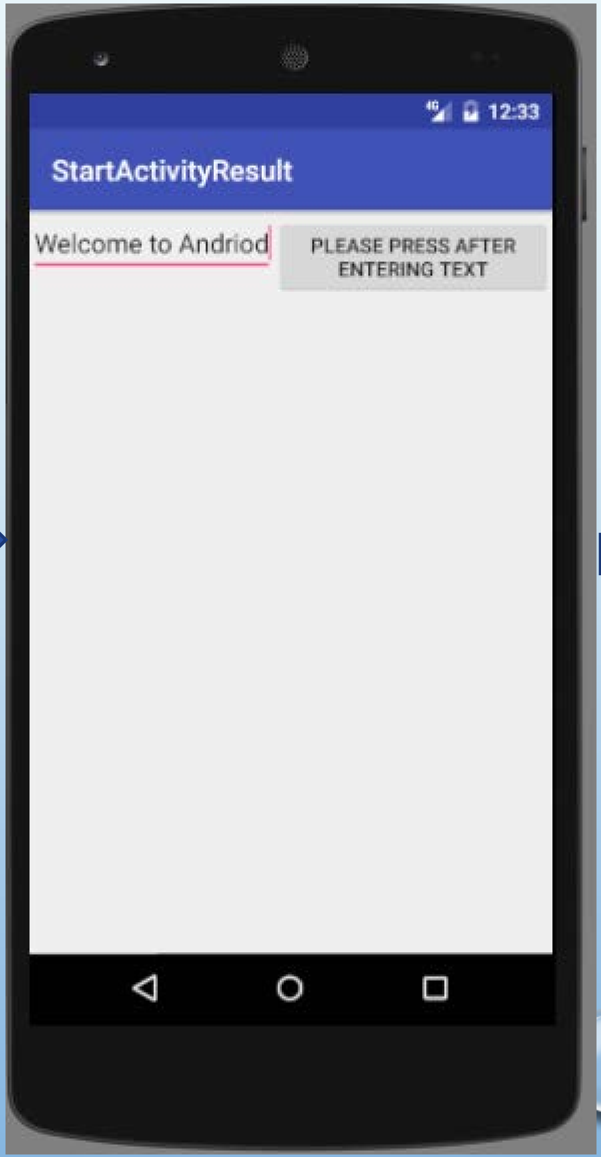
- Check resultCode for Andriod.RESULT_OK
- Check requestCode to see if it matches the int you passed to startActivityForResult.
- Retrieve the data from the intent, and use it.

Fetching Result from Activity B to Activity A

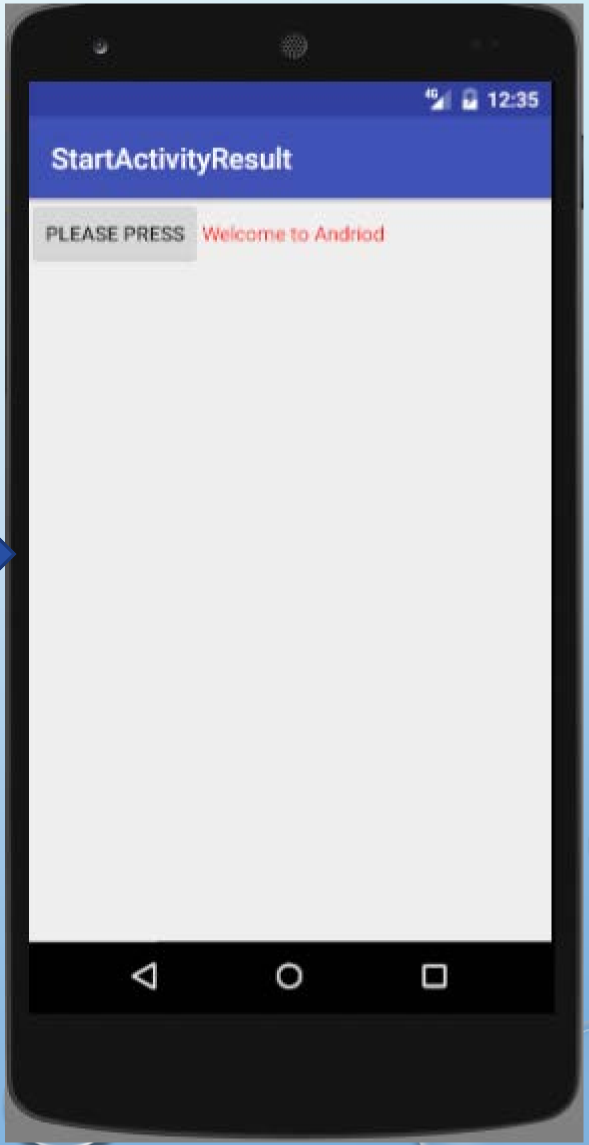
Activity A



Activity B



Activity A



Getting a result back from a child activity

```
// Partial Coding part from the MainActivity.java
View.OnClickListener bListener = new View.OnClickListener(){
        @Override
        public void onClick(View v)
        {
            Intent intent=new Intent(MainActivity.this,SecondActivity.class);
            startActivityForResult(intent, 1); //  Here 1 is the request code
        }
    };
// You should implement this below method in your MainActivity.java, file to need to retrieve the result
public void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == 1) {
            if (resultCode == RESULT_OK) {
                t.setTextColor(Color.RED);
                String returnedResult = data.getData().toString();
                t.setText(returnedResult);
            }
        }
    }
```

Getting a result back from a child activity

// Partial Coding part from the SecondActivity.java

```
View.OnClickListener bListener = new View.OnClickListener(){
    @Override
    public void onClick(View v)
    {
        Intent data = new Intent();
        String text = input.getText().toString();
        // fetch the result to Main Activity
        data.putExtra("Data",text); // Accept a key Value
        setResult(RESULT_OK,data);
       //---close the activity---
        finish();
    }
};
```

// Find the complete code from demo in Lesson4\StartActivityResult folder