

# Mobile App Development

## Intents and Intent Filters



**Dr. Christelle Scharff**  
**cscharff@pace.edu**  
**Pace University, USA**

Code with  
fragments

# Objectives


- Understand, define and use Intents
- Understand, define and use Intent filters
- Understand how to update the Manifest to deal with Intents
- Develop apps composed of several activities
- Understand, define and use menus

**Focus on Intent for  
activities\***

\* Can also be used with BroadcastReceiver

# Intents and Intent Filters

# Intents

- An Intent is a message notifying components (Activity, Service and BroadcastReceiver) of events and actions to generate
- An intent can:
  - **Launch an Activity** 
  - Be sent to any interested `BroadcastReceiver`
  - Communicate with a background `Service`
- <http://developer.android.com/reference/android/content/Intent.html>
- <http://developer.android.com/guide/components/intents-filters.html>

Intent class

# Intent Class Constructors


Public Constructors	
<code>Intent()</code>	Create an empty intent.
<code>Intent(Intent o)</code>	Copy constructor.
<code>Intent(String action)</code>	Create an intent with a given action.
<code>Intent(String action, Uri uri)</code>	Create an intent with a given action and for a given data url.
<code>Intent(Context packageContext, Class&lt;?&gt; cls)</code>	Create an intent for a specific component.
<code>Intent(String action, Uri uri, Context packageContext, Class&lt;?&gt; cls)</code>	Create an intent for a specific component with a specified action and data.

Java:

```
final Intent myIntent = new Intent();
```

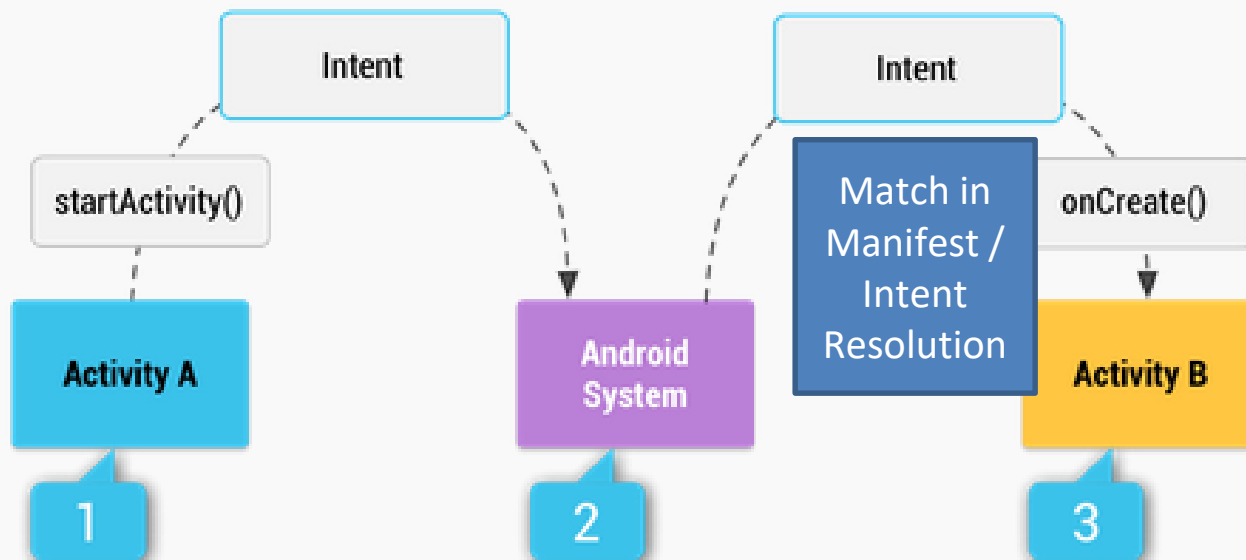
# Intent Filters

- An intent filter is an expression present in the app's **Manifest** file that specifies the type of intents that the component accepts to receive
- <http://developer.android.com/guide/components/intents-filters.html>



```
<activity
    android:name=".MainActivity"
    android:label="CS6392015HelloWorldFrag" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" /
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

# Processing of Intents



**Figure 1.** Illustration of how an implicit intent is delivered through the system to start another activity: **[1]** Activity A creates an `Intent` 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`.

`startActivity`  
method

# Intents' Properties

- An Intent is determined by:
  - An **action** to be performed (e.g., ACTION\_MAIN, ACTION\_VIEW, ACTION\_DIAL)
  - The **data** that the action needs to operate on
  - The data is generally expressed as a `Uri` to be parsed
  - <http://developer.android.com/reference/android/net/Uri.html>
- Examples of Uri:
  - tel: 12123461200 for a telephone number
  - http://google.com for a URL
  - content://contacts/people/1 for a contact
- <http://developer.android.com/reference/android/content/Intent.html>

setData method



# Intent Actions

- ACTION\_MAIN
- ACTION\_VIEW
- ACTION\_ATTACH\_DATA
- ACTION\_EDIT
- ACTION\_PICK
- ACTION\_CHOOSER
- ACTION\_GET\_CONTENT
- ACTION\_DIAL
- ACTION\_CALL
- ACTION\_SEND
- ACTION\_SENDTO
- ACTION\_ANSWER
- ACTION\_INSERT
- ACTION\_DELETE
- ACTION\_RUN
- ACTION\_SYNC
- ACTION\_PICK\_ACTIVITY
- ACTION\_SEARCH
- ACTION\_WEB\_SEARCH
- ACTION\_FACTORY\_TEST

- The constant value of Java ACTION\_MAIN in the Manifest file is:
  - android.intent.action.MAIN
- <http://developer.android.com/reference/android/content/Intent.html>

Java / XML

# Declaring and Starting an Intent

```
final Intent intent = new Intent(<ACTION>);  
  
intent.setData(<DATA>);  
  
startActivity(intent);
```

# Questions

- What action does ACTION\_MAIN perform?
- What action does ACTION\_CALL perform?
- What action does ACTION\_VIEW perform?

# Intents' Properties

- An Intent is, in addition, determined by:
  - A **category** to provides more information about the action to execute and the type of component that would handle the intent (e.g., CATEGORY\_LAUNCHER, CATEGORY\_DEFAULT)
  - A **type** to specify explicitly the type of the Intent data (MIME type) (e.g., plain/text)
  - A **component** to start
    - If specified, the Intent is **explicit**, otherwise it is **implicit**
  - **Extras** are key-value pairs data useful to accomplish the requested action
    - For ACTION\_SEND, you can specify the "to" recipient with the EXTRA\_EMAIL key

setCategory  
method

addType method

putExtra and  
putExtras  
methods

# Intent Category

- CATEGORY\_DEFAULT
- CATEGORY\_BROWSABLE
- CATEGORY\_TAB
- CATEGORY\_ALTERNATIVE
- CATEGORY\_SELECTED\_ALTERNATIVE
- CATEGORY\_LAUNCHER
- CATEGORY\_INFO
- CATEGORY\_HOME
- CATEGORY\_PREFERENCE
- CATEGORY\_TEST
- CATEGORY\_CAR\_DOCK
- CATEGORY\_DESK\_DOCK
- CATEGORY\_LE\_DESK\_DOCK
- CATEGORY\_HE\_DESK\_DOCK
- CATEGORY\_CAR\_MODE
- CATEGORY\_APP\_MARKET

- <http://developer.android.com/guide/topics/manifest/category-element.html>
- [http://developer.android.com/reference/android/content/Intent.html#CATEGORY\\_ALTERNATIVE](http://developer.android.com/reference/android/content/Intent.html#CATEGORY_ALTERNATIVE)

# Declaring and Starting an Intent

```
final Intent intent = new Intent(<ACTION>);  
  
intent.setData(<DATA>);  
  
intent.setCategory(<CATEGORY>);  
  
intent.addType(<TYPE>);  
  
intent.putExtra(<key>, <value>);  
  
startActivity(intent);
```

# Android Permissions

- Permissions are often required to launch particular actions (e.g., sending an SMS, making a call, using the Internet)
- Permissions are specified in the **Manifest** file
- Permissions are granted by the users when apps are installed, not while they are running



\$

```
<uses-permission android:name="string"  
                android:maxSdkVersion="integer" />
```

- <http://developer.android.com/guide/topics/manifest/permission-element.html>
- <http://developer.android.com/guide/topics/manifest/uses-permission-element.html>
- <http://developer.android.com/reference/android/Manifest.permission.html>

# Intent to Send an SMS

- Java

```
final Intent intent = new Intent(Intent.ACTION_SENDTO);  
intent.setData(Uri.parse("smsto:" + Uri.encode("12123630830")));  
intent.putExtra("sms_body", "hello");  
startActivity(intent);
```

- Permission in the Manifest file

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



Permission to send  
SMS



# Intent to Call a number

- Java

```
final Intent intent = new Intent(Intent.ACTION_DIAL);  
intent.setData(Uri.parse("tel:2123630830"));  
startActivity(intent);
```

- Permission in the Manifest file

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



Permission to call

# Intent to Launch the Browser

- Java

```
final Intent websiteIntent = new Intent(Intent.ACTION_VIEW,  
    Uri.parse("http://google.com"));  
startActivity(websiteIntent);
```

- Permission in the Manifest file

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



Permission to use  
the Internet

# Intent to Launch Google Maps

- There are 2 ways to view a location with Google Maps:
  - Use a MapView
  - Use a location Uri to launch the built-in Google Maps app with the specified location



```
String geoUri = String.format("geo:38.899533,-77.036476");
Uri geo = Uri.parse(geoUri);
Intent geoMap = new Intent(Intent.ACTION_VIEW, geo);
startActivity(geoMap);
```

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

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

# Lab

- What are the latitude and longitude of Pace University, One Pace Plaza, NYC, NY?

# Question

- What is/are the difference/s between ACCESS\_COARSE\_LOCATION and ACCESS\_FINE\_LOCATION?

# Overview of Common Intents

Alarm Clock

Calendar

Camera

Contacts/People App

Email

File Storage

Maps

Music or Video

Phone

Settings

Text Messaging

Web Browser

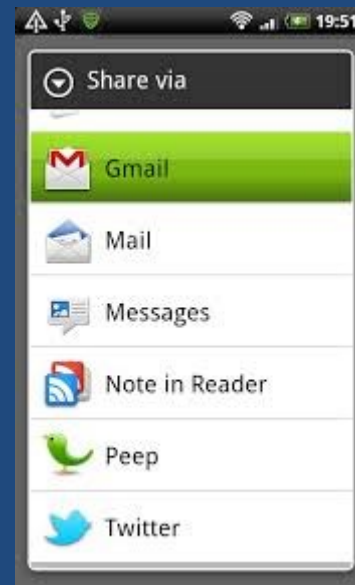
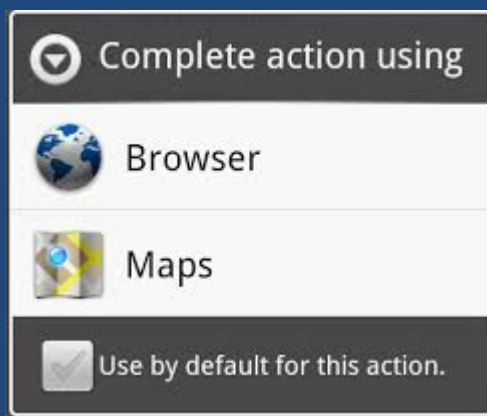
- <http://developer.android.com/guide/components/intents-common.html>

# Questions

- What is the String to use in `putExtra` (`String`, `Bundle`) of the class `Intent` to populate the subject of an SMS?
- What is the String to use in `putExtra` (`String`, `Bundle`) of the class `Intent` to populate the title of an activity chooser?

# Chooser Intent

- The chooser intent permits users to pick what they want to do before proceeding
- It is used when users are prompted to choose how they want to share, send etc information





# Chooser Intent

- Java

```
final Intent intent = new Intent(Intent.ACTION_SEND);  
intent.setType("text/plain");  
intent.putExtra(Intent.EXTRA_SUBJECT, "CS639");  
intent.putExtra(Intent.EXTRA_TEXT, "Join CS639");  
startActivity(Intent.createChooser(intent, "Share the love"));
```

- Permissions in the Manifest file

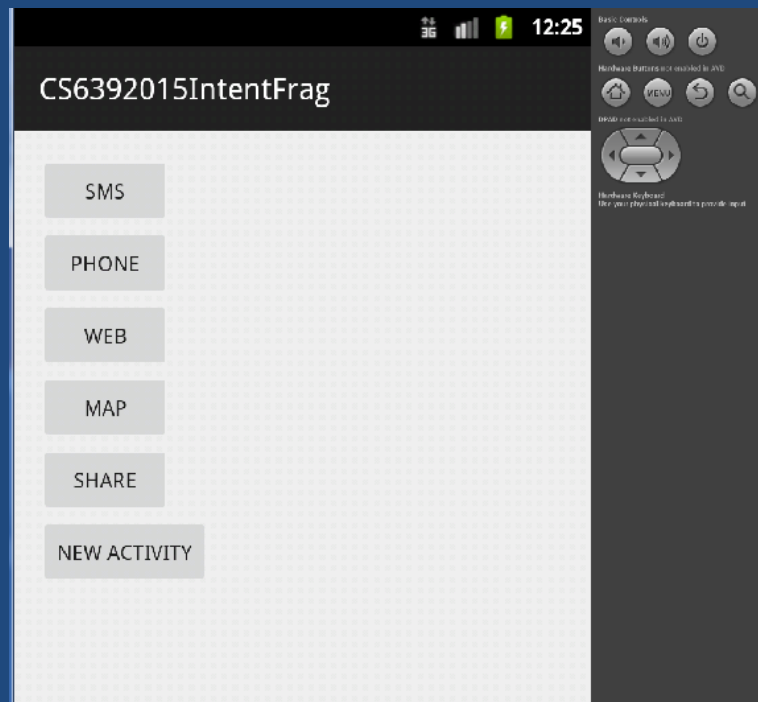


Add the required  
permissions

# Lab

(also your assignment)

- Create an app with buttons SMS, Phone, Web, Map, Share and New Activity that will launch the dedicated associated apps



# Apps with Several Activities

# Launching an Activity

MyActivity launches NewActivity

1<sup>st</sup> method

- Java

Code within an  
Activity

The manifest is  
automatically  
updated

```
final Intent intent = new Intent(getActivity(), NewActivity.class);  
startActivity(intent);
```

Code within a  
Fragment

- Manifest file

```
<activity android:name=".NewActivity"  
android:label="@string/title_activity_new" >  
</activity>
```

# Launching an Activity

## MyActivity launches NewActivity 2<sup>nd</sup> Method

- Java

```
final Intent intent = new Intent("com.cs619.tryintent");  
startActivity(intent);
```

The manifest is automatically updated

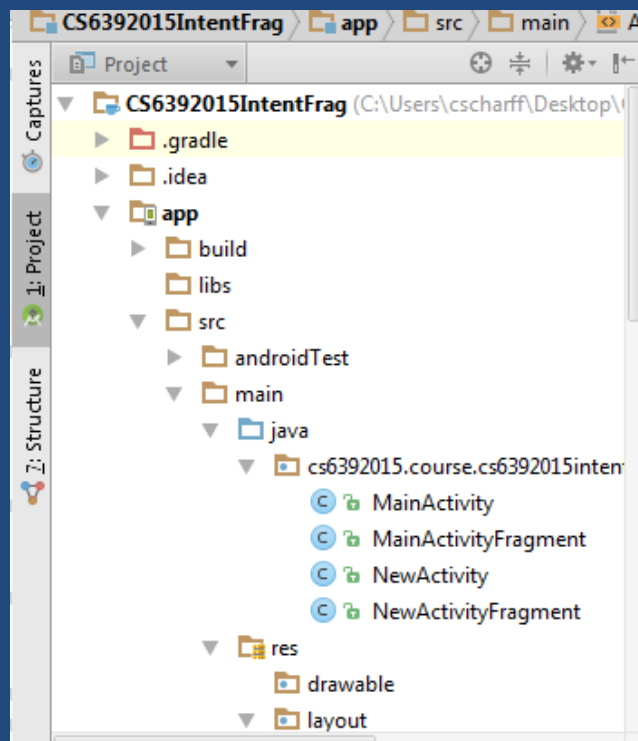
- Manifest file

Name provided by the developer

```
<activity  
    android:name=".NewActivity" ←  
    android:label="@string/title_activity_new" >  
    <intent-filter>  
        <action android:name="com.cs619.tryintent"></action> ←  
        <category android:name="android.intent.category.DEFAULT" />  
    </intent-filter>  
</activity>
```

To launch  
NewActivity

# Project and AndroidManifest.xml



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="cs6392015.course.cs6392015intentfrag" >

    <uses-permission android:name="android.permission.SEND_SMS" />
    <uses-permission android:name="android.permission.CALL_PHONE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity android:name=".NewActivity"
            android:label="@string/title_activity_new" >
        </activity>

    </application>

</manifest>
```

# Lab

(also part of your assignment)

- Create an app that launches an Activity called `NewActivity` when a button called *New Activity* is pressed

Menus

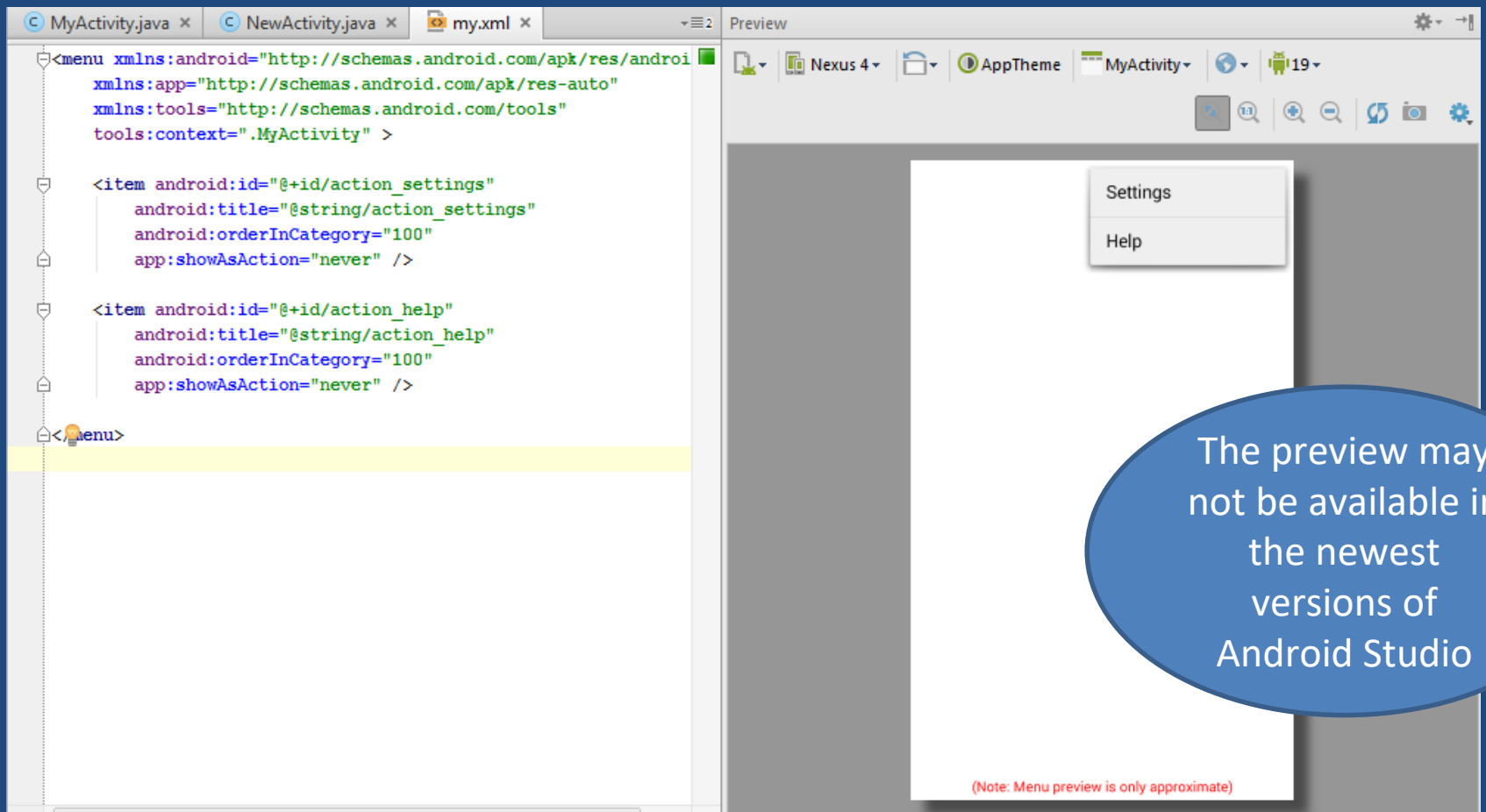


# Menus

- Menus can be associated to activities and views
- There are different types of menus:
  - **Options menu** – Primary collection of menu items of an activity. They appear in the action bar (at the bottom of the screen for earlier versions of Android)
  - **Context menu** – Floating menu that appears with a long click on an element
  - **Popup menu** – Menu associated to a view appearing as a list
- Menus are defined in XML files in /res/menu and are compiled into the application package at built time
- Menus can be dynamically changed based on application states

# Creating an Options Menu in XML

- In the res/menu XML files

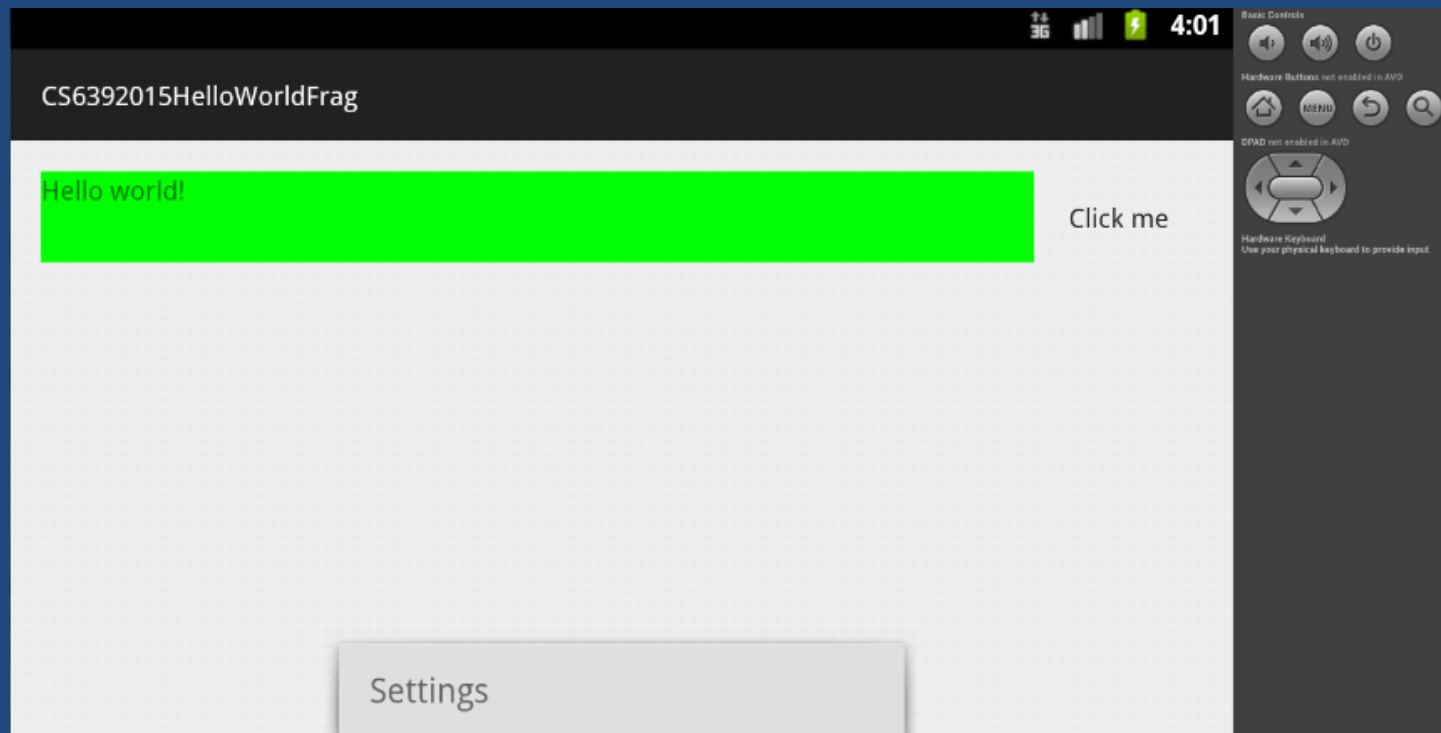


# Creating an Options Menu in Java

- The options menu can be declared in the activity or the fragment
- It is done in the method `onCreateOptionsMenu` of the `Activity` / `Fragment` class
- The XML code is inflated to create the menu

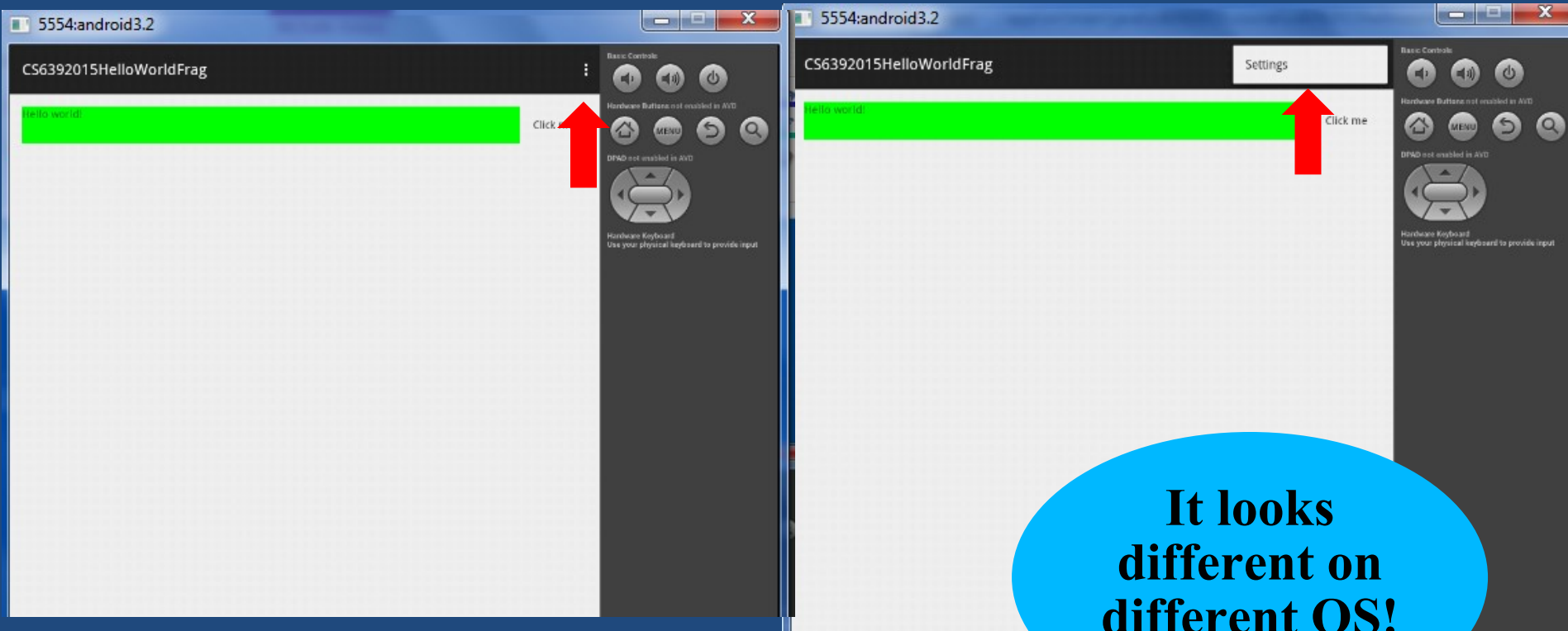
```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}
```

# An Option Menu on Android OS 2.3.3



**It looks  
different on  
different OS!**

# An Option Menu on Android OS 3.2



# Handling Clicks on an Option Menu

- It is done in the method `onOptionsItemSelected` of the `Activity / Fragment` class and by distinguishing the menu items

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        Toast t = Toast.makeText(this, "Settings", Toast.LENGTH_SHORT);
        t.show();
        return true;
    } else if (id == R.id.action_help) {
        Toast t = Toast.makeText(this, "Help", Toast.LENGTH_SHORT);
        t.show();
        return true;
    }

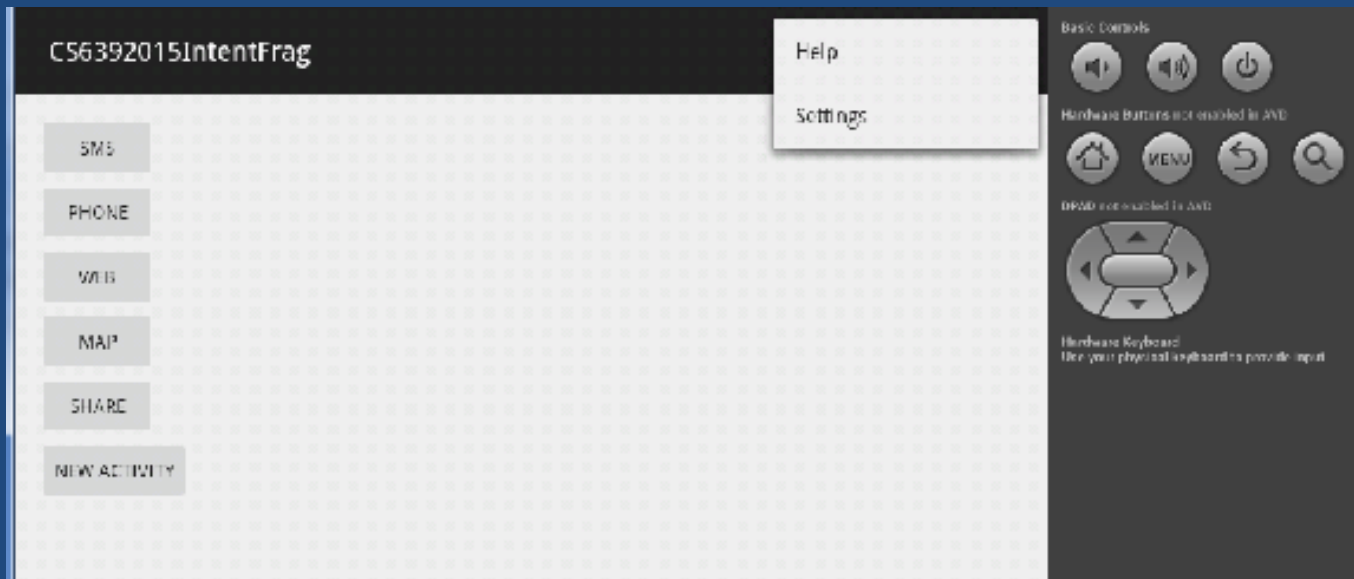
    return super.onOptionsItemSelected(item);
}
```

# Lab

(also part of your assignment)

- Create an app that launches an Activity called *HelpActivity* when a menu item called *Help* is pressed

# Creating a Menu





# Menus in Java

Type of menu	Defining the menu	Handling clicks in menu items
Options menu	<code>public boolean onCreateOptionsMenu( Menu menu)</code>	<code>public boolean onOptionsItemSelected( MenuItem item)</code>
Context menu	<code>public void onCreateContextMenu(C ontextMenu menu, View v, Context MenuInfo menuInfo)</code>	<code>public boolean onContextItemSelected( MenuItem item)</code>
Popup menu	<code>public void showPopup(View v)</code>	<code>public boolean onMenuItemClick(Men uItem item)</code>

# References

- <http://developer.android.com/reference/android/content/Intent.html>
- <http://developer.android.com/guide/components/intents-filters.html>
- <http://developer.android.com/guide/topics/ui/menus.html>
- <http://developer.android.com/reference/android/view/Menu.html>