CPSC 441 – Mobile App Development
Lecture #2

My first Application

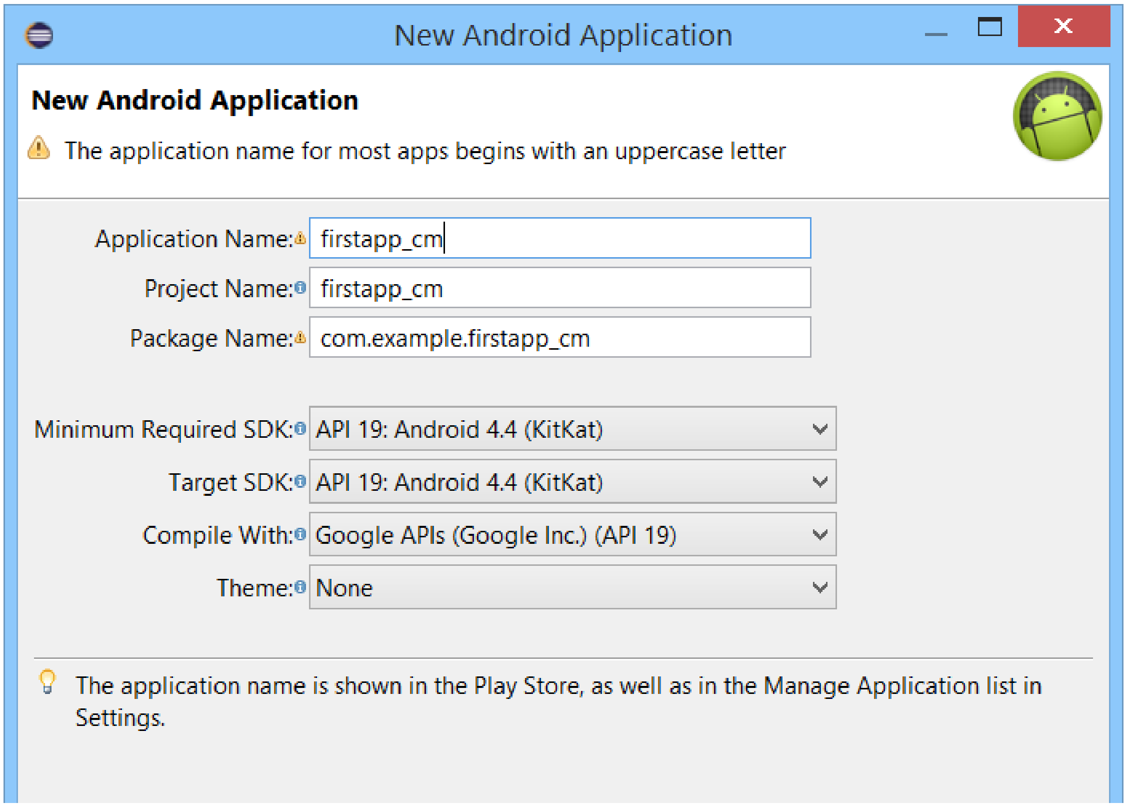
Goal: Create a simple application that shows your name, ID and picture.

Things to have done prior to this:

- Install android and run the hello world app (Lecture 1)

Steps:

Open eclipse-> File-> new->Project->android application project and use these values:



New Android Application

⚠ The application name for most apps begins with an uppercase letter

Application Name: firstapp_cm

Project Name: firstapp_cm

Package Name: com.example.firstapp_cm

Minimum Required SDK: API 19: Android 4.4 (KitKat)

Target SDK: API 19: Android 4.4 (KitKat)

Compile With: Google APIs (Google Inc.) (API 19)

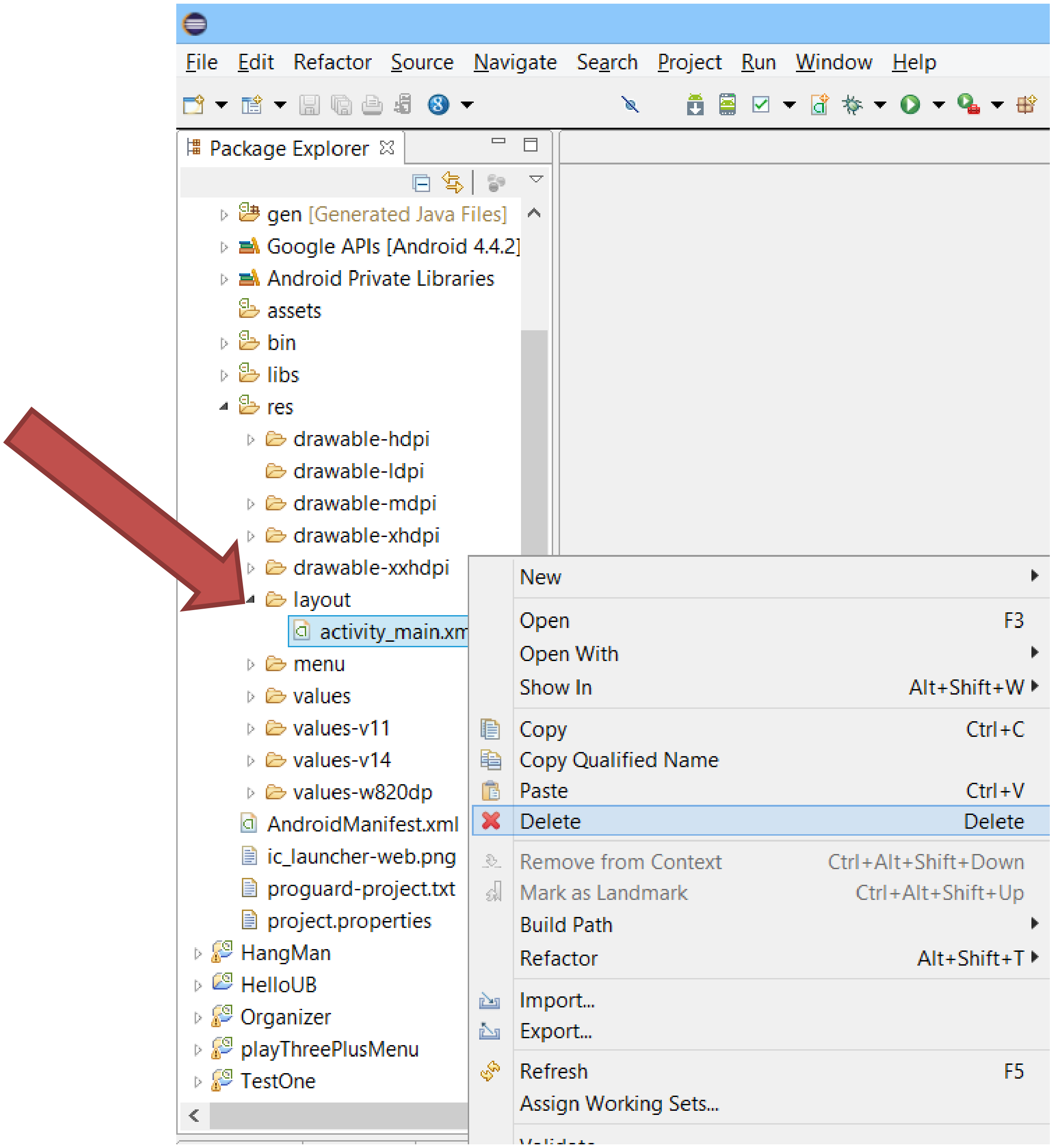
Theme: None

The application name is shown in the Play Store, as well as in the Manage Application list in Settings.

Note: the cm letters are the first and second letter of **YOUR NAME**. If your name is Bob Ferns then your app will be firstapp_bf

Click next and leave everything on default including the name of the activity to be MainActivity.

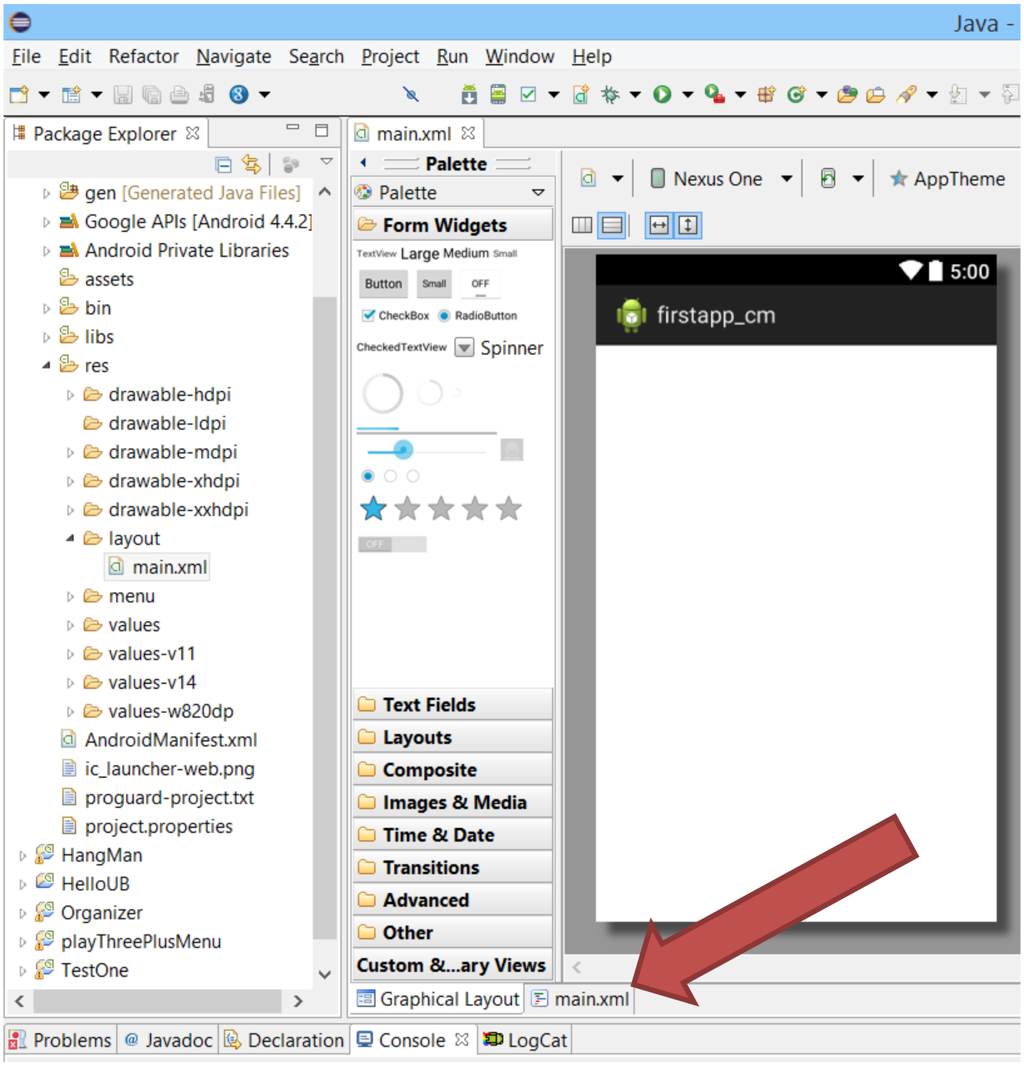
Click on the ‘res’ folder-> layout folder and delete the activity_main.xml either by clicking on it once and hit delete or right-clicking on it and selecting delete.



We are making a new layout were we will see the picture name and id.

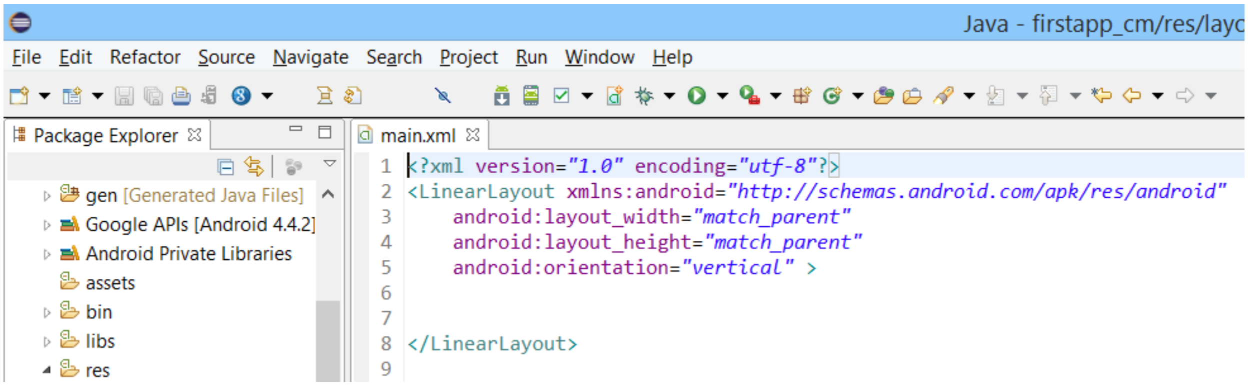
CPSC 441 – Mobile App Development
Lecture #2

Right-click the layout folder and select New->other->Android XML File then hit next and name it **main.xml** and hit finish**.** Double click on it to see the layout. To see the actual code that makes this layout we need to click on the bottom tab next to graphical layout in this case, main.xml.



Click on it and you should see this code in it. IF YOU DO NOT SEE THIS, ERASE WHAT YOU HAVE AND COPY THIS.

The default layout for 4.4 is relative layout but since we are starting, let's start with an easy one, linear layout ☺

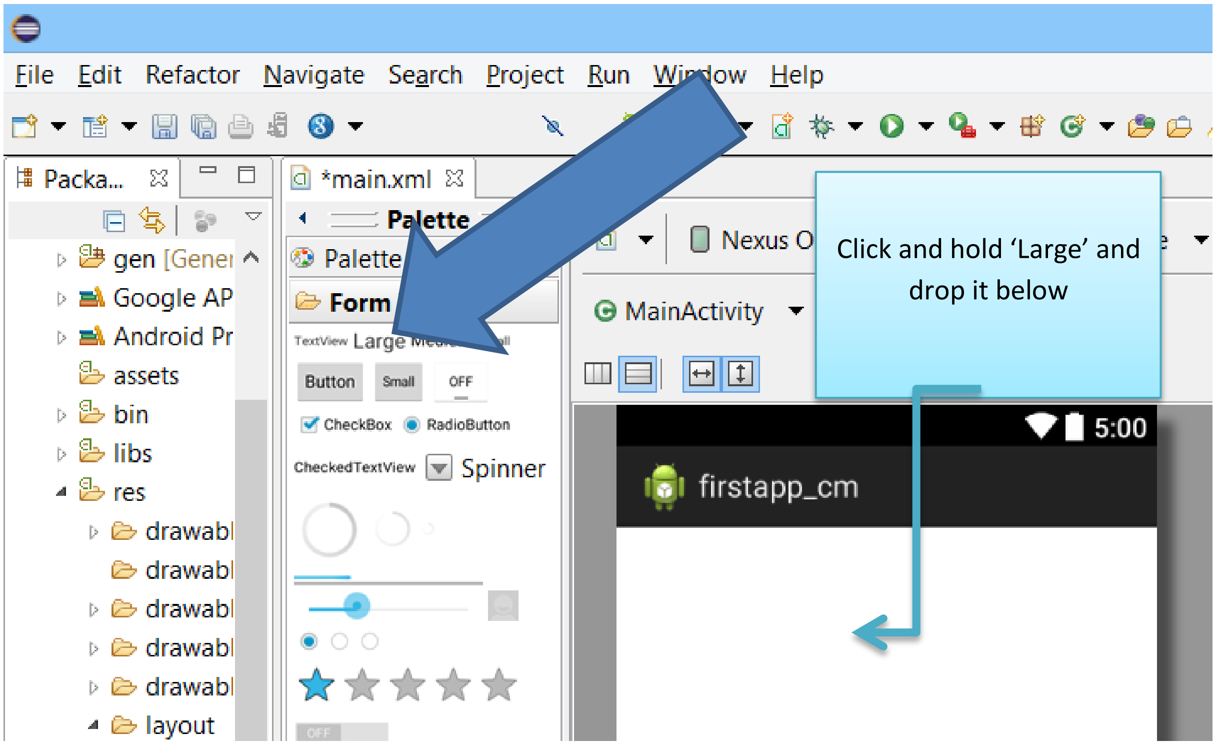

Java - firstapp_cm/res/layo
File Edit Refactor Source Navigate Search Project Run Window Help
Package Explorer
gen [Generated Java Files]
Google APIs [Android 4.4.2]
Android Private Libraries
assets
bin
libs
res
main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >


</LinearLayout>
```

Now we need to add the UI elements that make up the application that we are creating. These elements are:

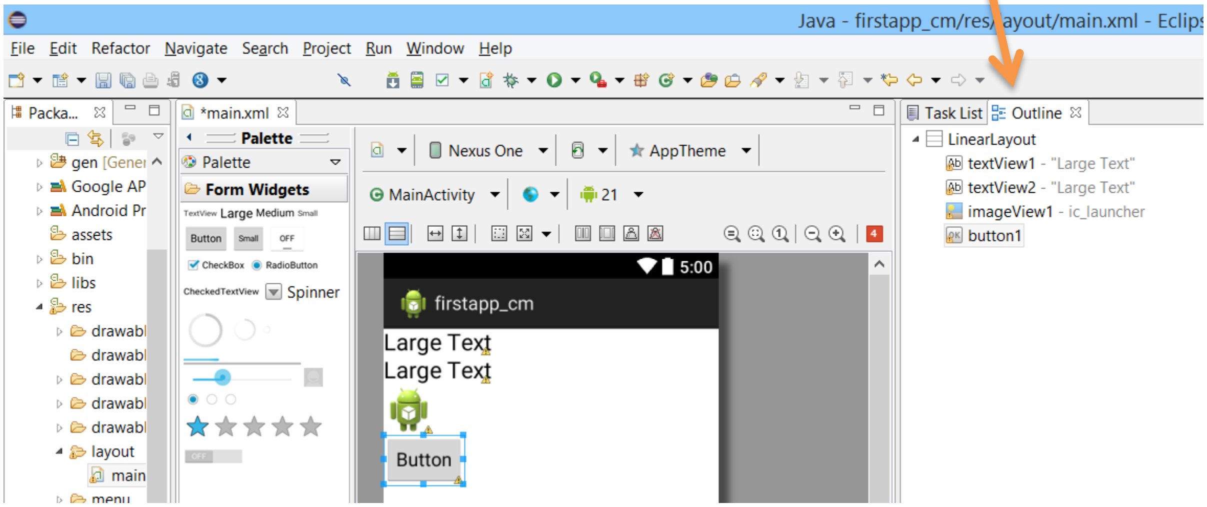
2 textViews	(under Form Widgets)
1 ImageView	(under Images & Media)
1 Button	(also under form widgets)

Click on graphical layout (bottom tab) and in order, drag and drop to the layout two text views, one Imageview and one button.



Click and hold ‘Large’ and drop it below

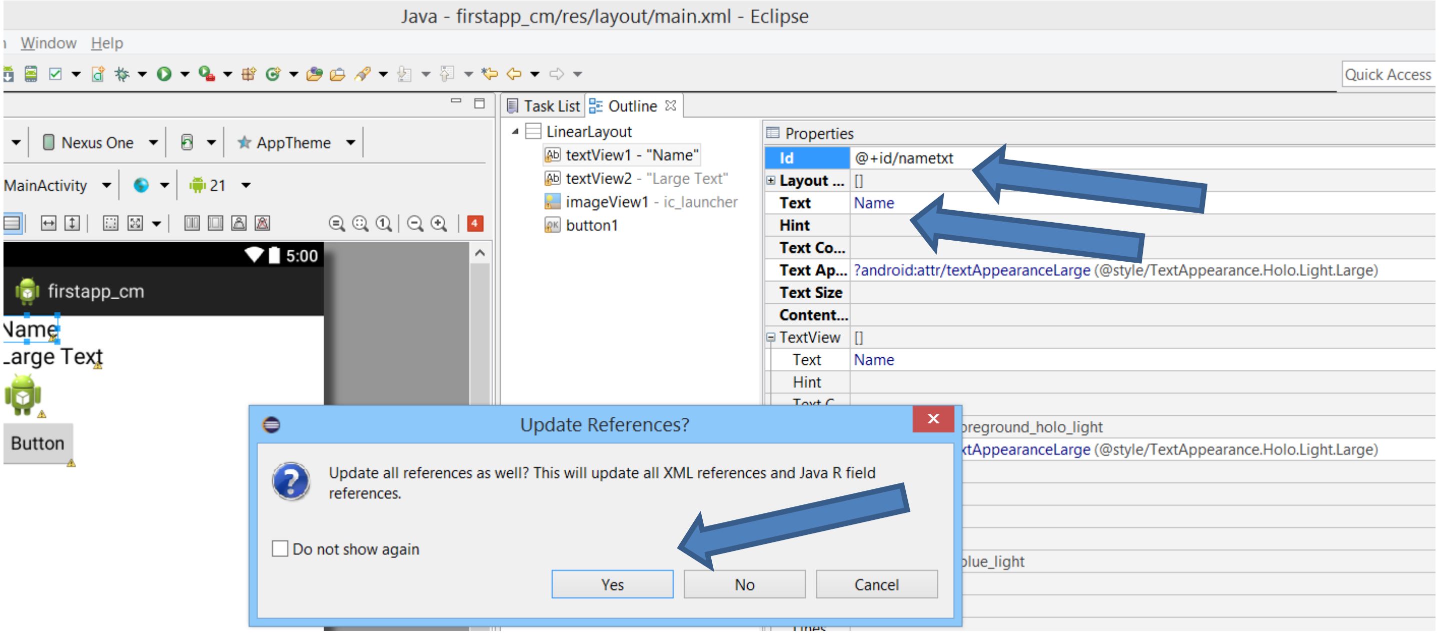
Once the four elements have been added, the order can be seen below. Notice under Outline, this tells you the order in a more detail fashion.



LinearLayout
- textView1 - "Large Text"
- textView2 - "Large Text"
- imageView1 - ic_launcher
- button1

Properties:

Every element has a substantial amount of properties. For this application we will only talk about a few. For the first textbox, we will change the Id and the text. Select the first textbox and under properties change the text to Name and the id to nametxt, finally say yes to update the references and yes to rename.



LinearLayout
- textView1 - "Name"
- textView2 - "Large Text"
- imageView1 - ic_launcher
- button1

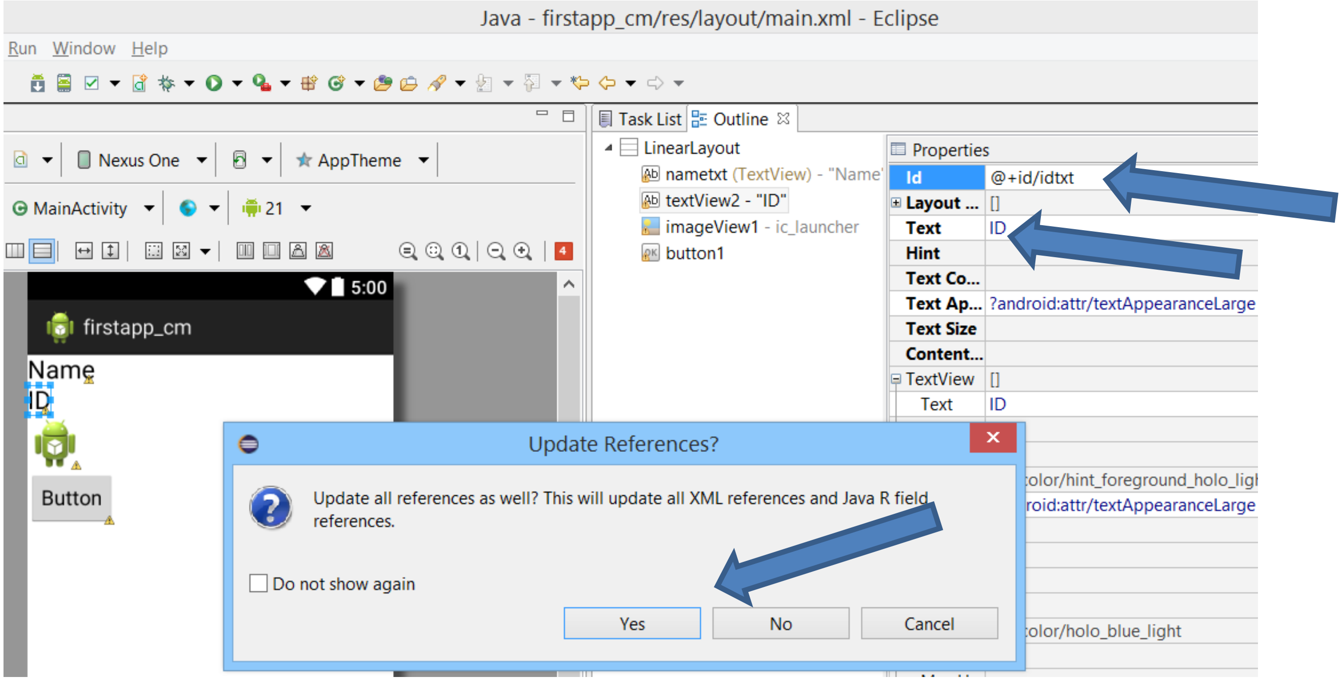
Id	@+id/nametxt
Layout ...	[]
Text	Name
Hint	
Text Co...	
Text Ap...	?android:attr/textAppearanceLarge (@style/TextAppearance.Holo.Light.Large)
Text Size	
Content...	
TextView	[]
Text	Name
Hint	

Update References?

Update all references as well? This will update all XML references and Java R field references.

☐ Do not show again

For the second textbox change the text to ID and the id to idtxt. Say yes to update reference and yes to rename.



Id	@+id/idtxt
Layout ...	[]
Text	ID
Hint	
Text Co...	
Text Ap...	?android:attr/textAppearanceLarge
Text Size	
Content...	
TextView	[]
Text	ID

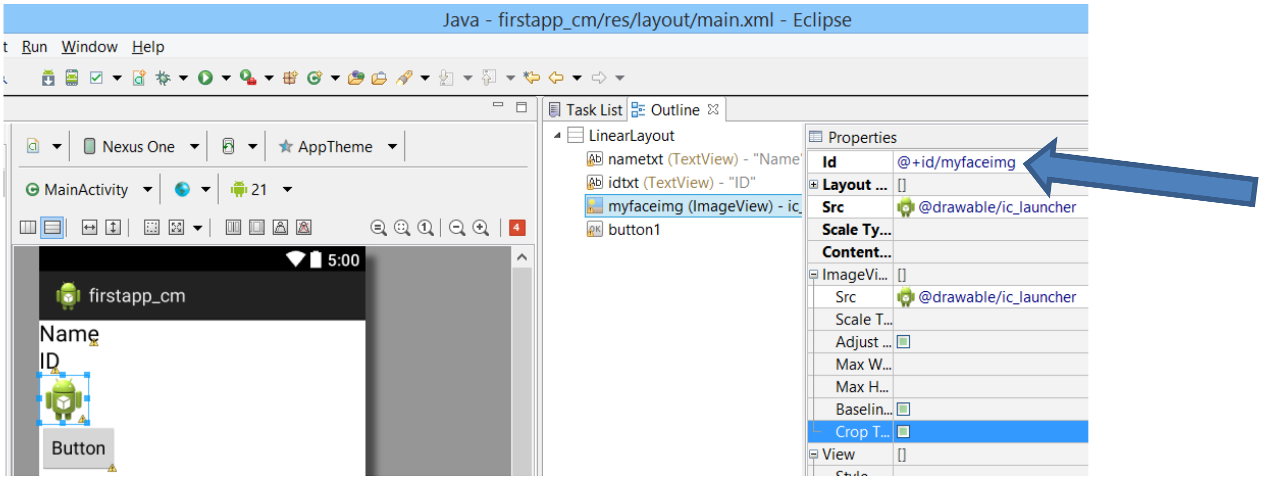
Update References?

Update all references as well? This will update all XML references and Java R field references.

□ Do not show again

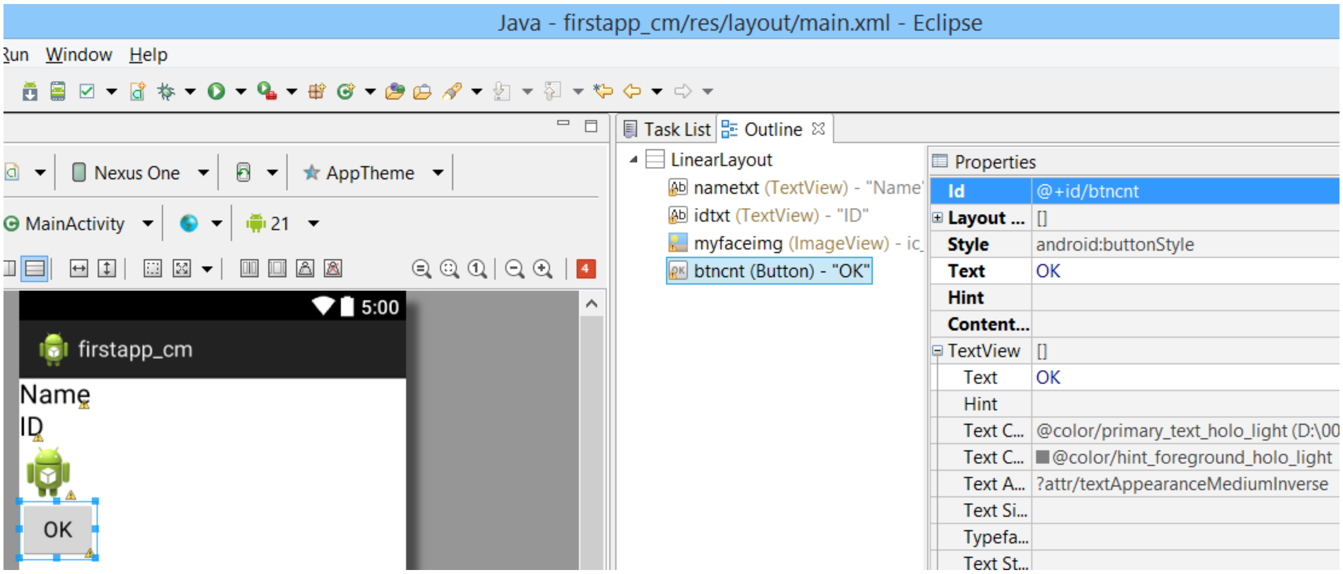
Remember to keep the *@+id/* part. This is needed as this is how android will know how to find this element. Every element is found and referenced by id. You will see this soon enough.

Click on the imageView1 and under properties change the Id to myfaceimg



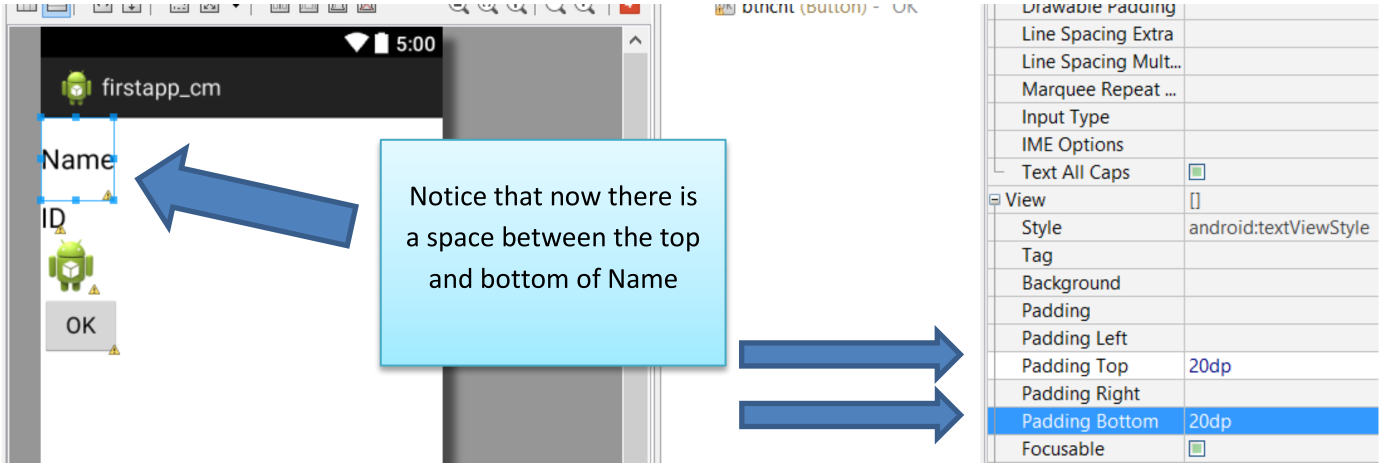
Id	@+id/myfaceimg
Layout ...	[]
Src	@drawable/ic_launcher
Scale Ty...	
Content...	
ImageVi...	[]
Src	@drawable/ic_launcher
Scale T...	
Adjust ...	□
Max W...	
Max H...	
Baselin...	□
Crop T...	□
View	[]

Say yes to update and yes to rename. Finally select button1 and change the text to OK and change the id to btncnt and say yes to update and yes to rename.



Id	@+id/btncnt
Layout ...	[]
Style	android:buttonStyle
Text	OK
Hint	
Content...	
TextView	[]
Text	OK
Hint	
Text C...	@color/primary_text_holo_light (D:\00
Text C...	@color/hint_foreground_holo_light
Text A...	?attr/textAppearanceMediumInverse
Text Si...	
Typefa...	
Text St...	

Ok now let's give it some space between the elements. Select nametxt (first one in the outline) and under properties go to view and type 20dp next to Padding top and Padding bottom

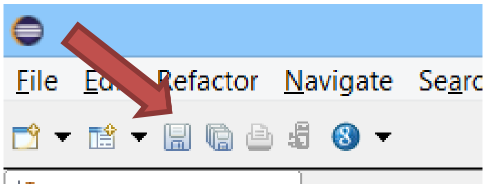


Notice that now there is a space between the top and bottom of Name

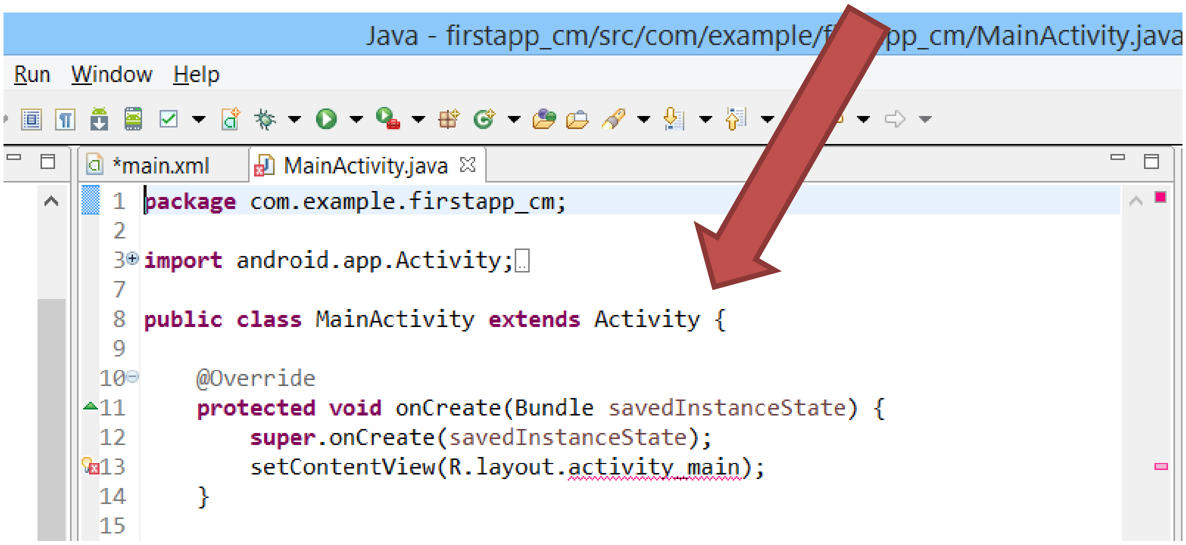
Drawable Padding	
Line Spacing Extra	
Line Spacing Mult...	
Marquee Repeat ...	
Input Type	
IME Options	
Text All Caps	□
View	[]
Style	android:textViewStyle
Tag	
Background	
Padding	
Padding Left	
Padding Top	20dp
Padding Right	
Padding Bottom	20dp
Focusable	□

Add 20dp to idtxt and 20dp to myfaceimg. This will create space between the elements.

Now Save it by clicking on the save icon or hitting CTRL+S (window users)

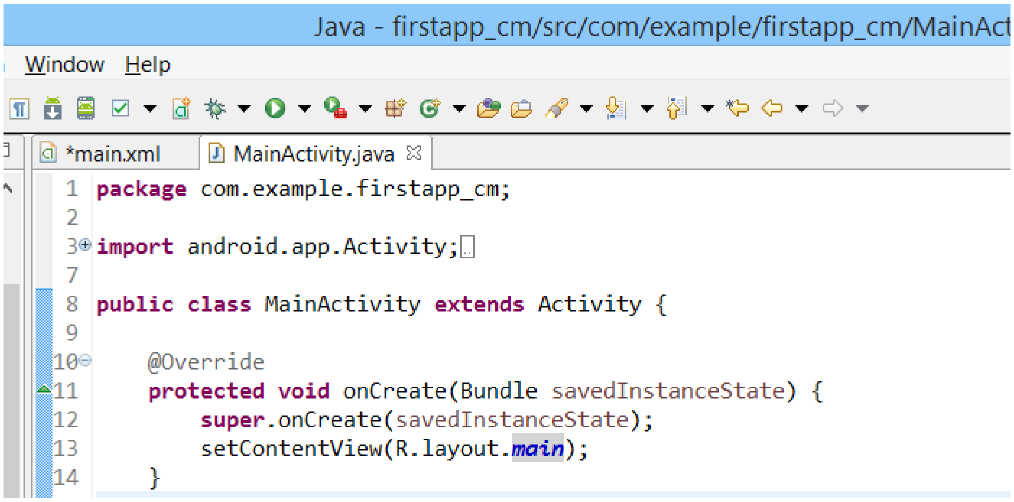


Now open up MainActivity.java located in the src-> com.example.firstapp_?? folder. Double click on MainActivity.java to see the contents of it.


Java - firstapp_cm/src/com/example/firstapp_cm/MainActivity.java
Run
Window
Help
*main.xml
MainActivity.java

```
package com.example.firstapp_cm;

import android.app.Activity;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

```

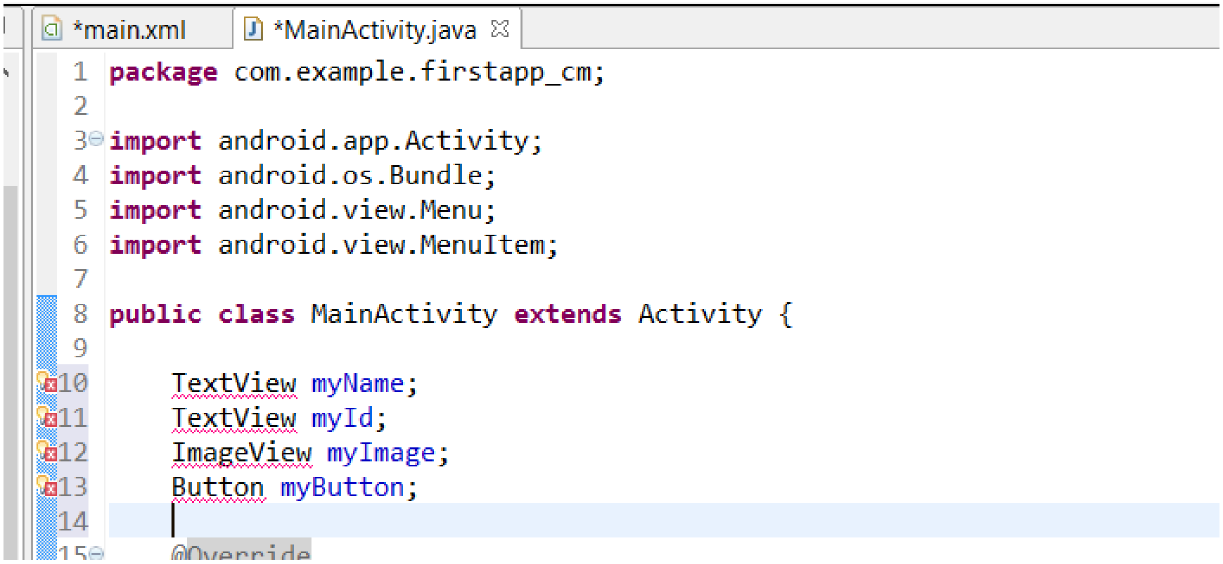
Since we deleted the layout and created a new one, this red line needs to be changed to the new layout name. Delete it and add the new one then save it.



```
package com.example.firstapp_cm;

import android.app.Activity;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
```

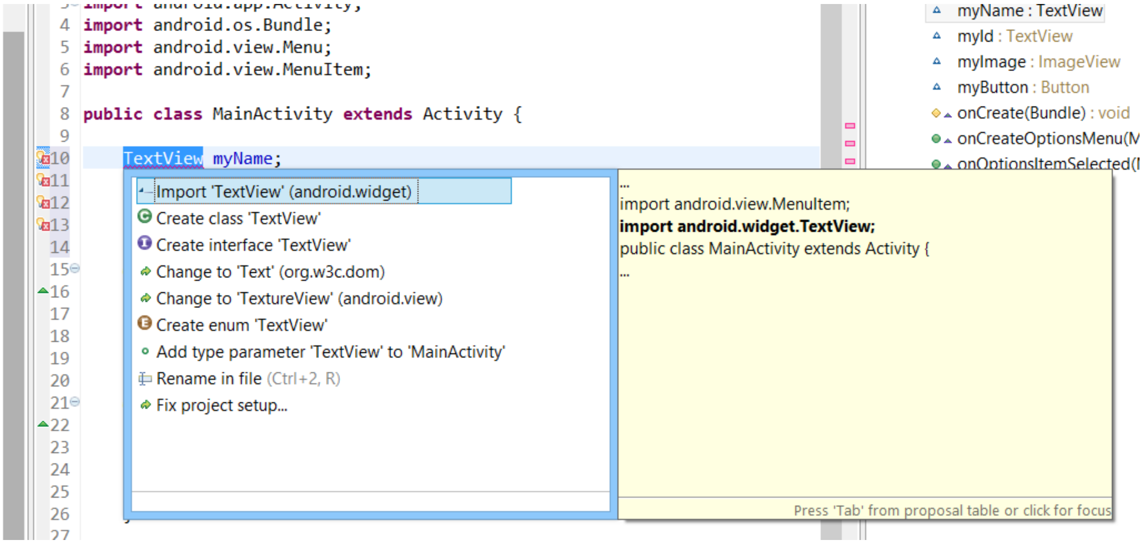
Access the elements:

The way that Android gets to change the element properties is by their respective ids. To do so first we must create objects to reference them.



```
package com.example.firstapp_cm;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends Activity {

    TextView myName;
    TextView myId;
    ImageView myImage;
    Button myButton;

    @Override
```

Put these four lines right under the class to make them globally available. The classes are red because we did not create them. To fix that we either create custom classes or in this case, we use the built in ones. To use the build in one, we just import them. Click on the yellow bulb and import the classes.

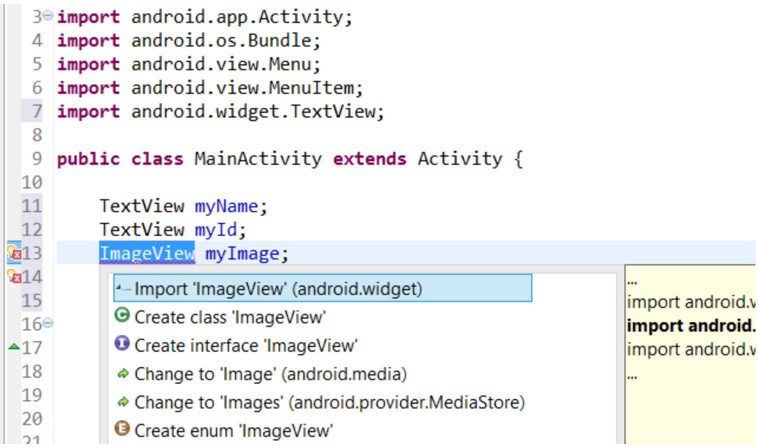


```
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends Activity {

    TextView myName;
```

Import 'TextView' (android.widget)
Create class 'TextView'
Create interface 'TextView'
Change to 'Text' (org.w3c.dom)
Change to 'TextureView' (android.view)
Create enum 'TextView'
Add type parameter 'TextView' to 'MainActivity'
Rename in file (Ctrl+2, R)
Fix project setup...

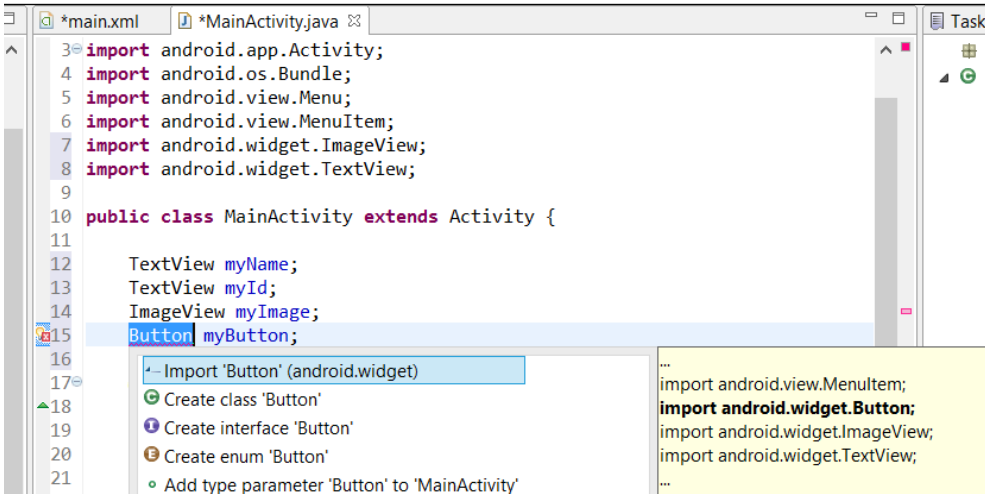
```
...
import android.view.MenuItem;
import android.widget.TextView;
public class MainActivity extends Activity {
...
```

Press 'Tab' from proposal table or click for focus
myName : TextView
myId : TextView
myImage : ImageView
myButton : Button
onCreate(Bundle) : void
onCreateOptionsMenu(M
onOptionsItemSelected(I



```
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;

public class MainActivity extends Activity {

    TextView myName;
    TextView myId;
    ImageView myImage;
```

Import 'ImageView' (android.widget)
Create class 'ImageView'
Create interface 'ImageView'
Change to 'Image' (android.media)
Change to 'Images' (android.provider.MediaStore)
Create enum 'ImageView'
...
import android.v
import android.
import android.v
...



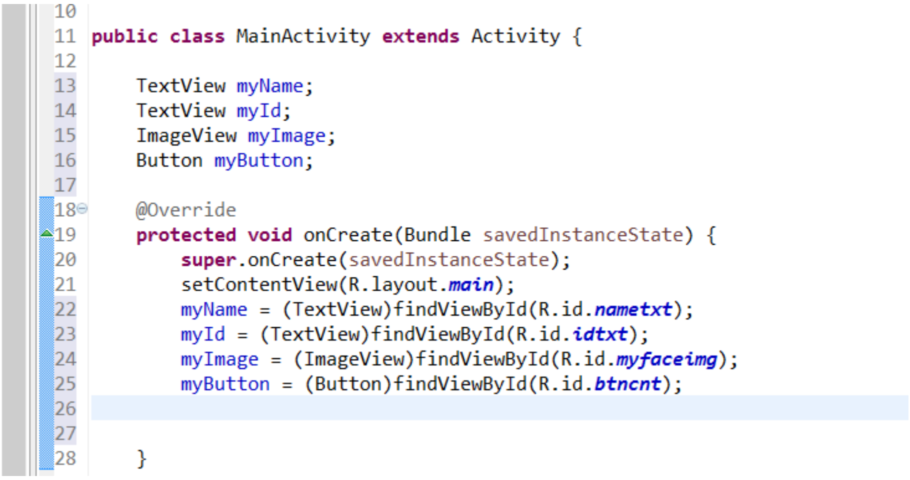
*main.xml
*MainActivity.java
Task

```
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.ImageView;
import android.widget.TextView;

public class MainActivity extends Activity {

    TextView myName;
    TextView myId;
    ImageView myImage;
    Button myButton;
```

Import 'Button' (android.widget)
Create class 'Button'
Create interface 'Button'
Create enum 'Button'
Add type parameter 'Button' to 'MainActivity'

```
...
import android.view.MenuItem;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
...
```

As they are imported they will show on the top of the class. Once they are all imported, we proceed to reference the elements from the layout. Add the following four lines in the OnCreate method.



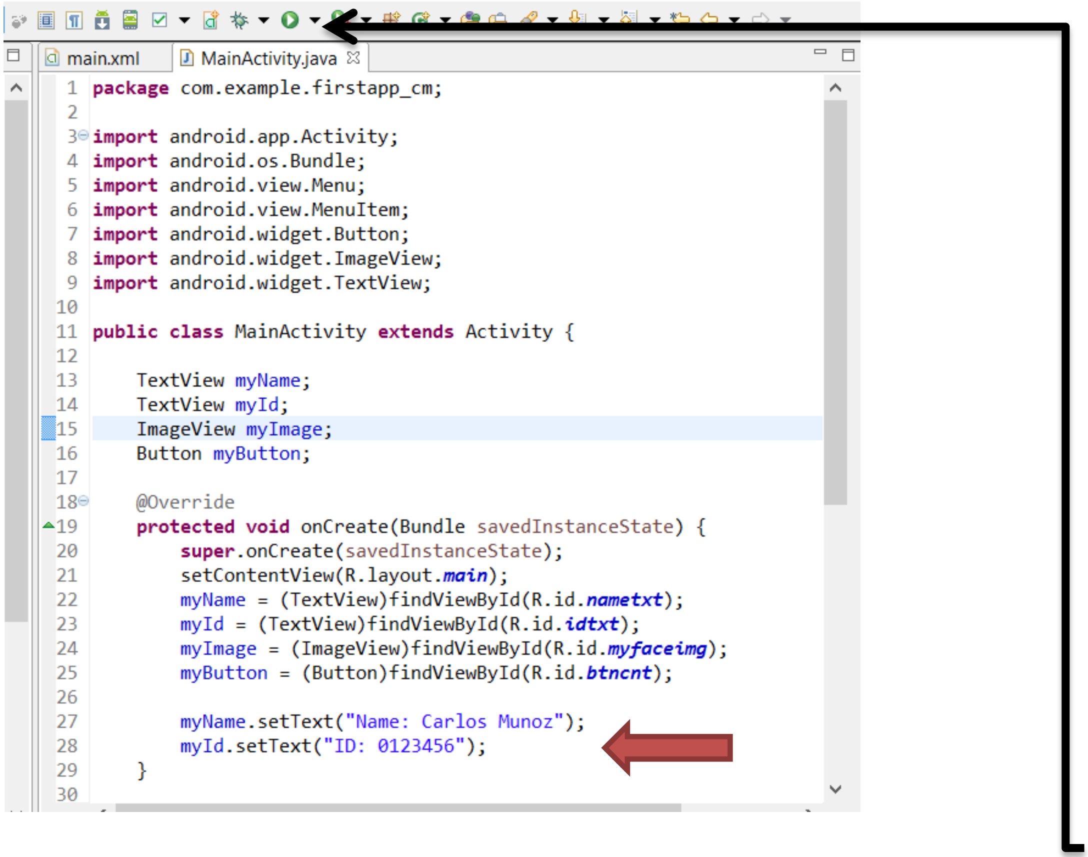
```

public class MainActivity extends Activity {

    TextView myName;
    TextView myId;
    ImageView myImage;
    Button myButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        myName = (TextView)findViewById(R.id.nametxt);
        myId = (TextView)findViewById(R.id.idtxt);
        myImage = (ImageView)findViewById(R.id.myfaceimg);
        myButton = (Button)findViewById(R.id.btncnt);


    }
```

CPSC 441 – Mobile App Development
Lecture #2

Why onCreate?

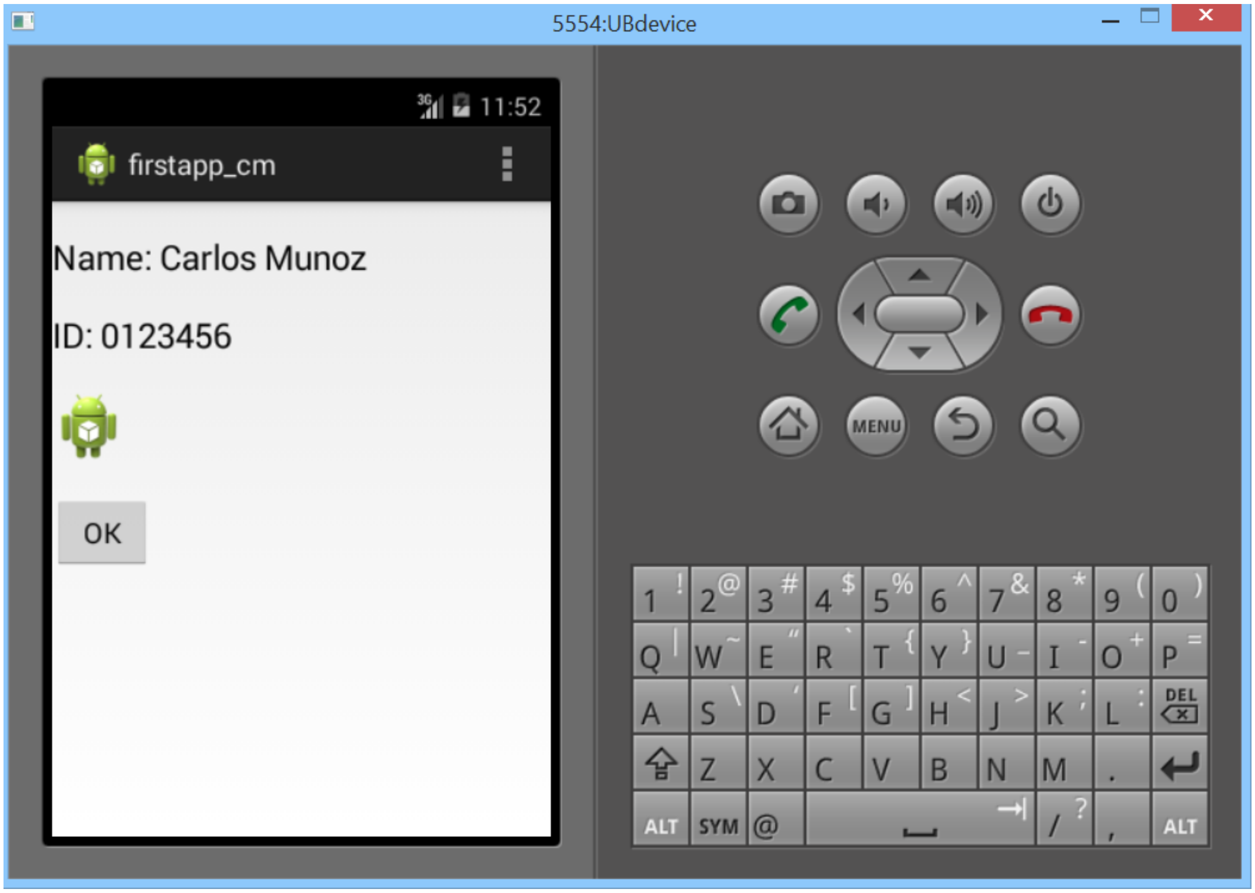
Every Activity (for now) starts on the onCreate Method. In this method the layout is set first then each element of the layout is found and referenced using their respective ids. The reason we need to cast is because much like Java and Objects being the top class, in Android every element's top class is the View Class. We cast them to their respective type in order to use them.

Now we can change each text like so:



```
package com.example.firstapp_cm;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

public class MainActivity extends Activity {

    TextView myName;
    TextView myId;
    ImageView myImage;
    Button myButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        myName = (TextView)findViewById(R.id.nametxt);
        myId = (TextView)findViewById(R.id.idtxt);
        myImage = (ImageView)findViewById(R.id.myfaceimg);
        myButton = (Button)findViewById(R.id.btncnt);

        myName.setText("Name: Carlos Munoz");
        myId.setText("ID: 0123456");
    }

```

Click on your app folder from the package Explorer list then hit the run button on the top to test what we have done so far. Save everything and select the UBdevice. Remember that we added that virtual device on lecture one. After 3-5 min you should see something similar to the image below.

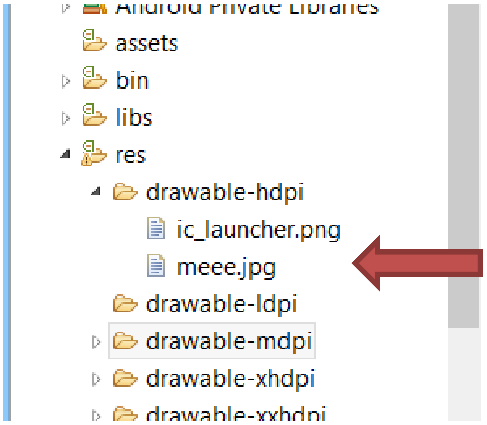


Name: Carlos Munoz

ID: 0123456

The Picture:

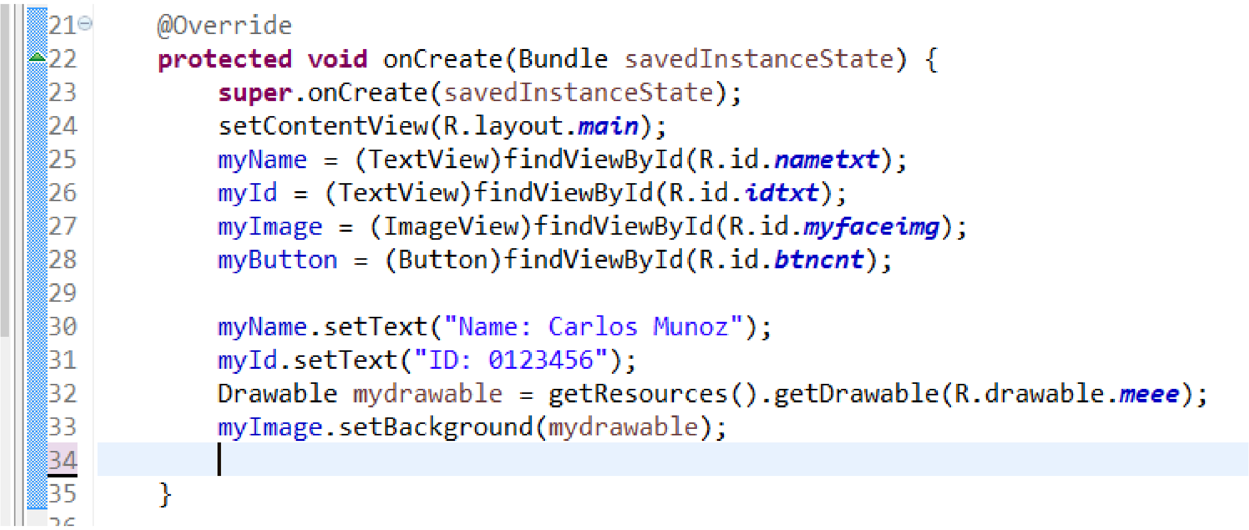
Images can be manipulated in many different ways. For now, we just want to change the little green guy to a picture of our face. Locate an image of you and crop it with photoshop, mspaint or any software that you have. Add the image to the drawable-hdpi folder. Make sure you select copy from the menu that pops up.



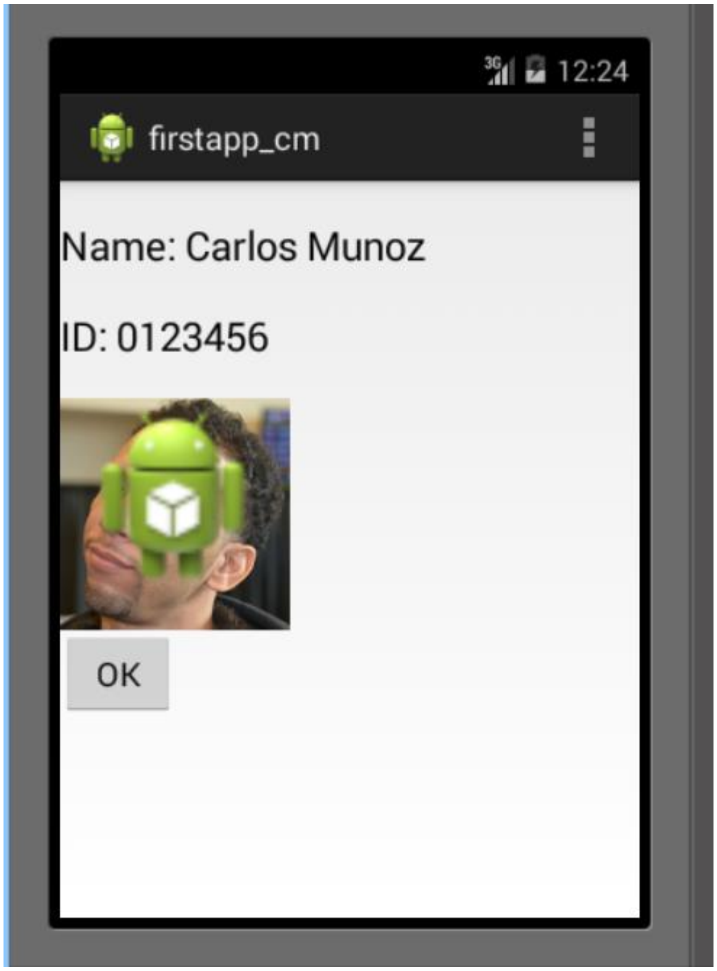
```
Android Private Libraries
assets
bin
libs
res
  drawable-hdpi
    ic_launcher.png
    meee.jpg
  drawable-ldpi
  drawable-mdpi
  drawable-xhdpi
  drawable-xxhdpi
```

I called it meee.jpg. Try to make the image width and height 192 pixels by 192 pixels

Add the following code to use the Image. The classes need to be imported just like before. Then run it.



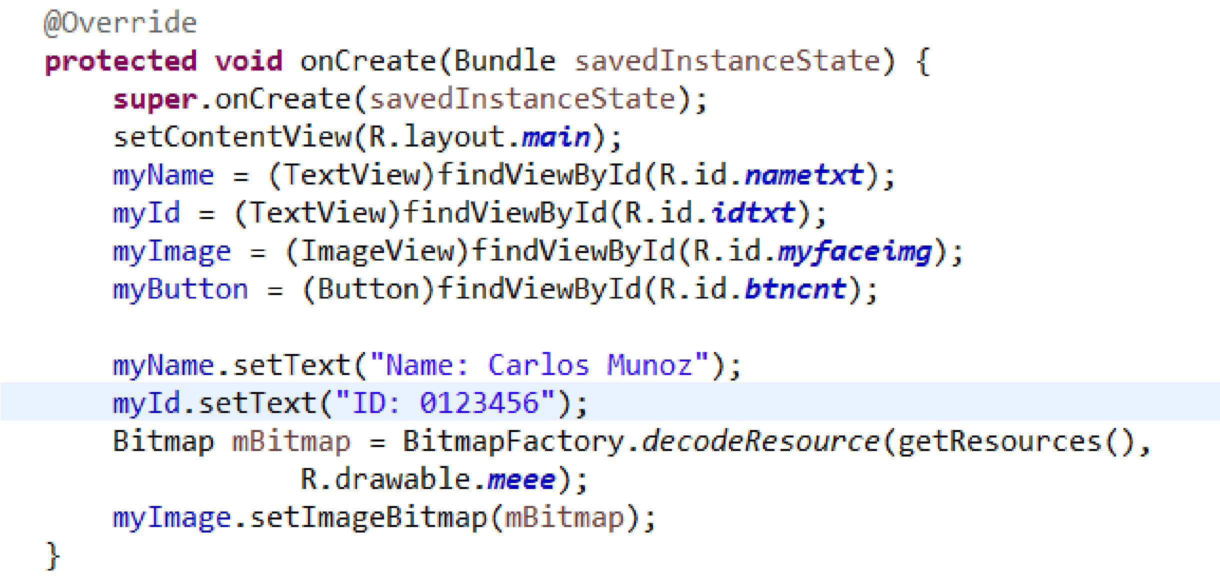
```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        myName = (TextView)findViewById(R.id.nametxt);
        myId = (TextView)findViewById(R.id.idtxt);
        myImage = (ImageView)findViewById(R.id.myfaceimg);
        myButton = (Button)findViewById(R.id.btncnt);

        myName.setText("Name: Carlos Munoz");
        myId.setText("ID: 0123456");
        Drawable mydrawable = getResources().getDrawable(R.drawable.meee);
        myImage.setBackground(mydrawable);

    }
```



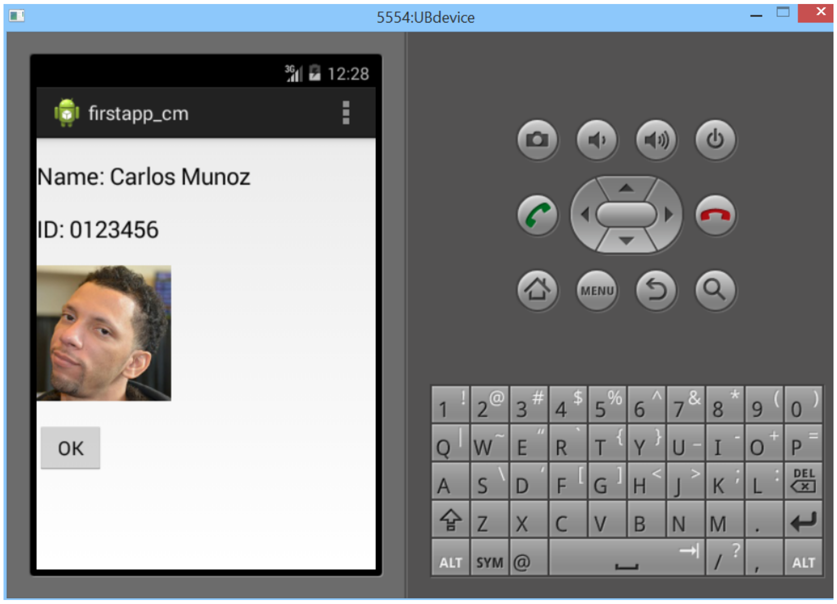
Name: Carlos Munoz

ID: 0123456

Hey, the green android is still there!!. Well, that is because we changed the background of the ImageView, not the contents of it. To get the desired behavior we can either remove the android or replace the actual contents. Replace the code with the code below to replace the actual contents of the ImageView. Import the classes if necessary.



```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    myName = (TextView)findViewById(R.id.nametxt);
    myId = (TextView)findViewById(R.id.idtxt);
    myImage = (ImageView)findViewById(R.id.myfaceimg);
    myButton = (Button)findViewById(R.id.btncnt);

    myName.setText("Name: Carlos Munoz");
    myId.setText("ID: 0123456");
    Bitmap mBitmap = BitmapFactory.decodeResource(getResources(),
                R.drawable.meee);
    myImage.setImageBitmap(mBitmap);
}
```



```
Name: Carlos Munoz

ID: 0123456
```

What about the ok button? We will use it later. Now that is all done, just go to your workspace folder, zip the folder and name it lecture2_yourname.zip and submit it to canvas.

Forgot where is your workspace? Click on Window->preferences->General

Click on the '>' on the left of general->startup and shutdown->workspaces and the path to your workspace is there.