Guide to running the game in the emulator

This installation guide document is not a substitute but a complementary resource for the videos provided in the course.

Please note that this is a working document and that errors may remain.

There are lot of things that are discussed in the videos that are not discussed here. This installation guide is only help you download, install and setup your system – we highly recommend you viewing the videos to understand the process fully. The installation guide combines install and configuring instructions from several MOOC resources without going into detail about other features discussed in videos.

Setting up the game framework

Now we are going to copy files from the downloaded and unzipped game framework bundle.

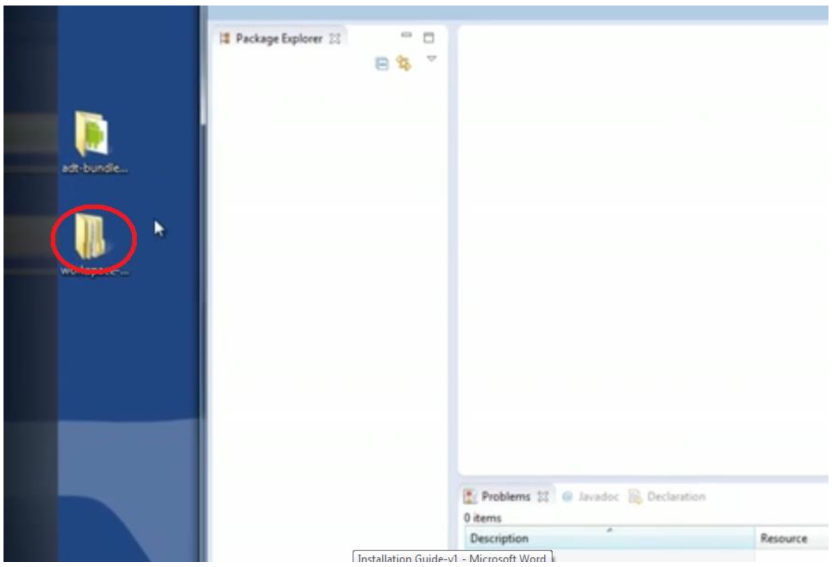
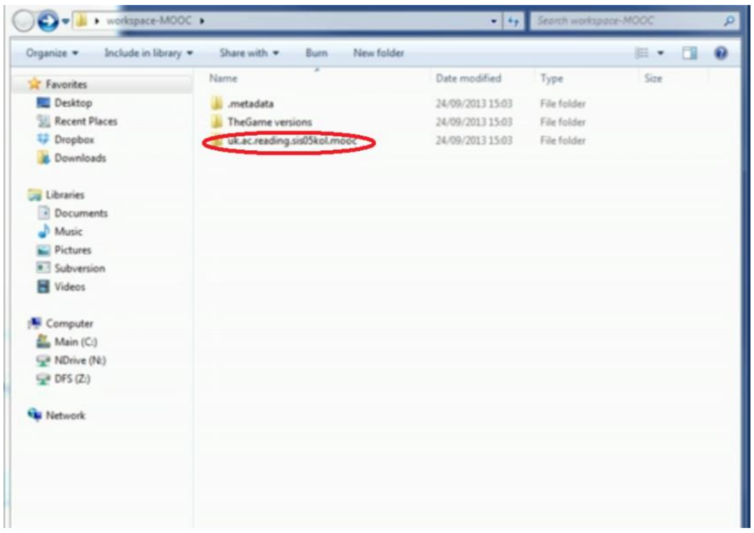


Figure 1: Locate Game Framework Files

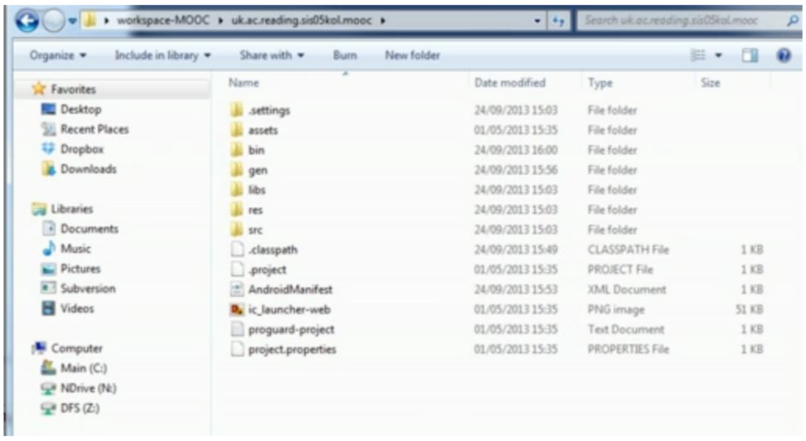
Double-click on the folder that contains the downloaded and unzipped game framework files.



Name	Date modified	Type	Size
.metadata	24/09/2013 15:03	File folder	
TheGame versions	24/09/2013 15:03	File folder	
uk.ac.reading.sis05kol.mooc	24/09/2013 15:03	File folder	

Figure 2: Game Framework folders

Go inside the folder ‘uk.ac.reading.sis05kol.mooc’. Its content is shown in Figure .



Name	Date modified	Type	Size
.settings	24/09/2013 15:03	File folder	
assets	01/05/2013 15:35	File folder	
bin	24/09/2013 16:00	File folder	
gen	24/09/2013 15:56	File folder	
libs	24/09/2013 15:03	File folder	
res	24/09/2013 15:03	File folder	
src	24/09/2013 15:03	File folder	
.classpath	24/09/2013 15:49	CLASSPATH File	1 KB
.project	01/05/2013 15:35	PROJECT File	1 KB
AndroidManifest	24/09/2013 15:53	XML Document	1 KB
ic_launcher-web	01/05/2013 15:35	PNG image	51 KB
proguard-project	01/05/2013 15:35	Text Document	1 KB
project.properties	01/05/2013 15:35	PROPERTIES File	1 KB

Figure 3: Content of the folder

Now we have to copy these files into an Android Project. For that we need to create an Android Project. Go to ADT and select ‘File’ and you will get a drop down menu.

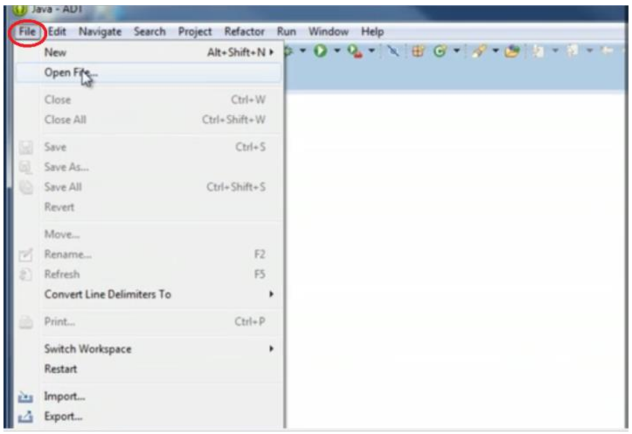


Figure 4: ADT File Menu

Under File menu select ‘New’ and under that select ‘Android Application Project’

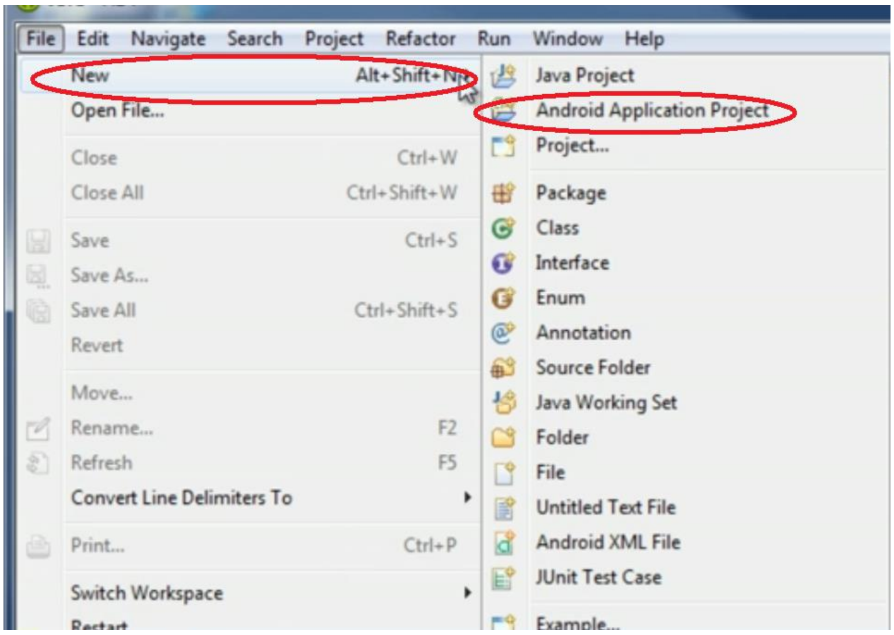
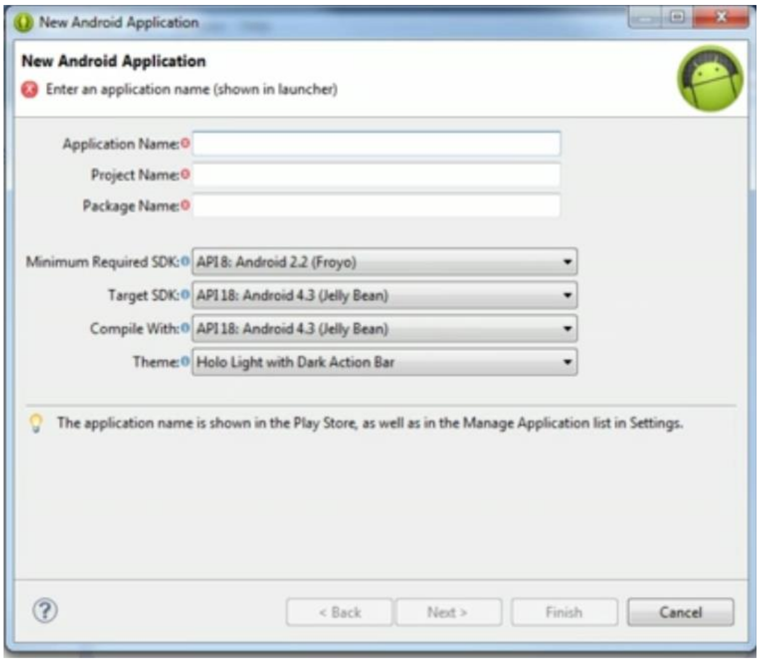


Figure 5: Create Android Application Project

This will open the New Android Application dialog. Here you need to give an Application Name. Type 'MOOC' as application name.



New Android Application

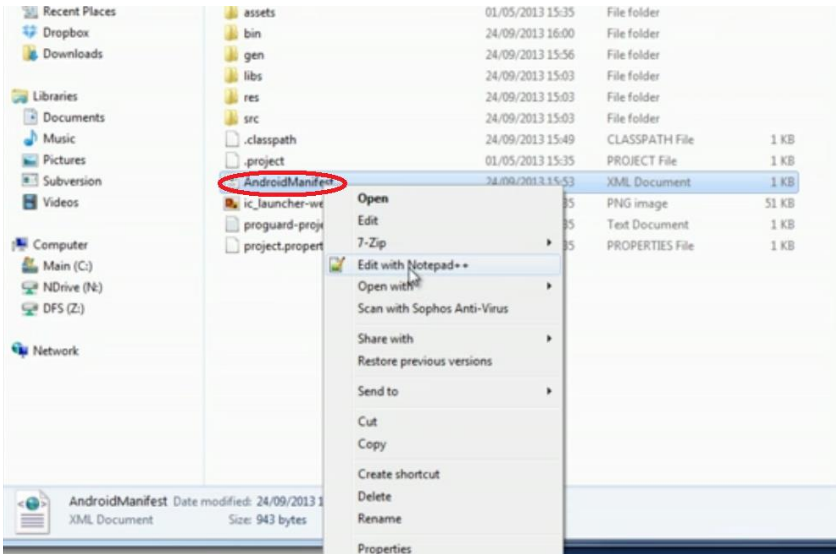
Enter an application name (shown in launcher)

Application Name:
Project Name:
Package Name:
Minimum Required SDK: API 8: Android 2.2 (Froyo)
Target SDK: API 18: Android 4.3 (Jelly Bean)
Compile With: API 18: Android 4.3 (Jelly Bean)
Theme: Holo Light with Dark Action Bar

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

Figure 6: New Android Application

Next we need to provide a 'Package Name'. The package name needs to be taken from the 'AndroidManifest' file in the downloaded game framework bundle.

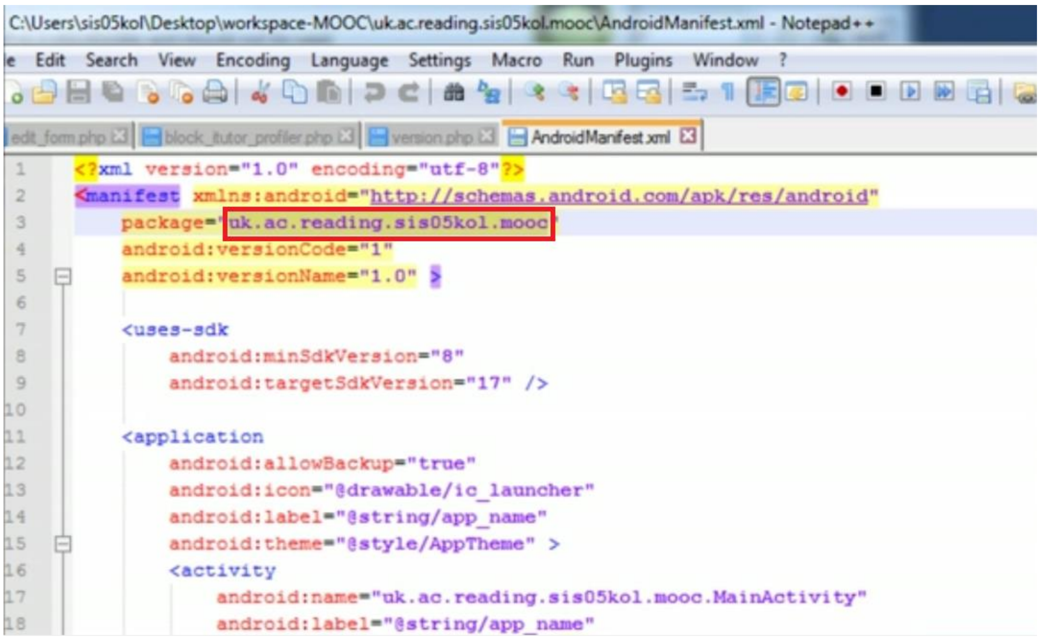


```
assets              01/05/2013 15:35   File folder
bin                 24/09/2013 16:00   File folder
gen                 24/09/2013 15:56   File folder
libs                24/09/2013 15:03   File folder
res                 24/09/2013 15:03   File folder
src                 24/09/2013 15:03   File folder
.classpath          24/09/2013 15:49   CLASSPATH File     1 KB
.project            01/05/2013 15:35   PROJECT File       1 KB
AndroidManifest     24/09/2013 15:53   XML Document       1 KB
ic_launcher-we                     35  PNG image         51 KB
proguard-proje                     35  Text Document      1 KB
project.propert                    35  PROPERTIES File    1 KB

AndroidManifest  Date modified: 24/09/2013 1
XML Document     Size: 943 bytes
```

Figure 7: Open AndroidManifest

Locate the file 'AndroidManifest' and right click it. From the dropdown menu select a text editor of your choice. Here we are using Notepad++, you could use notepad, TextEdit or gedit.

When opened AndroidManifest.xml file will look something like this.


C:\Users\sis05kol\Desktop\workspace-MOOC\uk.ac.reading.sis05kol.mooc\AndroidManifest.xml - Notepad++

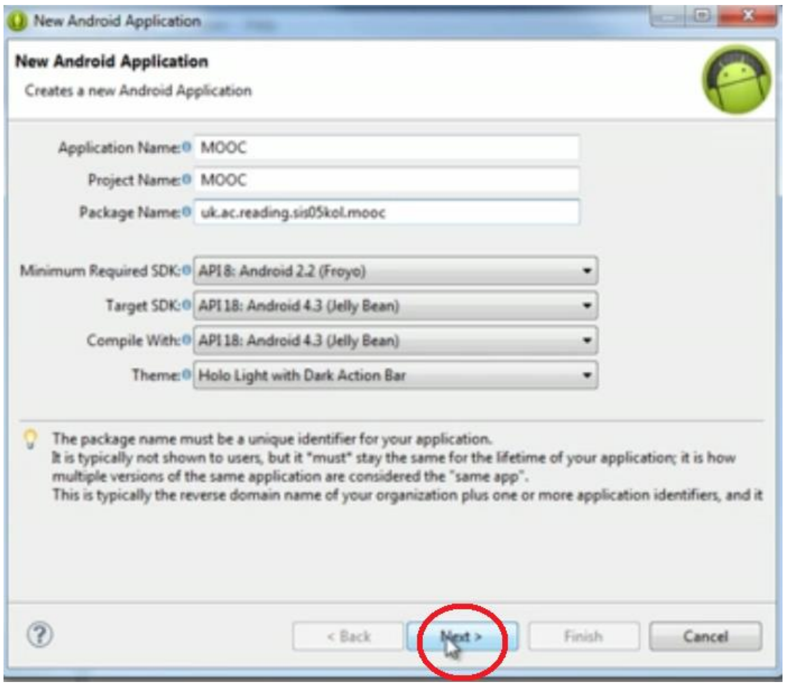
e Edit Search View Encoding Language Settings Macro Run Plugins Window ?
edit_form.php
block_tutor_profiler.php
version.php
AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="uk.ac.reading.sis05kol.mooc"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="uk.ac.reading.sis05kol.mooc.MainActivity"
            android:label="@string/app_name"
```

Figure 8: AndroidManifest.xml

Copy the package name from AndroidManifest and paste it in the New Android Application dialog's package name.

Now press 'Next' on the New Android Application Dialog.



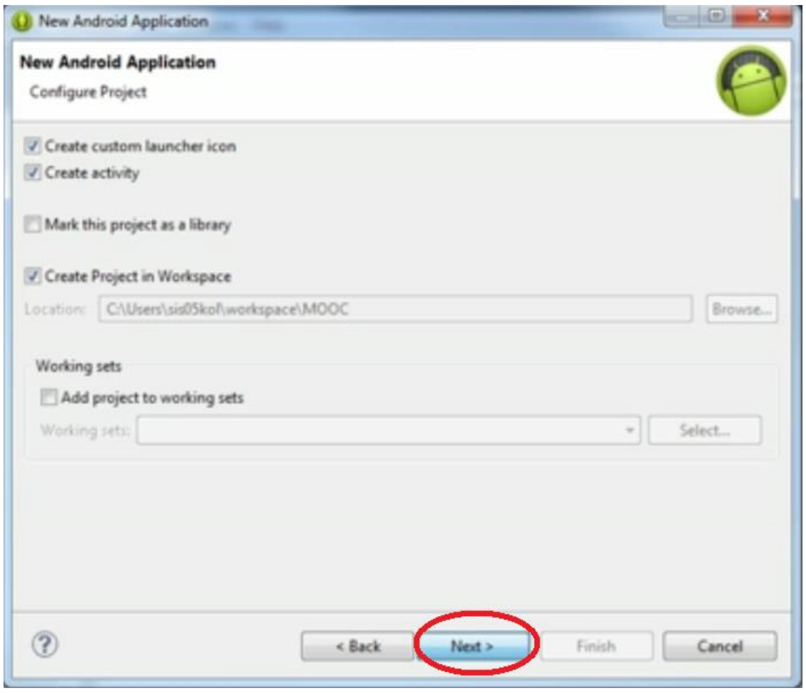
Application Name: MOOC
Project Name: MOOC
Package Name: uk.ac.reading.sis05kol.mooc

Minimum Required SDK: API 8: Android 2.2 (Froyo)
Target SDK: API 18: Android 4.3 (Jelly Bean)
Compile With: API 18: Android 4.3 (Jelly Bean)
Theme: Holo Light with Dark Action Bar

The package name must be a unique identifier for your application.
It is typically not shown to users, but it *must* stay the same for the lifetime of your application; it is how multiple versions of the same application are considered the "same app".
This is typically the reverse domain name of your organization plus one or more application identifiers, and it

Figure 9: New Android Application

Press 'Next' on the next three screens too.



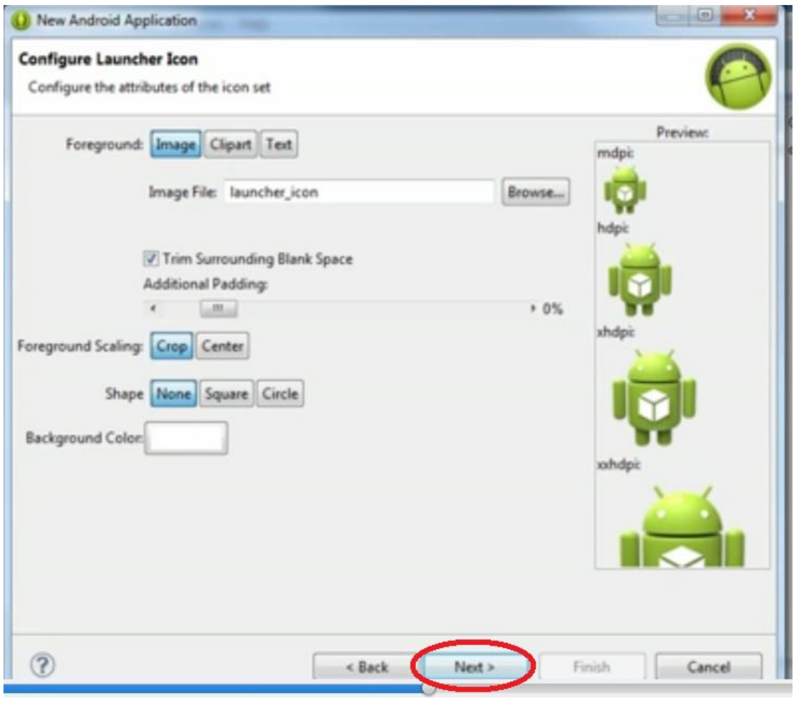
☑ Create custom launcher icon
☑ Create activity

☐ Mark this project as a library

☑ Create Project in Workspace
Location: C:\Users\sis05kol\workspace\MOOC

Working sets
☐ Add project to working sets
Working sets:

Figure 10:New Android Application dialog 2


New Android Application

Configure Launcher Icon

Configure the attributes of the icon set

Foreground: [✓ Image] Clipart Text

Image File: launcher_icon

☑ Trim Surrounding Blank Space

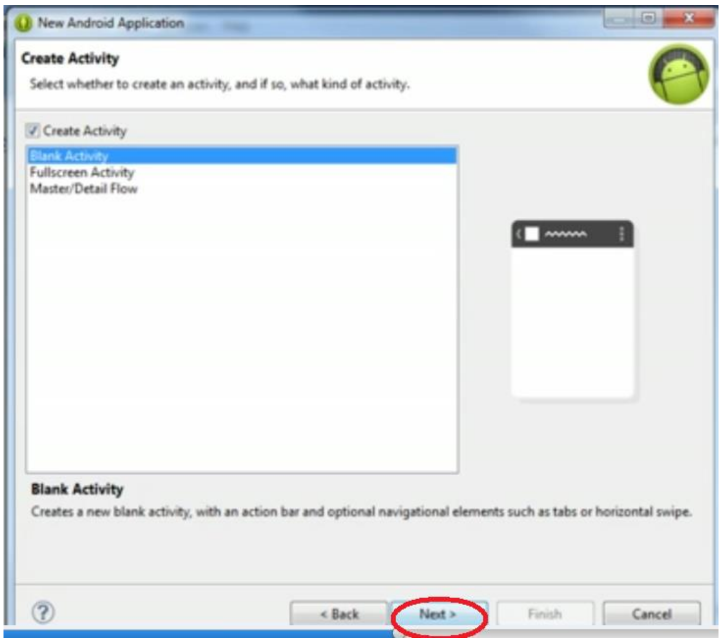
Additional Padding: 0%

Foreground Scaling: [✓ Crop] Center

Shape: [✓ None] Square Circle

Background Color:

Figure 11: New Android Application dialog 3


New Android Application

Create Activity

Select whether to create an activity, and if so, what kind of activity.

☑ Create Activity

[✓ Blank Activity]
Fullscreen Activity
Master/Detail Flow

Blank Activity

Creates a new blank activity, with an action bar and optional navigational elements such as tabs or horizontal swipe.

Figure 12: New Android Application dialog 4

In this final screen click ‘Finish’

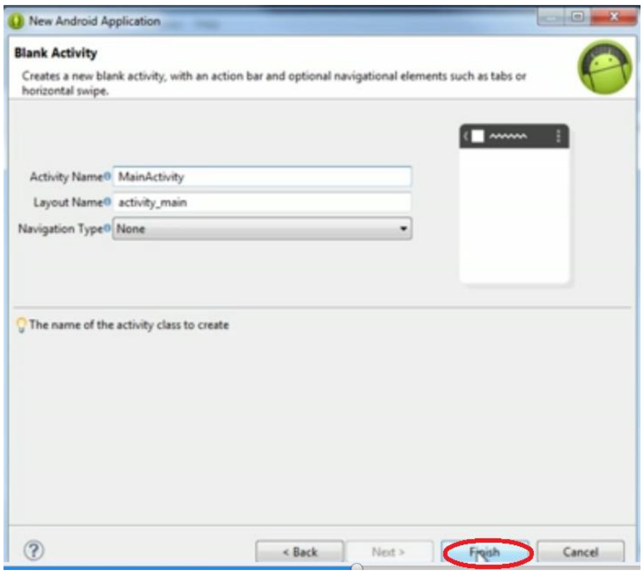

New Android Application
Blank Activity
Creates a new blank activity, with an action bar and optional navigational elements such as tabs or horizontal swipe.
Activity Name: MainActivity
Layout Name: activity_main
Navigation Type: None
The name of the activity class to create
< Back
Next >
Finish
Cancel

Figure 13: New Android Application dialog 5

Now it has created a new Project.

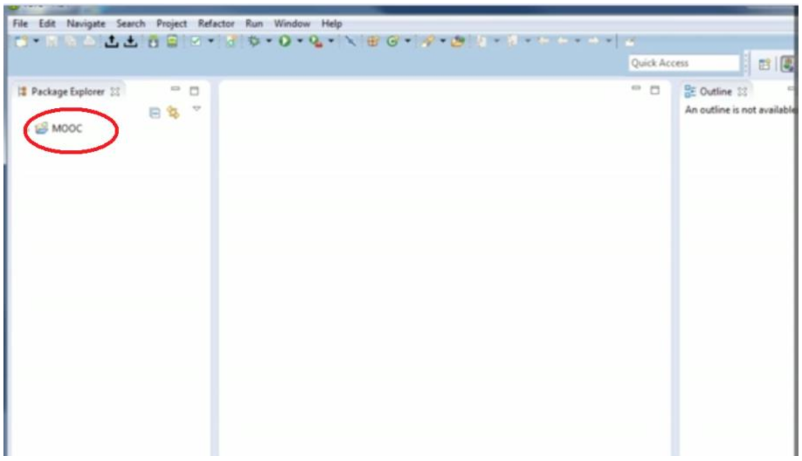
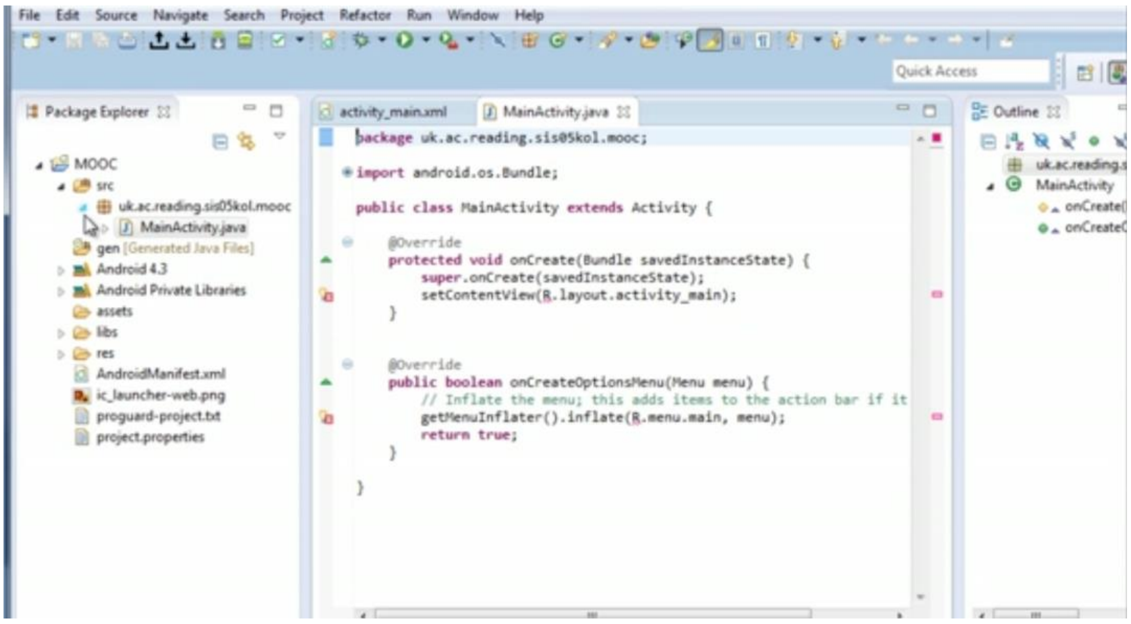

File Edit Navigate Search Project Refactor Run Window Help
Quick Access
Package Explorer
MOOC
Outline
An outline is not available

Figure 14: New project viewed on eclipse

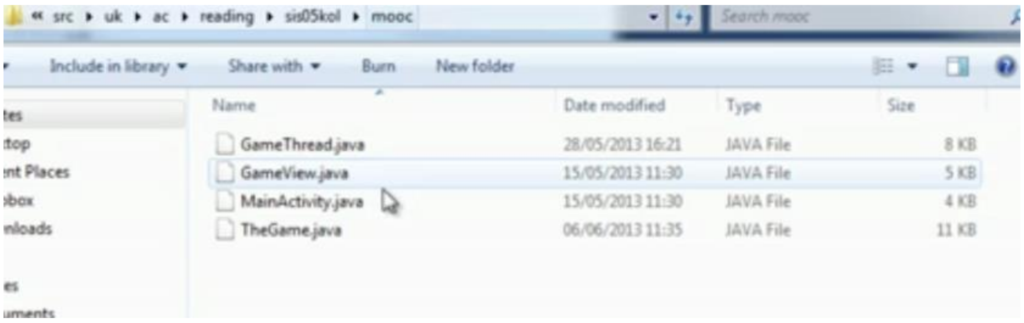
Click on the MOOC project and you will see lot of folders and files under the 'MOOC' project.



```
package uk.ac.reading.sis05kol.mooc;

import android.os.Bundle;

public class MainActivity extends Activity {

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


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

}
```

Figure 15: Project file structure - source (src)

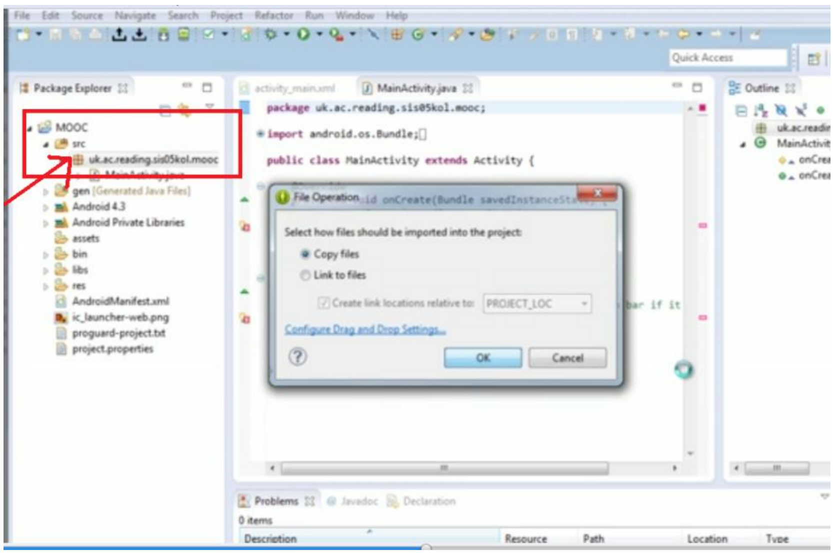
Now go into the Game Framework folder you downloaded and find the 'src' (we call it source) folder. Now go inside 'src' folder and you will see 'uk' folder. Go inside 'uk' → 'ac' → 'reading' →'sis05kol' → 'mooc'. The content of the MOOC folder will have four files.



Name	Date modified	Type	Size
GameThread.java	28/05/2013 16:21	JAVA File	8 KB
GameView.java	15/05/2013 11:30	JAVA File	5 KB
MainActivity.java	15/05/2013 11:30	JAVA File	4 KB
TheGame.java	06/06/2013 11:35	JAVA File	11 KB

Figure 16: Copy files from Game Framework (src)

Select all four files and copy them into the newly created 'MOOC' project. Be sure to copy it under the 'src' folder's 'uk.ac.reading.sis05kol.mooc



```
package uk.ac.reading.sis05kol.mooc;

import android.os.Bundle;

public class MainActivity extends Activity {
```

File Operation

Select how files should be imported into the project:

- (●) Copy files
- ( ) Link to files

Create link locations relative to: PROJECT_LOC

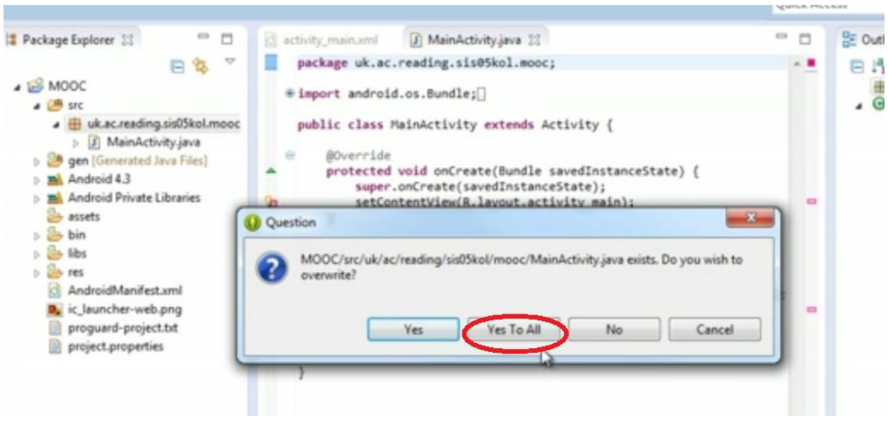
Configure Drag and Drop Settings...

OK | Cancel

Figure 17: copy files confirmation

Select 'Copy Files' and say 'OK'

Next screen will ask for confirmation for the actions.



```
package uk.ac.reading.sis05kol.mooc;

import android.os.Bundle;

public class MainActivity extends Activity {

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

Question

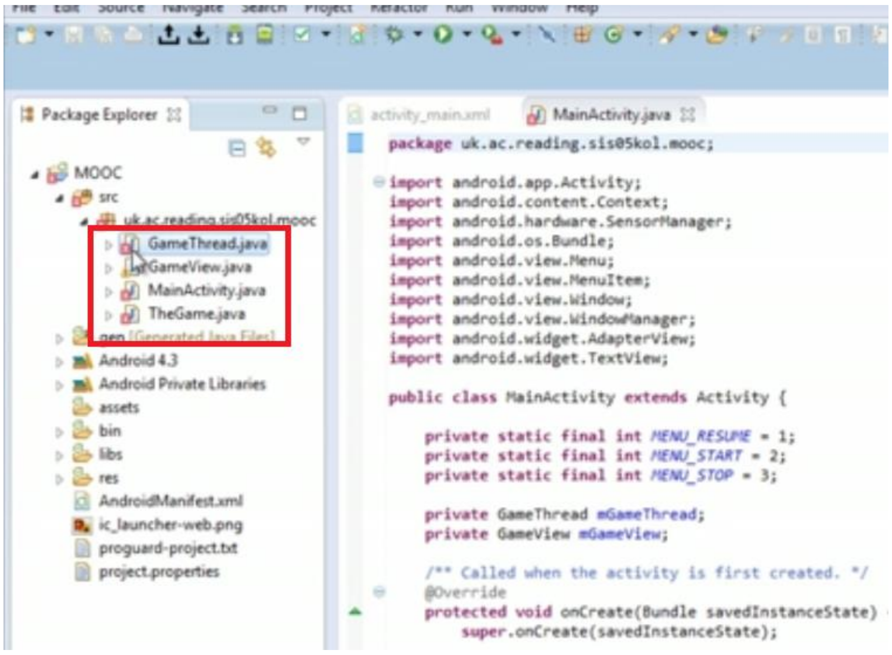
MOOC/src/uk/ac/reading/sis05kol/mooc/MainActivity.java exists. Do you wish to overwrite?

Yes | [✓ Yes To All] | No | Cancel

Figure 18: copy files confirmation

Select ‘Yes To All’.

Don’t worry about the red boxes here for the moment. They mean that there are errors but we will be adding more files into the project that are missing right now.



```java
package uk.ac.reading.sis05kol.mooc;

import android.app.Activity;
import android.content.Context;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.Window;
import android.view.WindowManager;
import android.widget.AdapterView;
import android.widget.TextView;

public class MainActivity extends Activity {

    private static final int MENU_RESUME = 1;
    private static final int MENU_START = 2;
    private static final int MENU_STOP = 3;

    private GameThread mGameThread;
    private GameView mGameView;

    /** Called when the activity is first created. */
    @Override
    protected void onCreate(Bundle savedInstanceState)
        super.onCreate(savedInstanceState);
```

Figure 19: Copied files (src) on project view

Now we are again going into the downloaded game framework files. This time to the ‘res’ folder.

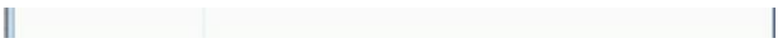


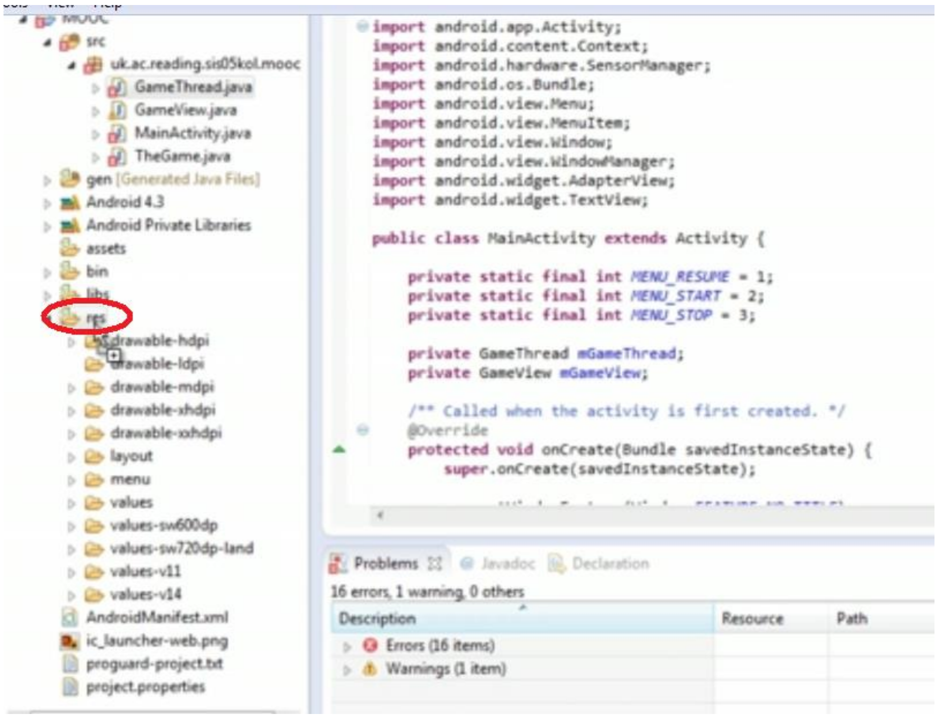
Figure 20: Copy files from Game framework (res)

This is where all resources ('res') are held.

Figure 21: Game framework drawable folder

Notice that there is a 'drawable' folder.

Now look at our newly created 'MOOC' project's folder structure. There is a 'res' folder but not a 'drawable' folder.



```
MOOC
  src
    uk.ac.reading.sis05kol.mooc
      GameThread.java
      GameView.java
      MainActivity.java
      TheGame.java
  gen [Generated Java Files]
  Android 4.3
  Android Private Libraries
  assets
  bin
  libs
  res
    drawable-hdpi
    drawable-ldpi
    drawable-mdpi
    drawable-xhdpi
    drawable-xxhdpi
    layout
    menu
    values
    values-sw600dp
    values-sw720dp-land
    values-v11
    values-v14
  AndroidManifest.xml
  ic_launcher-web.png
  proguard-project.txt
  project.properties
```

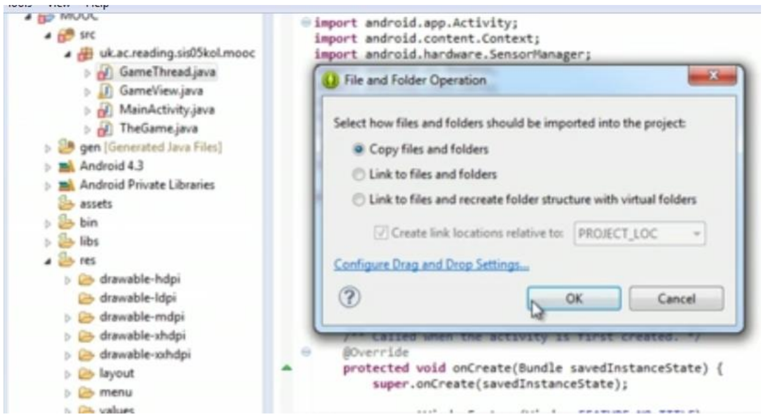
```
import android.app.Activity;
import android.content.Context;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.Window;
import android.view.WindowManager;
import android.widget.AdapterView;
import android.widget.TextView;

public class MainActivity extends Activity {

    private static final int MENU_RESUME = 1;
    private static final int MENU_START = 2;
    private static final int MENU_STOP = 3;

    private GameThread mGameThread;
    private GameView mGameView;

    /** Called when the activity is first created. */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

16 errors, 1 warning, 0 others

Description	Resource	Path
Errors (16 items)		
Warnings (1 item)		

Figure 22: Project view resources (res)

Now drag the 'drawable' folder from the downloaded code and put it under the 'res' folder of the created MOOC project. It will ask for confirmation and select 'Copy Files and folders'



```
import android.app.Activity;
import android.content.Context;
import android.hardware.SensorManager;
```

File and Folder Operation

Select how files and folders should be imported into the project:

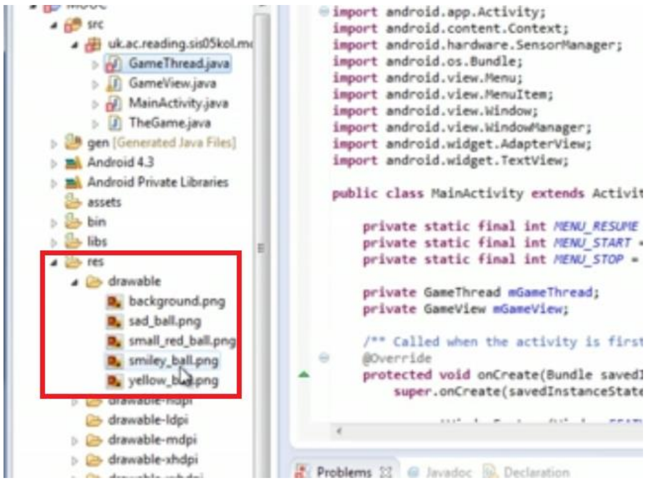
- [✓ Copy files and folders]
- Link to files and folders
- Link to files and recreate folder structure with virtual folders

Create link locations relative to: PROJECT_LOC

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

Figure 23: copy file confirmation dialog

Now your MOOC project should have a ‘drawable’ folder (containing couple of .png files) under ‘res’ folder.

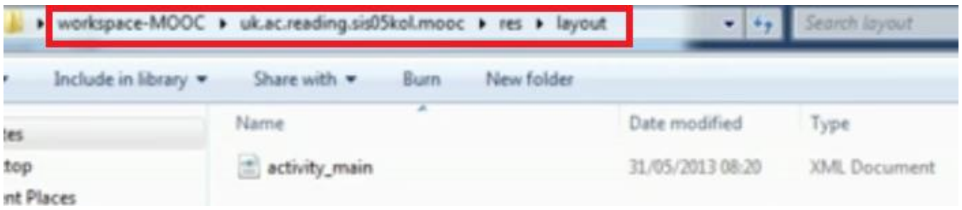


```
import android.app.Activity;
import android.content.Context;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.Window;
import android.view.WindowManager;
import android.widget.AdapterView;
import android.widget.TextView;

public class MainActivity extends Activit

    private static final int MENU_RESUME
    private static final int MENU_START
    private static final int MENU_STOP =

    private GameThread mGameThread;
    private GameView mGameView;

    /** Called when the activity is first
    @Override
    protected void onCreate(Bundle savedI
        super.onCreate(savedInstanceState
```

Figure 24: Copied files and folder drawable in Project view

If you want to change the Graphics in the game later on, this is where (res→drawable) you should do that.

Next we need to add some files to ‘layout’ of our newly created project. So go to downloaded Game framework’s res→ layout.

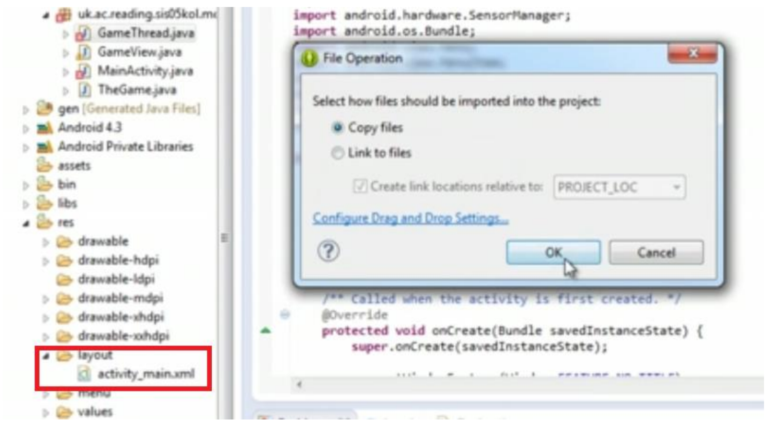


workspace-MOOC ▸ uk.ac.reading.sis05kol.mooc ▸ res ▸ layout

Name	Date modified	Type
activity_main	31/05/2013 08:20	XML Document

Figure 25: Game framework layout

We need to replace our project’s res→layout’s activity_main file with this one. So copy this file and paste it to the MOOC project’s res→layout.



```
import android.hardware.SensorManager;
import android.os.Bundle;
```

Select how files should be imported into the project:

- [✓ Copy files]
- Link to files

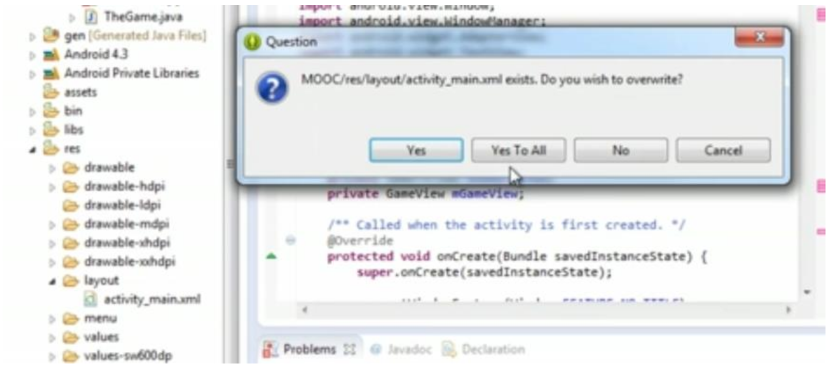
Create link locations relative to: PROJECT_LOC

Configure Drag and Drop Settings...

```
    /** Called when the activity is first created. */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

Figure 26: Copying to project resource layout

It will ask you for confirmation and click ‘OK’. Then from the next dialog select ‘Yes To All’.

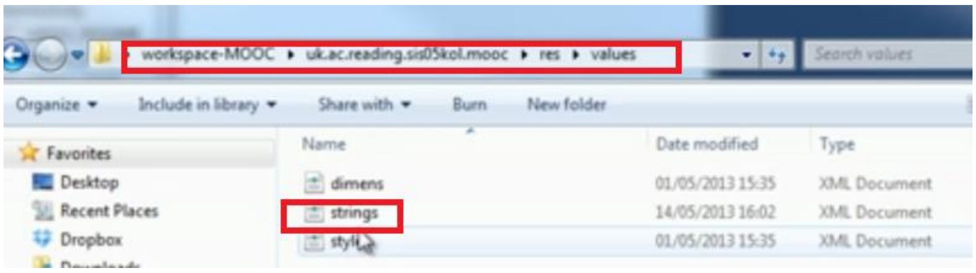


MOOC/res/layout/activity_main.xml exists. Do you wish to overwrite?

```
import android.view.WindowManager;
    private GameView mGameView;

    /** Called when the activity is first created. */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

Figure 27: Confirm file copy dialog

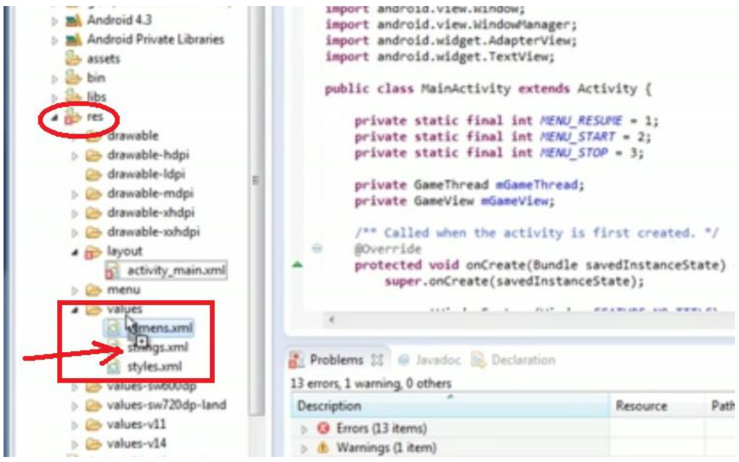
Next we need to copy the ‘Strings’ file from downloaded game framework to MOOC project. Strings file comes under the res→values. Copy it from the game framework



Name	Date modified	Type
dimens	01/05/2013 15:35	XML Document
strings	14/05/2013 16:02	XML Document
styles	01/05/2013 15:35	XML Document

Figure 28: Game framework resource values

Then paste it overwriting the existing file in the MOOC project.



```
import android.view.Window;
import android.view.WindowManager;
import android.widget.AdapterView;
import android.widget.TextView;

public class MainActivity extends Activity {

    private static final int MENU_RESUME = 1;
    private static final int MENU_START = 2;
    private static final int MENU_STOP = 3;

    private GameThread mGameThread;
    private GameView mGameView;

    /** Called when the activity is first created. */
    @Override
    protected void onCreate(Bundle savedInstanceState)
        super.onCreate(savedInstanceState);
```

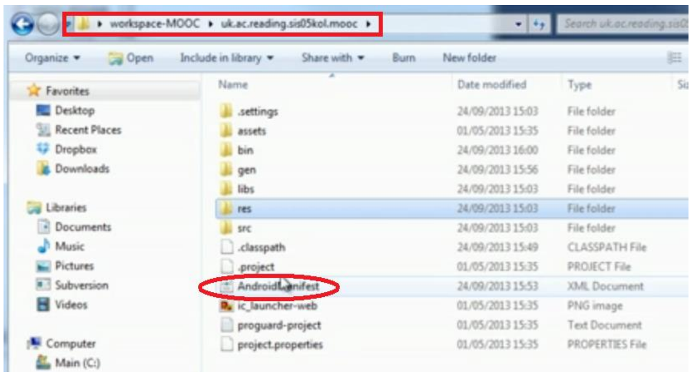
13 errors, 1 warning, 0 others

Description	Resource	Path
Errors (13 items)		
Warnings (1 item)		

Figure 29: Project resource values

Again it will ask you for confirmation and confirm that you want to copy it over.

Next we have to copy the AndroidManifest file from game framework to our MOOC project.



Name	Date modified	Type
.settings	24/09/2013 15:03	File folder
assets	01/05/2013 15:35	File folder
bin	24/09/2013 16:00	File folder
gen	24/09/2013 15:56	File folder
libs	24/09/2013 15:03	File folder
res	24/09/2013 15:03	File folder
src	24/09/2013 15:03	File folder
.classpath	24/09/2013 15:49	CLASSPATH File
.project	01/05/2013 15:35	PROJECT File
AndroidManifest	24/09/2013 15:53	XML Document
ic_launcher-web	01/05/2013 15:35	PNG image
proguard-project	01/05/2013 15:35	Text Document
project.properties	01/05/2013 15:35	PROPERTIES File

Figure 30: Game framework AndroidManifest

Now copy it over to the MOOC project and replace the existing AndroidManifest file. (You need to paste this file into the MOOC folder) .Again you will be asked to confirm.

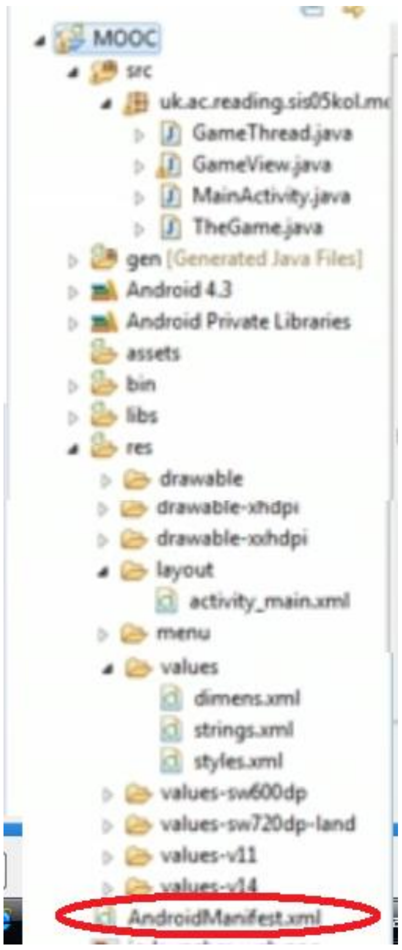


Figure 31: AndroidManifest on Project view

Now all the red cross marks appeared should be gone.

Next we are going to clean the project and do a fresh build. For this go to 'Project' on the menu bar and select 'Clean'.

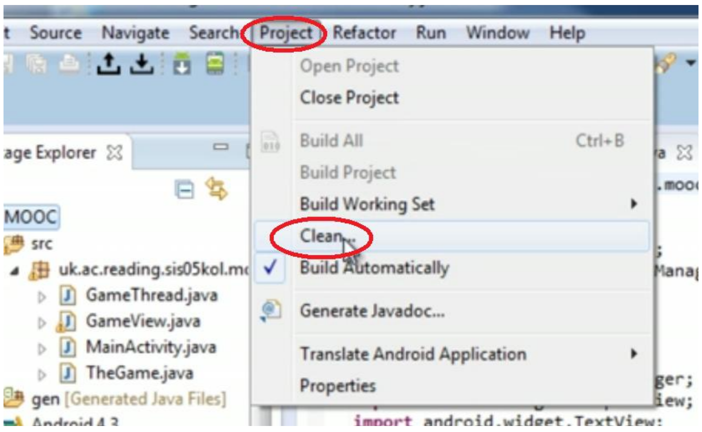
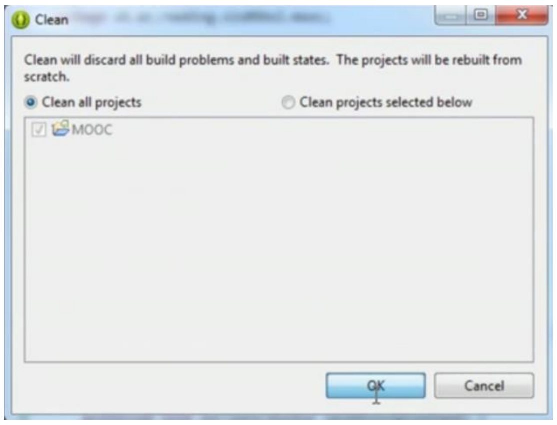


Figure 32: Clean project

Now Select 'OK'


Clean will discard all build problems and built states. The projects will be rebuilt from scratch.

[✓ Clean all projects] Clean projects selected below

MOOC

Figure 33: Clean project dialog

Now for the first time let us run the code.

For this go to the MOOC project. Right click the MOOC project and select Run As → Android Application.

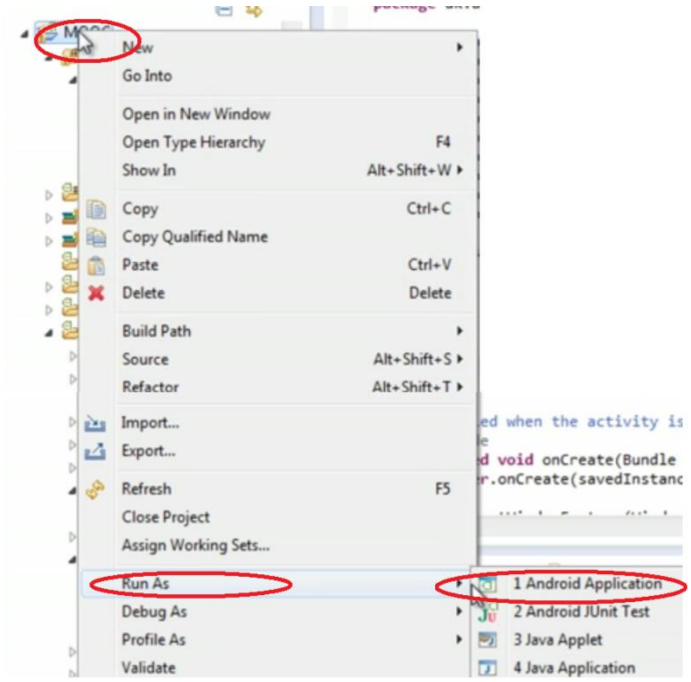


Figure 34: Run project as Android Application

Now click on the minimized Emulator window to get it to the foreground. There will be a bit of a wait but then it appears on the emulator!

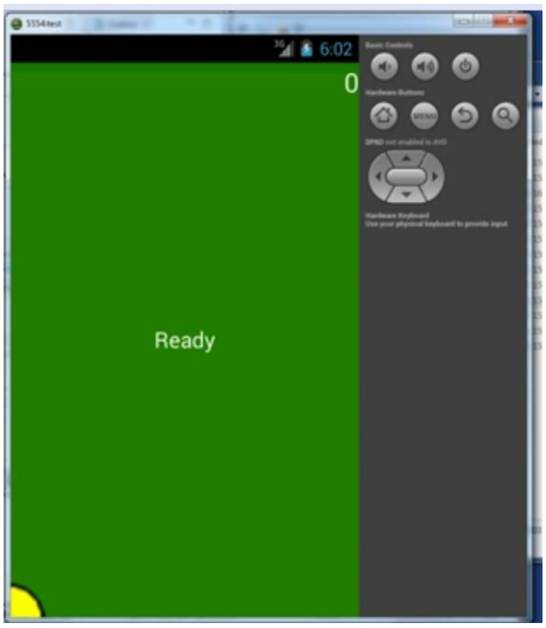
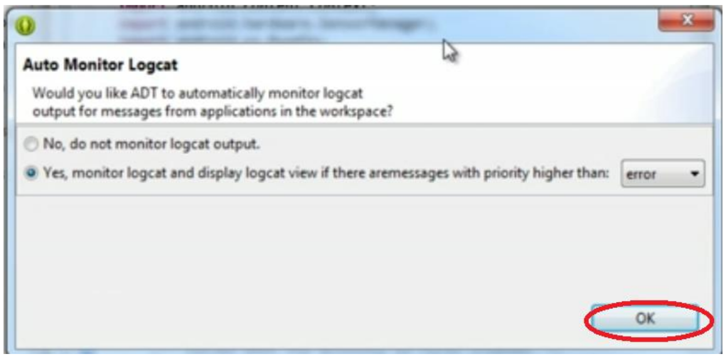


Figure 35:Running on emulator

Now it will show you a dialog to which you need to select ‘OK’



Auto Monitor Logcat

Would you like ADT to automatically monitor logcat output for messages from applications in the workspace?

- No, do not monitor logcat output.
- [✓ Yes, monitor logcat and display logcat view if there aremessages with priority higher than:] error

OK

Figure 36: Auto Monitor Logcat

Now we are going to add some code into our game.

Go into the downloaded game framework folder. Go into TheGameVersions. Here you will see some java files. Right click on v1.java and open with a text editor. I am using Notepad++

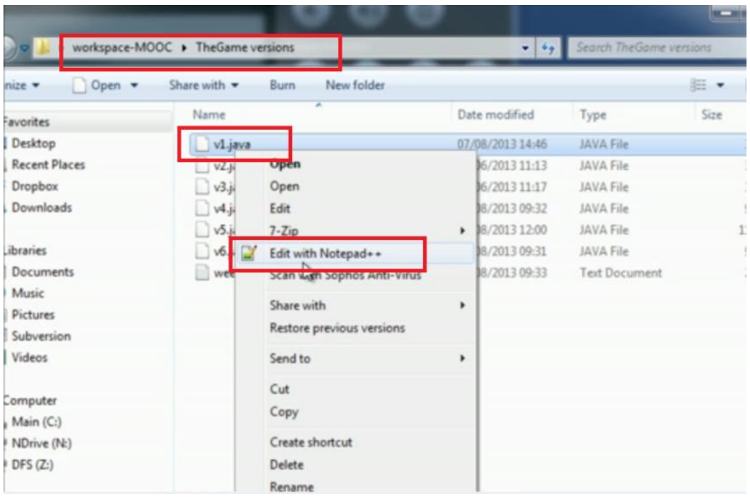
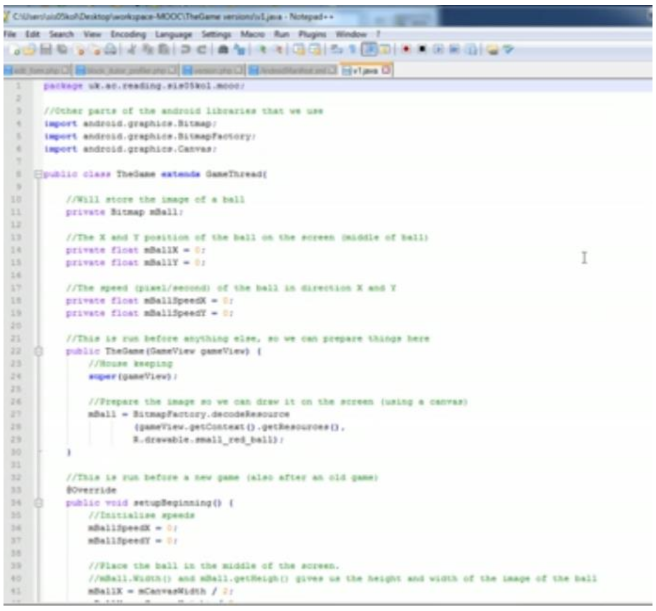


Figure 37: Game framework - Game version - 1

When opened you will see the contents of file v1. Java

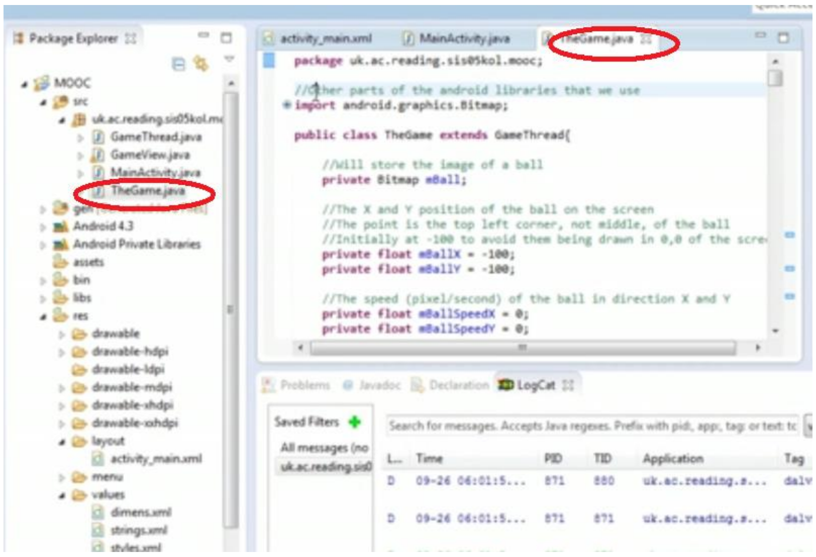


```
package uk.ac.reading.sis05kol.mooc;

//Other parts of the android libraries that we use
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;

public class TheGame extends GameThread{

    //Will store the image of a ball
    private Bitmap mBall;

    //The X and Y position of the ball on the screen (middle of ball)
    private float mBallX = 0;
    private float mBallY = 0;

    //The speed (pixel/second) of the ball in direction X and Y
    private float mBallSpeedX = 0;
    private float mBallSpeedY = 0;

    //This is run before anything else, so we can prepare things here
    public TheGame(GameView gameView) {
        //House keeping
        super(gameView);

        //Prepare the image so we can draw it on the screen (using a canvas)
        mBall = BitmapFactory.decodeResource
                (gameView.getContext().getResources(),
                R.drawable.small_red_ball);
    }

    //This is run before a new game (also after an old game)
    @Override
    public void setupBeginning() {
        //Initialise speeds
        mBallSpeedX = 0;
        mBallSpeedY = 0;

        //Place the ball in the middle of the screen.
        //mBall.Width() and mBall.getHeigh() gives us the height and width of the image of the ball
        mBallX = mCanvasWidth / 2;
```

Figure 38: file v1.java

Select all of its text (CTRL + A) and copy it to clipboard (CTRL+C). Then go to the MOOC project.

The contents of the file needs to be copied to TheGame.java file located in the 'src' folder. Now select all from this file and delete it then past the copied content to this file.



```
package uk.ac.reading.sis05kol.mooc;

//Other parts of the android libraries that we use
import android.graphics.Bitmap;

public class TheGame extends GameThread{

    //Will store the image of a ball
    private Bitmap mBall;

    //The X and Y position of the ball on the screen
    //The point is the top left corner, not middle, of the ball
    //Initially at -100 to avoid them being drawn in 0,0 of the scre
    private float mBallX = -100;
    private float mBallY = -100;

    //The speed (pixel/second) of the ball in direction X and Y
    private float mBallSpeedX = 0;
    private float mBallSpeedY = 0;
```

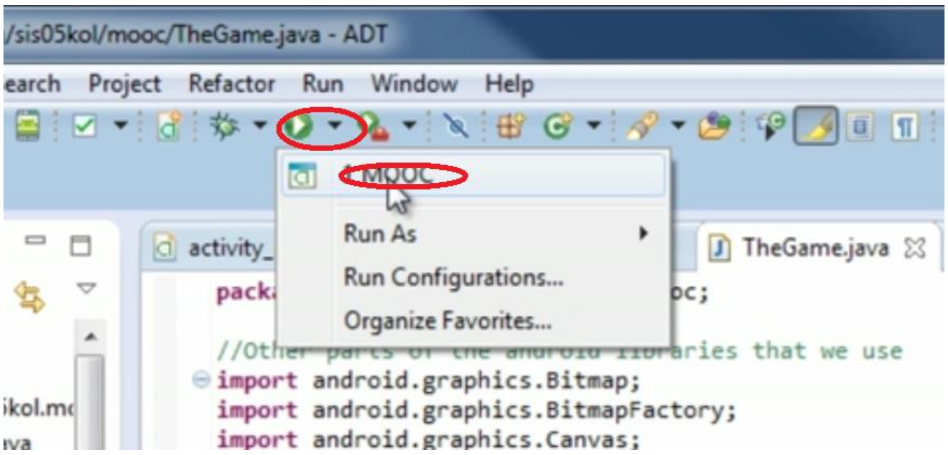
```
Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: tc

L...  Time               PID   TID   Application           Tag
D     09-26 06:01:5...   871   880   uk.ac.reading.s...    dalv
D     09-26 06:01:5...   871   871   uk.ac.reading.s...    dalv
```

Figure 39: Project TheGame.java

Now we have Week 1 code in our TheGame.java file.

Let us run the project.

For this you need to go to Click on the down arrow on Run icon and then select the project name 'MOOC'.



```
pack                                        oc;

//Other parts of the android libraries that we use
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
```

Figure 40: Run project

As we previously did, bring the emulator to foreground.

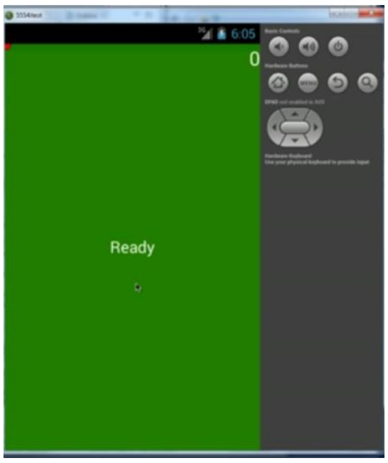


Figure 41: Run on emulator

Clicking on the screen will bring a red ball to the screen.

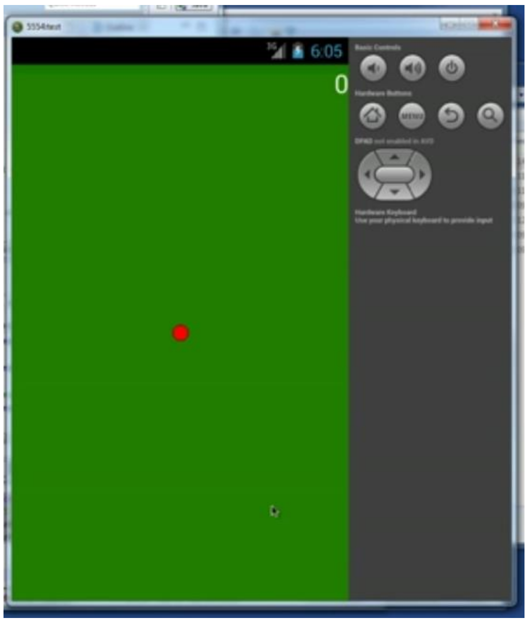
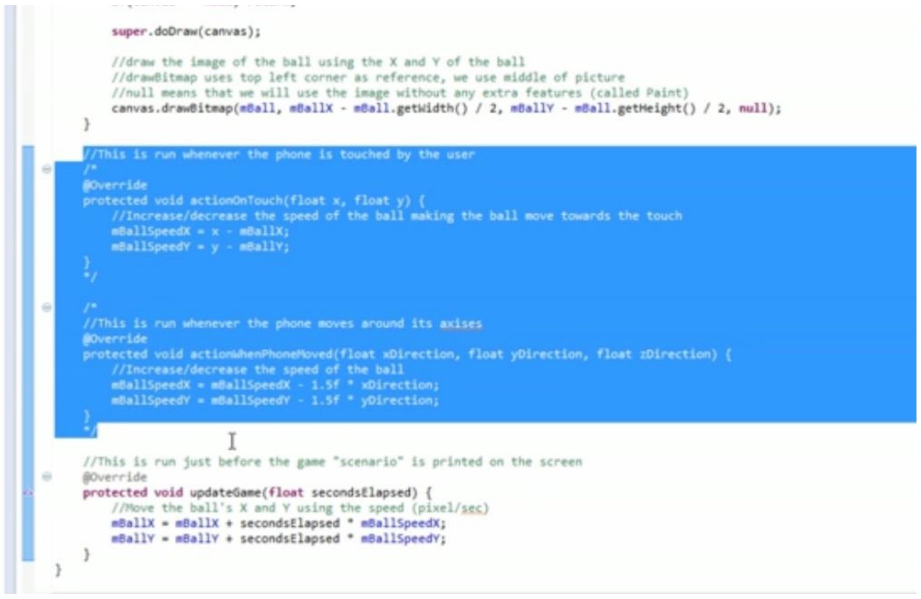


Figure 42: Run on Emulator - red ball

This is a boring game because nothing happens when I click on the screen.

At this point we are introducing ‘comments’ in coding. Comments are used to describe complex logic in a program or source code to improve human readability. Comments are ignored by the computer when executing the code. However, if you go down on TheGame.java file in the MOOC project you can see there is a lot of code (shown highlighted in the figure) that is in green colour. In this text editor comments are shown in green.



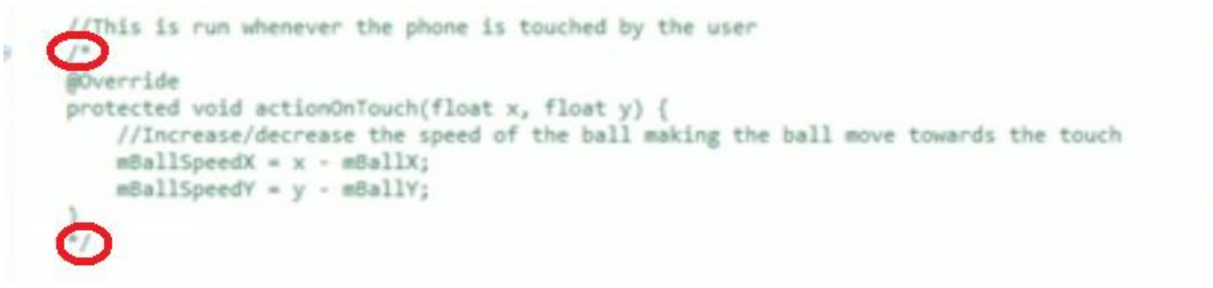
```
        super.doDraw(canvas);

        //draw the image of the ball using the X and Y of the ball
        //drawBitmap uses top left corner as reference, we use middle of picture
        //null means that we will use the image without any extra features (called Paint)
        canvas.drawBitmap(mBall, mBallX - mBall.getWidth() / 2, mBallY - mBall.getHeight() / 2, null);
    }

    //This is run whenever the phone is touched by the user
    /*
    @Override
    protected void actionOnTouch(float x, float y) {
        //Increase/decrease the speed of the ball making the ball move towards the touch
        mBallSpeedX = x - mBallX;
        mBallSpeedY = y - mBallY;
    }
    */

    /*
    //This is run whenever the phone moves around its axises
    @Override
    protected void actionWhenPhoneMoved(float xDirection, float yDirection, float zDirection) {
        //Increase/decrease the speed of the ball
        mBallSpeedX = mBallSpeedX - 1.5f * xDirection;
        mBallSpeedY = mBallSpeedY - 1.5f * yDirection;
    }
    */

    //This is run just before the game "scenario" is printed on the screen
    @Override
    protected void updateGame(float secondsElapsed) {
        //Move the ball's X and Y using the speed (pixel/sec)
        mBallX = mBallX + secondsElapsed * mBallSpeedX;
        mBallY = mBallY + secondsElapsed * mBallSpeedY;
    }
}
```

Figure 43: TheGame.java comments

Actually what we have done here is commenting out some of the code we have written in the game and that is why the instructions are ignored by the computer. Hence nothing happens in the game. Now let us un-comment this block of instructions to see what happens.

Concentrate on the two blocks of code ‘actionOnTouch’ and ‘actionWhenPhoneMoved’. To run the code in emulator we need to remove the /* and */ symbols that are placed between ‘actionOnTouch’ block. If you are running it on the phone /* and */ symbols that are placed between ‘actionWhenPhoneMoved’ should be removed. We will learn about comments later but what happens when you use /* and */ is that anything placed between these two sets of symbols is considered a comment. So because ‘actionOnTouch’ is between the opening (/*) and closing (*/) comments symbols the computer completely ignores what it should do when ‘actionOnTouch’ happens.

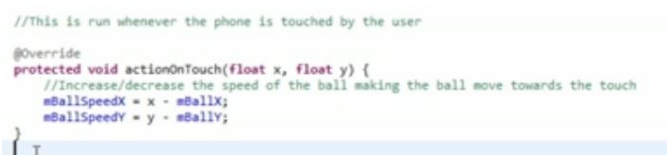
Now that we know a bit about comments let’s un-comment this block. For that delete the highlighted bits from the file.



```
//This is run whenever the phone is touched by the user
/*
@Override
protected void actionOnTouch(float x, float y) {
    //Increase/decrease the speed of the ball making the ball move towards the touch
    mBallSpeedX = x - mBallX;
    mBallSpeedY = y - mBallY;
}
*/
```

Figure 44: Comments in code

Did you notice that the colour of the text changed from being green?



```
//This is run whenever the phone is touched by the user

@Override
protected void actionOnTouch(float x, float y) {
    //Increase/decrease the speed of the ball making the ball move towards the touch
    mBallSpeedX = x - mBallX;
    mBallSpeedY = y - mBallY;
}
```

Figure 45: Uncommented block of code

Now run the project again (just like you did before) and see what the difference is. Now it will respond when you click on the screen!

If you have reached this point successfully that means you have fully setup your working enviornment and you are ready for Week 2. If you have an Android phone or tablet you could try installing the game into it using the guide we have provided in 1.10.