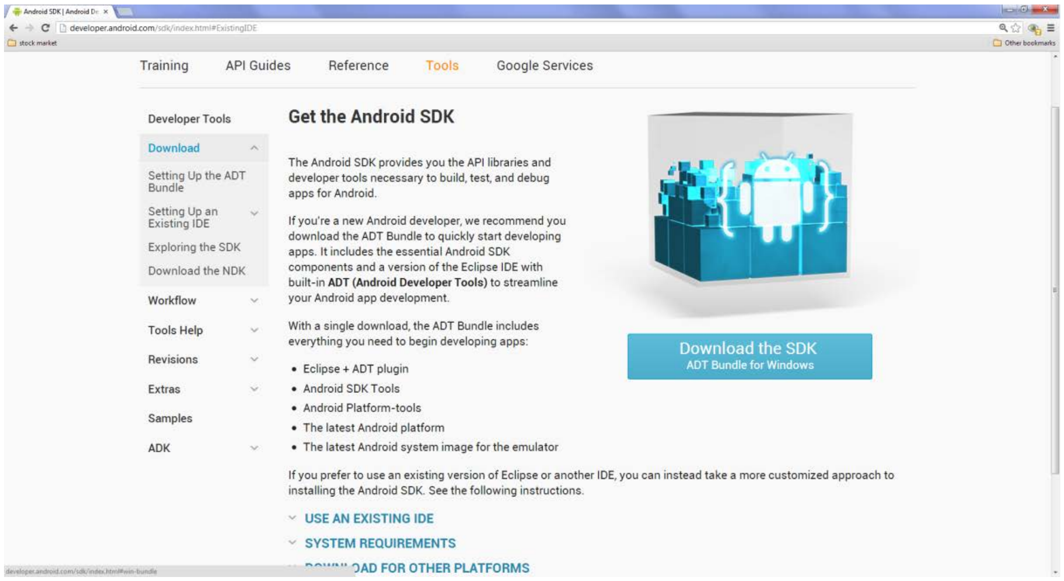
Code Readme

Please follow these steps for setting up Eclipse, obtaining Android SDK platforms, and obtaining AndEngine libraries. This code readme file will help you to successfully set up your environment and then execute all the code provided along with this book.

Here are the steps:

1. Download the Android ADT Bundle from `http://developer.android.com/sdk/index.html#ExistingIDE`, clicking the large blue button with the text, **Download the SDK**.



Get the Android SDK

The Android SDK provides you the API libraries and developer tools necessary to build, test, and debug apps for Android.

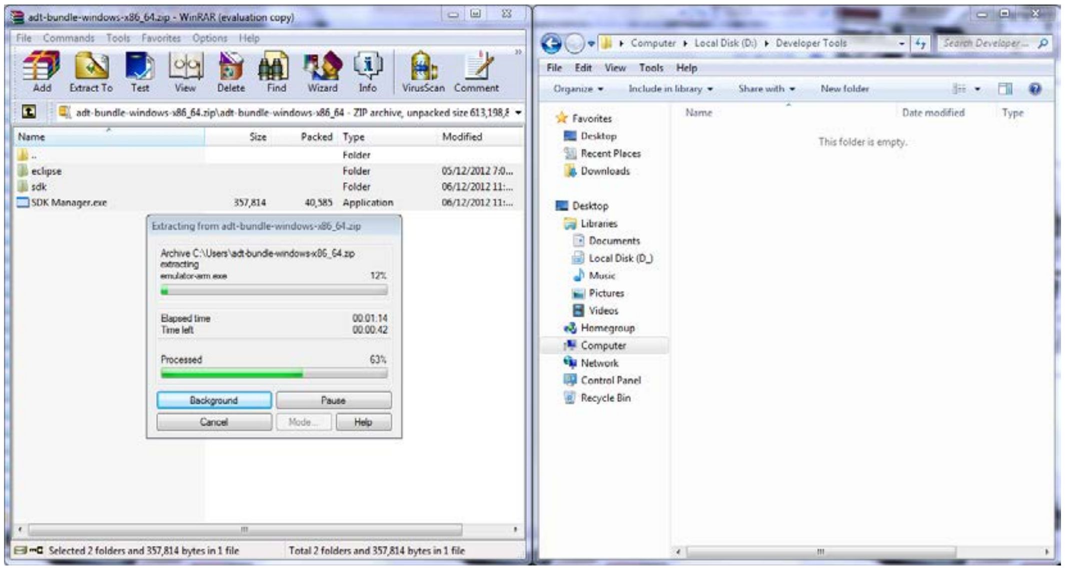
If you're a new Android developer, we recommend you download the ADT Bundle to quickly start developing apps. It includes the essential Android SDK components and a version of the Eclipse IDE with built-in **ADT (Android Developer Tools)** to streamline your Android app development.

With a single download, the ADT Bundle includes everything you need to begin developing apps:

- Eclipse + ADT plugin
- Android SDK Tools
- Android Platform-tools
- The latest Android platform
- The latest Android system image for the emulator

If you prefer to use an existing version of Eclipse or another IDE, you can instead take a more customized approach to installing the Android SDK. See the following instructions.

2. Extract the contents of the downloaded `.zip` file to a folder of your choice. The contents of the `.zip` file does not include any installers, so be sure not to place them in a temporary folder.


adt-bundle-windows-x86_64.zip - WinRAR (evaluation copy)
File Commands Tools Favorites Options Help
Add Extract To Test View Delete Find Wizard Info VirusScan Comment
Background Pause Cancel Mode... Help
Computer ▸ Local Disk (D:) ▸ Developer Tools
Search Developer...
File Edit View Tools Help
Organize Include in library Share with New folder
Favorites
Desktop
Recent Places
Downloads
Desktop
Libraries
Documents
Local Disk (D_)
Music
Pictures
Videos
Homegroup
Computer
Network
Control Panel
Recycle Bin

adt-bundle-windows-x86_64.zip\adt-bundle-windows-x86_64 - ZIP archive, unpacked size 613,198,8

Name	Size	Packed	Type	Modified
..			Folder	
eclipse			Folder	05/12/2012 7:0...
sdk			Folder	06/12/2012 11:...
SDK Manager.exe	357,814	40,585	Application	06/12/2012 11:...

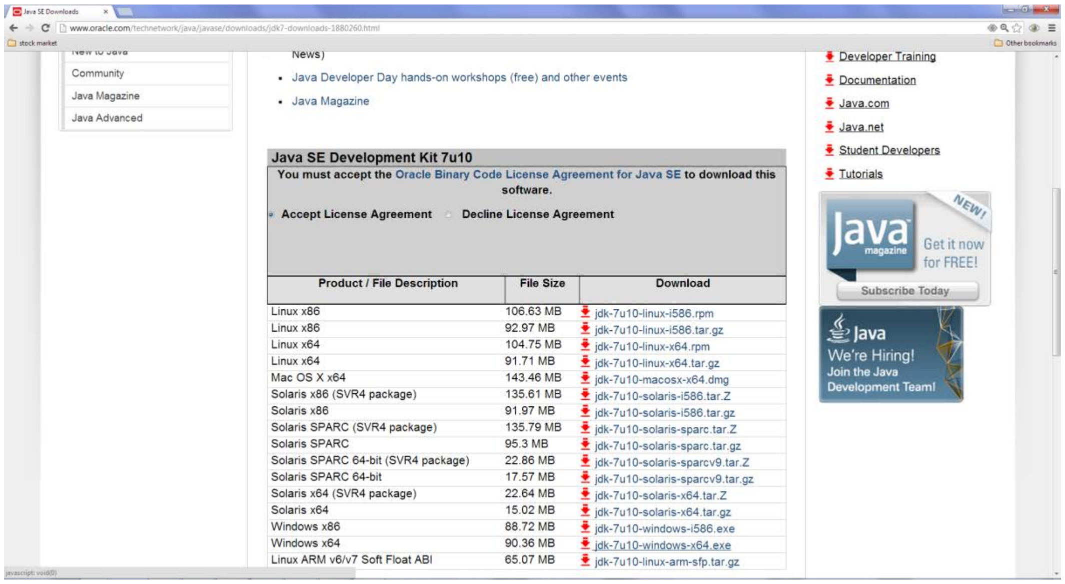
```
Extracting from adt-bundle-windows-x86_64.zip

Archive C:\Users\adt-bundle-windows-x86_64.zip
extracting
emulator-arm.exe                         12%

Elapsed time                        00:01:14
Time left                           00:00:42

Processed                                63%
```

Selected 2 folders and 357,814 bytes in 1 file | Total 2 folders and 357,814 bytes in 1 file

Name	Date modified	Type

This folder is empty.

3. Download and install a copy of the Java SE Development Kit 7u10 (JDK) from `http://oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html`.



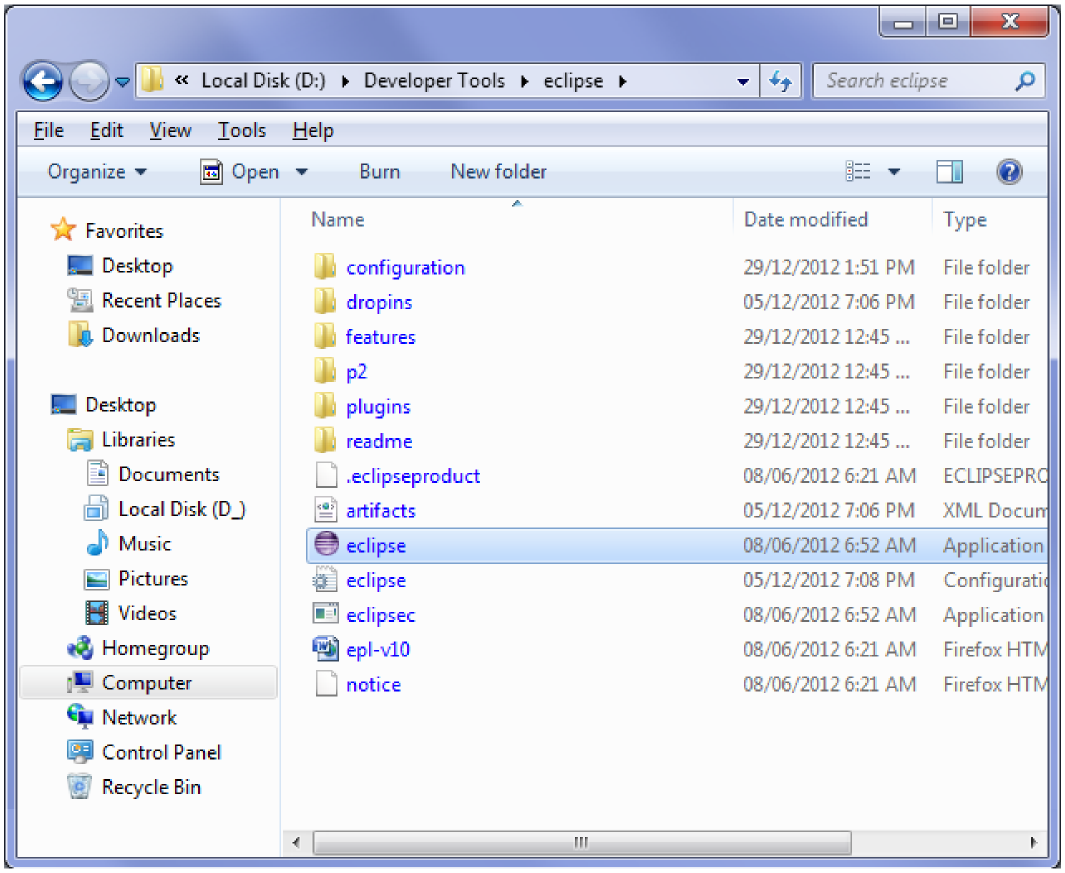
Java SE Development Kit 7u10

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

[✓ Accept License Agreement] Decline License Agreement

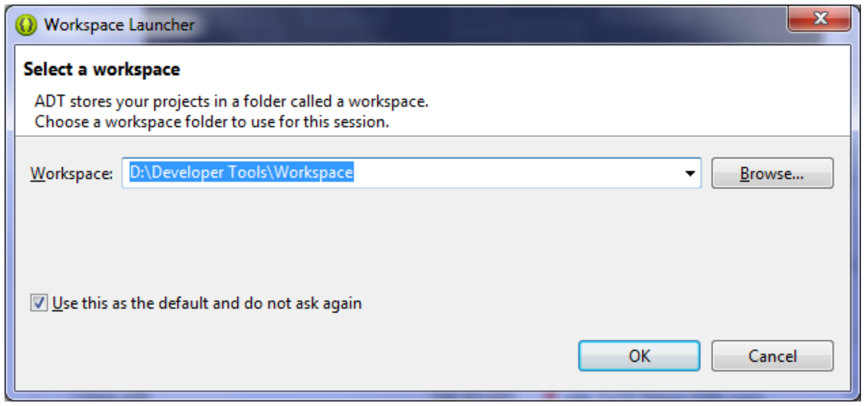
Product / File Description	File Size	Download
Linux x86	106.63 MB	jdk-7u10-linux-i586.rpm
Linux x86	92.97 MB	jdk-7u10-linux-i586.tar.gz
Linux x64	104.75 MB	jdk-7u10-linux-x64.rpm
Linux x64	91.71 MB	jdk-7u10-linux-x64.tar.gz
Mac OS X x64	143.46 MB	jdk-7u10-macosx-x64.dmg
Solaris x86 (SVR4 package)	135.61 MB	jdk-7u10-solaris-i586.tar.Z
Solaris x86	91.97 MB	jdk-7u10-solaris-i586.tar.gz
Solaris SPARC (SVR4 package)	135.79 MB	jdk-7u10-solaris-sparc.tar.Z
Solaris SPARC	95.3 MB	jdk-7u10-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	22.86 MB	jdk-7u10-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	17.57 MB	jdk-7u10-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	22.64 MB	jdk-7u10-solaris-x64.tar.Z
Solaris x64	15.02 MB	jdk-7u10-solaris-x64.tar.gz
Windows x86	88.72 MB	jdk-7u10-windows-i586.exe
Windows x64	90.36 MB	jdk-7u10-windows-x64.exe
Linux ARM v6/v7 Soft Float ABI	65.07 MB	jdk-7u10-linux-arm-sfp.tar.gz

4. Visit the folder that the `.zip` file contents were extracted to in step 2. Navigate to the **eclipse** folder and launch the **eclipse** application.



Name	Date modified	Type
configuration	29/12/2012 1:51 PM	File folder
dropins	05/12/2012 7:06 PM	File folder
features	29/12/2012 12:45 ...	File folder
p2	29/12/2012 12:45 ...	File folder
plugins	29/12/2012 12:45 ...	File folder
readme	29/12/2012 12:45 ...	File folder
.eclipseproduct	08/06/2012 6:21 AM	ECLIPSEPRO
artifacts	05/12/2012 7:06 PM	XML Docum
eclipse	08/06/2012 6:52 AM	Application
eclipse	05/12/2012 7:08 PM	Configurati
eclipsec	08/06/2012 6:52 AM	Application
epl-v10	08/06/2012 6:21 AM	Firefox HTM
notice	08/06/2012 6:21 AM	Firefox HTM

5. The Eclipse IDE will prompt you to create a new workspace. Define a path of your choice and click **OK**.



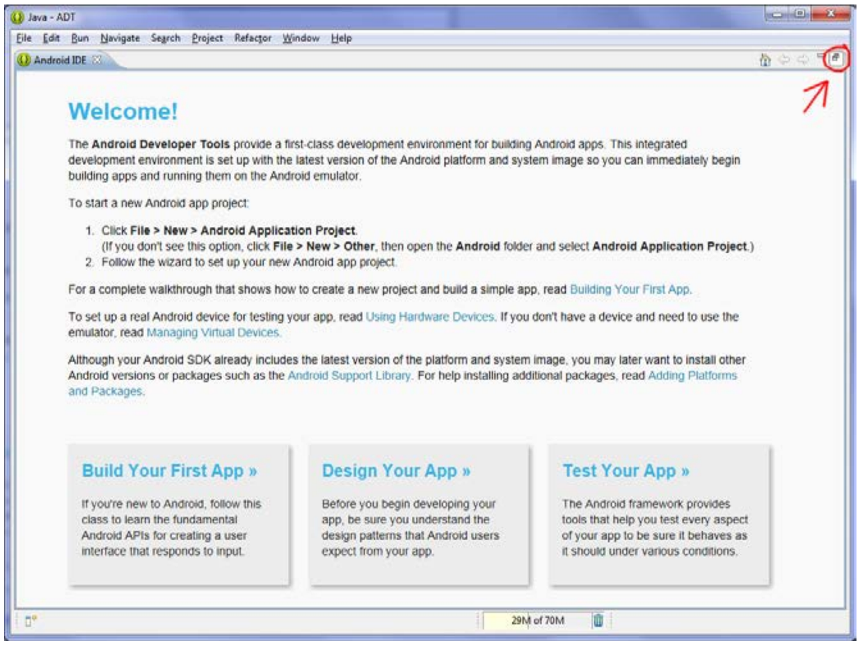
Select a workspace

ADT stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: D:\Developer Tools\Workspace

☑ Use this as the default and do not ask again

6. Once Eclipse is established and running, we will be taken to the **Welcome!** screen on first launch. Feel free to read over the contents of this page—click on the minimize icon in the top-right corner of the eclipse workspace.



Welcome!

The **Android Developer Tools** provide a first-class development environment for building Android apps. This integrated development environment is set up with the latest version of the Android platform and system image so you can immediately begin building apps and running them on the Android emulator.

To start a new Android app project:

1. Click **File > New > Android Application Project**.
   (If you don't see this option, click **File > New > Other**, then open the **Android** folder and select **Android Application Project**.)
2. Follow the wizard to set up your new Android app project.

For a complete walkthrough that shows how to create a new project and build a simple app, read Building Your First App.

To set up a real Android device for testing your app, read Using Hardware Devices. If you don't have a device and need to use the emulator, read Managing Virtual Devices.

Although your Android SDK already includes the latest version of the platform and system image, you may later want to install other Android versions or packages such as the Android Support Library. For help installing additional packages, read Adding Platforms and Packages.

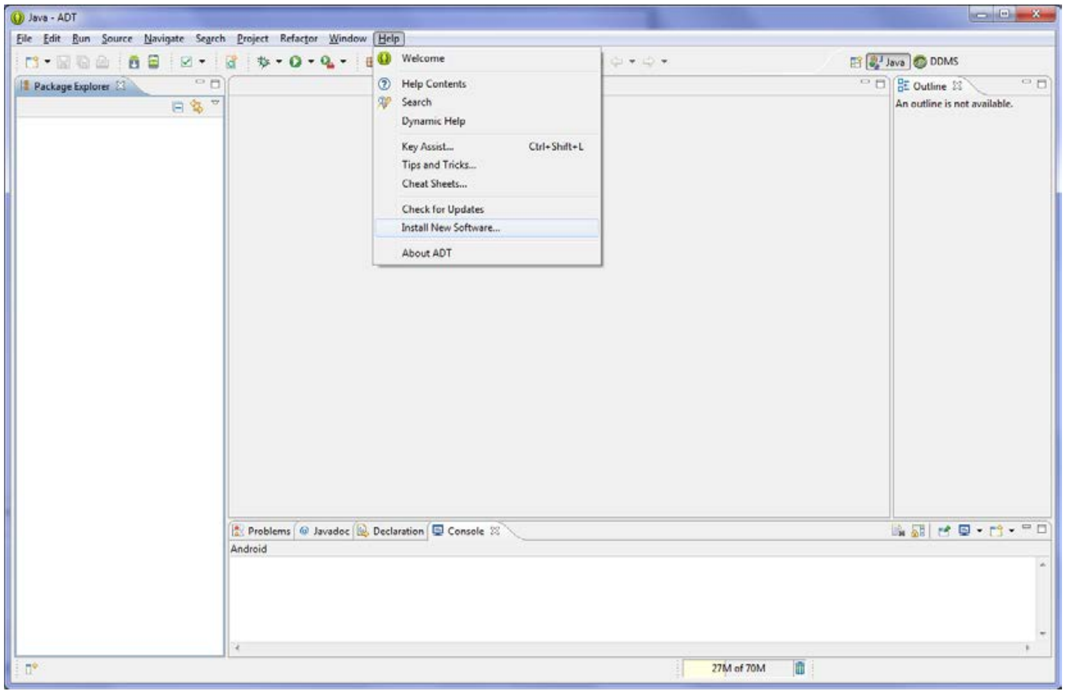
Build Your First App »
If you're new to Android, follow this class to learn the fundamental Android APIs for creating a user interface that responds to input.

Design Your App »
Before you begin developing your app, be sure you understand the design patterns that Android users expect from your app.

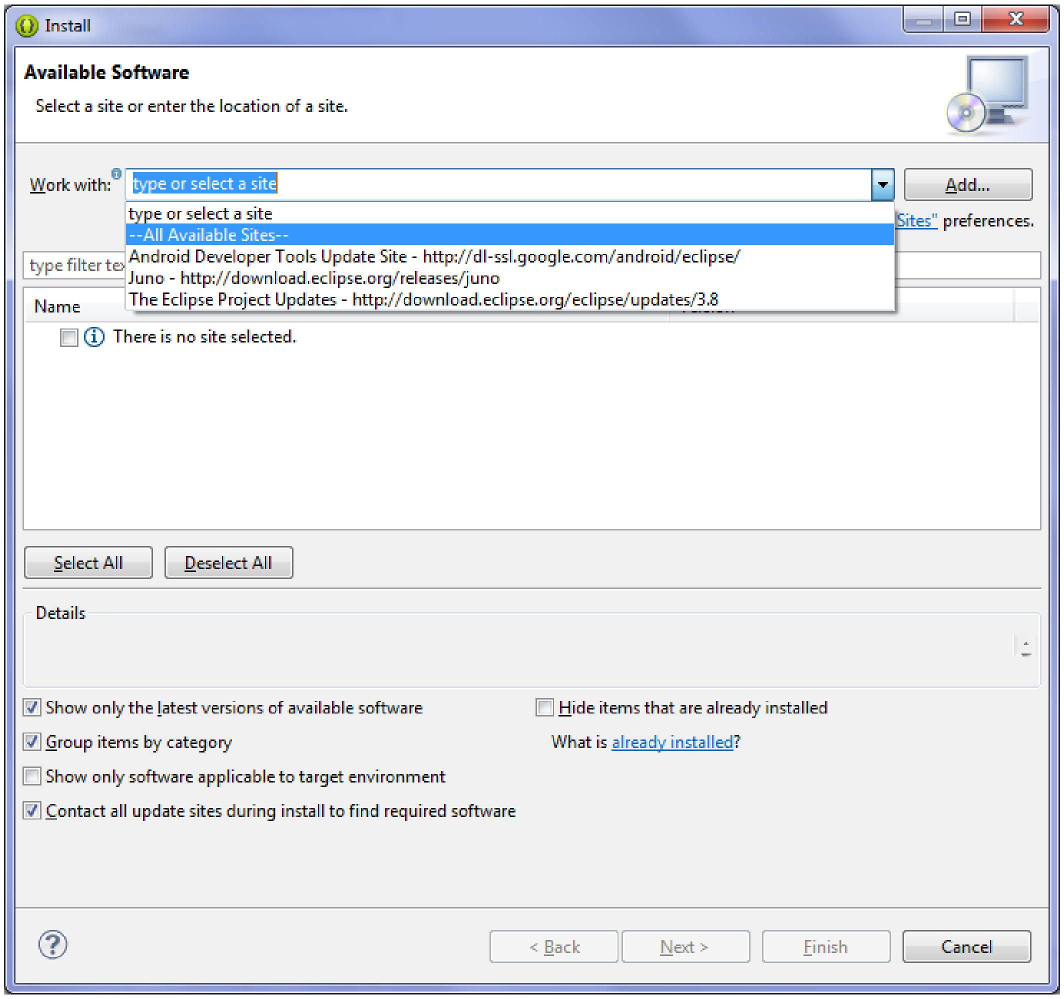
Test Your App »
The Android framework provides tools that help you test every aspect of your app to be sure it behaves as it should under various conditions.

29M of 70M

7. We must install the necessary plug-in which will allow us to pull the AndEngine library projects needed to execute the recipes in the book. Select **Help** from the Eclipse toolbar and then select **Install New Software**.



8. In order to locate the GitHub plug-in, we will search all of the available sites from the drop-down selection.


Install

Available Software

Select a site or enter the location of a site.

Work with: type or select a site

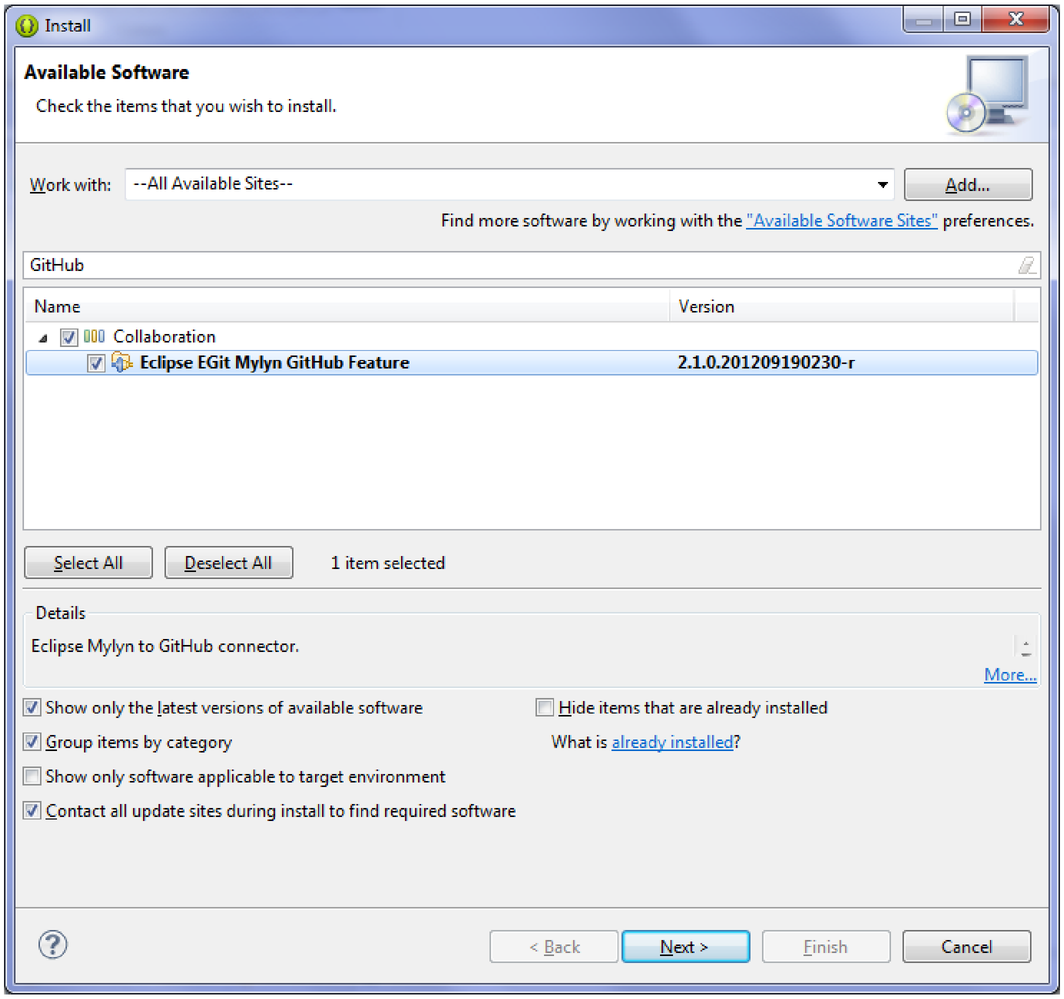
- type or select a site
- --All Available Sites--
- Android Developer Tools Update Site - http://dl-ssl.google.com/android/eclipse/
- Juno - http://download.eclipse.org/releases/juno
- The Eclipse Project Updates - http://download.eclipse.org/eclipse/updates/3.8

There is no site selected.

- [✓] Show only the latest versions of available software
- [✓] Group items by category
- [ ] Show only software applicable to target environment
- [✓] Contact all update sites during install to find required software
- [ ] Hide items that are already installed

What is already installed?

9. Enter **GitHub** into the search text field. The plugin wizard will automatically update the list of plugins, which will return the **Eclipse Egit Mylyn GitHub Feature** plugin. Select the plugin and click **Next**. Follow through with the wizard to finish installing the plugin.



Install

Available Software

Check the items that you wish to install.

Work with: --All Available Sites--

Find more software by working with the "Available Software Sites" preferences.

GitHub

Name	Version
☑ Collaboration	
☑ Eclipse EGit Mylyn GitHub Feature	2.1.0.201209190230-r

1 item selected

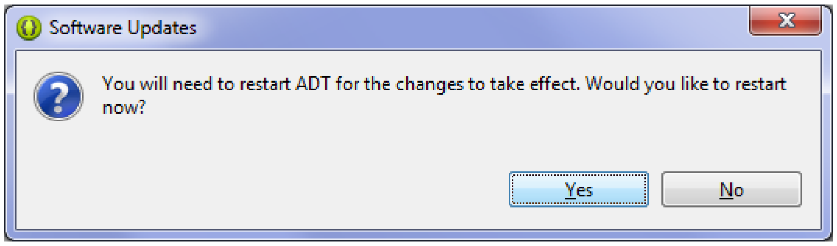
Details

Eclipse Mylyn to GitHub connector.

- ☑ Show only the latest versions of available software
- ☑ Group items by category
- ☐ Show only software applicable to target environment
- ☑ Contact all update sites during install to find required software
- ☐ Hide items that are already installed

What is already installed?

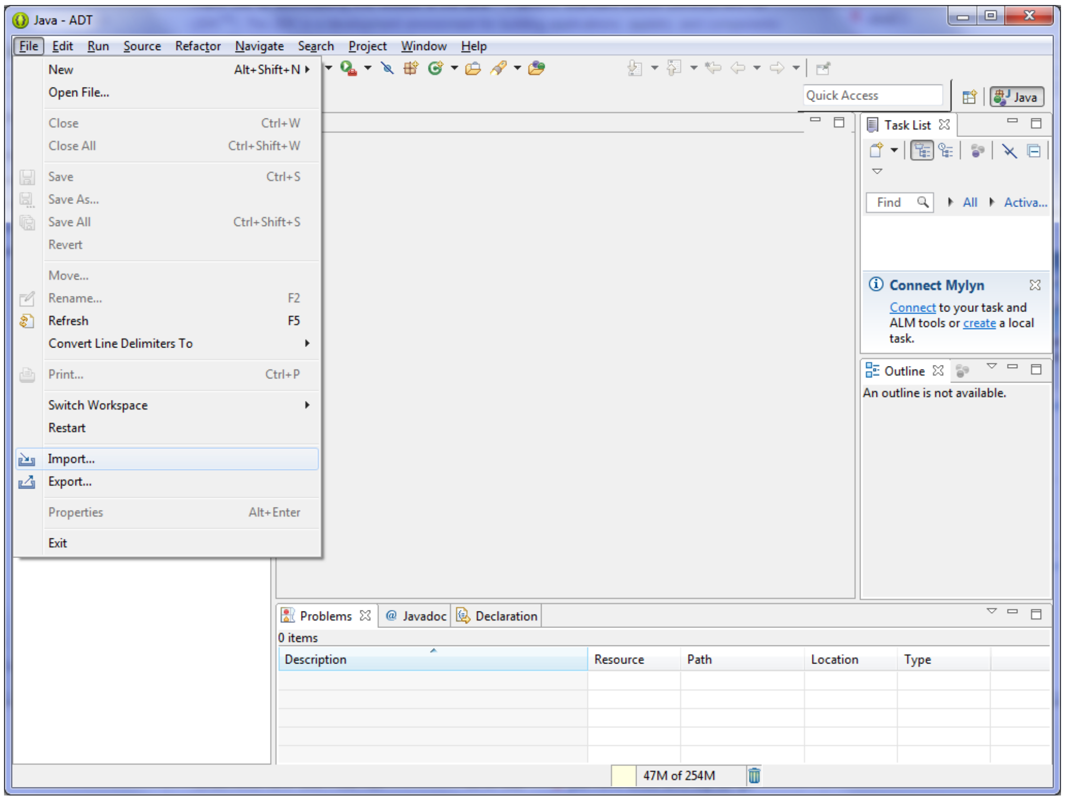
10. In order to make the plugin available to us, we must restart the Eclipse IDE. Select **Yes** when prompted to restart Eclipse.



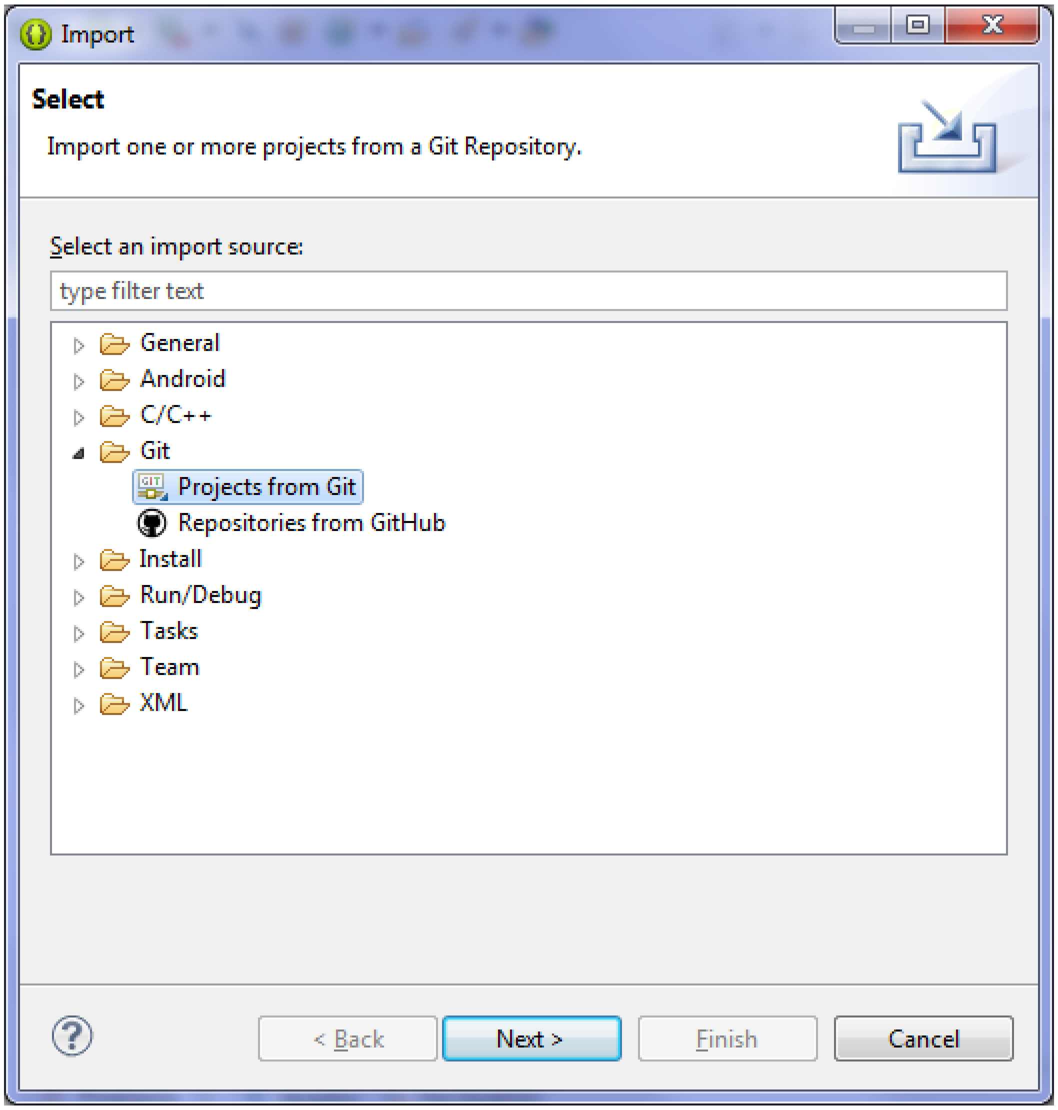
Software Updates

You will need to restart ADT for the changes to take effect. Would you like to restart now?

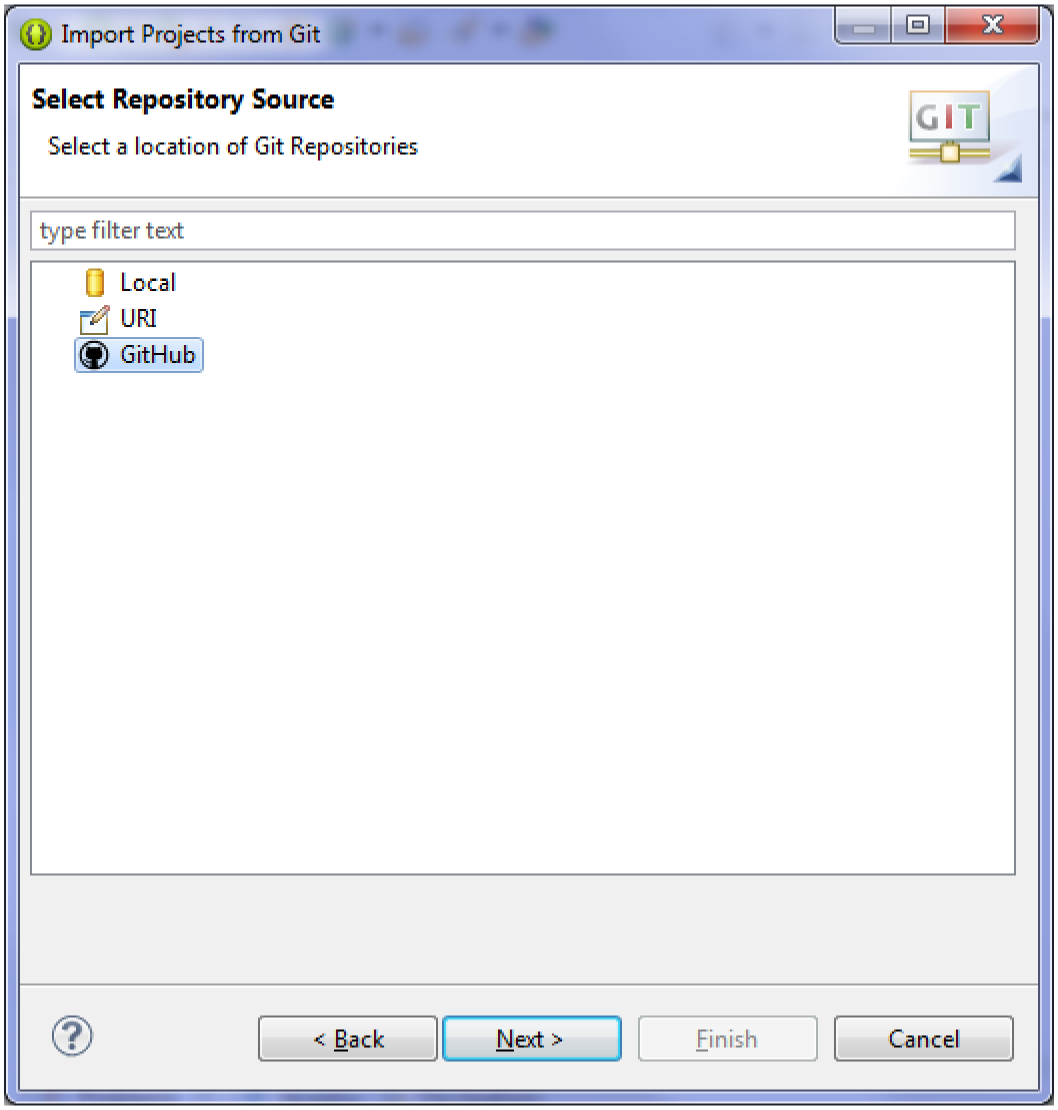
11. Once Eclipse has restarted, select **File** from the Eclipse toolbar. Navigate to the **Import...** selection and click to proceed.



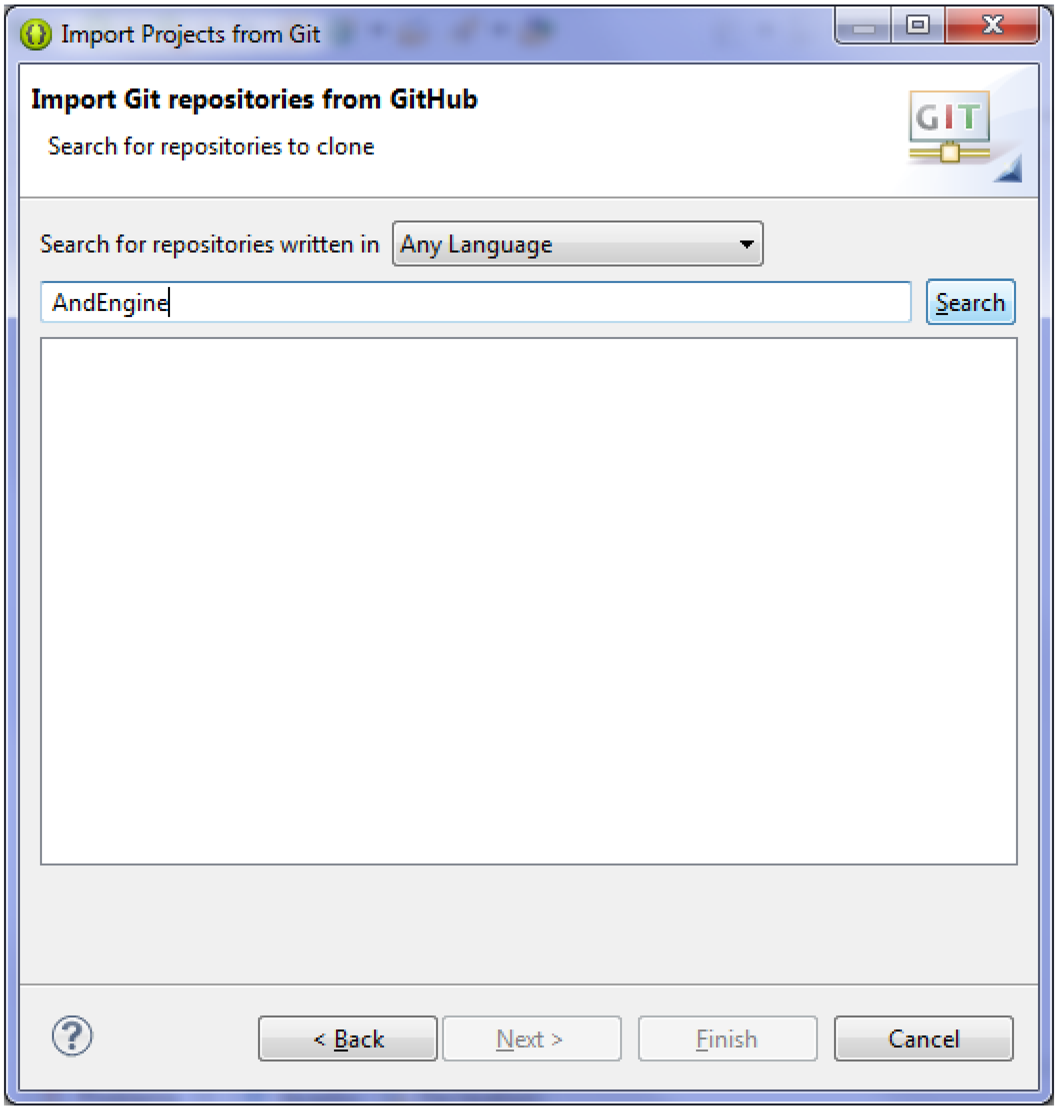
12. Open the **Git** folder and select **Projects from Git**. Click **Next**.



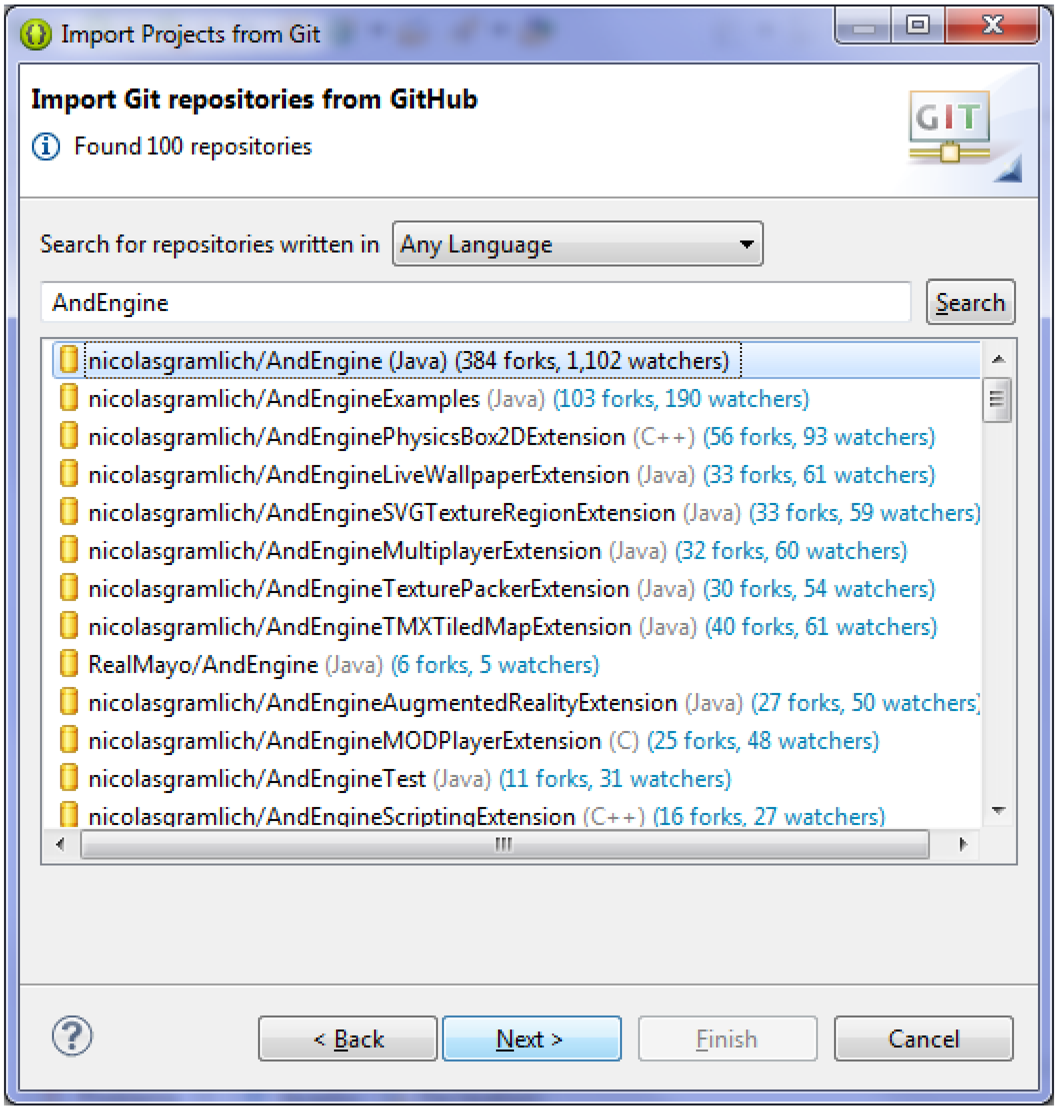
13. Select **GitHub** from the list of selections displayed then click **Next**.



14. Enter **AndEngine** in the search text field then click the **Search** button.

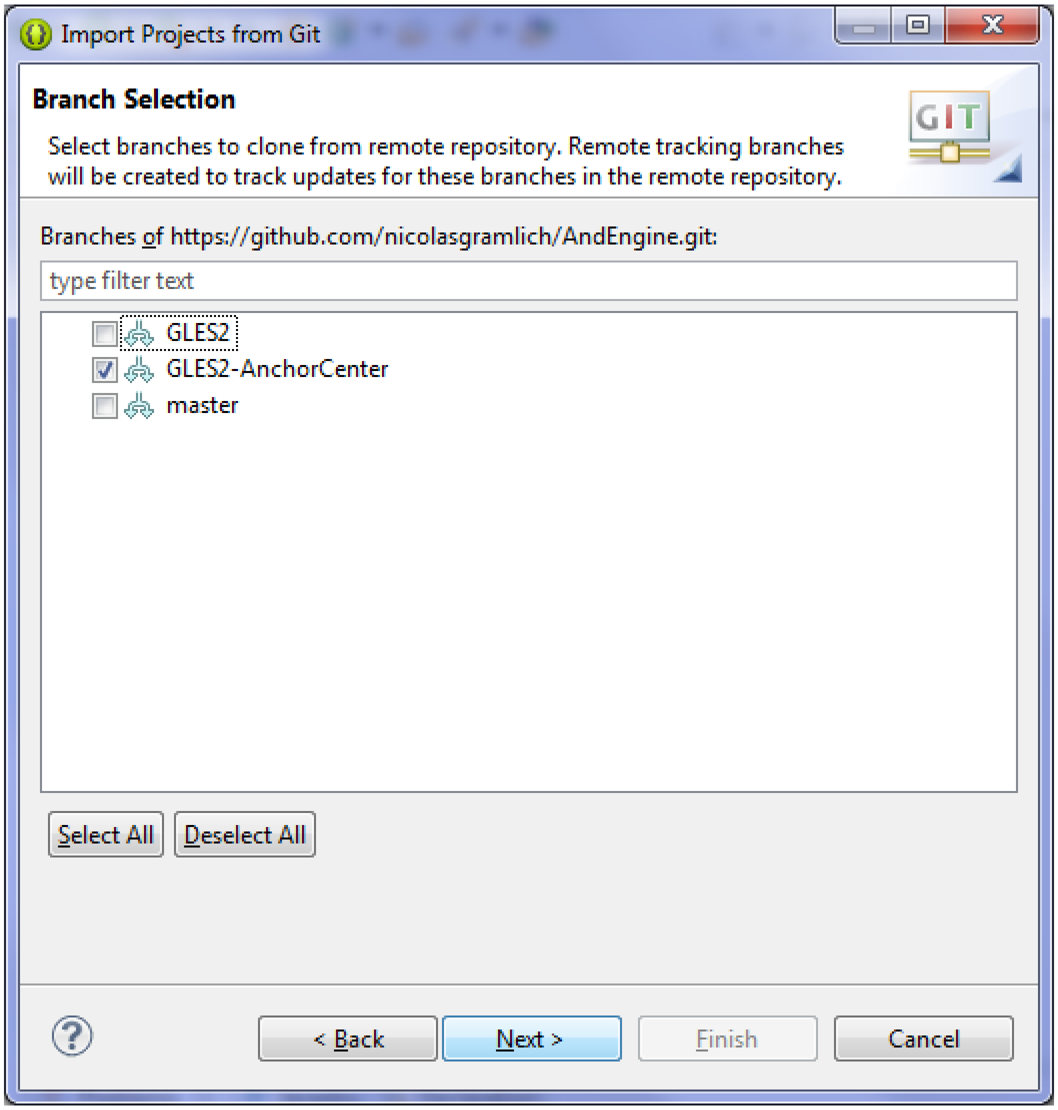

Import Projects from Git
Import Git repositories from GitHub
Search for repositories to clone
Search for repositories written in
Any Language
AndEngine
Search
< Back
Next >
Finish
Cancel

15. Once the list of libraries appears, select the main AndEngine library named **nicholasgramlich/AndEngine (Java)**. Once selected, click **Next**. Once we start working with AndEngine extensions within the book, we will need to revisit this step for finding and downloading the necessary extensions.



nicolasgramlich/AndEngine (Java) (384 forks, 1,102 watchers)
nicolasgramlich/AndEngineExamples (Java) (103 forks, 190 watchers)
nicolasgramlich/AndEnginePhysicsBox2DExtension (C++) (56 forks, 93 watchers)
nicolasgramlich/AndEngineLiveWallpaperExtension (Java) (33 forks, 61 watchers)
nicolasgramlich/AndEngineSVGTextureRegionExtension (Java) (33 forks, 59 watchers)
nicolasgramlich/AndEngineMultiplayerExtension (Java) (32 forks, 60 watchers)
nicolasgramlich/AndEngineTexturePackerExtension (Java) (30 forks, 54 watchers)
nicolasgramlich/AndEngineTMXTiledMapExtension (Java) (40 forks, 61 watchers)
RealMayo/AndEngine (Java) (6 forks, 5 watchers)
nicolasgramlich/AndEngineAugmentedRealityExtension (Java) (27 forks, 50 watchers
nicolasgramlich/AndEngineMODPlayerExtension (C) (25 forks, 48 watchers)
nicolasgramlich/AndEngineTest (Java) (11 forks, 31 watchers)
nicolasgramlich/AndEngineScriptingExtension (C++) (16 forks, 27 watchers)

16. Since we'll be working solely with the GLES2-AnchorCenter branch throughout the recipes in the book, select only the **GLES2-AnchorCenter** branch from the list.


Import Projects from Git

Branch Selection

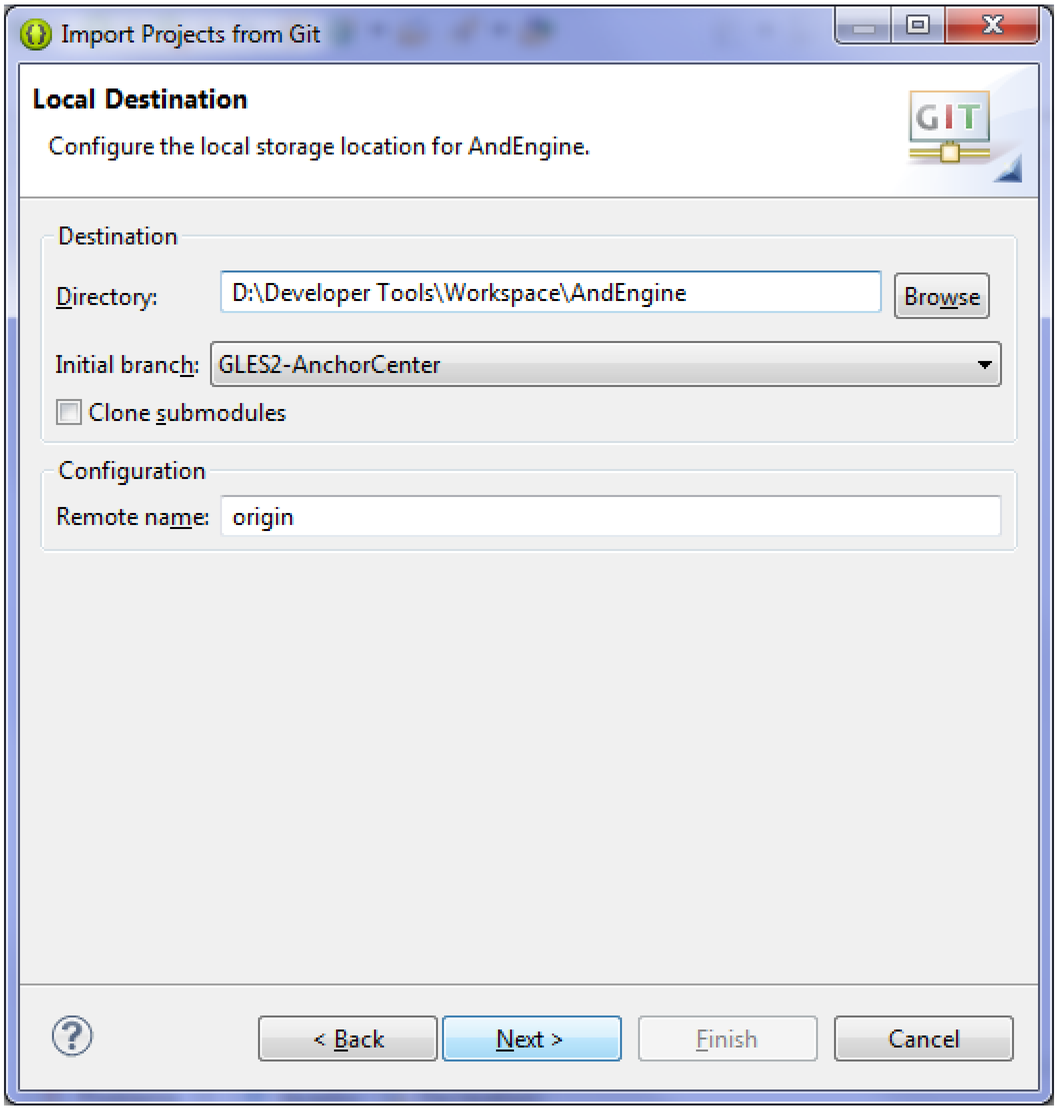
Select branches to clone from remote repository. Remote tracking branches will be created to track updates for these branches in the remote repository.

Branches of https://github.com/nicolasgramlich/AndEngine.git:

type filter text

- [ ] GLES2
- [✓] GLES2-AnchorCenter
- [ ] master

17. Make sure the directory for the library is pointing to the same drive as our workspace. This is very important as library projects tend to cause project errors if the library project is on a separate drive than the project we're working on. The safest route is to save the library projects within our eclipse workspace. Once the directory has been provided, click **Next**.


Import Projects from Git

Local Destination

Configure the local storage location for AndEngine.

Destination

Directory: D:\Developer Tools\Workspace\AndEngine

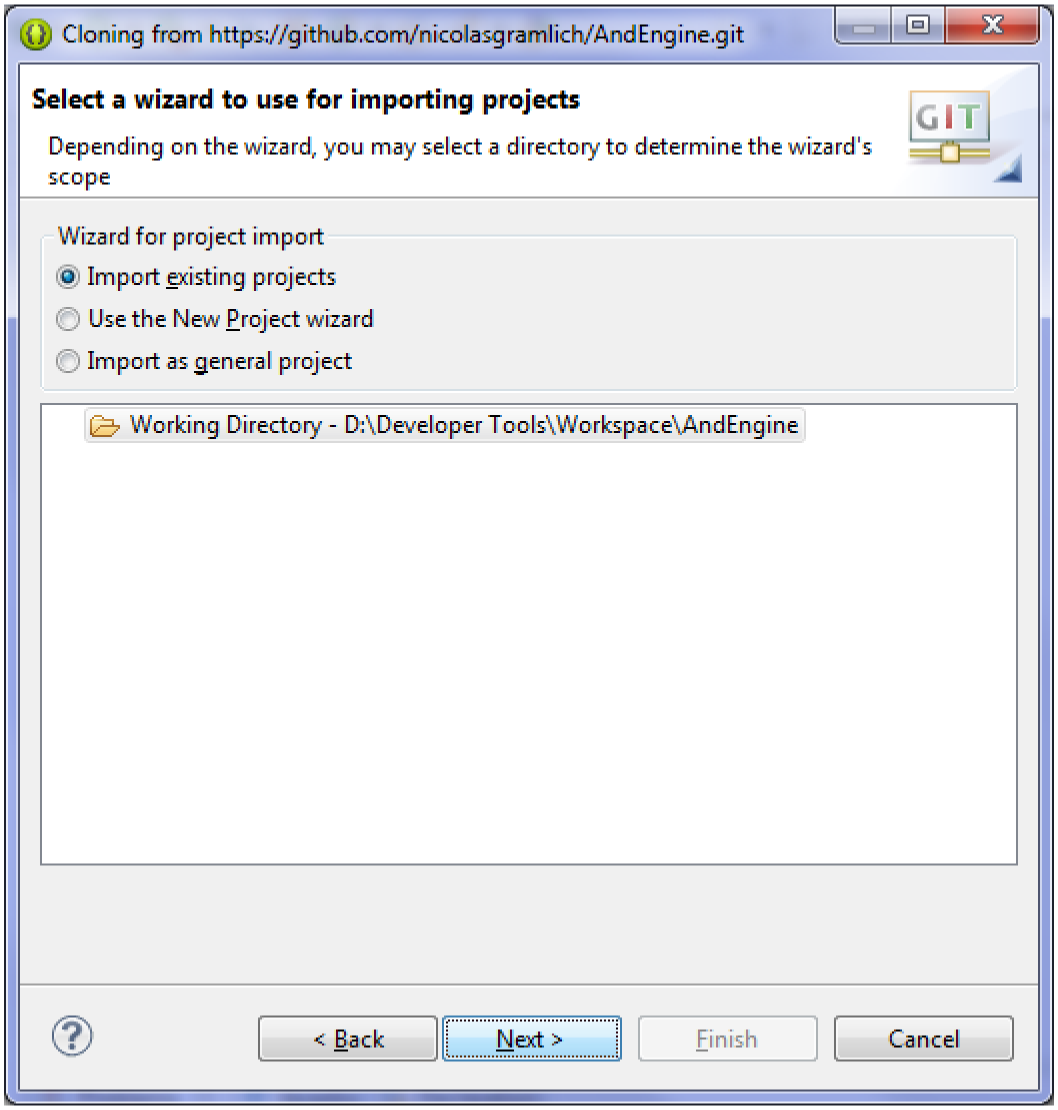
Initial branch: GLES2-AnchorCenter

☐ Clone submodules

Configuration

Remote name: origin

18. Click **Next**.


Cloning from https://github.com/nicolasgramlich/AndEngine.git

Select a wizard to use for importing projects

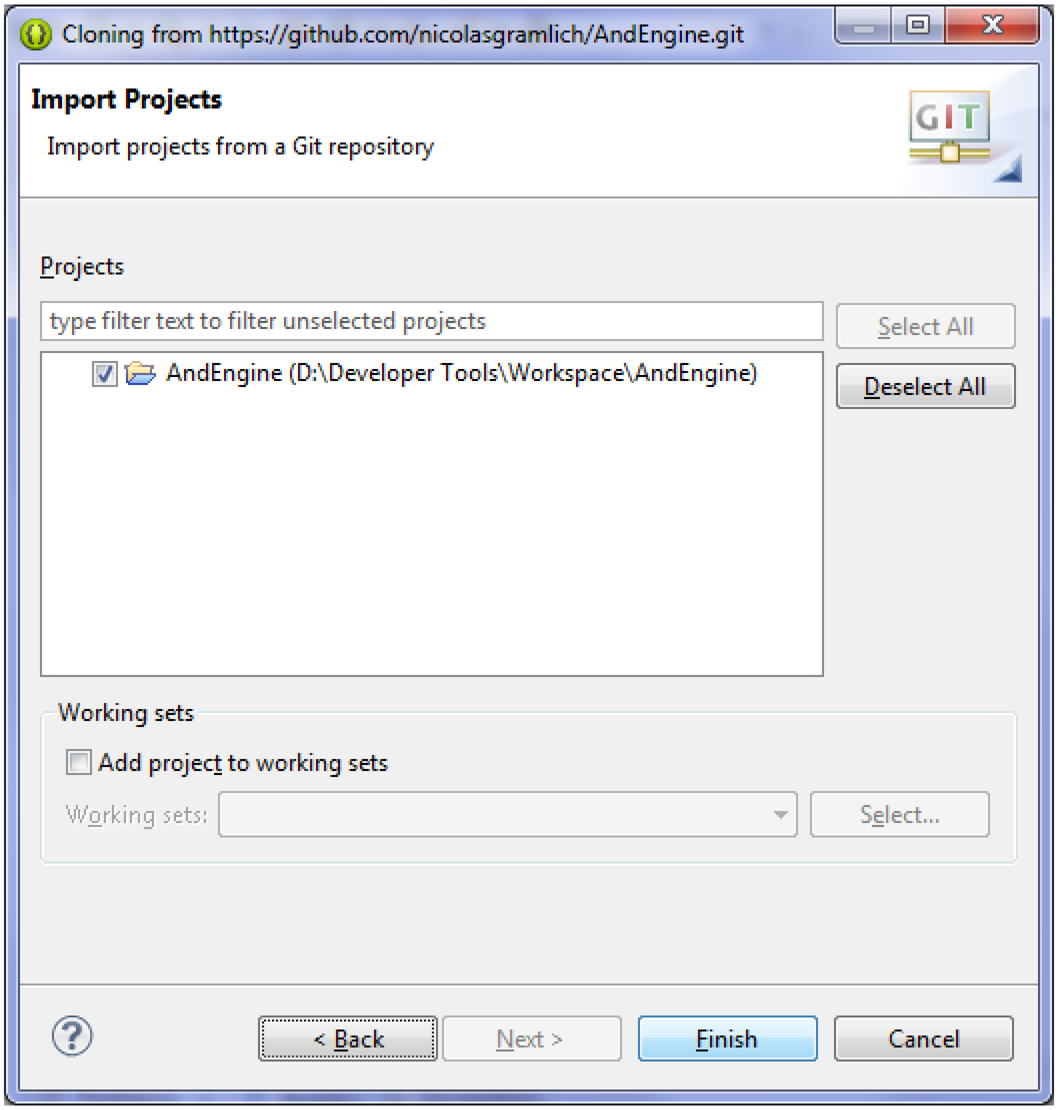
Depending on the wizard, you may select a directory to determine the wizard's scope

Wizard for project import

[● Import existing projects] ○ Use the New Project wizard ○ Import as general project

Working Directory - D:\Developer Tools\Workspace\AndEngine

19. Click **Finish**.


Cloning from https://github.com/nicolasgramlich/AndEngine.git

Import Projects

Import projects from a Git repository

Projects

type filter text to filter unselected projects

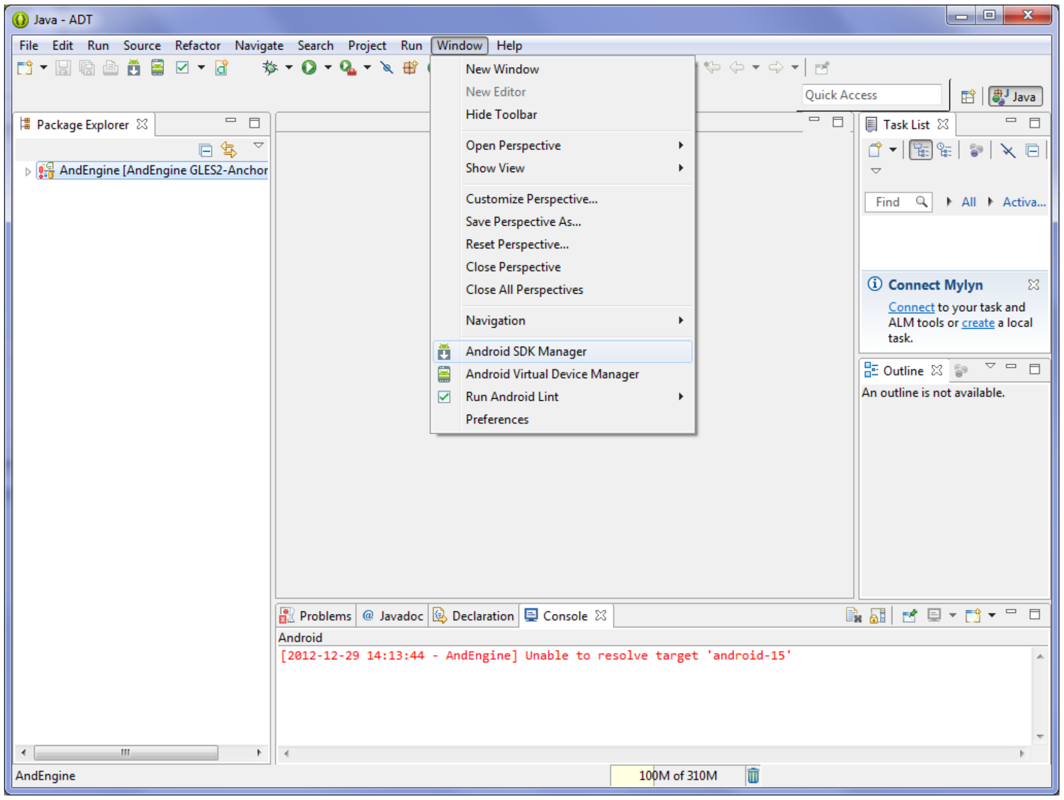
☑ AndEngine (D:\Developer Tools\Workspace\AndEngine)

Working sets

☐ Add project to working sets

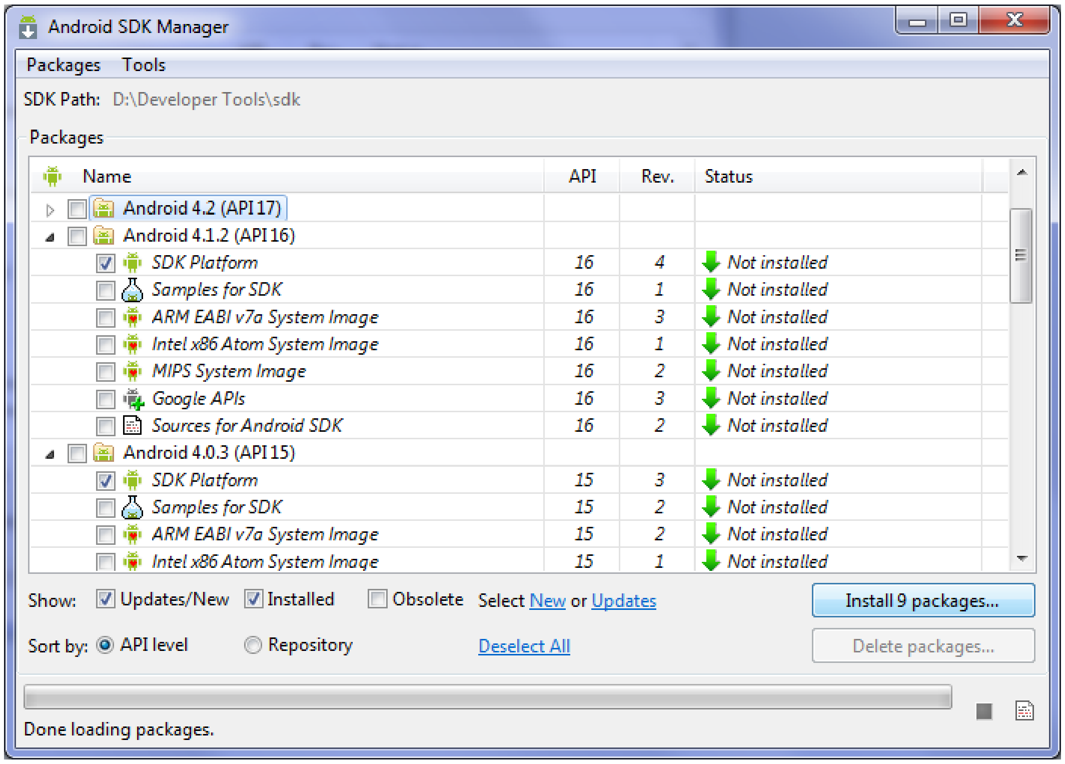
Working sets:

20. The AndEngine library will now be visible within our workspace, however we'll need to obtain the necessary Android platform SDK's before we can satisfy the library. Select **Window** from the Eclipse toolbar then navigate to the **Android SDK Manager** selection.



Android
```
[2012-12-29 14:13:44 - AndEngine] Unable to resolve target 'android-15'
```

21. To be safe, make sure to put a check in the **SDK Platform** selection checkboxes from API 8 all the way up to API 16 (API 17's SDK Platform comes with the ADT bundle). This is to ensure that our games/recipes will be available on all devices ranging from API 8 through to API 17 for max compatibility. Once the boxes have been selected, click **Install x packages...** where x is the number of packages to download and install.



Android SDK Manager

Packages Tools

SDK Path: D:\Developer Tools\sdk

Packages

Name	API	Rev.	Status
☐ Android 4.2 (API 17)			
☐ Android 4.1.2 (API 16)			
☑ *SDK Platform*	*16*	*4*	*Not installed*
☐ *Samples for SDK*	*16*	*1*	*Not installed*
☐ *ARM EABI v7a System Image*	*16*	*3*	*Not installed*
☐ *Intel x86 Atom System Image*	*16*	*1*	*Not installed*
☐ *MIPS System Image*	*16*	*2*	*Not installed*
☐ *Google APIs*	*16*	*3*	*Not installed*
☐ *Sources for Android SDK*	*16*	*2*	*Not installed*
☐ Android 4.0.3 (API 15)			
☑ *SDK Platform*	*15*	*3*	*Not installed*
☐ *Samples for SDK*	*15*	*2*	*Not installed*
☐ *ARM EABI v7a System Image*	*15*	*2*	*Not installed*
☐ *Intel x86 Atom System Image*	*15*	*1*	*Not installed*

Show: ☑ Updates/New ☑ Installed ☐ Obsolete Select New or Updates

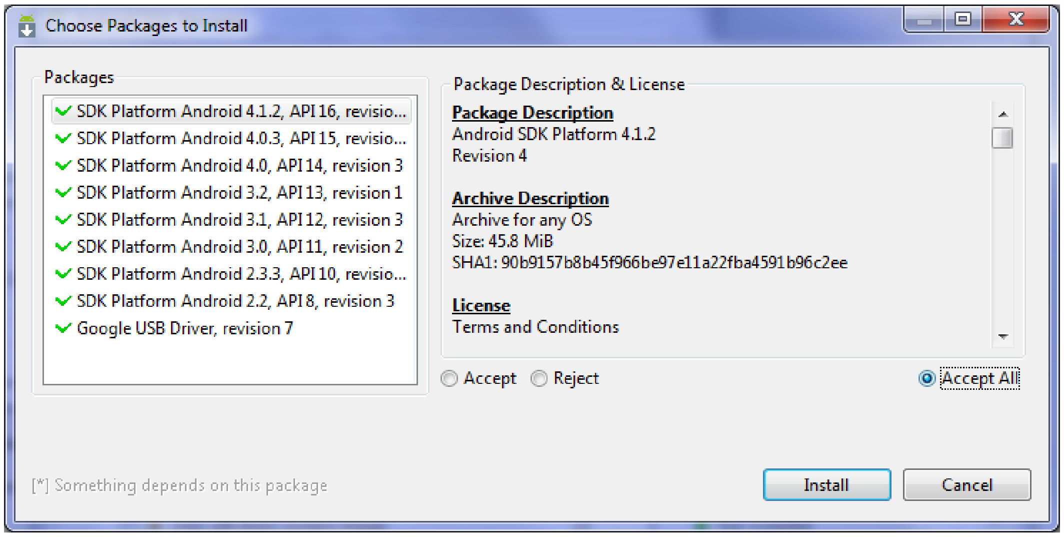
Sort by: ● API level ○ Repository Deselect All

Install 9 packages...

Delete packages...

Done loading packages.

22. Make sure to read through the terms and conditions of the various packages, then click **Accept** for each or **Accept All** if you agree to all of the licenses. Next, click **Install**. We must wait for the packages to download and install—meanwhile, close the Eclipse IDE as it will require a restart once all of the packages are installed. Once the package installations have completed, reopen the Eclipse IDE.



✓ SDK Platform Android 4.1.2, API 16, revisio...
✓ SDK Platform Android 4.0.3, API 15, revisio...
✓ SDK Platform Android 4.0, API 14, revision 3
✓ SDK Platform Android 3.2, API 13, revision 1
✓ SDK Platform Android 3.1, API 12, revision 3
✓ SDK Platform Android 3.0, API 11, revision 2
✓ SDK Platform Android 2.3.3, API 10, revisio...
✓ SDK Platform Android 2.2, API 8, revision 3
✓ Google USB Driver, revision 7

Package Description
Android SDK Platform 4.1.2
Revision 4

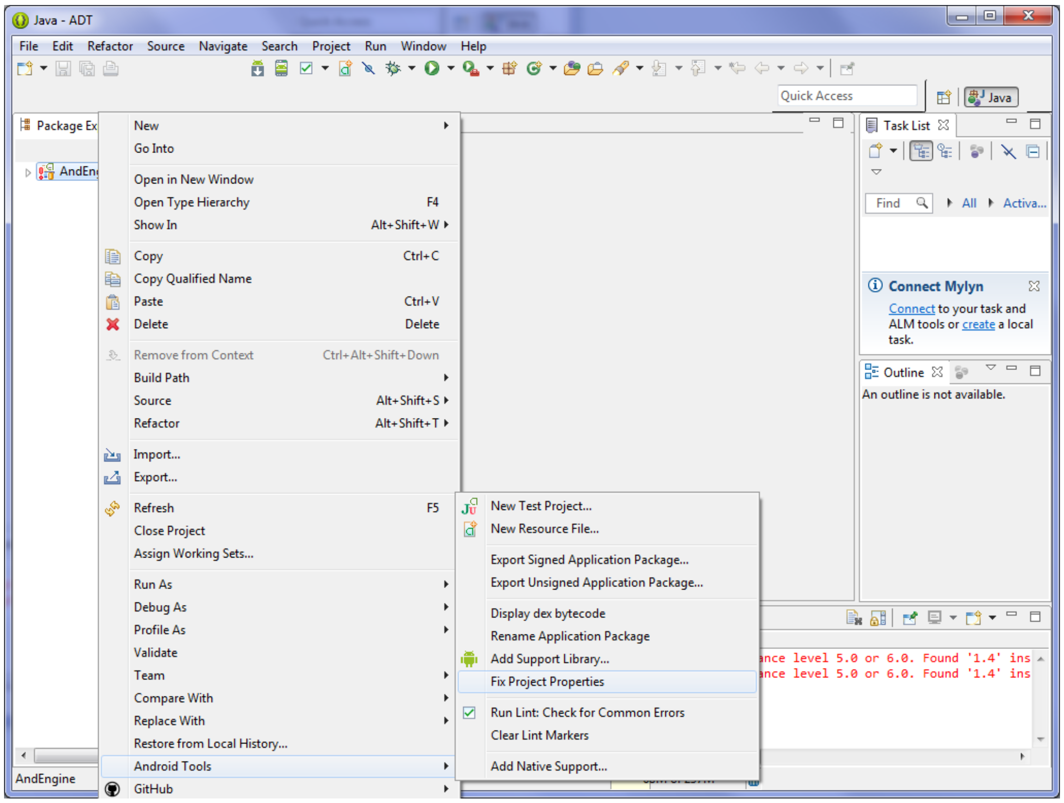
Archive Description
Archive for any OS
Size: 45.8 MiB
SHA1: 90b9157b8b45f966be97e11a22fba4591b96c2ee

License
Terms and Conditions

Accept Reject [✓ Accept All]

[*] Something depends on this package

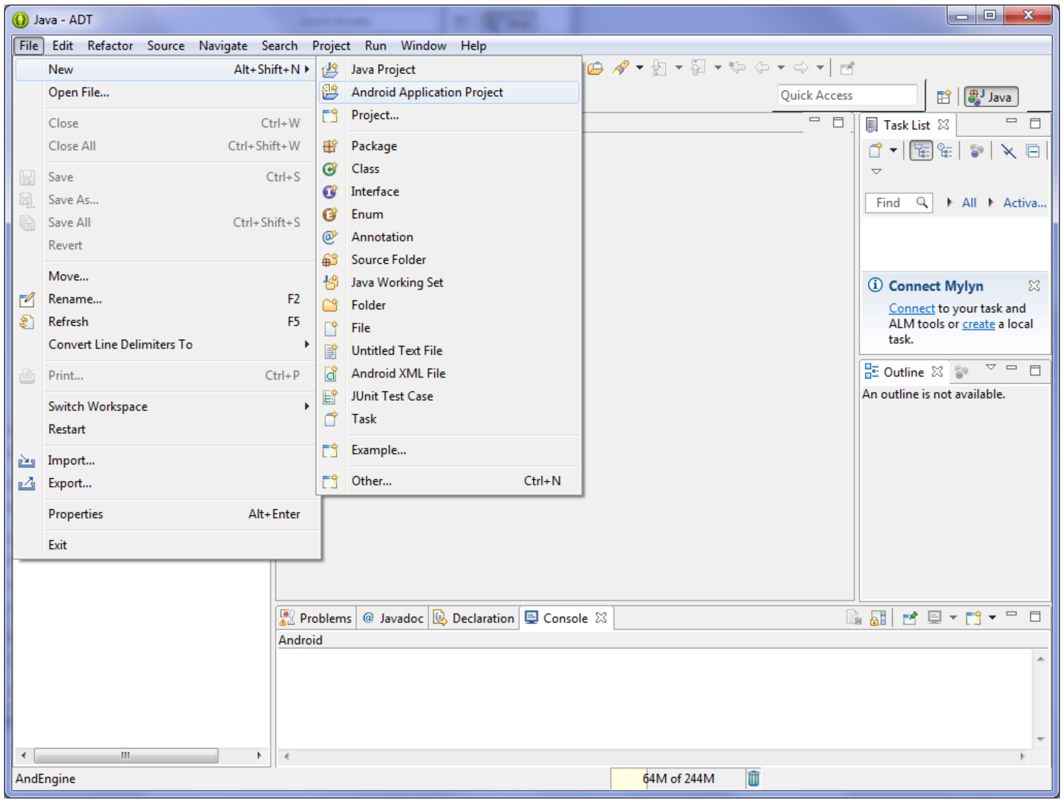
23. The last step we must take before the AndEngine library is ready for use is to fix the project properties. Right-click on the AndEngine project in the Package Explorer and navigate to the **Android Tools** selection. From here, select **Fix Project Properties**.



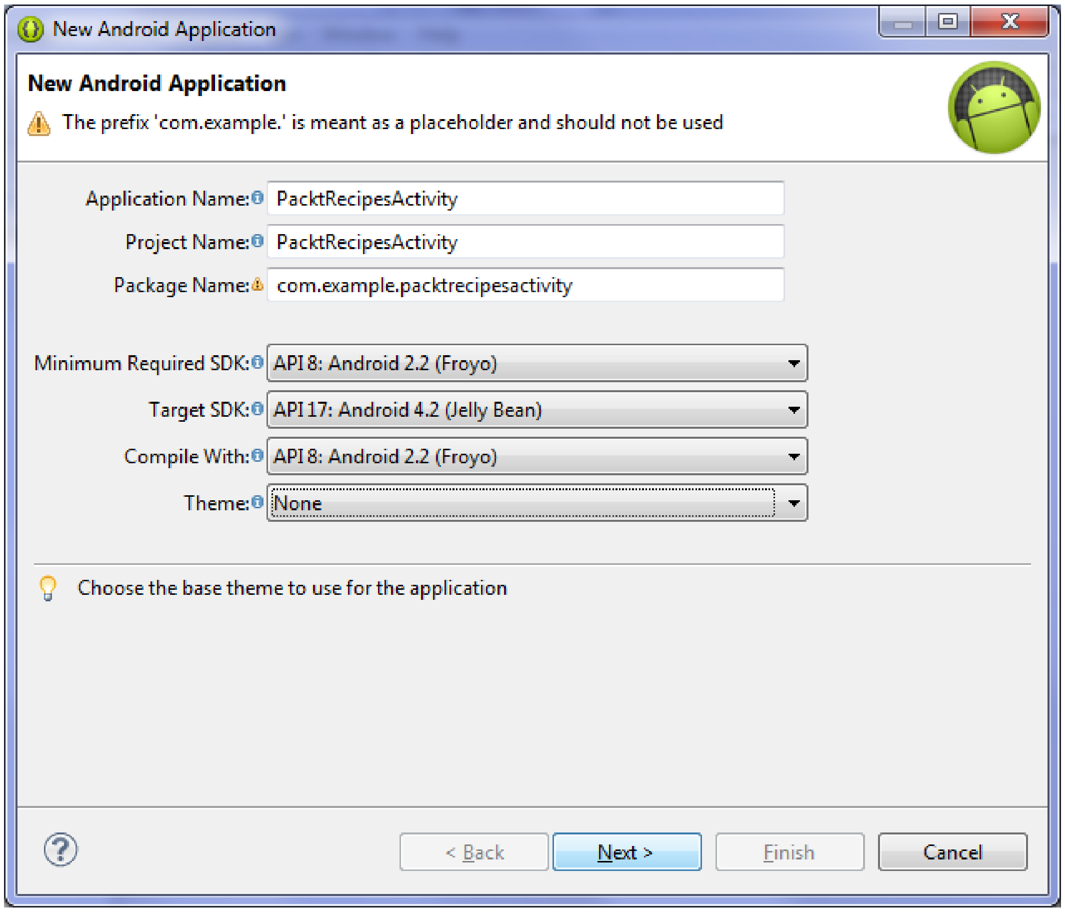
```
ance level 5.0 or 6.0. Found '1.4' ins
ance level 5.0 or 6.0. Found '1.4' ins
```

Steps involved for setting up a base AndEngine project:

1. Select **File** from the Eclipse toolbar and click **Android Application Project** to begin the Android project wizard.



2. Enter the recipe name in the **Application Name** text field. The following text fields will fill themselves in automatically. For maximum compatibility, select the API 8 in the **Minimum Required SDK** drop-down list, API 17 in the **Target SDK** drop-down list. It is wise to set the **Compile With** selection to that of the minimum required SDK selection as to disallow compilation when a class or method is used that is not available to lesser API levels. Once the fields have been filled out, click **Next**.



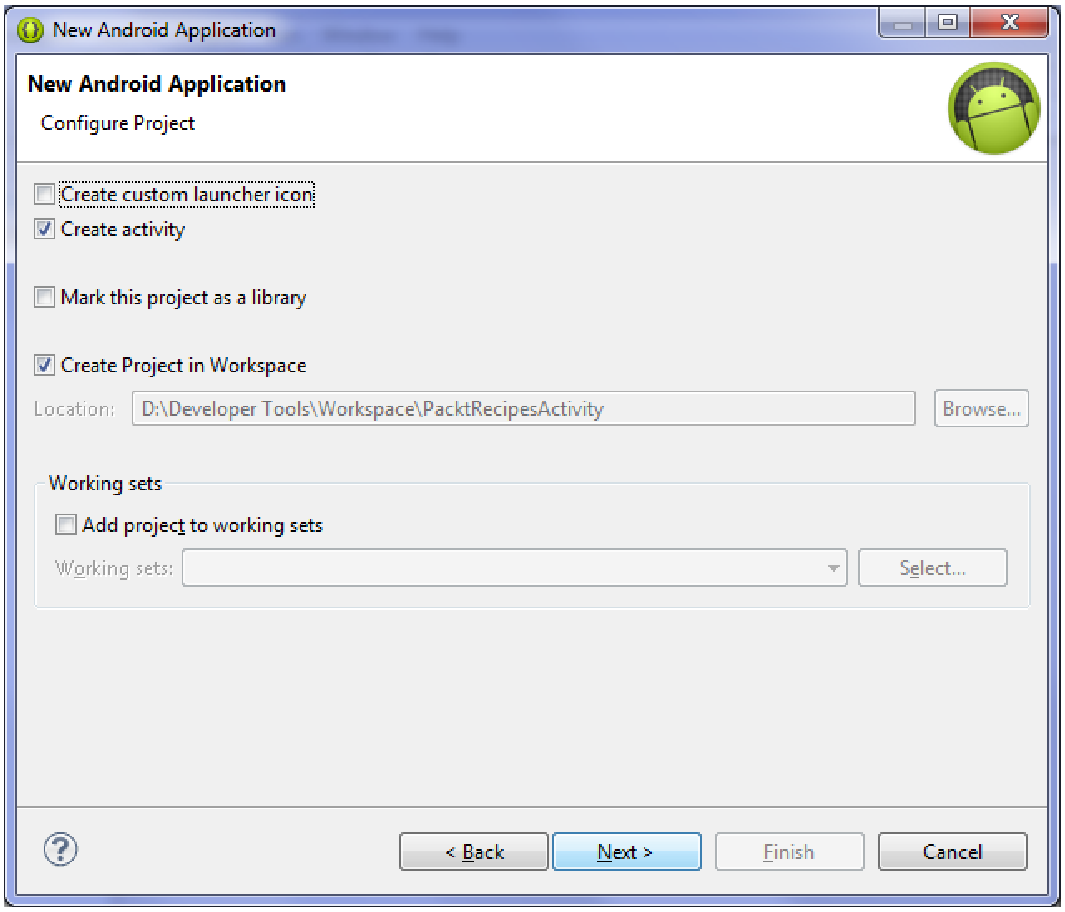
New Android Application

The prefix 'com.example.' is meant as a placeholder and should not be used

Application Name:	PacktRecipesActivity
Project Name:	PacktRecipesActivity
Package Name:	com.example.packtrecipesactivity
Minimum Required SDK:	API 8: Android 2.2 (Froyo)
Target SDK:	API 17: Android 4.2 (Jelly Bean)
Compile With:	API 8: Android 2.2 (Froyo)
Theme:	None

Choose the base theme to use for the application

3. Remove the check from the **Create custom launcher icon** checkbox as they are not necessary for recipes. Click **Next**.



New Android Application

New Android Application

Configure Project

- [ ] Create custom launcher icon
- [✓] Create activity
- [ ] Mark this project as a library
- [✓] Create Project in Workspace

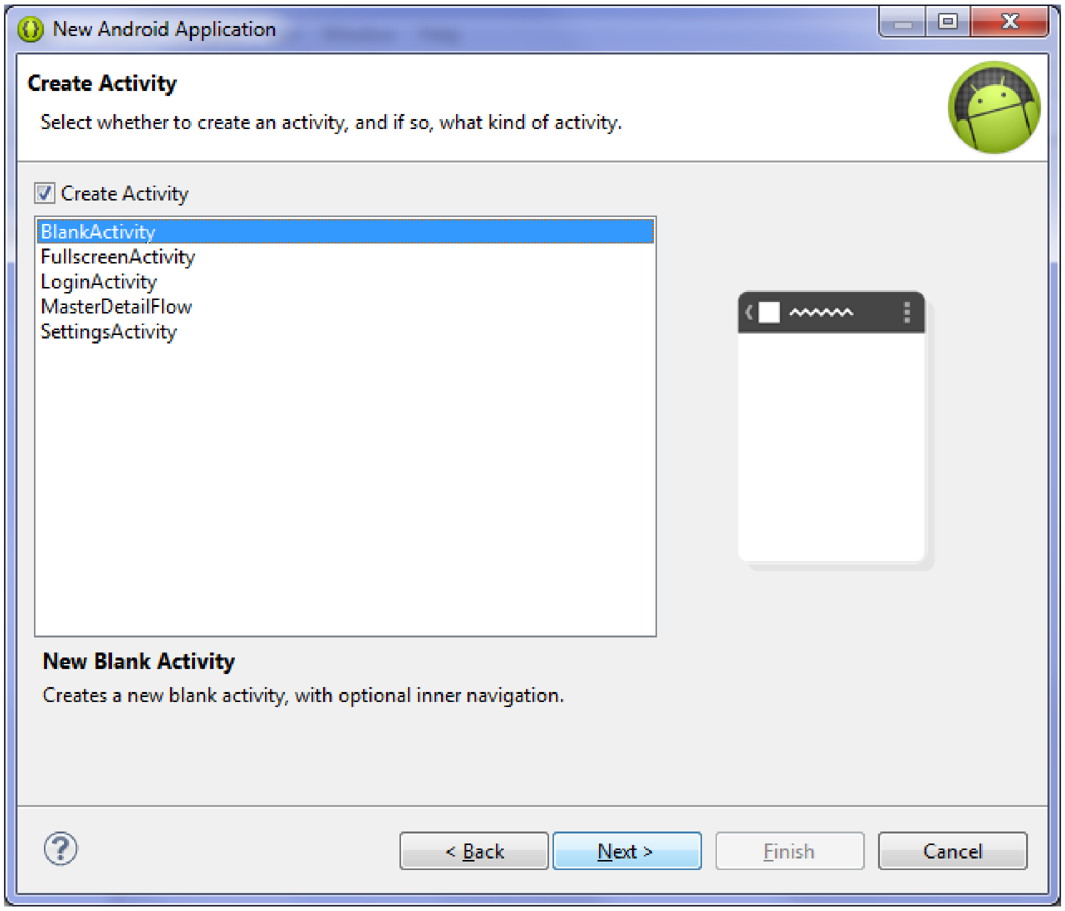
Location: D:\Developer Tools\Workspace\PacktRecipesActivity

Working sets

- [ ] Add project to working sets

Working sets:

4. Click **Next**.



New Android Application

Create Activity

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

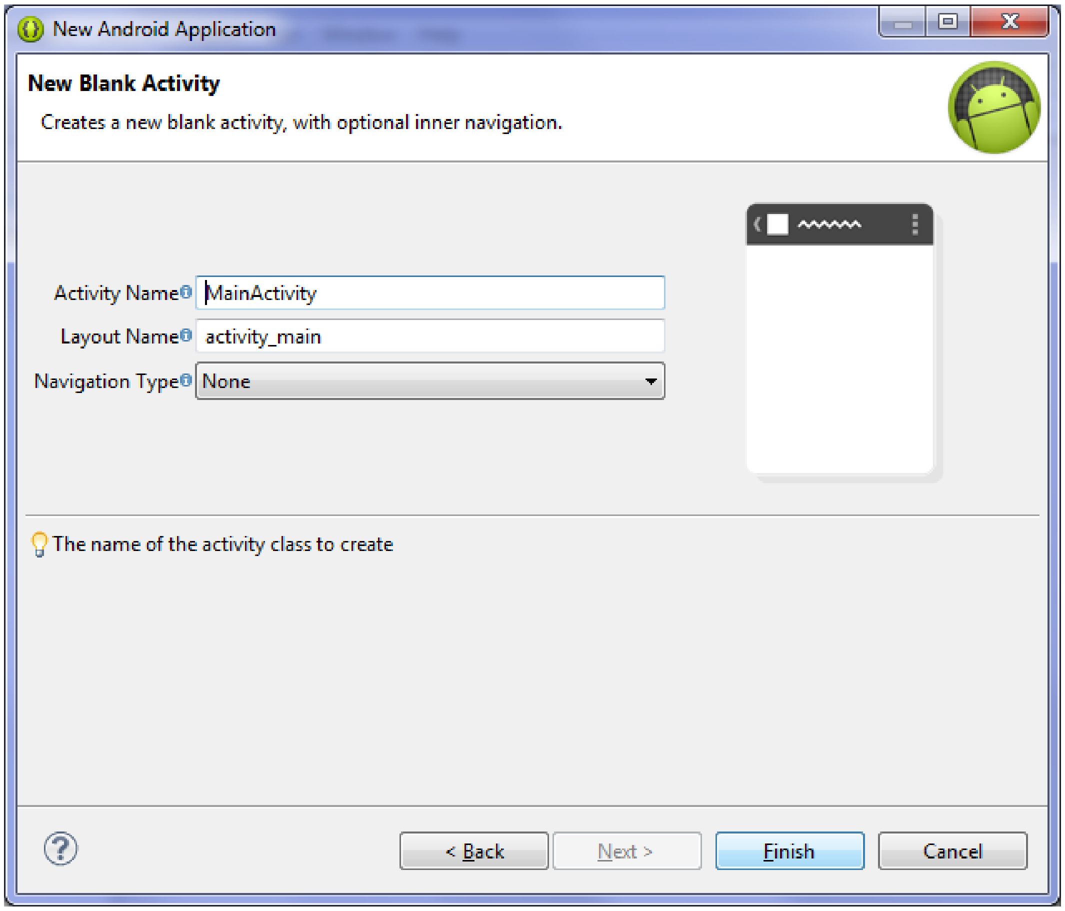
☑ Create Activity

- BlankActivity
- FullscreenActivity
- LoginActivity
- MasterDetailFlow
- SettingsActivity

New Blank Activity

Creates a new blank activity, with optional inner navigation.

5. Click **Finish**.



New Android Application

New Blank Activity

Creates a new blank activity, with optional inner navigation.

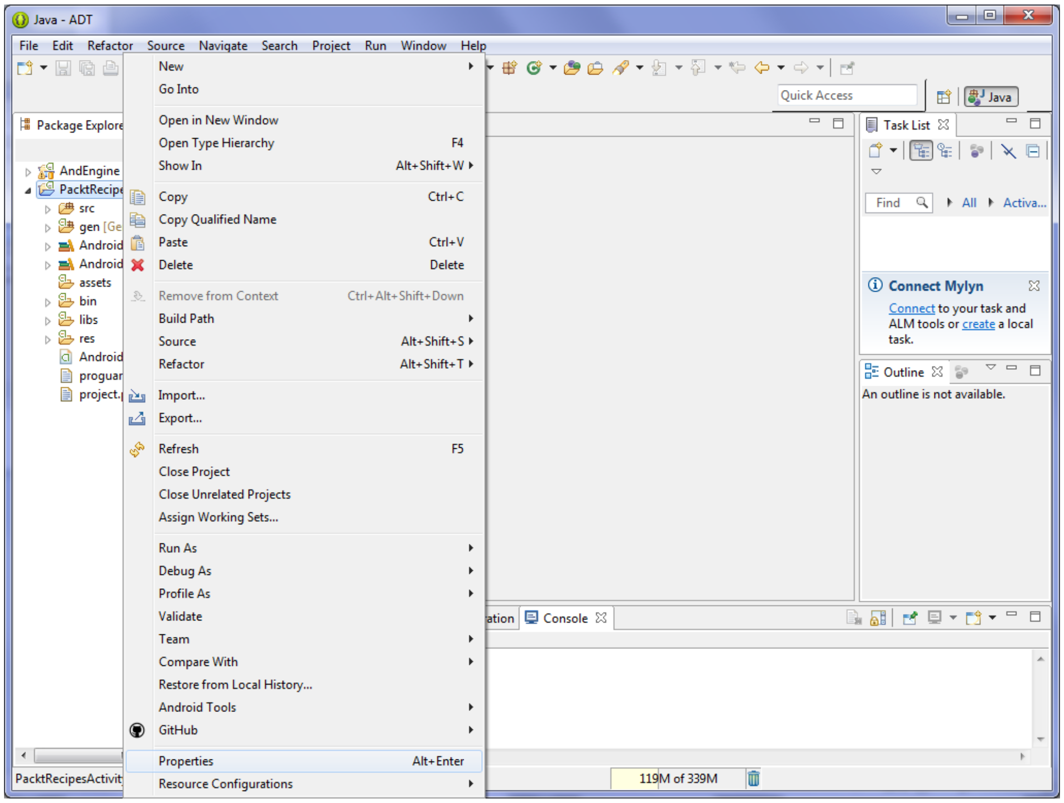
Activity Name: MainActivity

Layout Name: activity_main

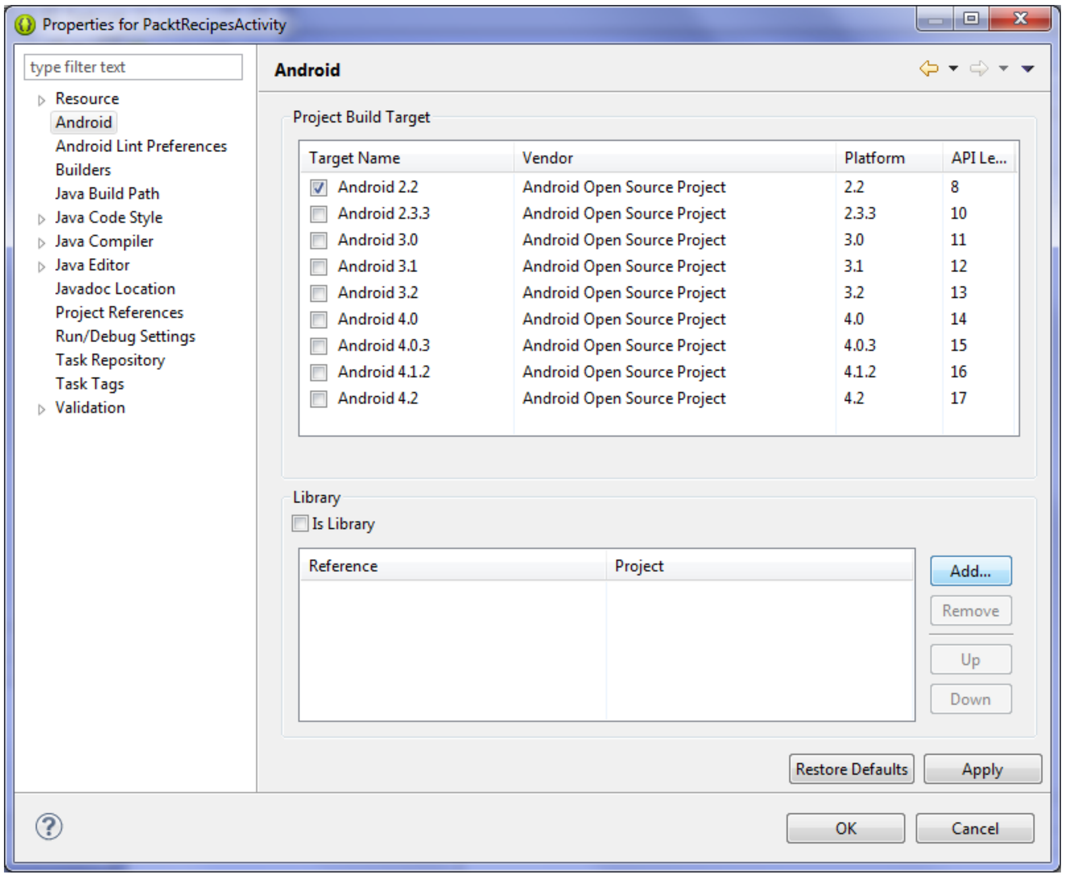
Navigation Type: None

The name of the activity class to create

6. We will now add the AndEngine library to our new project. Right-click on the project you wish to add the AndEngine library to and select **Properties**.



7. In the left-hand side of the properties window, select the **Android** group. Next, click the **Add...** button in the **Library** section.

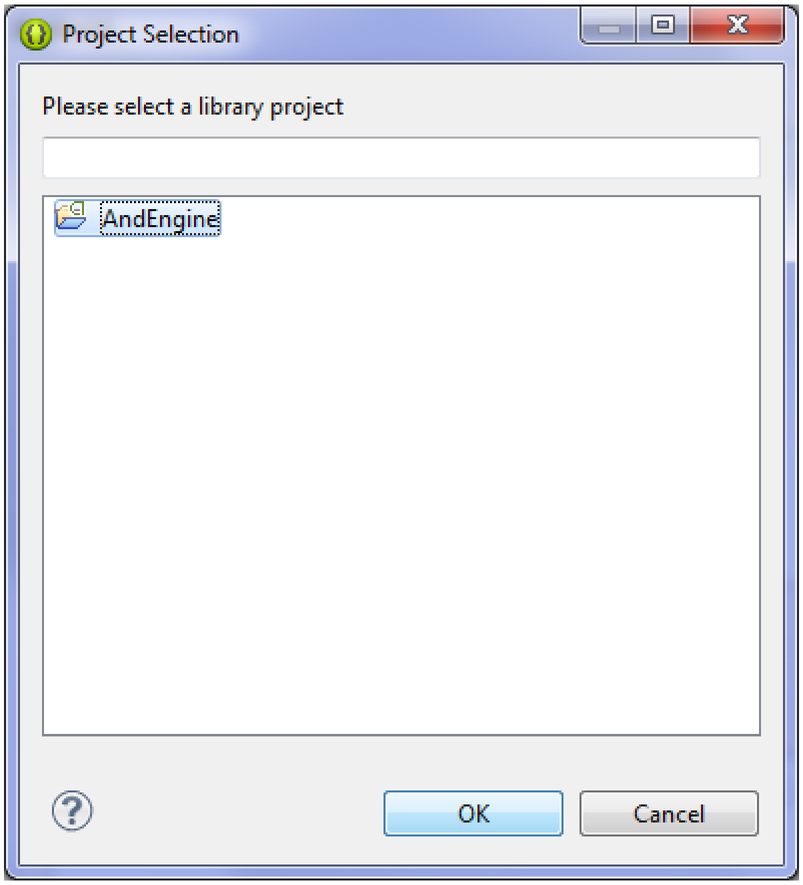


Target Name	Vendor	Platform	API Le...
✓ Android 2.2	Android Open Source Project	2.2	8
Android 2.3.3	Android Open Source Project	2.3.3	10
Android 3.0	Android Open Source Project	3.0	11
Android 3.1	Android Open Source Project	3.1	12
Android 3.2	Android Open Source Project	3.2	13
Android 4.0	Android Open Source Project	4.0	14
Android 4.0.3	Android Open Source Project	4.0.3	15
Android 4.1.2	Android Open Source Project	4.1.2	16
Android 4.2	Android Open Source Project	4.2	17

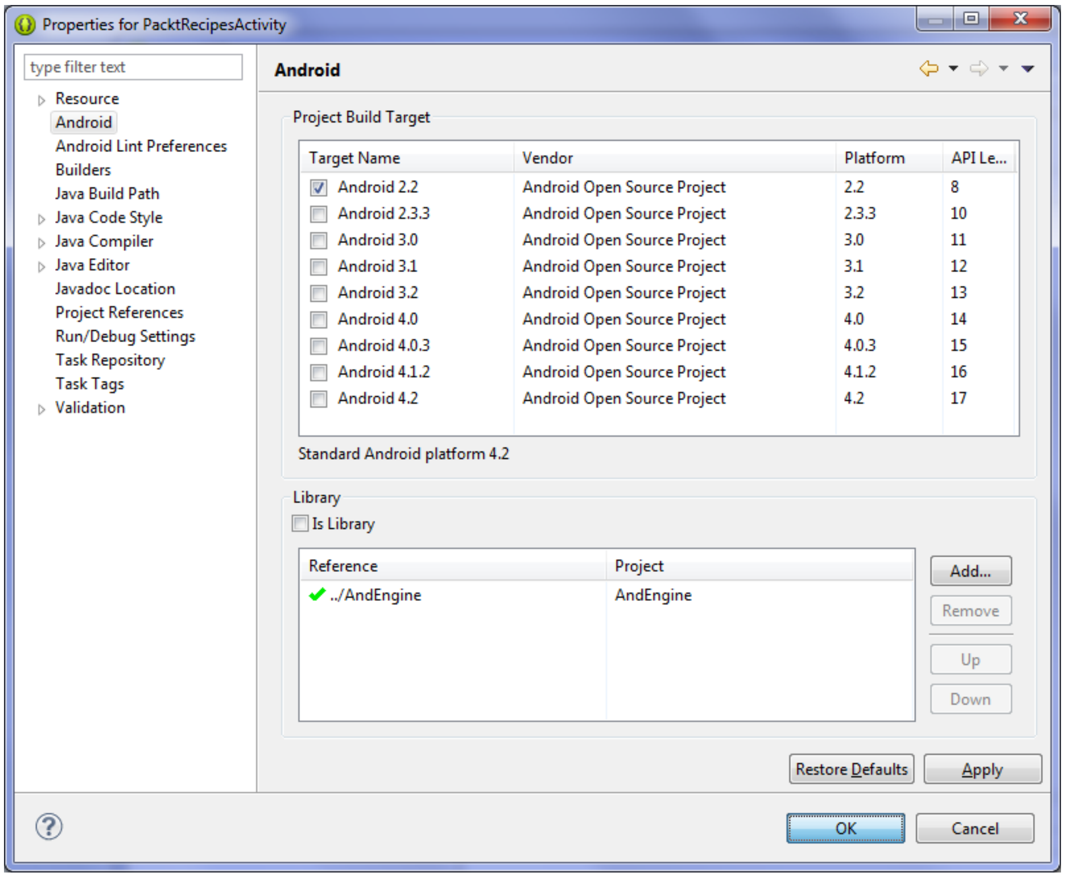
☐ Is Library

Reference	Project

8. Select **AndEngine** from the project selection window that appears, then click **OK**.



9. We should now see a reference to the AndEngine library in our project's properties. Click **OK** to save and dismiss the properties.



Project Build Target

Target Name	Vendor	Platform	API Le...
✓ Android 2.2	Android Open Source Project	2.2	8
Android 2.3.3	Android Open Source Project	2.3.3	10
Android 3.0	Android Open Source Project	3.0	11
Android 3.1	Android Open Source Project	3.1	12
Android 3.2	Android Open Source Project	3.2	13
Android 4.0	Android Open Source Project	4.0	14
Android 4.0.3	Android Open Source Project	4.0.3	15
Android 4.1.2	Android Open Source Project	4.1.2	16
Android 4.2	Android Open Source Project	4.2	17

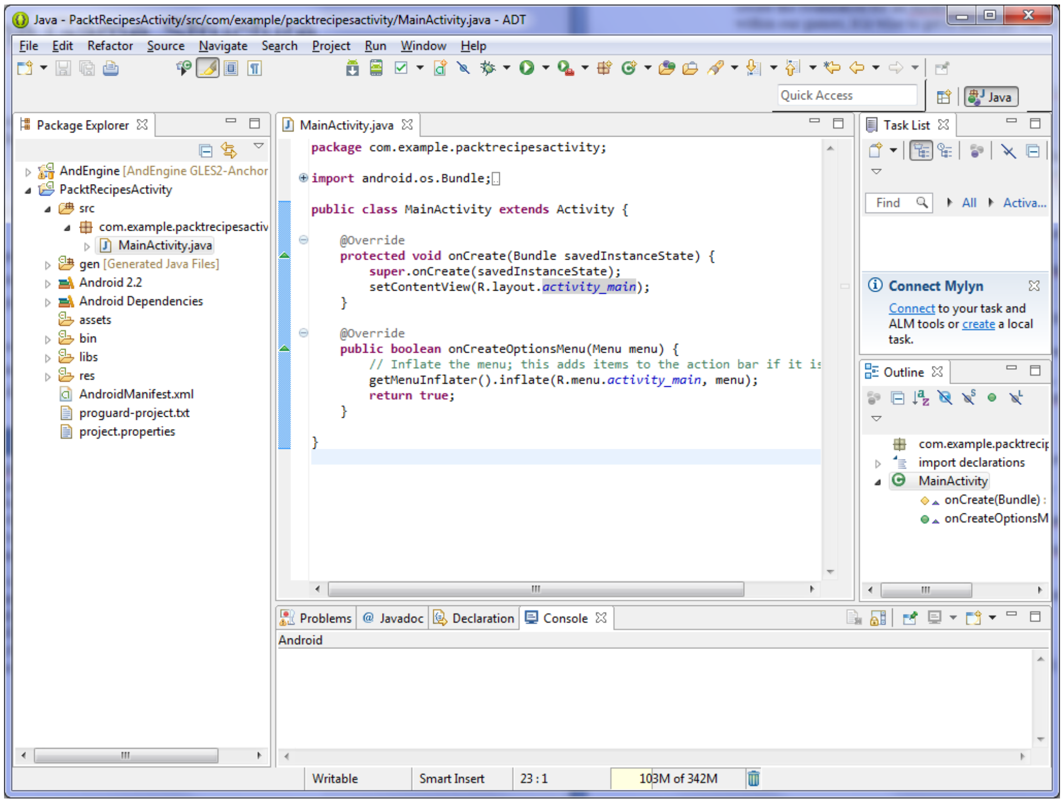
Standard Android platform 4.2

Library

Is Library

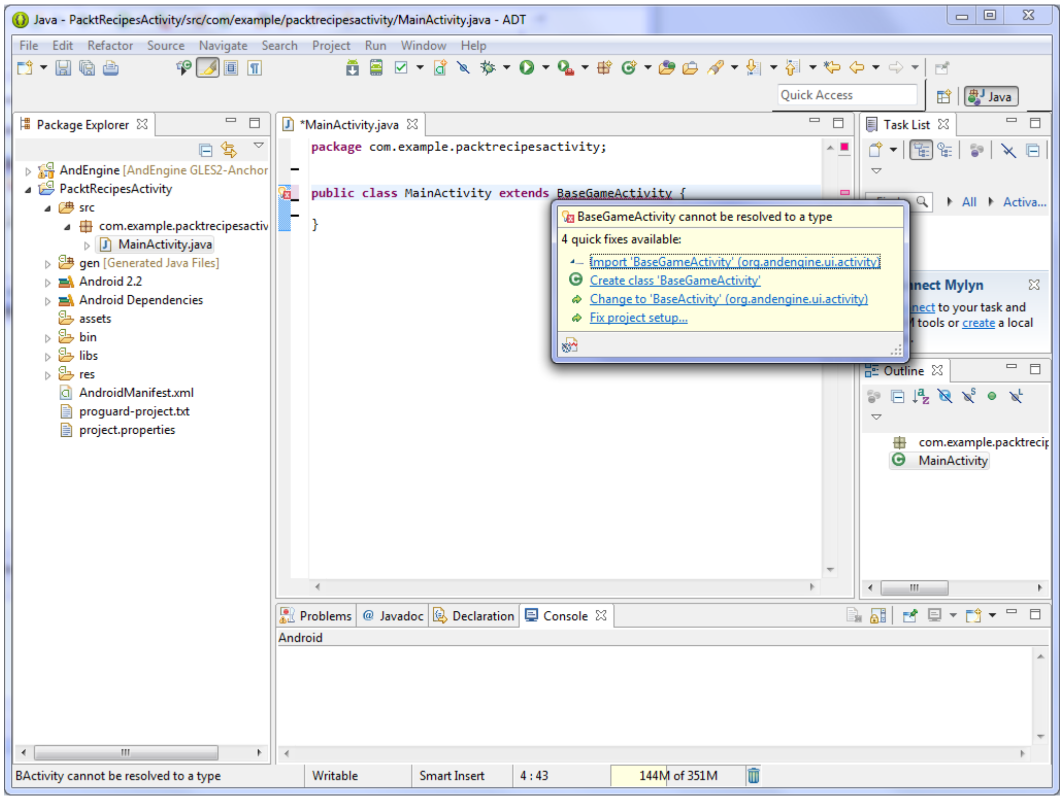
Reference	Project
✓ ../AndEngine	AndEngine

10. Open up the project's activity class within the project's **src** folder. This will be the only file within the project's **src** folder for now.



```
package com.example.packtrecipesactivity;

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 is
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }

}
```

11. Remove all code between the first and last brackets of the activity class. Once the specified code has been removed, instead of extending the `Activity` class, we will extend AndEngine's `BaseGameActivity` class. Next, we can either mouse-over the **BaseGameActivity** text within the code to import the class to our project, or simply press the *Ctrl* + *Shift* + *o* hotkey combination.

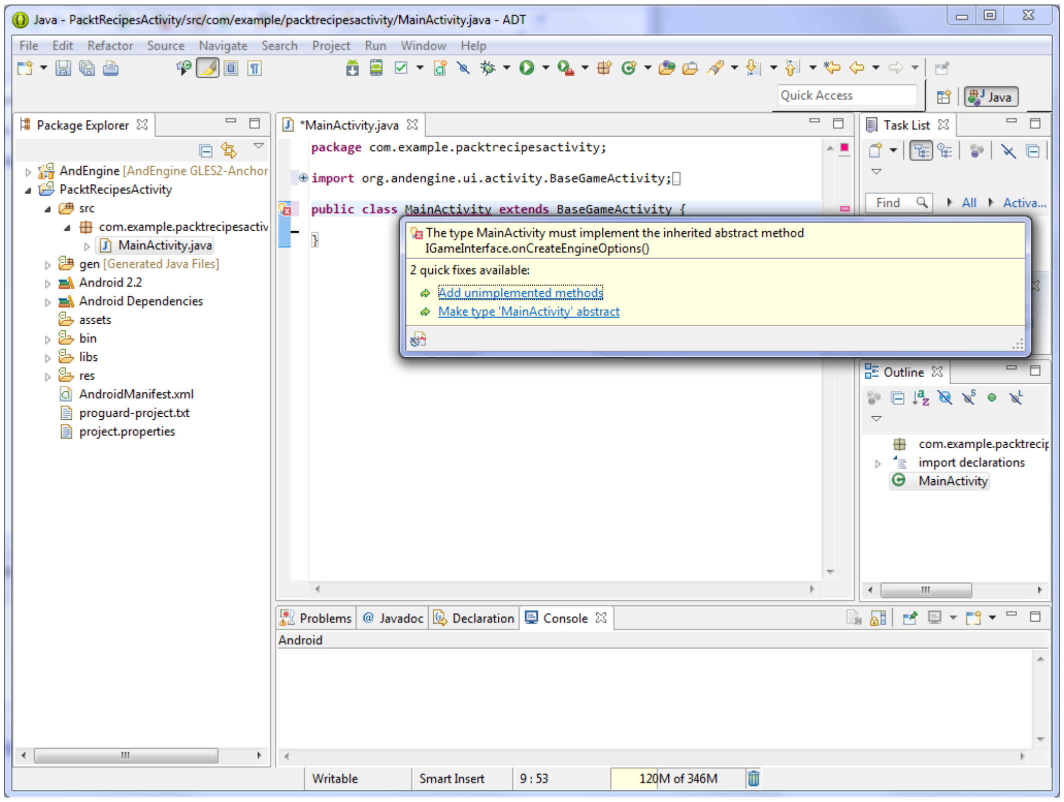


```
package com.example.packtrecipesactivity;

public class MainActivity extends BaseGameActivity {

}
```

BaseGameActivity cannot be resolved to a type
4 quick fixes available:
- Import 'BaseGameActivity' (org.andengine.ui.activity)
- Create class 'BaseGameActivity'
- Change to 'BaseActivity' (org.andengine.ui.activity)
- Fix project setup...

BActivity cannot be resolved to a type | Writable | Smart Insert | 4 : 43 | 144M of 351M

12. Once the `BaseGameAcivity` class has been imported to our project, we must add its necessary interface methods. Mouse-over the **MainActivity** (depending on your naming convention) text within the code and select **Add unimplemented methods** from the available fixes list that appears.



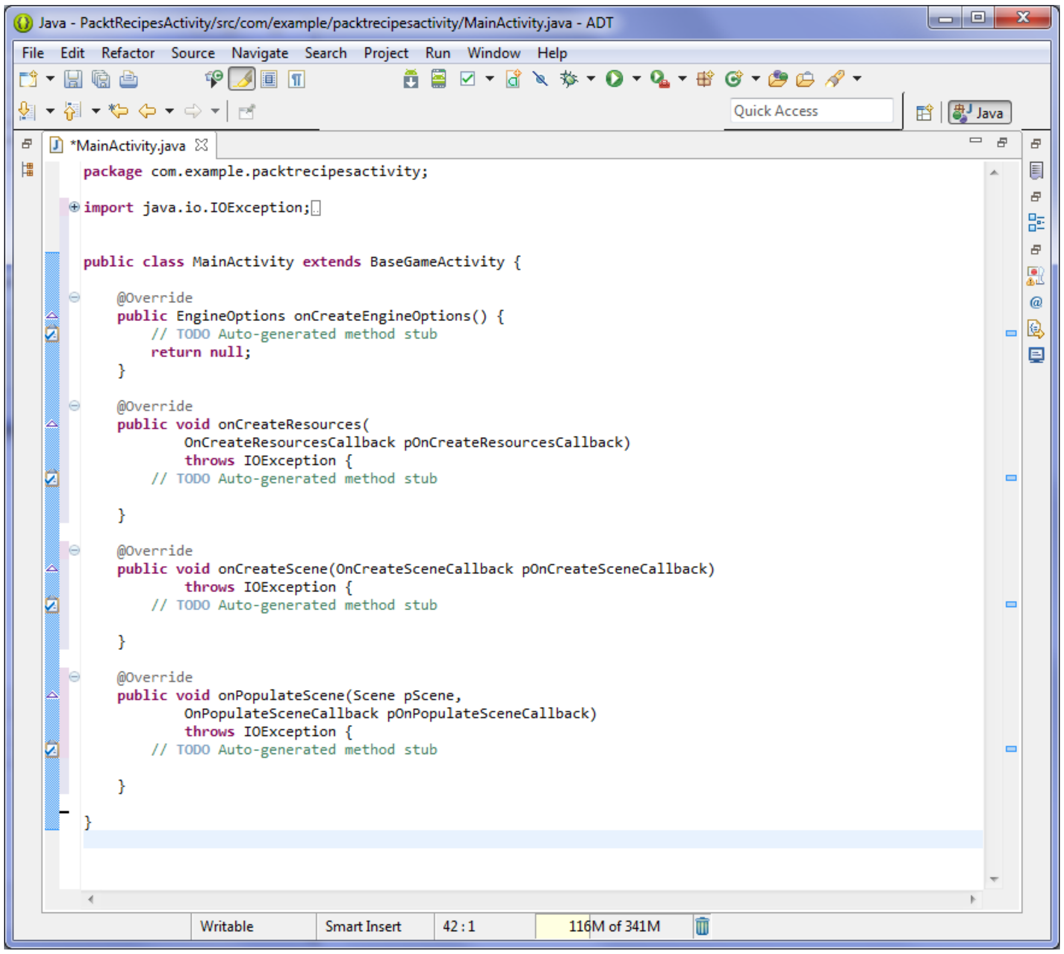
```
package com.example.packtrecipesactivity;

import org.andengine.ui.activity.BaseGameActivity;

public class MainActivity extends BaseGameActivity {

}
```

The type MainActivity must implement the inherited abstract method IGameInterface.onCreateEngineOptions()

2 quick fixes available:

- Add unimplemented methods
- Make type 'MainActivity' abstract

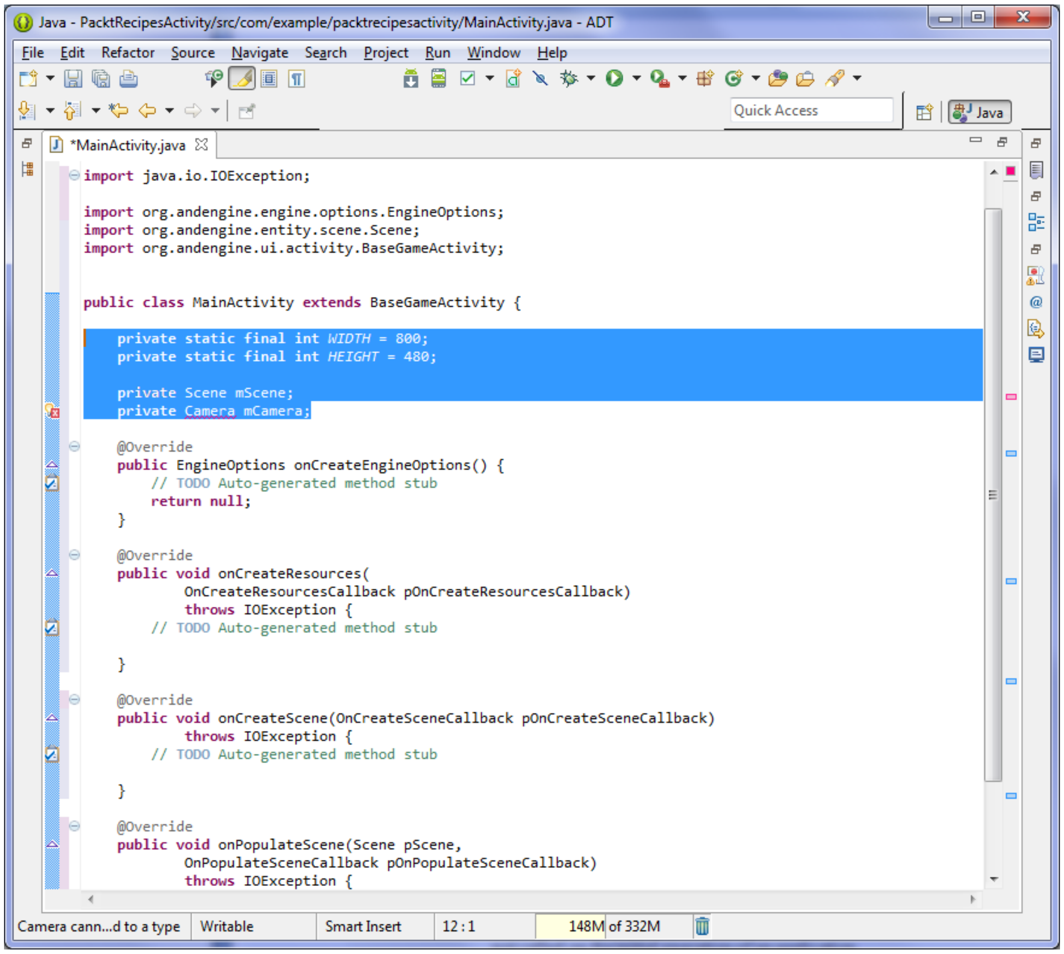
13. (Screenshot provided but no action necessary by the reader). Feel free to remove.


Java - PacktRecipesActivity/src/com/example/packtrecipesactivity/MainActivity.java - ADT
File Edit Refactor Source Navigate Search Project Run Window Help
Quick Access
Java
*MainActivity.java
Writable
Smart Insert
42 : 1
116M of 341M

```
package com.example.packtrecipesactivity;

import java.io.IOException;

public class MainActivity extends BaseGameActivity {

    @Override
    public EngineOptions onCreateEngineOptions() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public void onCreateResources(
            OnCreateResourcesCallback pOnCreateResourcesCallback)
            throws IOException {
        // TODO Auto-generated method stub

    }

    @Override
    public void onCreateScene(OnCreateSceneCallback pOnCreateSceneCallback)
            throws IOException {
        // TODO Auto-generated method stub

    }

    @Override
    public void onPopulateScene(Scene pScene,
            OnPopulateSceneCallback pOnPopulateSceneCallback)
            throws IOException {
        // TODO Auto-generated method stub

    }

}
```

Steps involved for executing the first recipe in *Chapter 1*, *AndEngine Game Structure*:

1. Introduce the main fields needed for any AndEngine project to the activity class.



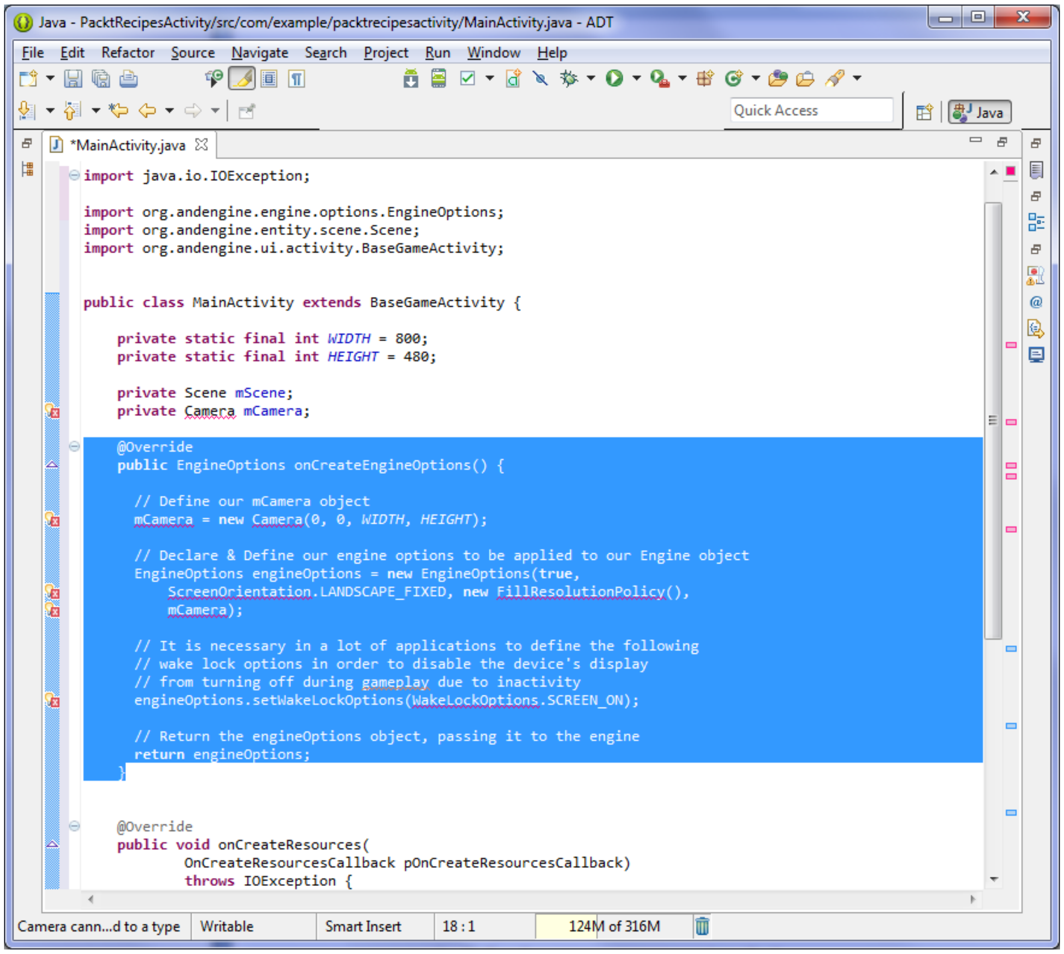
Java - PacktRecipesActivity/src/com/example/packtrecipesactivity/MainActivity.java - ADT

*MainActivity.java

```
import java.io.IOException;

import org.andengine.engine.options.EngineOptions;
import org.andengine.entity.scene.Scene;
import org.andengine.ui.activity.BaseGameActivity;


public class MainActivity extends BaseGameActivity {

    private static final int WIDTH = 800;
    private static final int HEIGHT = 480;

    private Scene mScene;
    private Camera mCamera;

    @Override
    public EngineOptions onCreateEngineOptions() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public void onCreateResources(
            OnCreateResourcesCallback pOnCreateResourcesCallback)
            throws IOException {
        // TODO Auto-generated method stub

    }

    @Override
    public void onCreateScene(OnCreateSceneCallback pOnCreateSceneCallback)
            throws IOException {
        // TODO Auto-generated method stub

    }

    @Override
    public void onPopulateScene(Scene pScene,
            OnPopulateSceneCallback pOnPopulateSceneCallback)
            throws IOException {
```

Camera cann...d to a type | Writable | Smart Insert | 12 : 1 | 148M of 332M

2. Populate the `onCreateEngineOptions()` method in order to properly set up our game's `Camera` object, create our `EngineOptions` object, set the screen to stay awake while our game is running, and return the newly created `EngineOptions` object to the game engine.



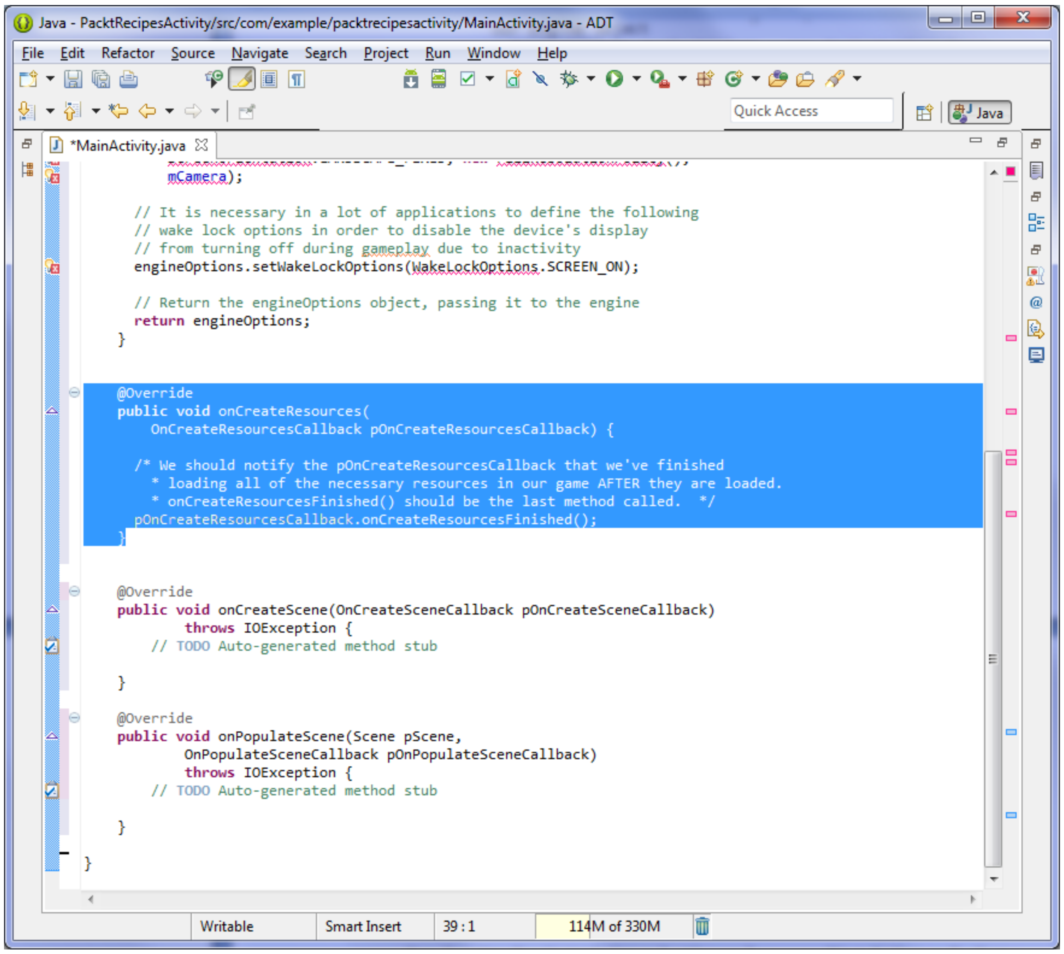
Java - PacktRecipesActivity/src/com/example/packtrecipesactivity/MainActivity.java - ADT

*MainActivity.java

```java
import java.io.IOException;

import org.andengine.engine.options.EngineOptions;
import org.andengine.entity.scene.Scene;
import org.andengine.ui.activity.BaseGameActivity;


public class MainActivity extends BaseGameActivity {

    private static final int WIDTH = 800;
    private static final int HEIGHT = 480;

    private Scene mScene;
    private Camera mCamera;

    @Override
    public EngineOptions onCreateEngineOptions() {

      // Define our mCamera object
      mCamera = new Camera(0, 0, WIDTH, HEIGHT);

      // Declare & Define our engine options to be applied to our Engine object
      EngineOptions engineOptions = new EngineOptions(true,
          ScreenOrientation.LANDSCAPE_FIXED, new FillResolutionPolicy(),
          mCamera);

      // It is necessary in a lot of applications to define the following
      // wake lock options in order to disable the device's display
      // from turning off during gameplay due to inactivity
      engineOptions.setWakeLockOptions(WakeLockOptions.SCREEN_ON);

      // Return the engineOptions object, passing it to the engine
      return engineOptions;
    }


    @Override
    public void onCreateResources(
            OnCreateResourcesCallback pOnCreateResourcesCallback)
            throws IOException {
```

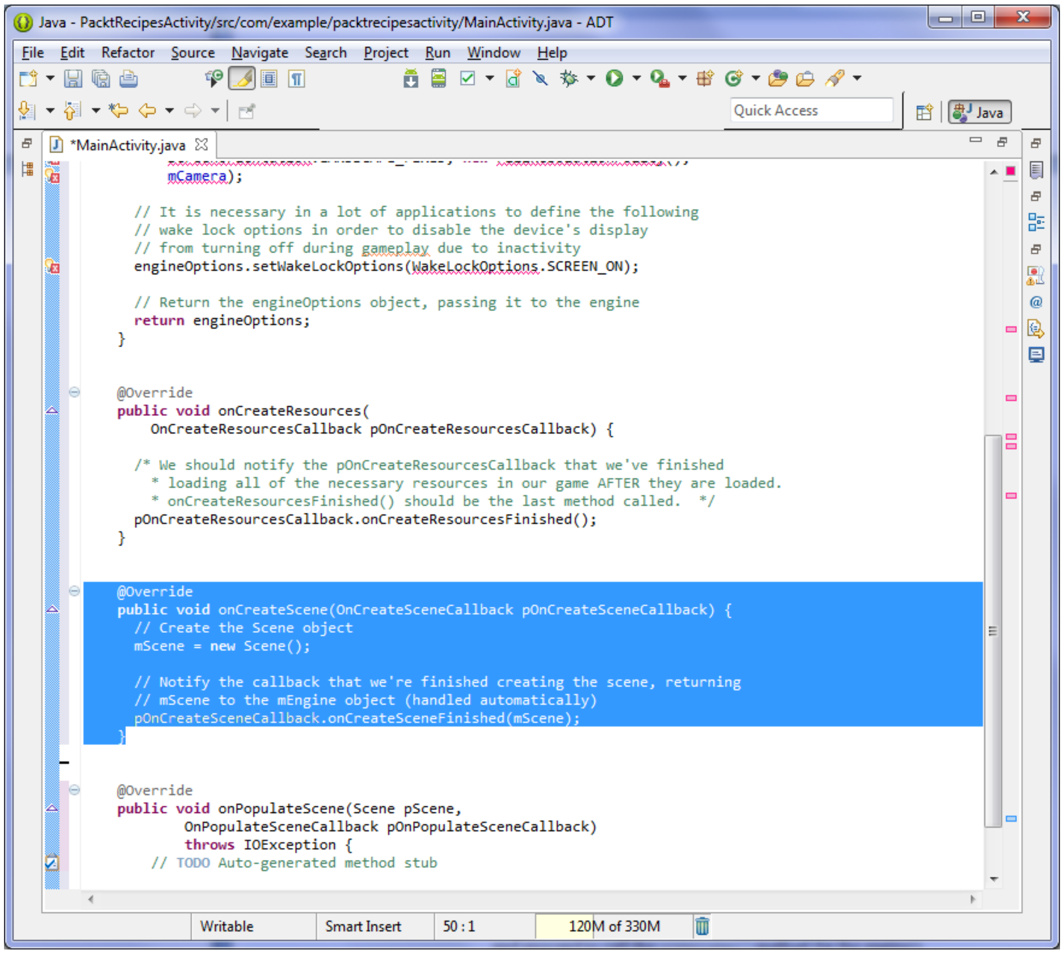
Camera cann...d to a type | Writable | Smart Insert | 18 : 1 | 124M of 316M

3. Populate the `onCreateResources()` method. At the very least, we will be required to make a call to the callback passed in via the method parameter.


Java - PacktRecipesActivity/src/com/example/packtrecipesactivity/MainActivity.java - ADT
File Edit Refactor Source Navigate Search Project Run Window Help
Quick Access
Java
*MainActivity.java
Writable
Smart Insert
39 : 1
114M of 330M

```
            mCamera);

        // It is necessary in a lot of applications to define the following
        // wake lock options in order to disable the device's display
        // from turning off during gameplay due to inactivity
        engineOptions.setWakeLockOptions(WakeLockOptions.SCREEN_ON);

        // Return the engineOptions object, passing it to the engine
        return engineOptions;
    }

    @Override
    public void onCreateResources(
        OnCreateResourcesCallback pOnCreateResourcesCallback) {

        /* We should notify the pOnCreateResourcesCallback that we've finished
         * loading all of the necessary resources in our game AFTER they are loaded.
         * onCreateResourcesFinished() should be the last method called.  */
        pOnCreateResourcesCallback.onCreateResourcesFinished();
    }


    @Override
    public void onCreateScene(OnCreateSceneCallback pOnCreateSceneCallback)
            throws IOException {
        // TODO Auto-generated method stub

    }

    @Override
    public void onPopulateScene(Scene pScene,
            OnPopulateSceneCallback pOnPopulateSceneCallback)
            throws IOException {
        // TODO Auto-generated method stub

    }
}
```

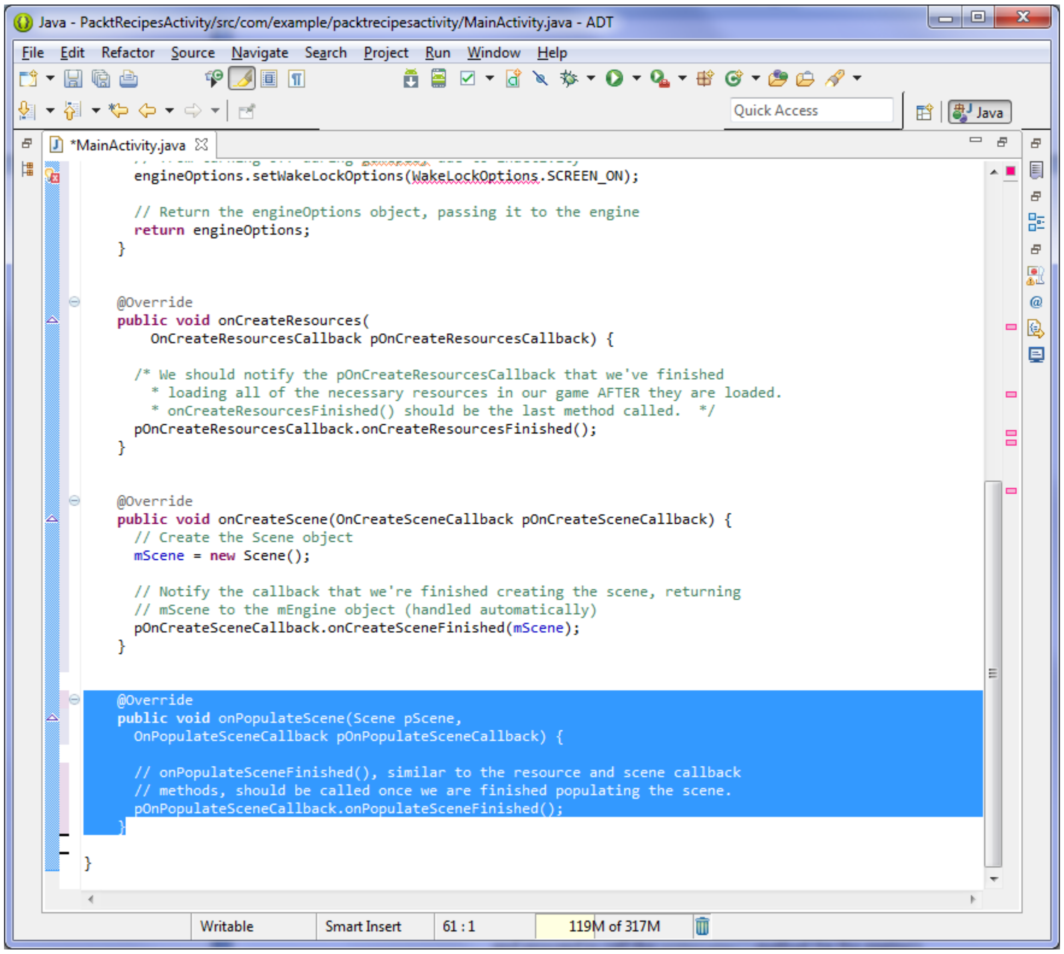
4. Populate the `onCreateScene()` method, creating our `Scene` object and making a call to the callback passed in via the method parameter.



Java - PacktRecipesActivity/src/com/example/packtrecipesactivity/MainActivity.java - ADT

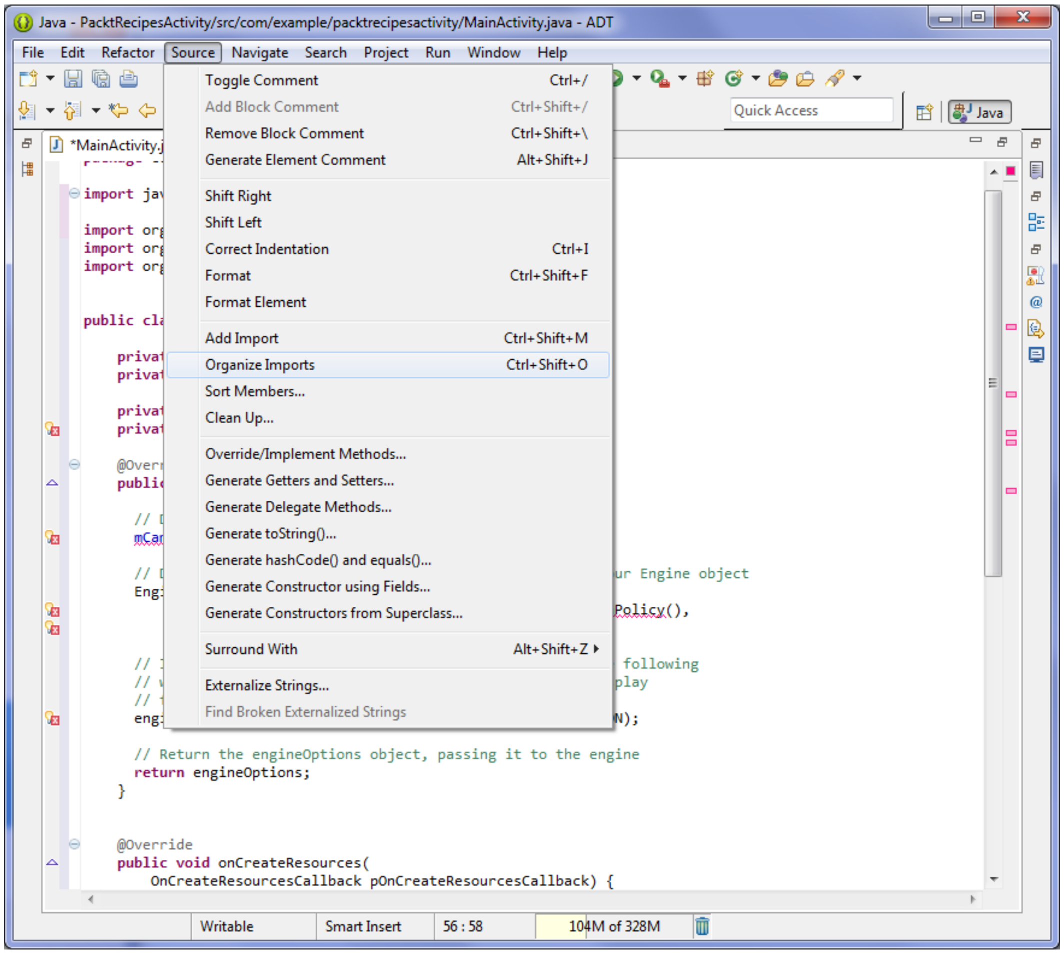
```
            mCamera);

        // It is necessary in a lot of applications to define the following
        // wake lock options in order to disable the device's display
        // from turning off during gameplay due to inactivity
        engineOptions.setWakeLockOptions(WakeLockOptions.SCREEN_ON);

        // Return the engineOptions object, passing it to the engine
        return engineOptions;
    }

    @Override
    public void onCreateResources(
            OnCreateResourcesCallback pOnCreateResourcesCallback) {

        /* We should notify the pOnCreateResourcesCallback that we've finished
         * loading all of the necessary resources in our game AFTER they are loaded.
         * onCreateResourcesFinished() should be the last method called.  */
        pOnCreateResourcesCallback.onCreateResourcesFinished();
    }

    @Override
    public void onCreateScene(OnCreateSceneCallback pOnCreateSceneCallback) {
        // Create the Scene object
        mScene = new Scene();

        // Notify the callback that we're finished creating the scene, returning
        // mScene to the mEngine object (handled automatically)
        pOnCreateSceneCallback.onCreateSceneFinished(mScene);
    }

    @Override
    public void onPopulateScene(Scene pScene,
            OnPopulateSceneCallback pOnPopulateSceneCallback)
            throws IOException {
        // TODO Auto-generated method stub
```

5. Populate the `onPopulateScene()` method. At the very least, we will be required to make a call to the callback passed in via the method parameter.


Java - PacktRecipesActivity/src/com/example/packtrecipesactivity/MainActivity.java - ADT
File Edit Refactor Source Navigate Search Project Run Window Help
Quick Access
Java
*MainActivity.java

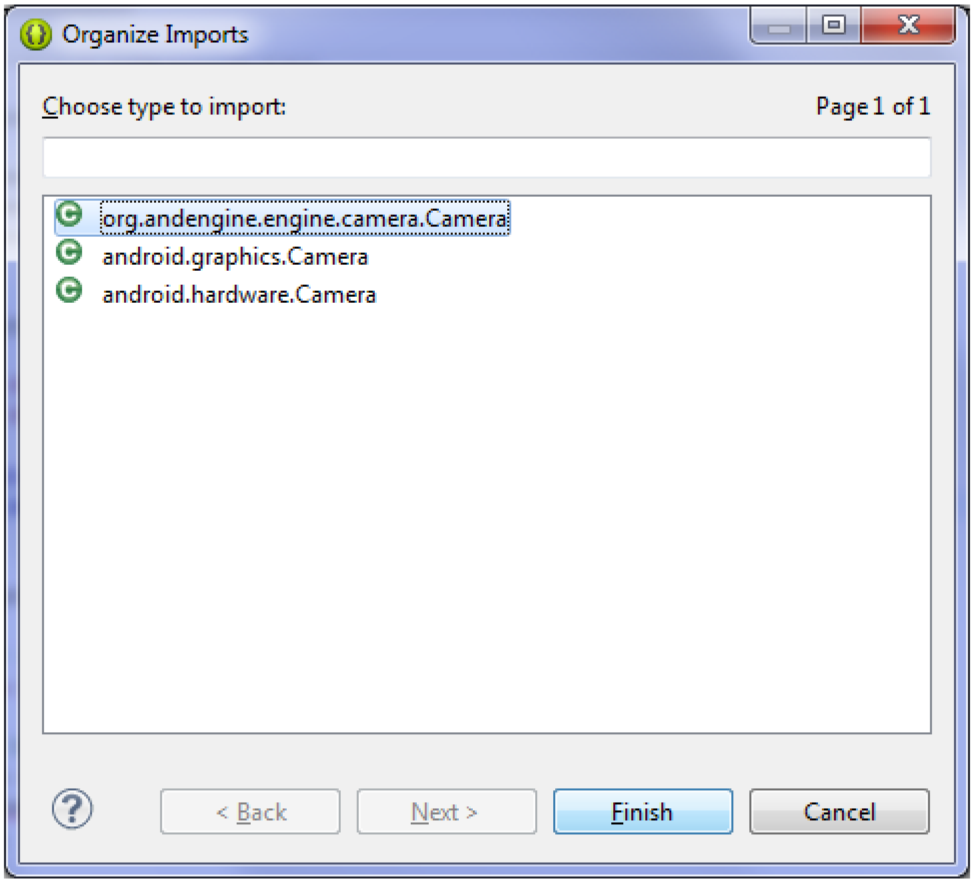
```
    engineOptions.setWakeLockOptions(WakeLockOptions.SCREEN_ON);

    // Return the engineOptions object, passing it to the engine
    return engineOptions;
}

@Override
public void onCreateResources(
        OnCreateResourcesCallback pOnCreateResourcesCallback) {

    /* We should notify the pOnCreateResourcesCallback that we've finished
     * loading all of the necessary resources in our game AFTER they are loaded.
     * onCreateResourcesFinished() should be the last method called.  */
    pOnCreateResourcesCallback.onCreateResourcesFinished();
}

@Override
public void onCreateScene(OnCreateSceneCallback pOnCreateSceneCallback) {
    // Create the Scene object
    mScene = new Scene();

    // Notify the callback that we're finished creating the scene, returning
    // mScene to the mEngine object (handled automatically)
    pOnCreateSceneCallback.onCreateSceneFinished(mScene);
}

@Override
public void onPopulateScene(Scene pScene,
    OnPopulateSceneCallback pOnPopulateSceneCallback) {

    // onPopulateSceneFinished(), similar to the resource and scene callback
    // methods, should be called once we are finished populating the scene.
    pOnPopulateSceneCallback.onPopulateSceneFinished();
}

}
```

6. Once we've added the necessary code for our base AndEngine project foundation, we must include any imports needed. Simply press *Ctrl* + *Shift* + *o* to quickly import all missing imports within a class.


Java - PacktRecipesActivity/src/com/example/packtrecipesactivity/MainActivity.java - ADT
File Edit Refactor Source Navigate Search Project Run Window Help
Quick Access
Java
*MainActivity.j
Toggle Comment Ctrl+/
Add Block Comment Ctrl+Shift+/
Remove Block Comment Ctrl+Shift+\
Generate Element Comment Alt+Shift+J
Shift Right
Shift Left
Correct Indentation Ctrl+I
Format Ctrl+Shift+F
Format Element
Add Import Ctrl+Shift+M
Organize Imports Ctrl+Shift+O
Sort Members...
Clean Up...
Override/Implement Methods...
Generate Getters and Setters...
Generate Delegate Methods...
Generate toString()...
Generate hashCode() and equals()...
Generate Constructor using Fields...
Generate Constructors from Superclass...
Surround With Alt+Shift+Z
Externalize Strings...
Find Broken Externalized Strings
Writable
Smart Insert
56 : 58
104M of 328M

```
import ja

import or
import or
import or

public cla

    privat
    privat

    privat
    privat

    @Over
    publi

        // 
        mCa

        // ur Engine object
        Eng Policy(),

        // following
        // play
        //
        eng N);

        // Return the engineOptions object, passing it to the engine
        return engineOptions;
    }

    @Override
    public void onCreateResources(
        OnCreateResourcesCallback pOnCreateResourcesCallback) {
```

7. Depending on the name of the class needing importing, we may be prompted to select from a list of similarly-named classes. In most cases, we will simply select the **org.andengine...** packaged classes while working with AndEngine. Be careful not to import classes from unwanted packages.



org.andengine.engine.camera.Camera
android.graphics.Camera
android.hardware.Camera