Android Tutorial

References

- This tutorial is a brief overview of some major concepts…Android is much richer and more complex
- Developer's Guide
  - http://developer.android.com/guide/index.html
- API Reference
  - http://developer.android.com/reference/packages.html

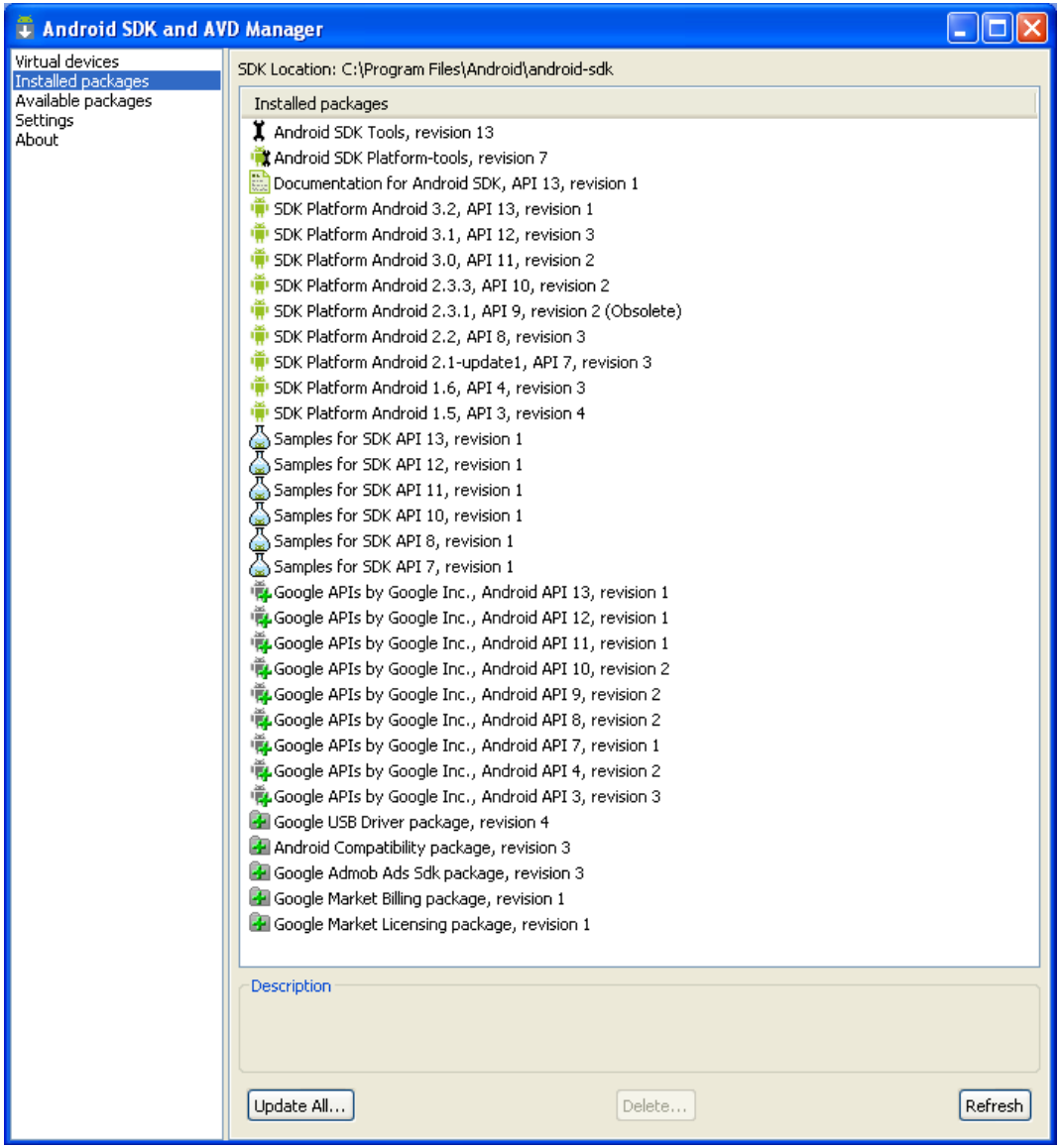
Tools

- Phone
- Eclipse (http://www.eclipse.org/downloads/)
  - Android Plugin (ADT)
- Android SDK (http://developer.android.com/sdk/index.html)
  - Install everything except Additional SDK Platforms, unless you want to
- Windows Users: may need to install Motorola Driver directly (http://www.motorola.com/Support/US-EN/Support-Homepage/Software_and_Drivers/USB-and-PC-Charging-Drivers)

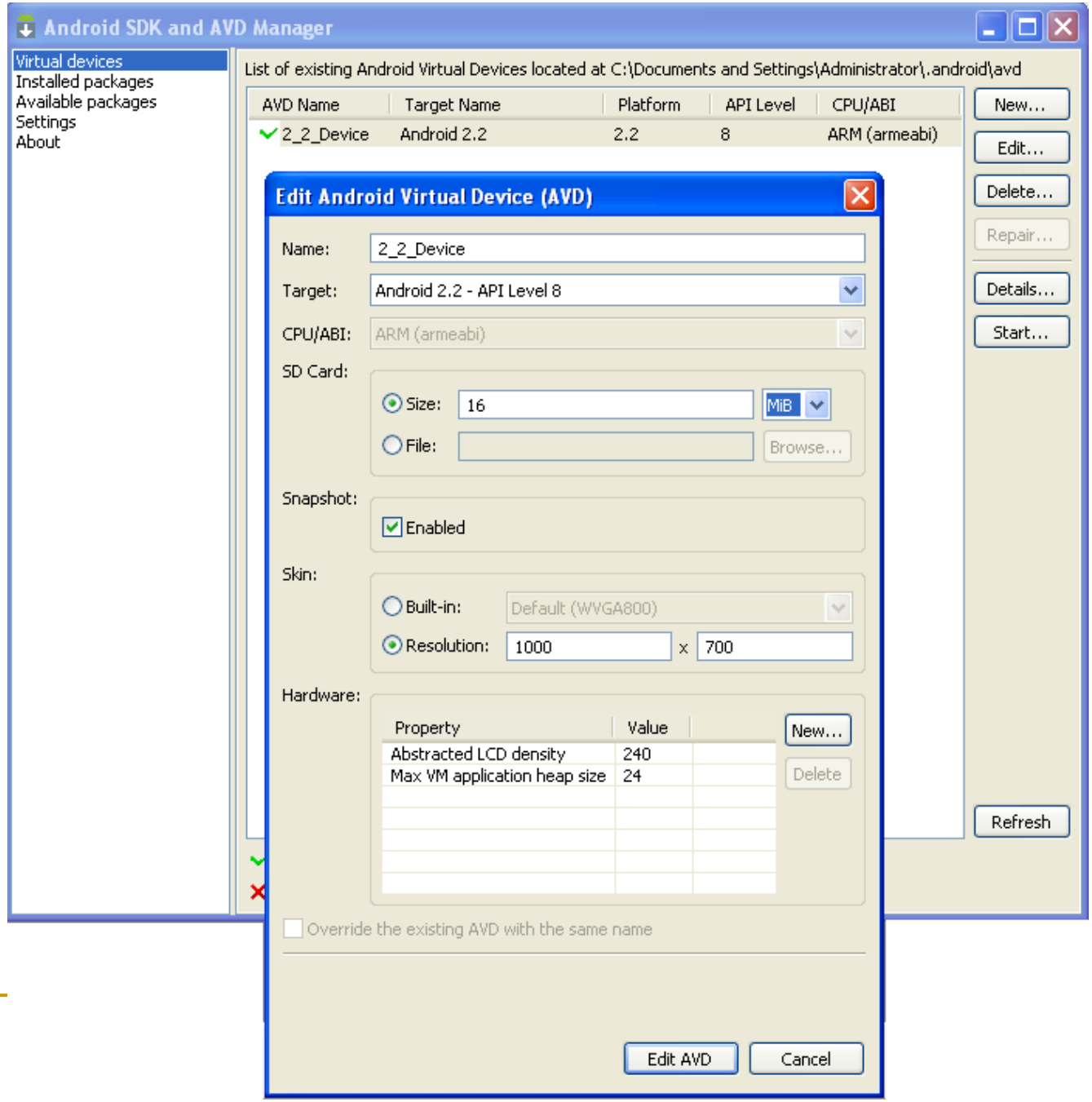
Android SDK

- Once installed open the SDK Manager
- Install the desired packages
- Create an Android Virtual Device (AVD)

SDK Manager


Android SDK and AVD Manager
Virtual devices
Installed packages
Available packages
Settings
About
SDK Location: C:\Program Files\Android\android-sdk
Installed packages
Android SDK Tools, revision 13
Android SDK Platform-tools, revision 7
Documentation for Android SDK, API 13, revision 1
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, revision 2
SDK Platform Android 2.3.1, API 9, revision 2 (Obsolete)
SDK Platform Android 2.2, API 8, revision 3
SDK Platform Android 2.1-update1, API 7, revision 3
SDK Platform Android 1.6, API 4, revision 3
SDK Platform Android 1.5, API 3, revision 4
Samples for SDK API 13, revision 1
Samples for SDK API 12, revision 1
Samples for SDK API 11, revision 1
Samples for SDK API 10, revision 1
Samples for SDK API 8, revision 1
Samples for SDK API 7, revision 1
Google APIs by Google Inc., Android API 13, revision 1
Google APIs by Google Inc., Android API 12, revision 1
Google APIs by Google Inc., Android API 11, revision 1
Google APIs by Google Inc., Android API 10, revision 2
Google APIs by Google Inc., Android API 9, revision 2
Google APIs by Google Inc., Android API 8, revision 2
Google APIs by Google Inc., Android API 7, revision 1
Google APIs by Google Inc., Android API 4, revision 2
Google APIs by Google Inc., Android API 3, revision 3
Google USB Driver package, revision 4
Android Compatibility package, revision 3
Google Admob Ads Sdk package, revision 3
Google Market Billing package, revision 1
Google Market Licensing package, revision 1
Description
Update All...
Delete...
Refresh

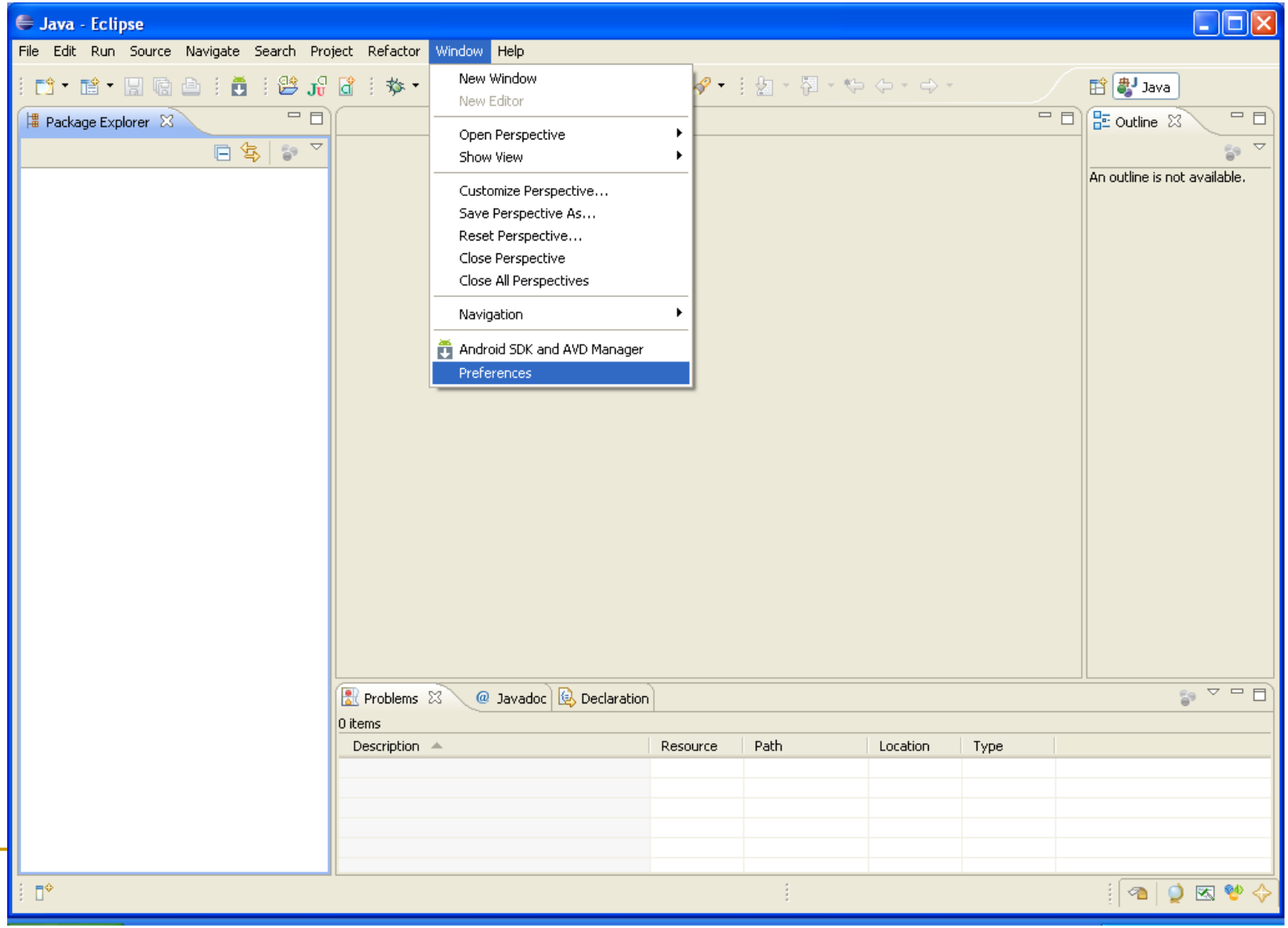
AVD


Android SDK and AVD Manager
Virtual devices
Installed packages
Available packages
Settings
About
List of existing Android Virtual Devices located at C:\Documents and Settings\Administrator\.android\avd
AVD Name
Target Name
Platform
API Level
CPU/ABI
2_2_Device
Android 2.2
2.2
8
ARM (armeabi)
New...
Edit...
Delete...
Repair...
Details...
Start...
Refresh
Edit Android Virtual Device (AVD)
Name:
2_2_Device
Target:
Android 2.2 - API Level 8
CPU/ABI:
ARM (armeabi)
SD Card:
Size:
16
MiB
File:
Browse...
Snapshot:
Enabled
Skin:
Built-in:
Default (WVGA800)
Resolution:
1000
x
700
Hardware:
Property
Value
Abstracted LCD density
240
Max VM application heap size
24
New...
Delete
Override the existing AVD with the same name
Edit AVD
Cancel

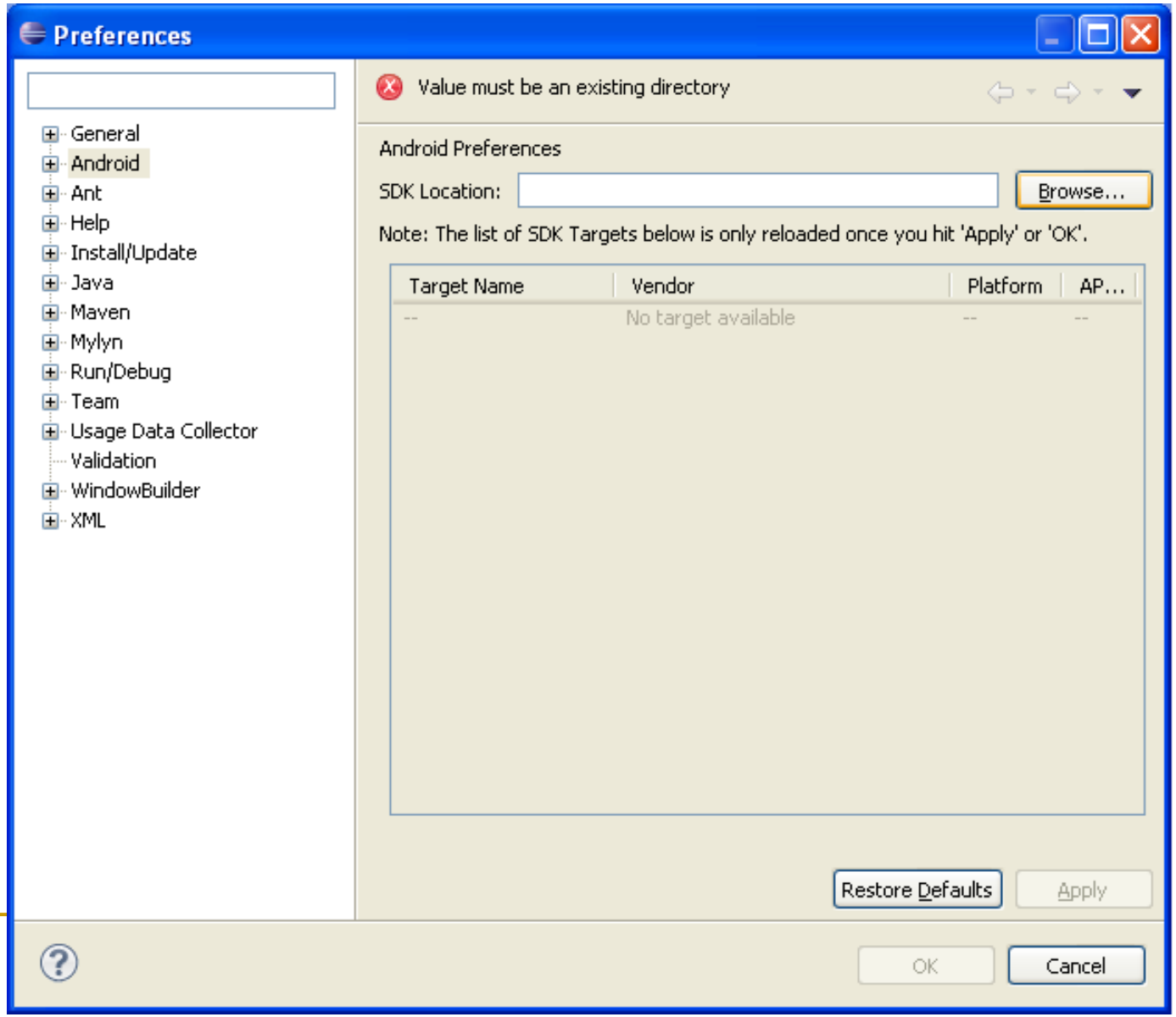
ADT Plugin (1)

- In Eclipse, go to Help -> Install New Software
- Click ‘Add’ in top right
- Enter:
  - Name: ADT Plugin
  - Location: https://dl-ssl.google.com/android/eclipse/
- Click OK, then select ‘Developer Tools’, click Next
- Click Next and then Finish
- Afterwards, restart Eclipse
- Specify SDK location (next 3 slides)
  - Must do this every time start a new project in a new location (at least in Windows)

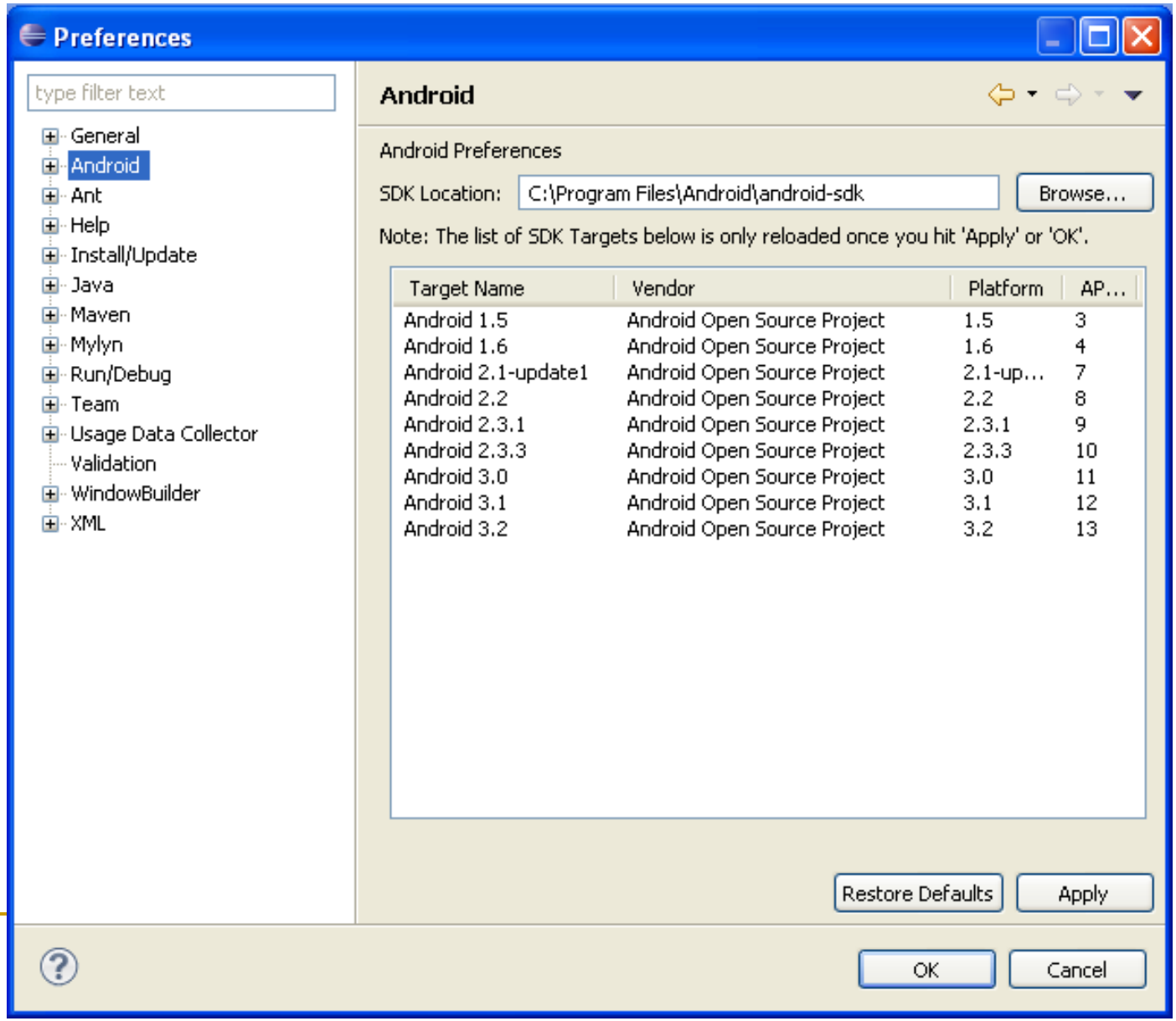
ADT Plugin (2)


Java - Eclipse
File Edit Run Source Navigate Search Project Refactor Window Help
New Window
New Editor
Open Perspective
Show View
Customize Perspective...
Save Perspective As...
Reset Perspective...
Close Perspective
Close All Perspectives
Navigation
Android SDK and AVD Manager
Preferences
Java
Package Explorer
Outline
An outline is not available.
Problems
Javadoc
Declaration
0 items
Description
Resource
Path
Location
Type

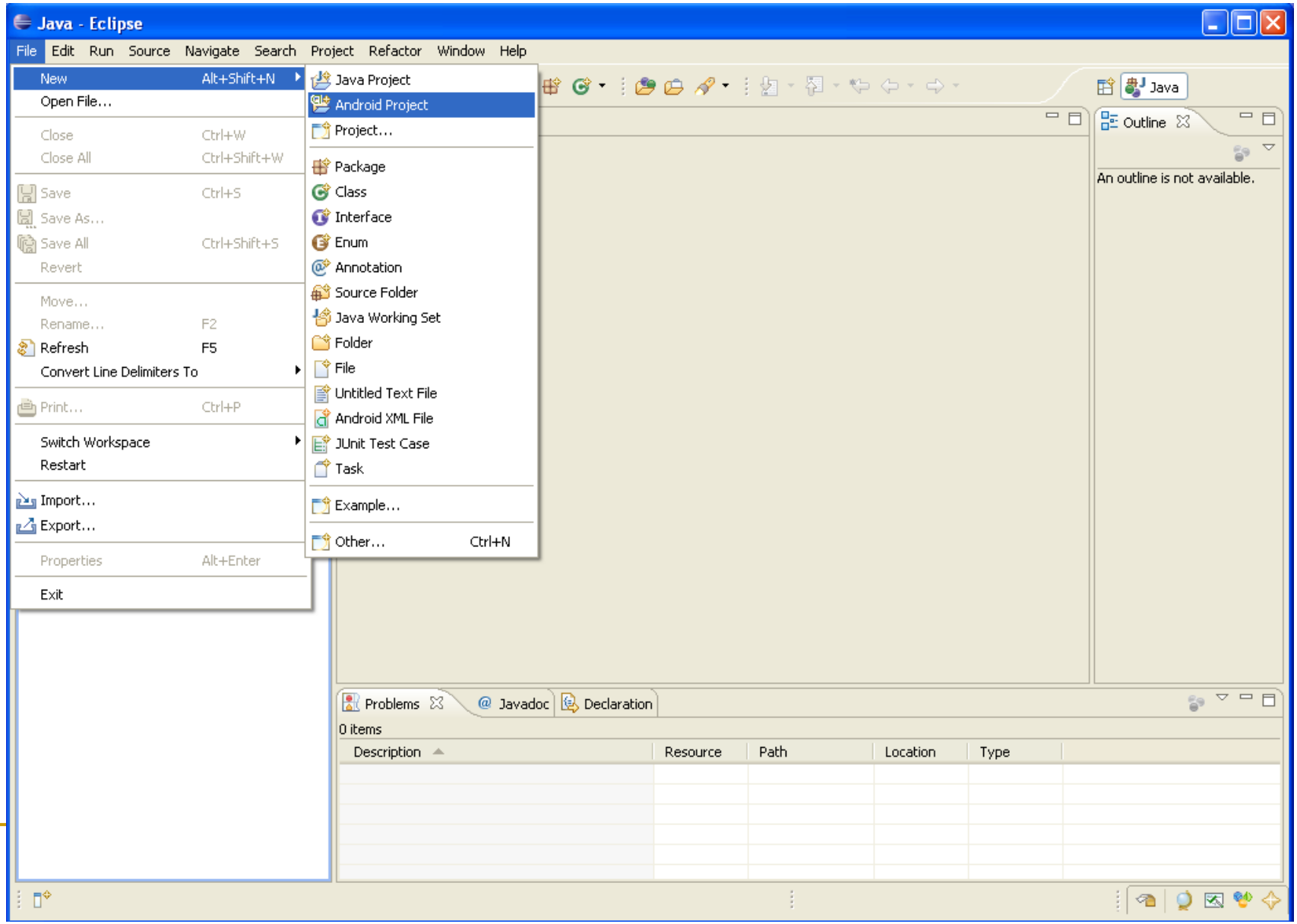
ADT Plugin (3)


Preferences
General
Android
Ant
Help
Install/Update
Java
Maven
Mylyn
Run/Debug
Team
Usage Data Collector
Validation
WindowBuilder
XML
Value must be an existing directory
Android Preferences
SDK Location:
Browse...
Note: The list of SDK Targets below is only reloaded once you hit 'Apply' or 'OK'.
Target Name
Vendor
Platform
AP...
No target available
Restore Defaults
Apply
OK
Cancel

ADT Plugin (4)


Preferences
type filter text
General
Android
Ant
Help
Install/Update
Java
Maven
Mylyn
Run/Debug
Team
Usage Data Collector
Validation
WindowBuilder
XML
Android
Android Preferences
SDK Location:
C:\Program Files\Android\android-sdk
Browse...
Note: The list of SDK Targets below is only reloaded once you hit 'Apply' or 'OK'.
Target Name
Vendor
Platform
AP...
Android 1.5
Android Open Source Project
1.5
3
Android 1.6
Android Open Source Project
1.6
4
Android 2.1-update1
Android Open Source Project
2.1-up...
7
Android 2.2
Android Open Source Project
2.2
8
Android 2.3.1
Android Open Source Project
2.3.1
9
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
Restore Defaults
Apply
OK
Cancel

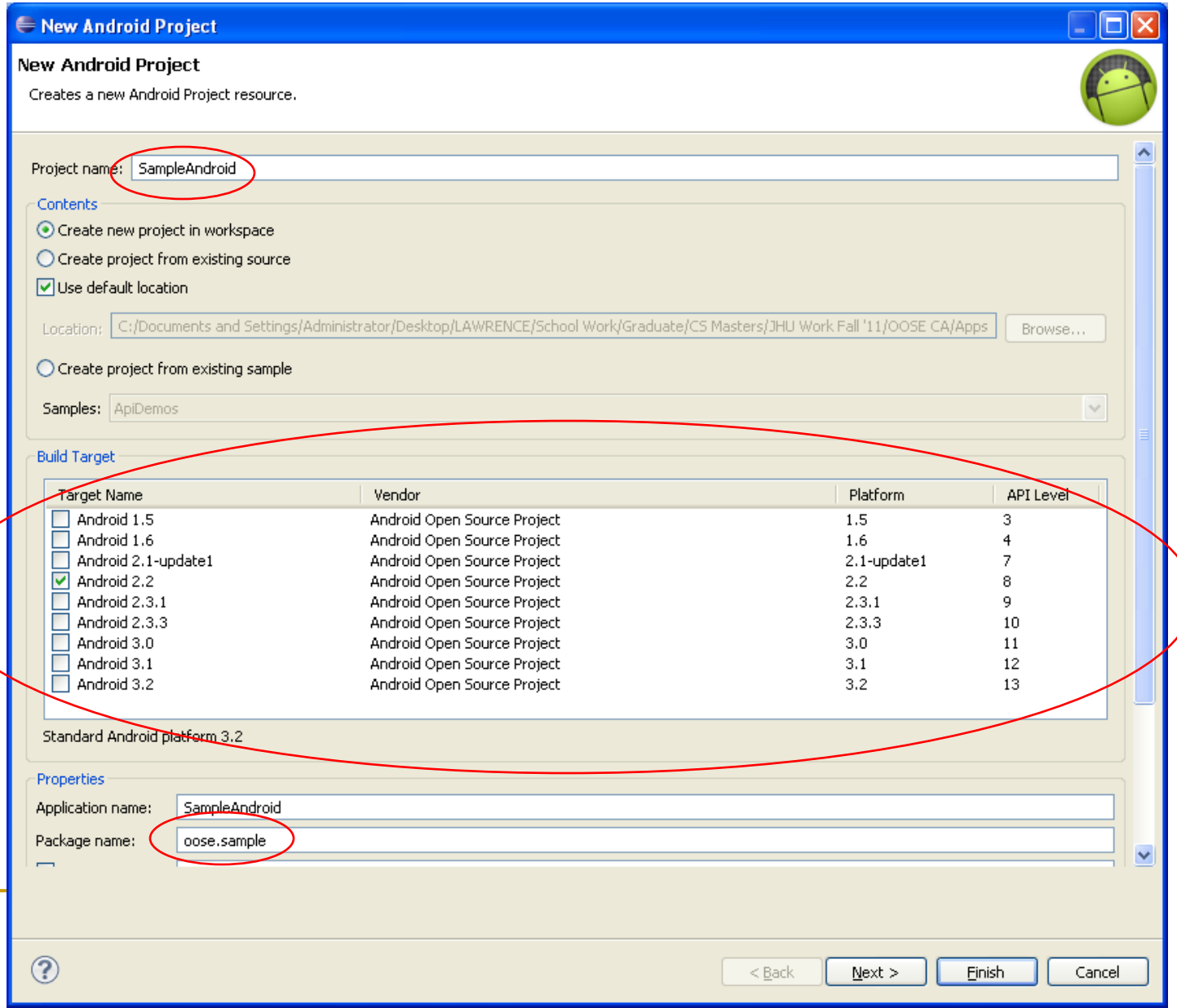
Creating a Project (1)



Creating a Project (2)

Need the items circled

Then click Finish



Project Components

- src – your source code
- gen – auto-generated code (usually just R.java)
- Included libraries
- Resources
  - Drawables (like .png images)
  - Layouts
  - Values (like strings)
- Manifest file

XML

- Used to define some of the resources
  - Layouts (UI)
  - Strings
- Manifest file
- Shouldn’t usually have to edit it directly, Eclipse can do that for you
- Preferred way of creating UIs
  - Separates the description of the layout from any actual code that controls it
  - Can easily take a UI from one platform to another

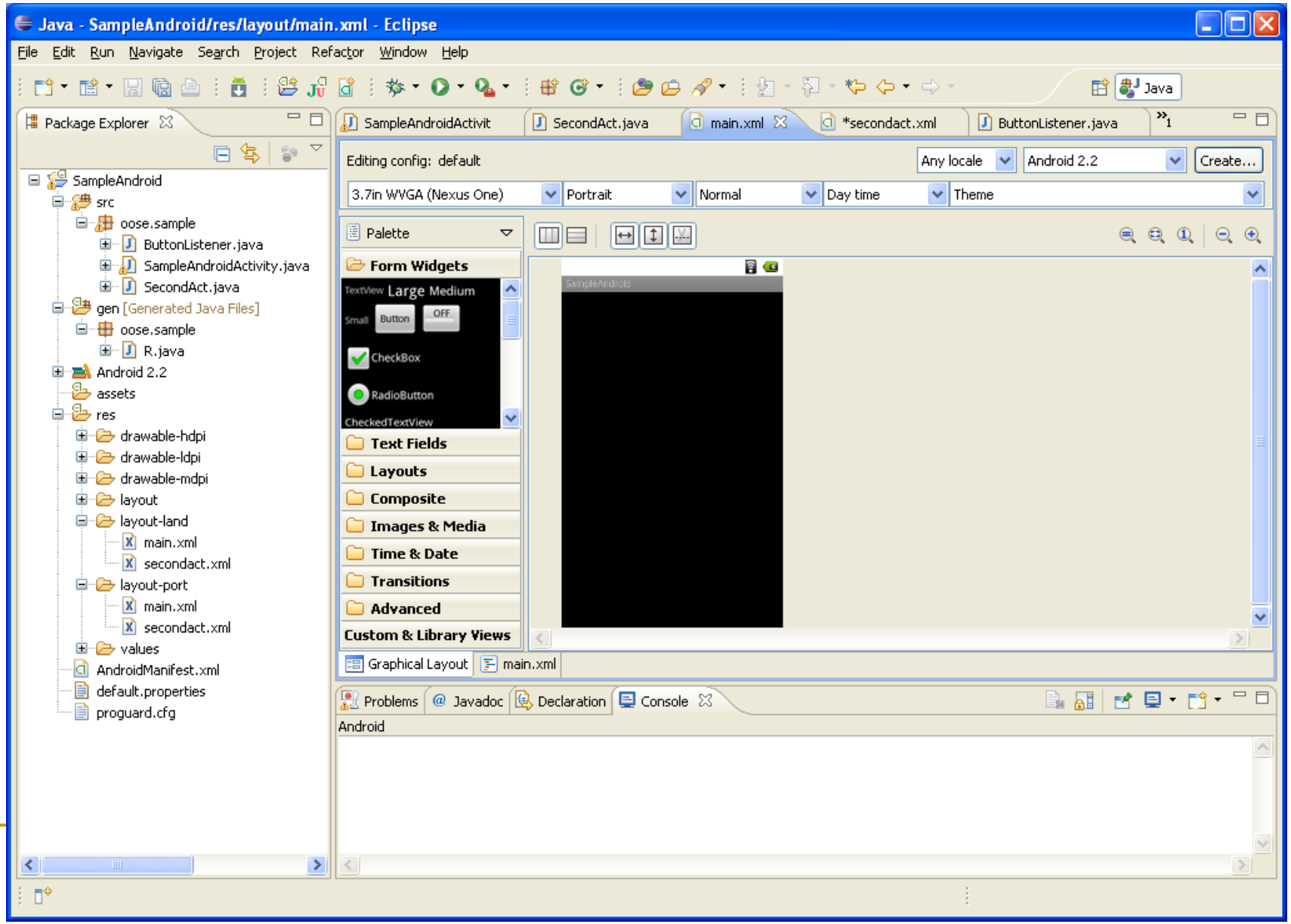
R Class

- Auto-generated: you shouldn't edit it
- Contains IDs of the project resources
- Enforces good software engineering
- Use findViewById and Resources object to get access to the resources
  - Ex. Button b = (Button)findViewById(R.id.button1)
  - Ex. getResources().getString(R.string.*hello*));

Layouts (1)

- Eclipse has a great UI creator
  - Generates the XML for you
- Composed of *View* objects
- Can be specified for portrait and landscape mode
  - Use same file name, so can make completely different UIs for the orientations without modifying any code

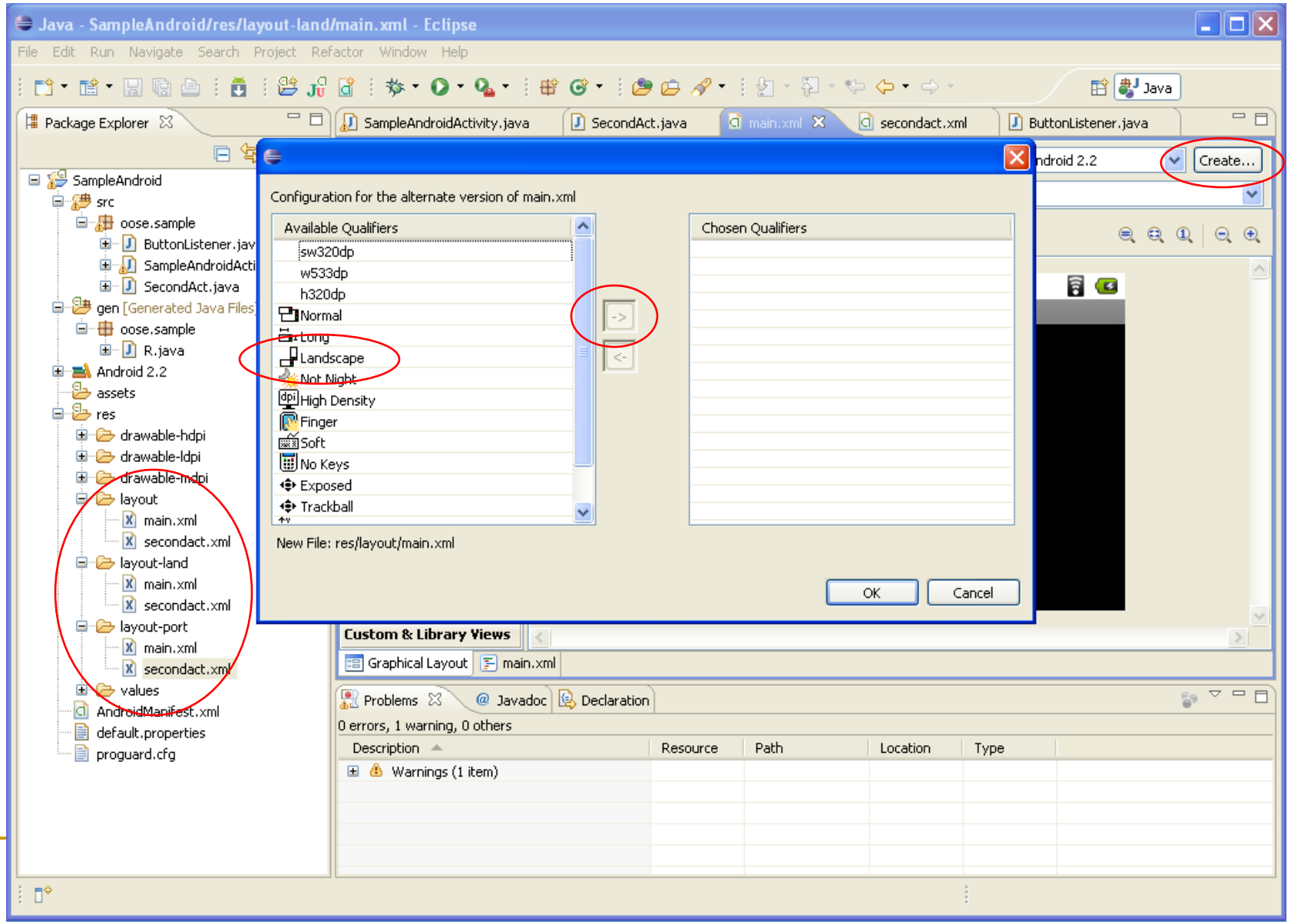
Layouts (2)


Java - SampleAndroid/res/layout/main.xml - Eclipse
File Edit Run Navigate Search Project Refactor Window Help
Java
Package Explorer
SampleAndroid
src
oose.sample
ButtonListener.java
SampleAndroidActivity.java
SecondAct.java
gen [Generated Java Files]
oose.sample
R.java
Android 2.2
assets
res
drawable-hdpi
drawable-ldpi
drawable-mdpi
layout
layout-land
main.xml
secondact.xml
layout-port
main.xml
secondact.xml
values
AndroidManifest.xml
default.properties
proguard.cfg
SampleAndroidActivit
SecondAct.java
main.xml
*secondact.xml
ButtonListener.java
Editing config: default
Any locale
Android 2.2
Create...
3.7in WVGA (Nexus One)
Portrait
Normal
Day time
Theme
Palette
Form Widgets
TextView Large Medium
Small Button OFF
CheckBox
RadioButton
CheckedTextView
Text Fields
Layouts
Composite
Images & Media
Time & Date
Transitions
Advanced
Custom & Library Views
SampleAndroid
Graphical Layout
main.xml
Problems
Javadoc
Declaration
Console
Android

Layouts (3)

- Click ‘Create’ to make layout modifications
- When in portrait mode can select ‘Portrait’ to make a res sub folder for portrait layouts
  - Likewise for Landscape layouts while in landscape mode
  - Will create folders titled ‘layout-port’ and ‘layout-land’
- Note: these ‘port’ and ‘land’ folders are examples of ‘alternate layouts’, see here for more info
  - http://developer.android.com/guide/topics/resources/providing-resources.html
- Avoid errors by making sure components have the same id in both orientations, and that you’ve tested each orientation thoroughly

Layouts (4)


Java - SampleAndroid/res/layout-land/main.xml - Eclipse
File Edit Run Navigate Search Project Refactor Window Help
Java
Package Explorer
SampleAndroidActivity.java
SecondAct.java
main.xml
secondact.xml
ButtonListener.java
SampleAndroid
src
oose.sample
ButtonListener.jav
SampleAndroidActi
SecondAct.java
gen [Generated Java Files]
oose.sample
R.java
Android 2.2
assets
res
drawable-hdpi
drawable-ldpi
drawable-mdpi
layout
main.xml
secondact.xml
layout-land
main.xml
secondact.xml
layout-port
main.xml
secondact.xml
values
AndroidManifest.xml
default.properties
proguard.cfg
Configuration for the alternate version of main.xml
Available Qualifiers
sw320dp
w533dp
h320dp
Normal
Long
Landscape
Not Night
High Density
Finger
Soft
No Keys
Exposed
Trackball
Chosen Qualifiers
New File: res/layout/main.xml
OK
Cancel
ndroid 2.2
Create...
Custom & Library Views
Graphical Layout
main.xml
Problems
Javadoc
Declaration
0 errors, 1 warning, 0 others
Description
Resource
Path
Location
Type
Warnings (1 item)

Strings

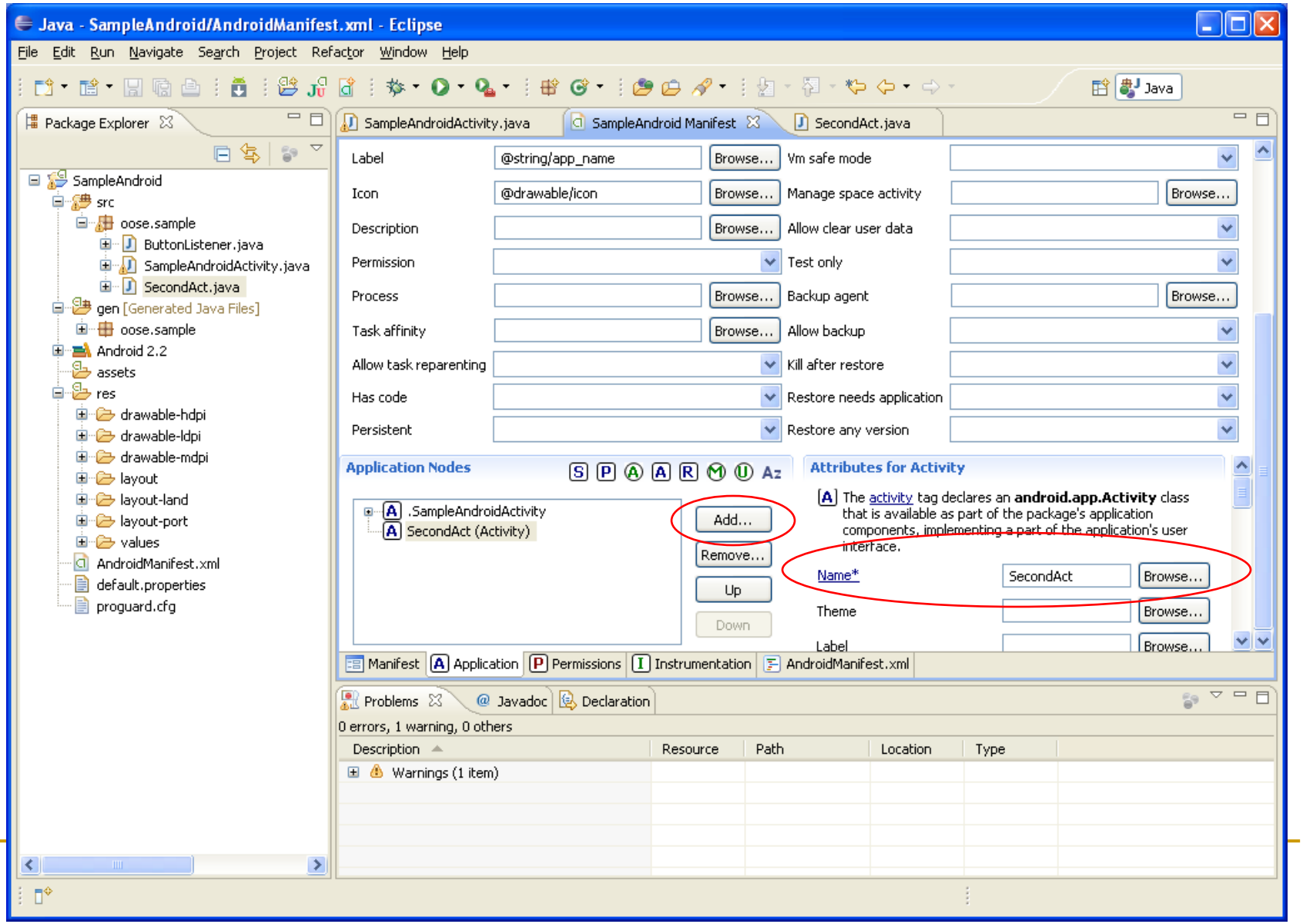
- In res/values
  - strings.xml
- Application wide available strings
- Promotes good software engineering
- UI components made in the UI editor should have text defined in strings.xml

- Strings are just one kind of ‘Value’ there are many others

Manifest File (1)

- Contains characteristics about your application
- When have more than one Activity in app, NEED to specify it in manifest file
  - Go to graphical view of the manifest file
  - Add an Activity in the bottom right
  - Browse for the name of the activity
- Need to specify Services and other components too
- Also important to define permissions and external libraries, like Google Maps API

Manifest File (2) – Adding an Activity



Android Programming Components

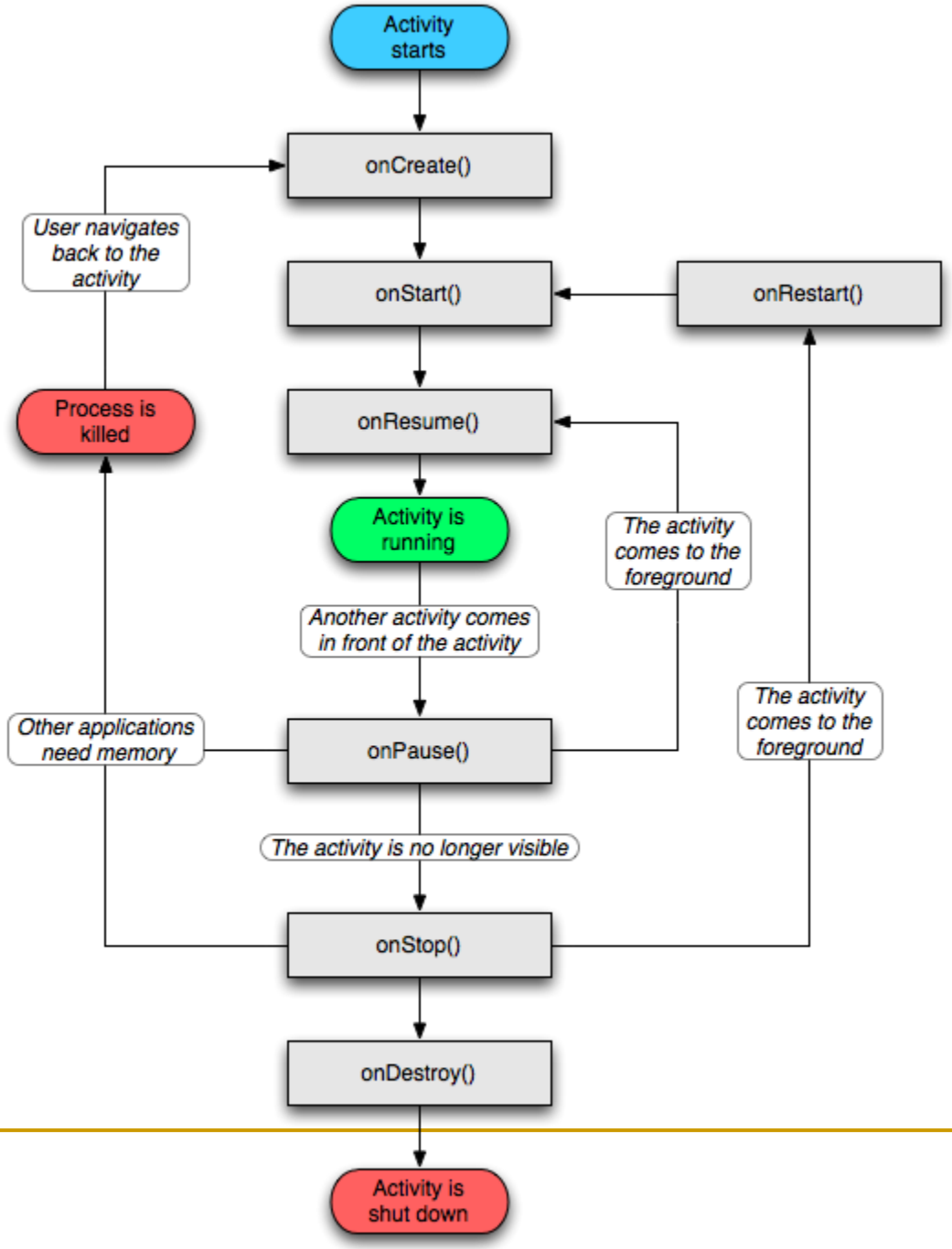
- Activity
  - http://developer.android.com/guide/topics/fundamentals/activities.html
- Service
  - http://developer.android.com/guide/topics/fundamentals/services.html
- Content Providers
- Broadcast Receivers

- Android in a nutshell:
  - http://developer.android.com/guide/topics/fundamentals.html

Activities (1)

- The basis of android applications
- A single Activity defines a single viewable screen
  - the actions, not the layout
- Can have multiple per application
- Each is a separate entity
- They have a structured life cycle
  - Different events in their life happen either via the user touching buttons or programmatically

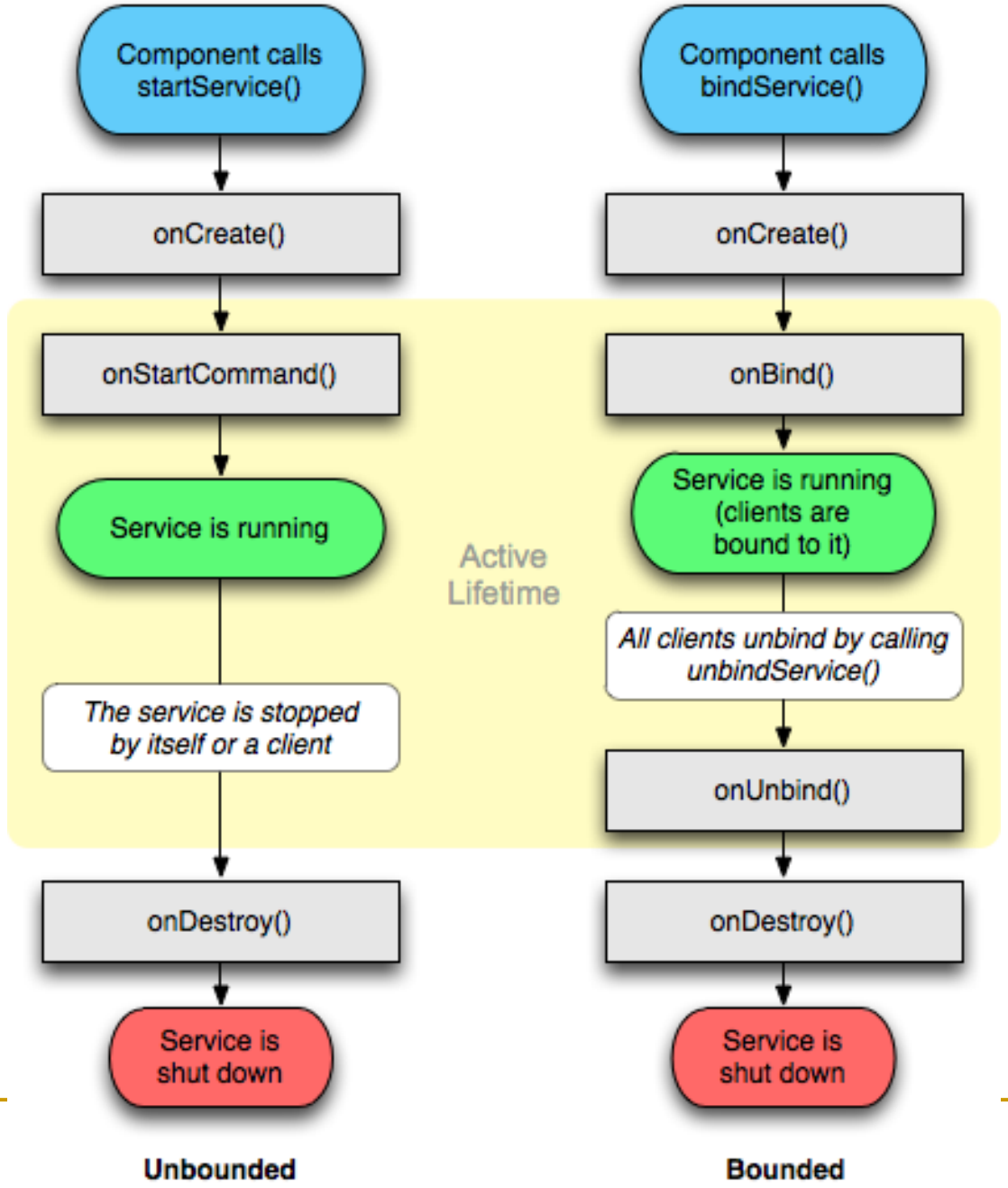
Activities (2)



Services (1)

- Run in the background
  - Can continue even if Activity that started it dies
  - Should be used if something needs to be done while the user is not interacting with application
    - Otherwise, a thread is probably more applicable
  - Should create a new thread in the service to do work in, since the service runs in the main thread
- Can be bound to an application
  - In which case will terminate when all applications bound to it unbind
  - Allows multiple applications to communicate with it via a common interface
- Needs to be declared in manifest file
- Like Activities, has a structured life cycle

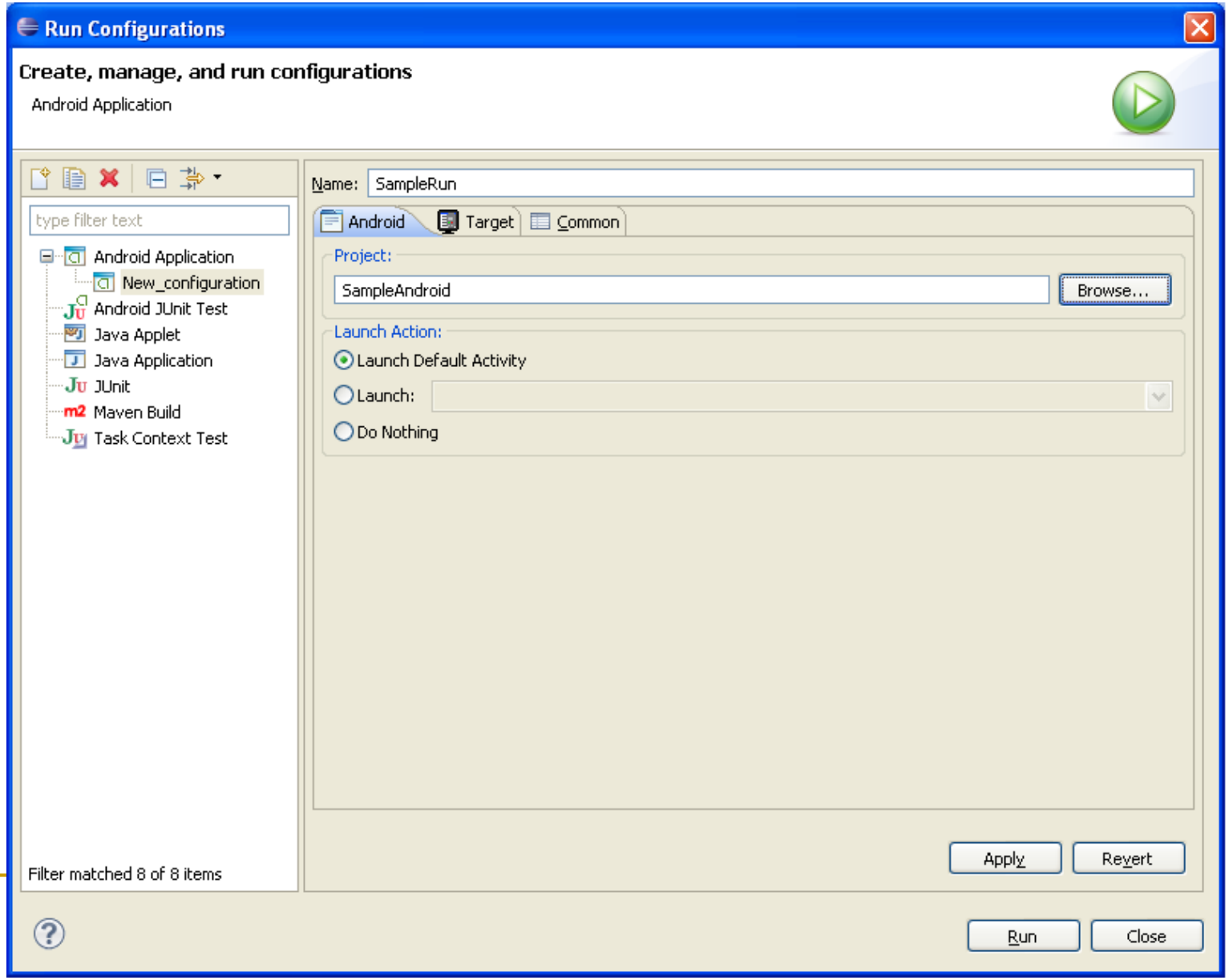
Services (2)


Component calls startService()
onCreate()
onStartCommand()
Service is running
The service is stopped by itself or a client
onDestroy()
Service is shut down
Unbounded
Active Lifetime
Component calls bindService()
onCreate()
onBind()
Service is running (clients are bound to it)
All clients unbind by calling unbindService()
onUnbind()
onDestroy()
Service is shut down
Bounded

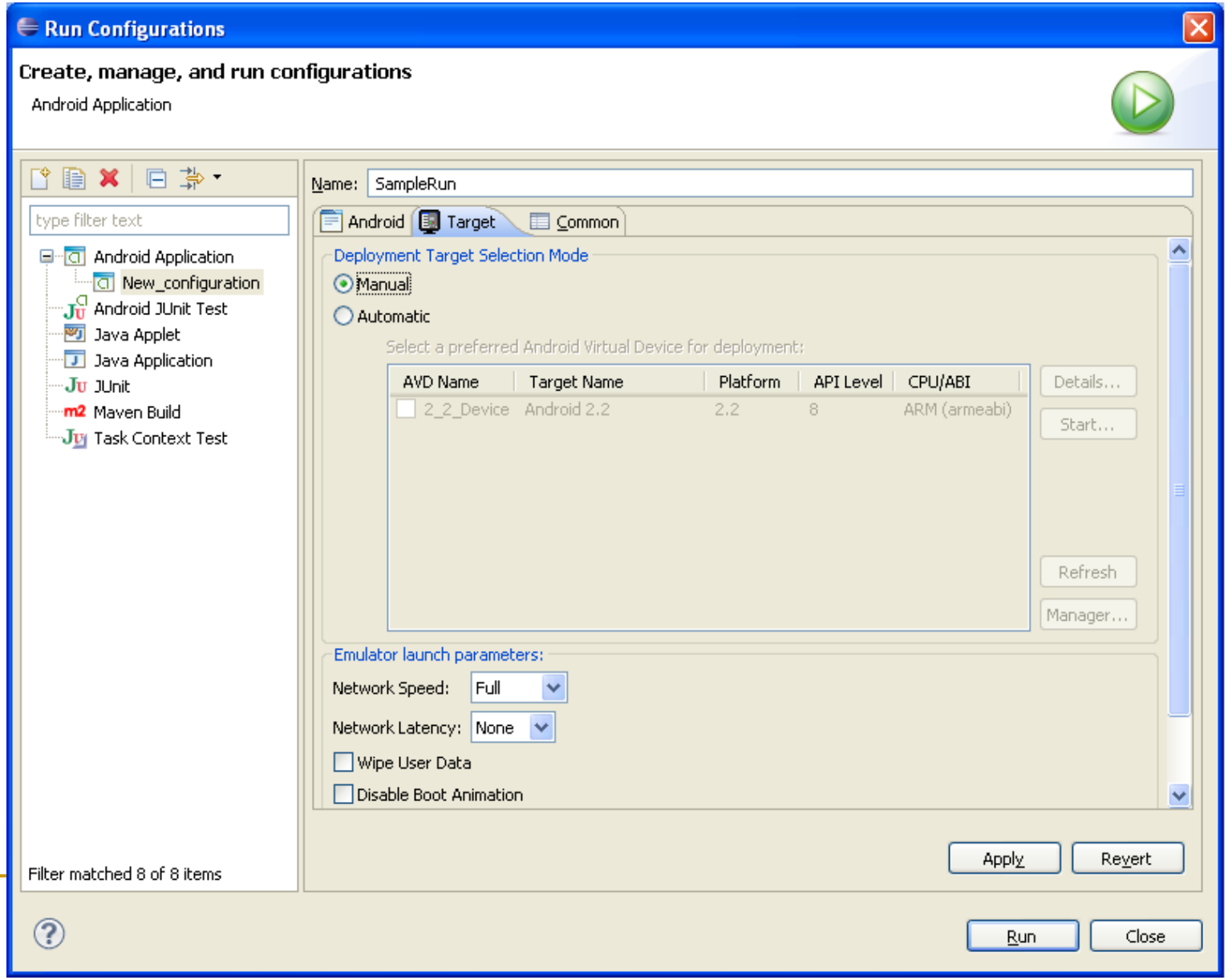
Running in Eclipse (1)

- Similar to launching a regular Java app, use the launch configurations
- Specify an Android Application and create a new one
- Specify activity to be run
- Can select a manual option, so each time program is run, you are asked whether you want to use the actual phone or the emulator
  - Otherwise, it should be smart and use whichever one is available

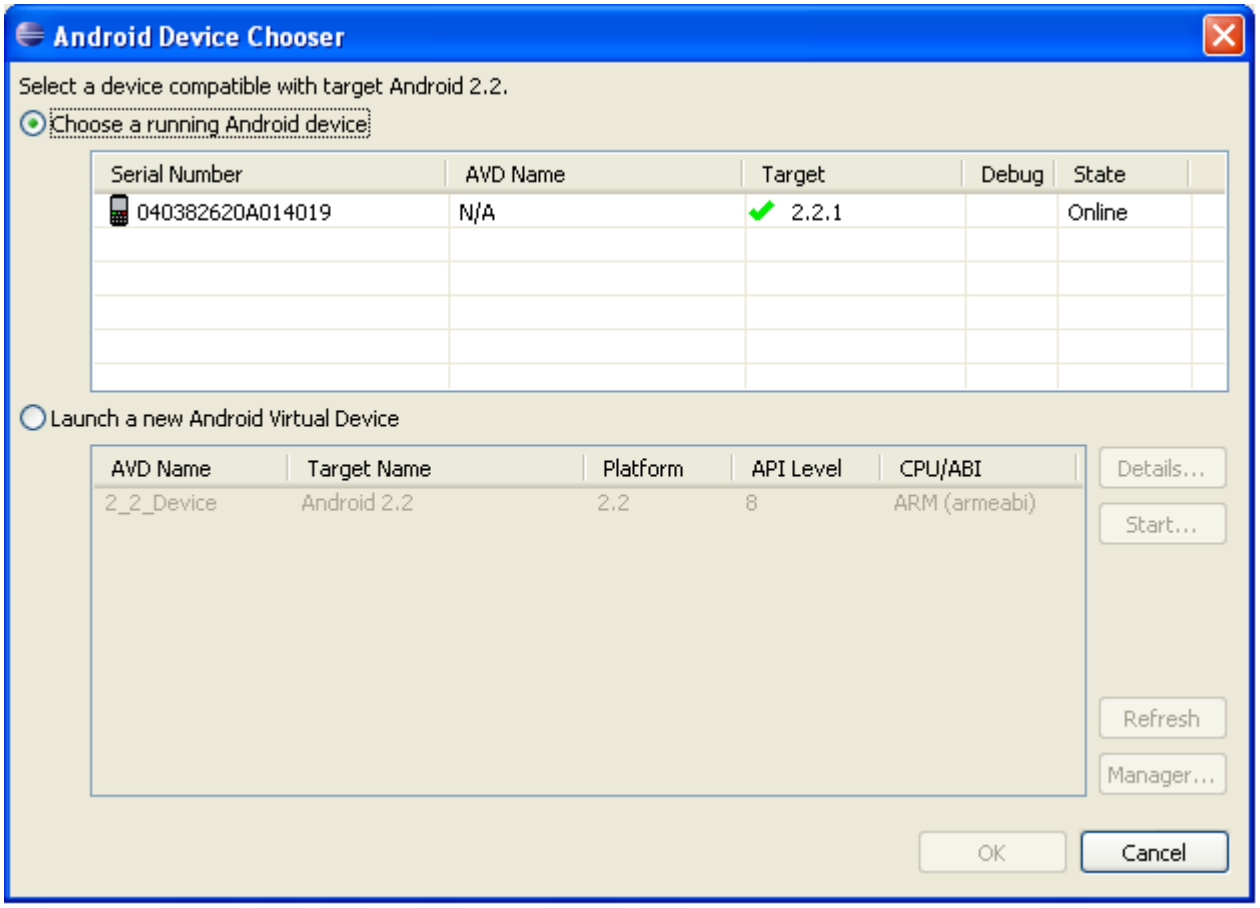
Running in Eclipse (2)



Running in Eclipse (3)



Running in Eclipse (4)



USB Debugging

- Should be enabled on phone to use developer features
- In the main apps screen select Settings -> Applications -> Development -> USB debugging (it needs to be checked)

Android Debug Bridge

- Used for a wide variety of developer tasks
  - Read from the log file
  - Show what android devices are available
  - Install android applications (.apk files)
- In the ‘platform-tools’ directory of the main android sdk directory
  - Recommend putting this directory and the ‘tools’ directory on the system path
- adb.exe

Debugging

- Instead of using traditional System.out.println, use the Log class
  - Imported with android.util.Log
  - Multiple types of output (debug, warning, error, …)
  - Log.d(<tag>,<string>)
- Can be read using logcat.
  - Print out the whole log, which auto-updates
    - adb logcat
  - Erase log
    - adb logcat –c
  - Filter output via tags
    - adb logcat <tag>:<msg type> *:S
    - can have multiple <tag>:<msg type> filters
    - <msg type> corresponds to debug, warning, error, etc.
    - If use Log.d(), then <msg type> = D
- Reference
  - http://developer.android.com/guide/developing/debugging/debugging-log.html

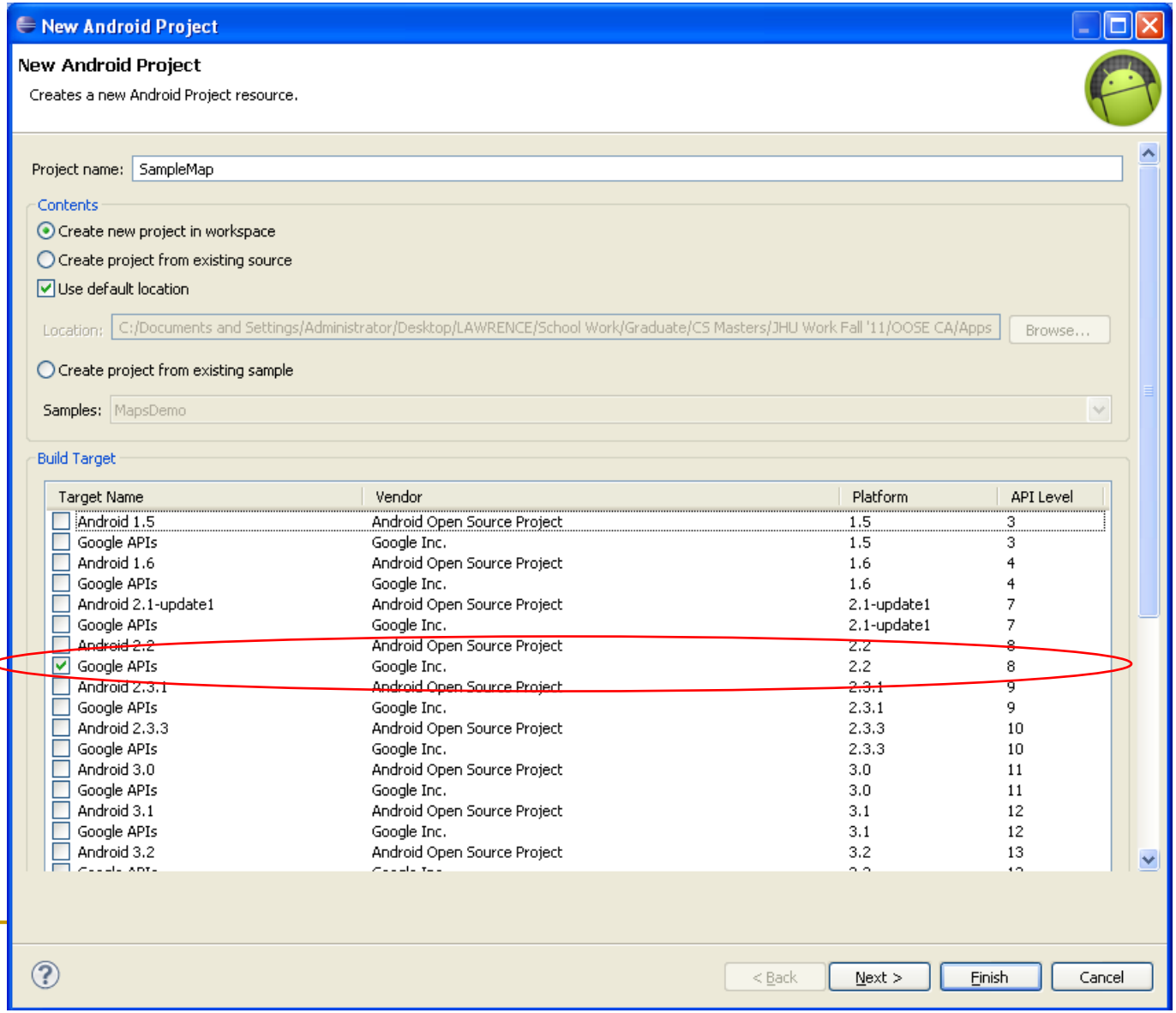
Screen Shots

- Some say you need to root the phone – that is not true
- One option: Android Screen Capture
  - http://www.mightypocket.com/2010/08/android-screenshots-screen-capture-screen-cast/
  - It's slow, but fine for screenshots of applications whose screens aren't changing fast
  - Read their installation help, following the extra steps if need be (I had to *copy* adb.exe and some dll files, as they explain)

Maps Example (1)

- Using Google Maps in your app
- Setup project to use ‘Google API’ version
- Edit Manifest file
  - To indicate the app will use maps and the internet
- Get a maps API key
- Note: Google Maps API can display a map and draw overlays, but is not the full Google Maps experience you enjoy on the web
  - For example, there does not seem to be inherent support for drawing routes between points (if you find it let me know)…however, you can draw lines between points and almost any type of overlay, but that’s different than street routes
  - The directions API is a web service, which is different, among several other Google web services
- Read the Google API terms of use

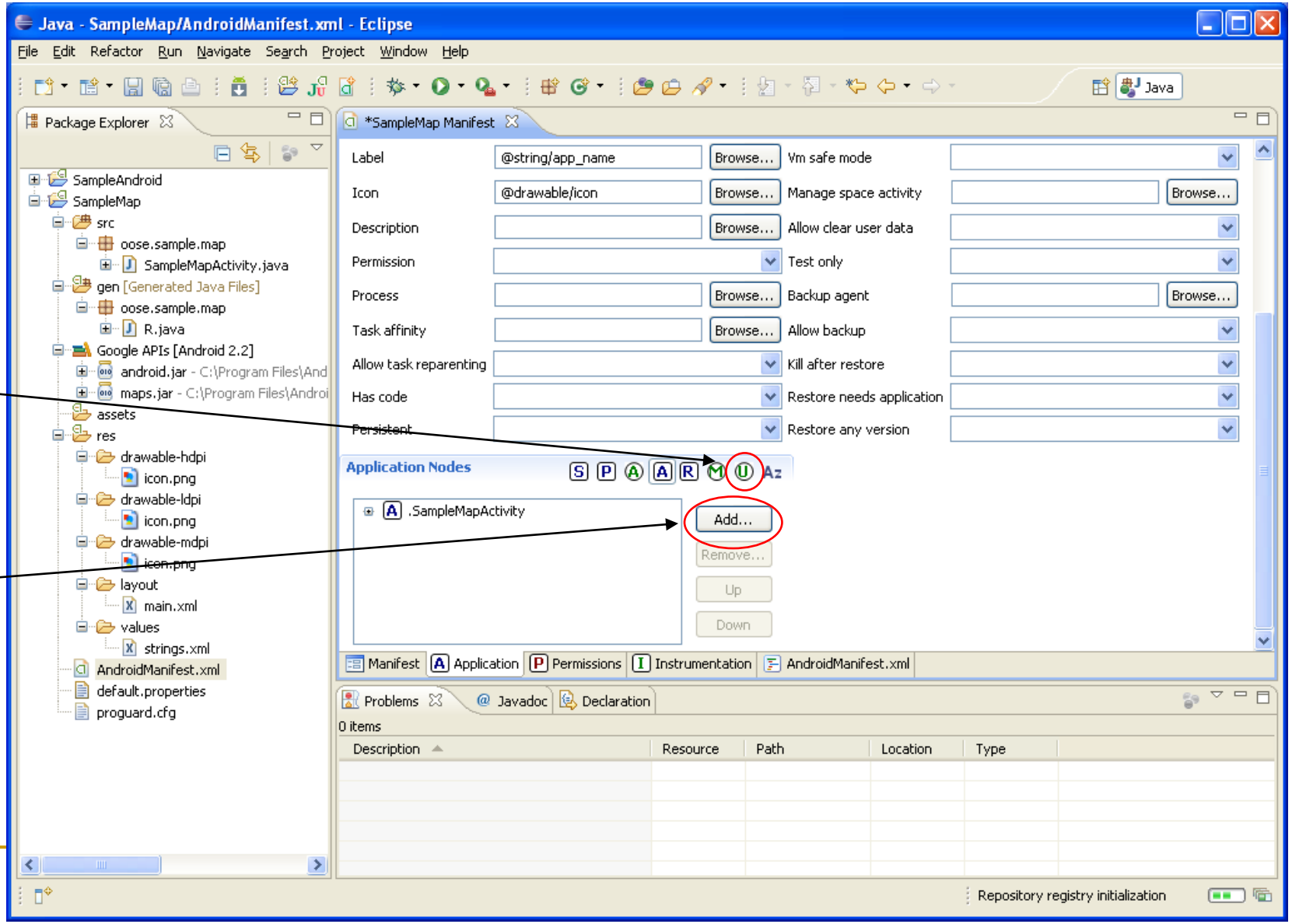
Maps Example (2)



Maps Example (3) – Manifest (1)

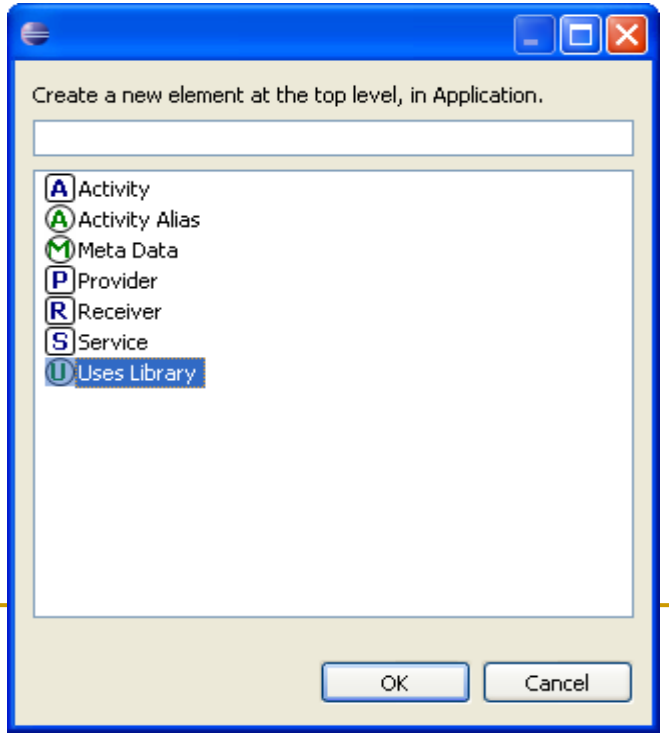
- Open Manifest file
- Add map library tag
  - Add the ‘Uses Library’ com.google.android.maps
- Indicate the app will access the internet
  - Add the ‘Permission’ android.permission.INTERNET
- End goal is to add the following two lines to XML file, under the <manifest> and <application tags>, respectively
  - Under the <manifest> tag
    - <uses-permission android:name=*"android.permission.INTERNET"*></uses-permission>
  - Under the <application> tag
    - <uses-library android:name=*"com.google.android.maps"*></uses-library>
- Following is GUI way to add them

Maps Example (4) – Manifest (2)

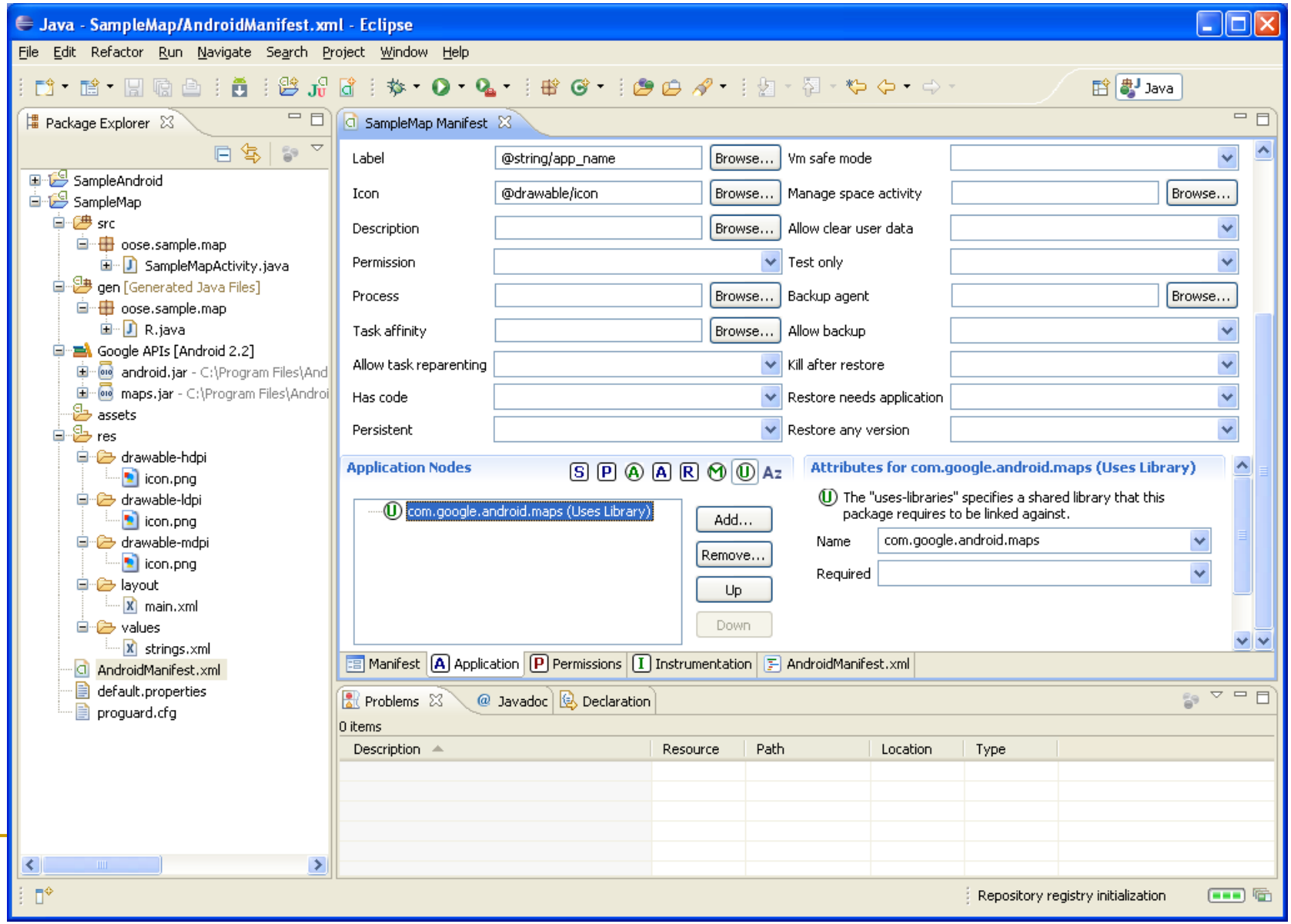


Maps Example (5) – Manifest (3)

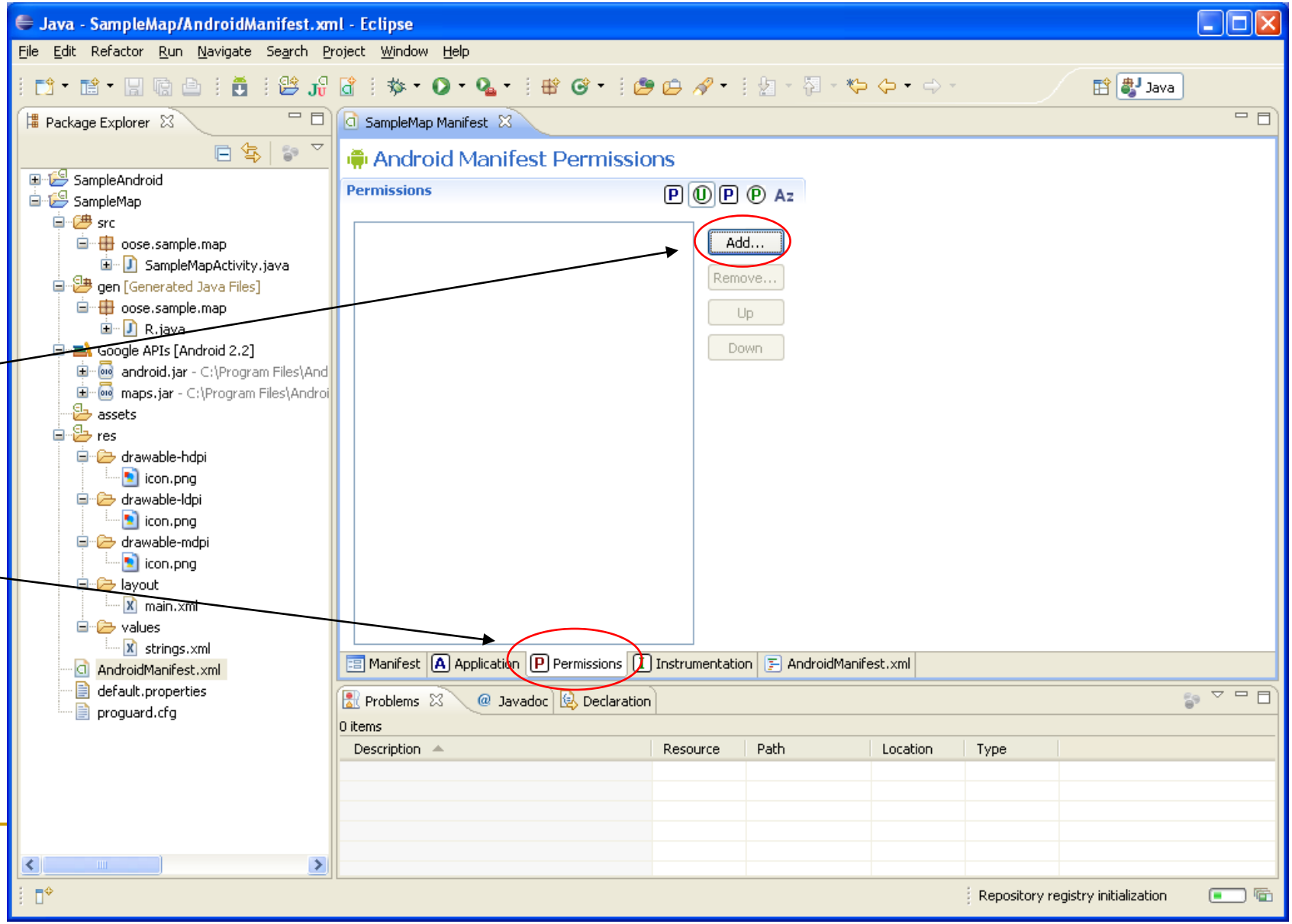
- Select ‘Add’ under ‘Uses Library’ (last slide)
- Then select ‘Uses Library at this prompt
- Set name as: com.google.android.maps (next slide) and save



Maps Example (6) – Manifest (4)

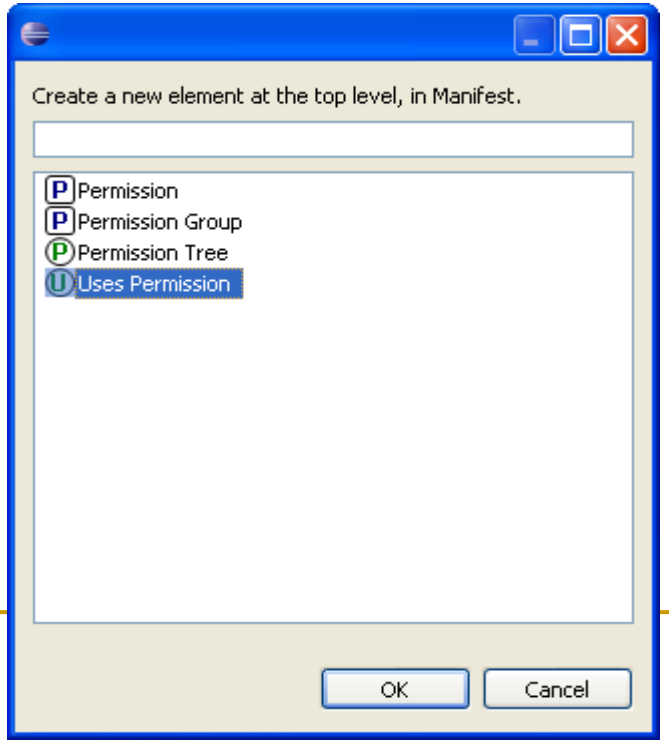


Maps Example (7) – Manifest (5)

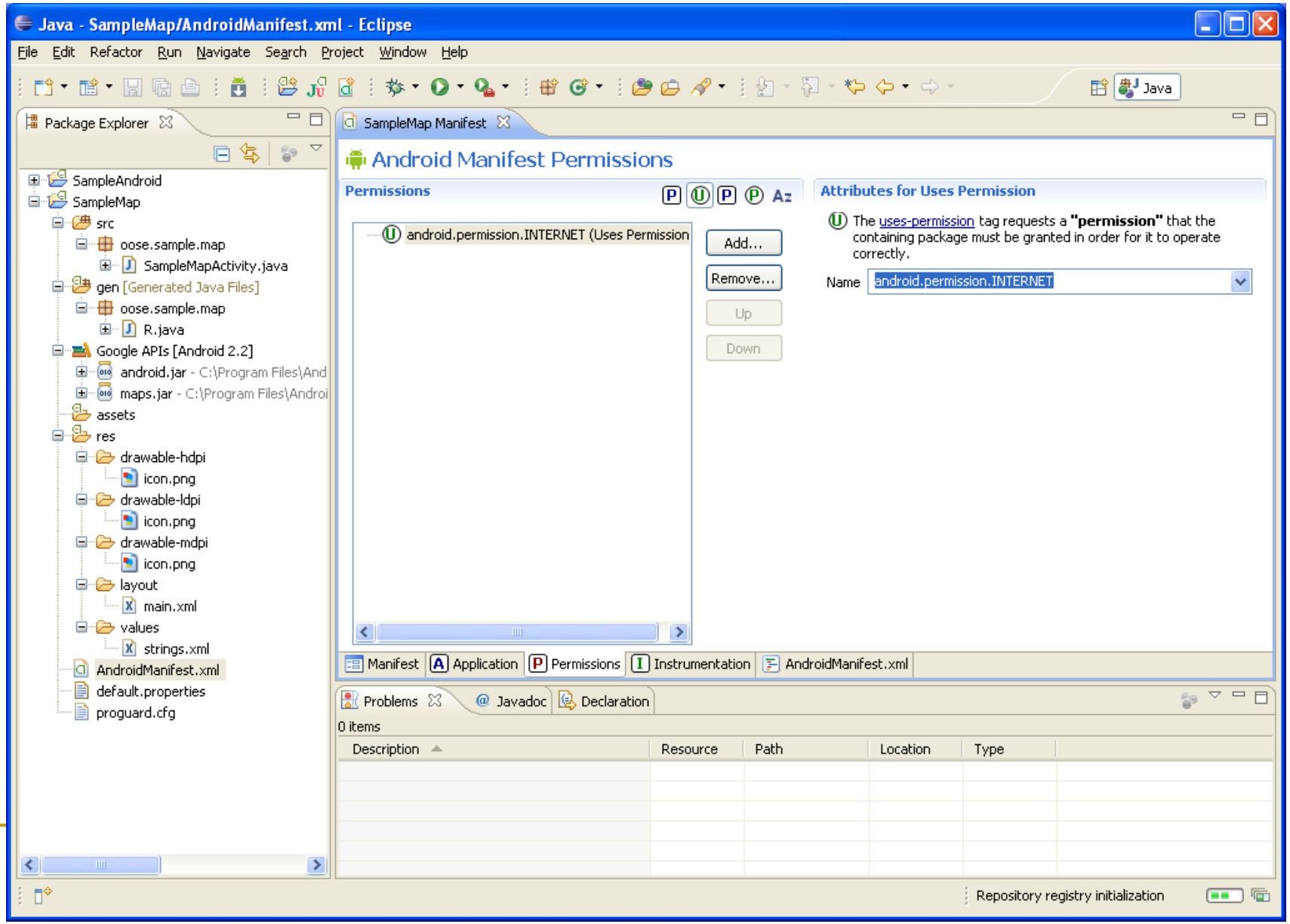


Maps Example (8) – Manifest (6)

- Select ‘Permissions’ and then ‘Add’ (last slide)
- Select ‘Uses Permissions’ at this prompt
- Set name to: android.permission.INTERNET and save (next slide)



Maps Example (9) – Manifest (7)



Maps Example (10) – Maps API Key (1)

- All Android applications need to be signed
  - The debug mode signs for you with special debug certificate
- All MapView elements in map applications need to have an API key associated with them
  - That key must be registered with the certificate used to sign the app
- When releasing app, need to sign with a release certificate and get a new API Key

Maps Example (11) – Maps API Key (2)

- For debug mode, get the MD5 fingerprint of the debug certificate
  - Locate the 'keystore'
    - Windows Vista: C:\Users\<user>\.android\debug.keystore
    - Windows XP: C:\Documents and Settings\<user>\.android\debug.keystore
    - OS X and Linux: ~/.android/debug.keystore
  - Use Keytool (comes with Java, in the bin directory with the other Java tools, should put that dir on system PATH) to get fingerprint
    - keytool -list –v -alias androiddebugkey -keystore "<path_to_debug_keystore>" -storepass android -keypass android
      - If don't include –v option, then will probably get only 1 fingerprint, and if it's not MD5, then need –v (Java 7 needs –v)
    - Extract the MD5 fingerprint, SHA will not work unfortunately
- Go to https://code.google. com/android/maps-api-signup.html , agree to terms and paste MD5 fingerprint, you will then be given an API Key

Maps Example (12)

- Need to put MapView tag in XML
  - com.google.android.maps.MapView
  - MapView is the basic view that represents a Google Map display
  - Must include API Key in XML, inside a layout
    - <com.google.android.maps.MapView
      android:id="@+id/mapview"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:clickable="true"
      android:apiKey="<api key>"/>
- Maps API Reference
  - http://code.google.com/android/add-ons/google-apis/reference/index.html

Thanks

- Larry Walters for insights and wonderful compilation of ideas