
Android Tutorial

Larry Walters

OOSE Fall 2011

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

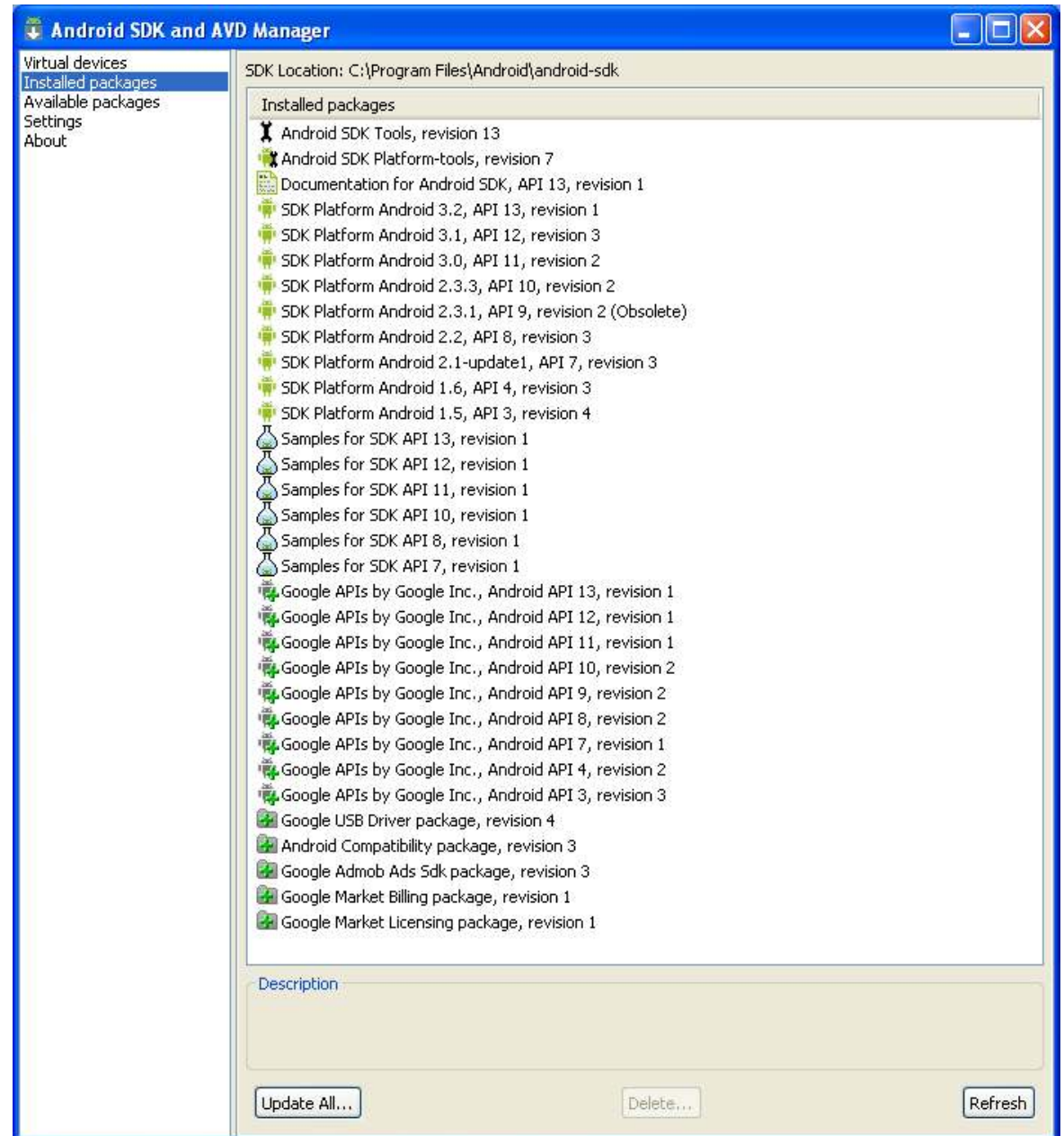
- Phone
 - Eclipse (<http://www.eclipse.org/downloads/>)
 - Android Plugin (ADT)
 - Android SDK (<http://developer.android.com/sdk/index.html>)
 - Install everything except Additional SDK Platforms, unless you want to
 - Windows Users: may need to install Motorola Driver directly ([http://www.motorola.com/Support/US-EN/Support-Homepage/Software and Drivers/USB-and-PC-Charging-Drivers](http://www.motorola.com/Support/US-EN/Support-Homepage/Software%20and%20Drivers/USB-and-PC-Charging-Drivers))
-

Android SDK

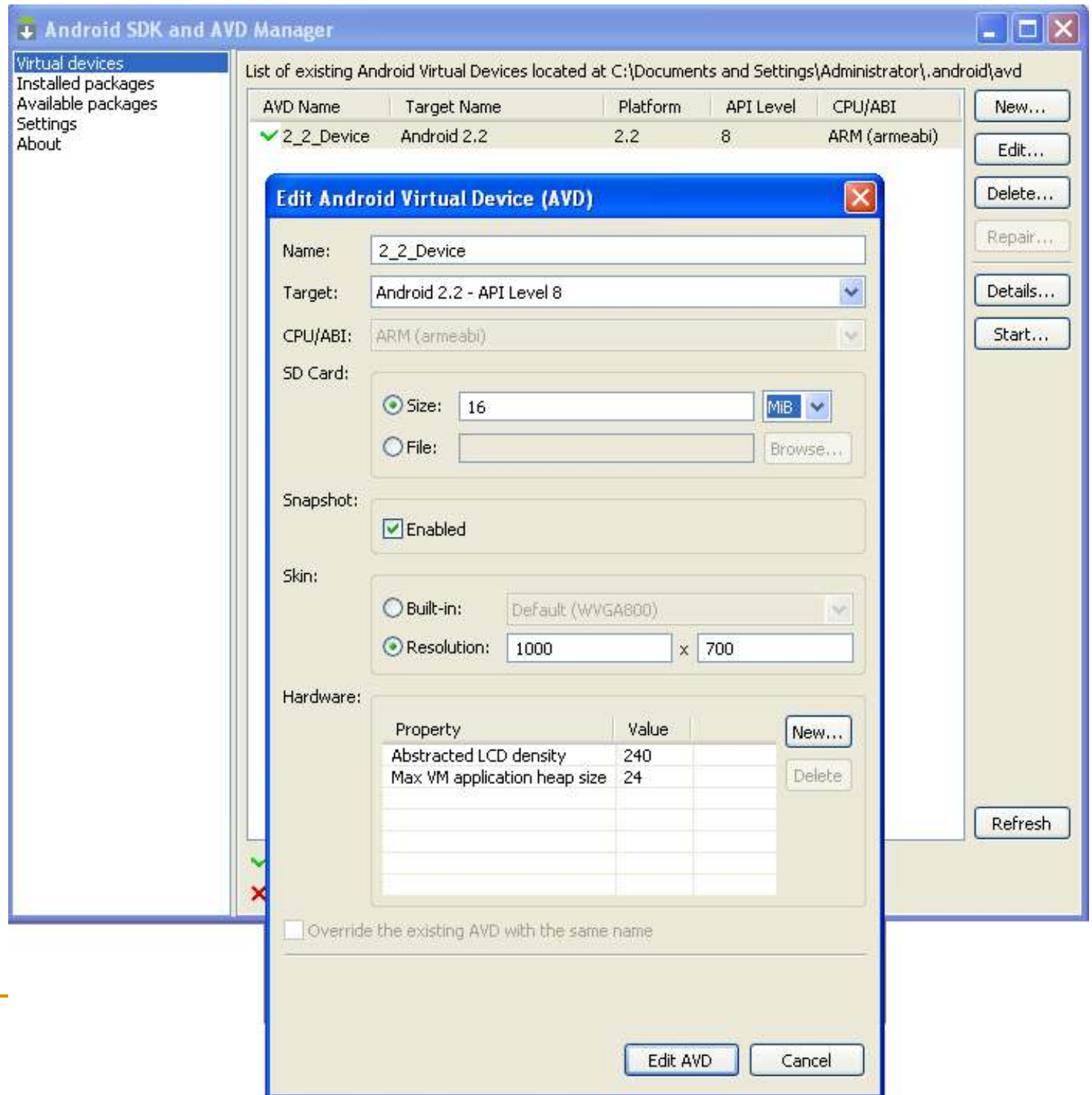
- Once installed open the SDK Manager
- Install the desired packages
- Create an Android Virtual Device (AVD)



SDK Manager



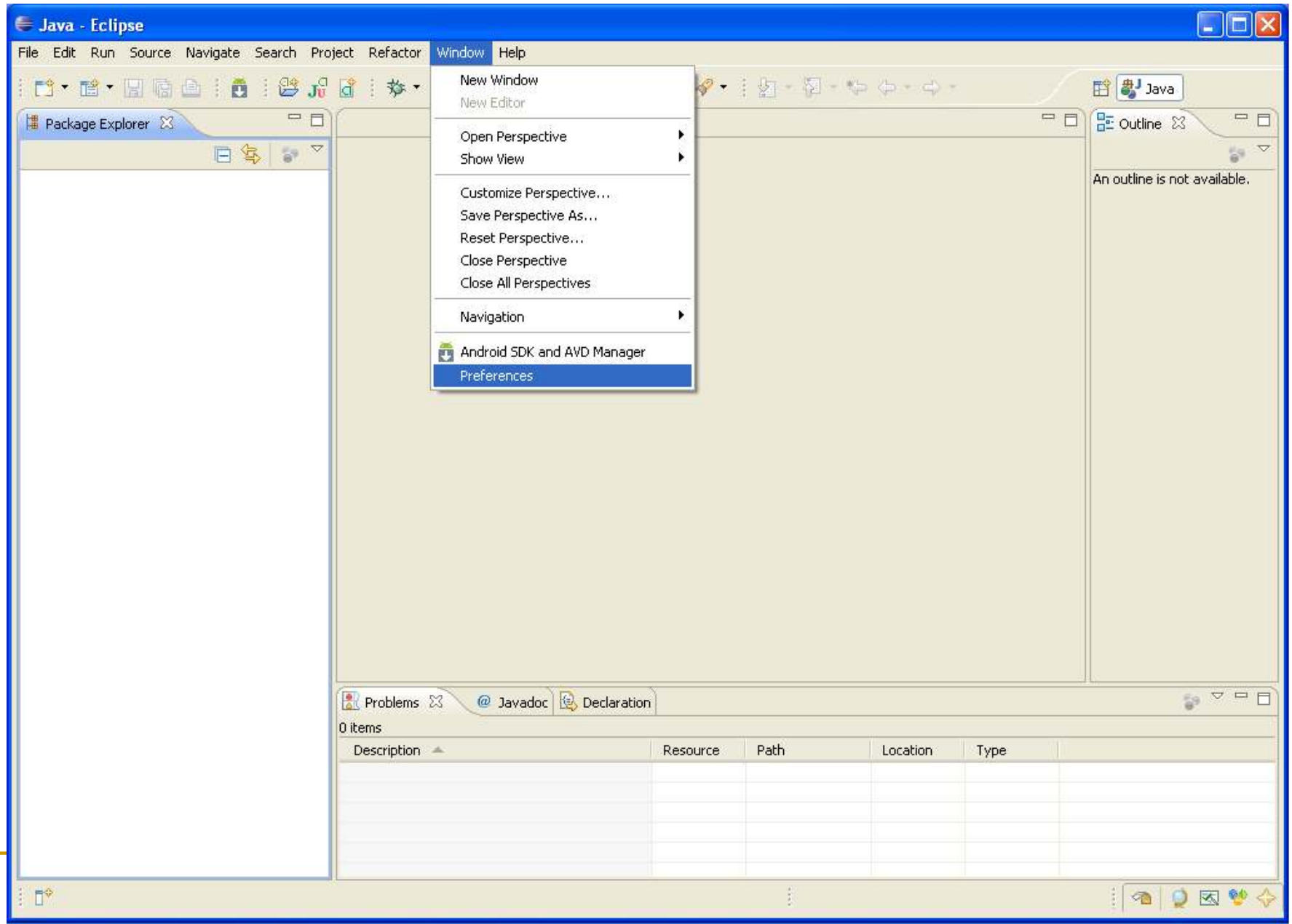
AVD



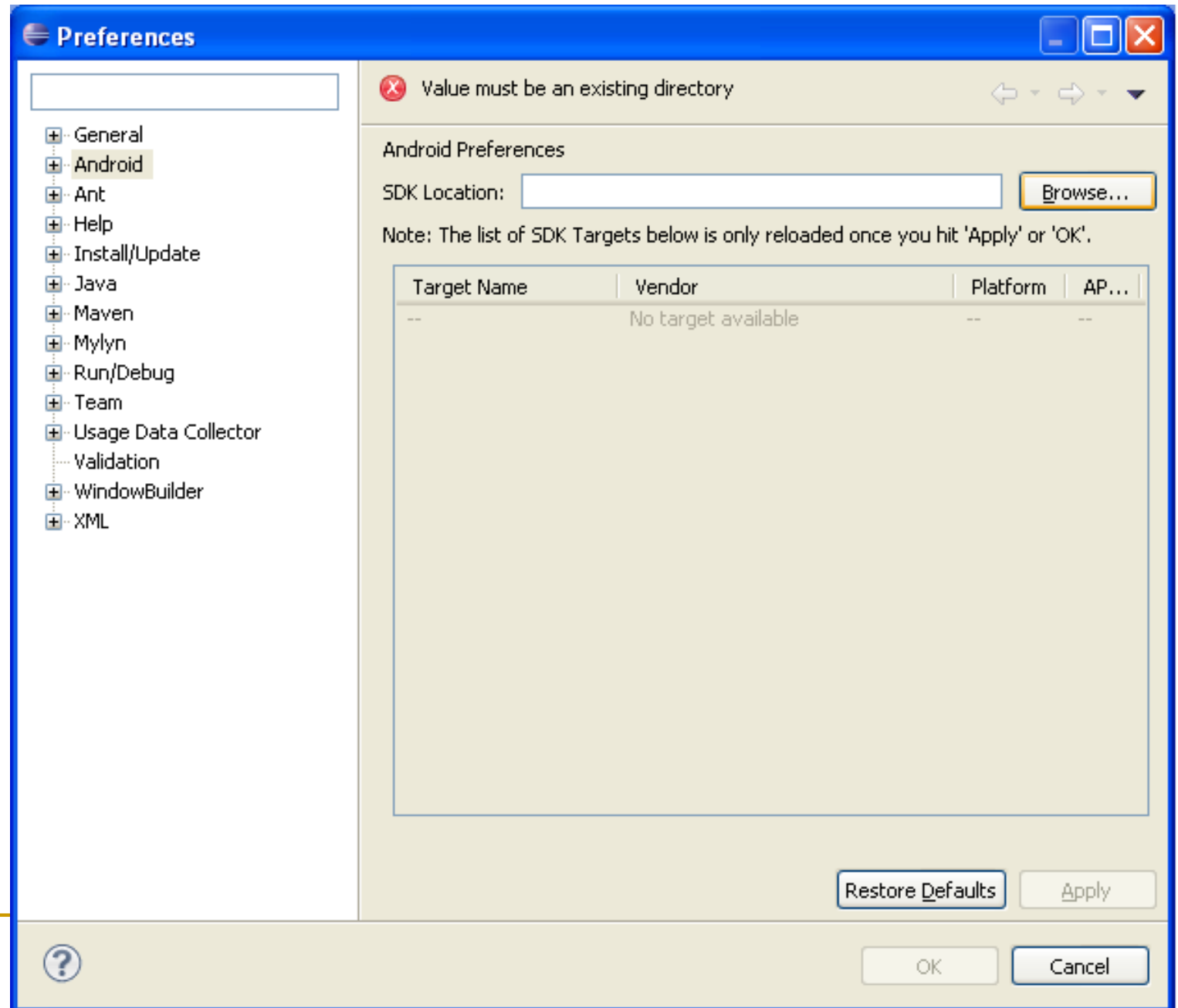
ADT Plugin (1)

- 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)
-

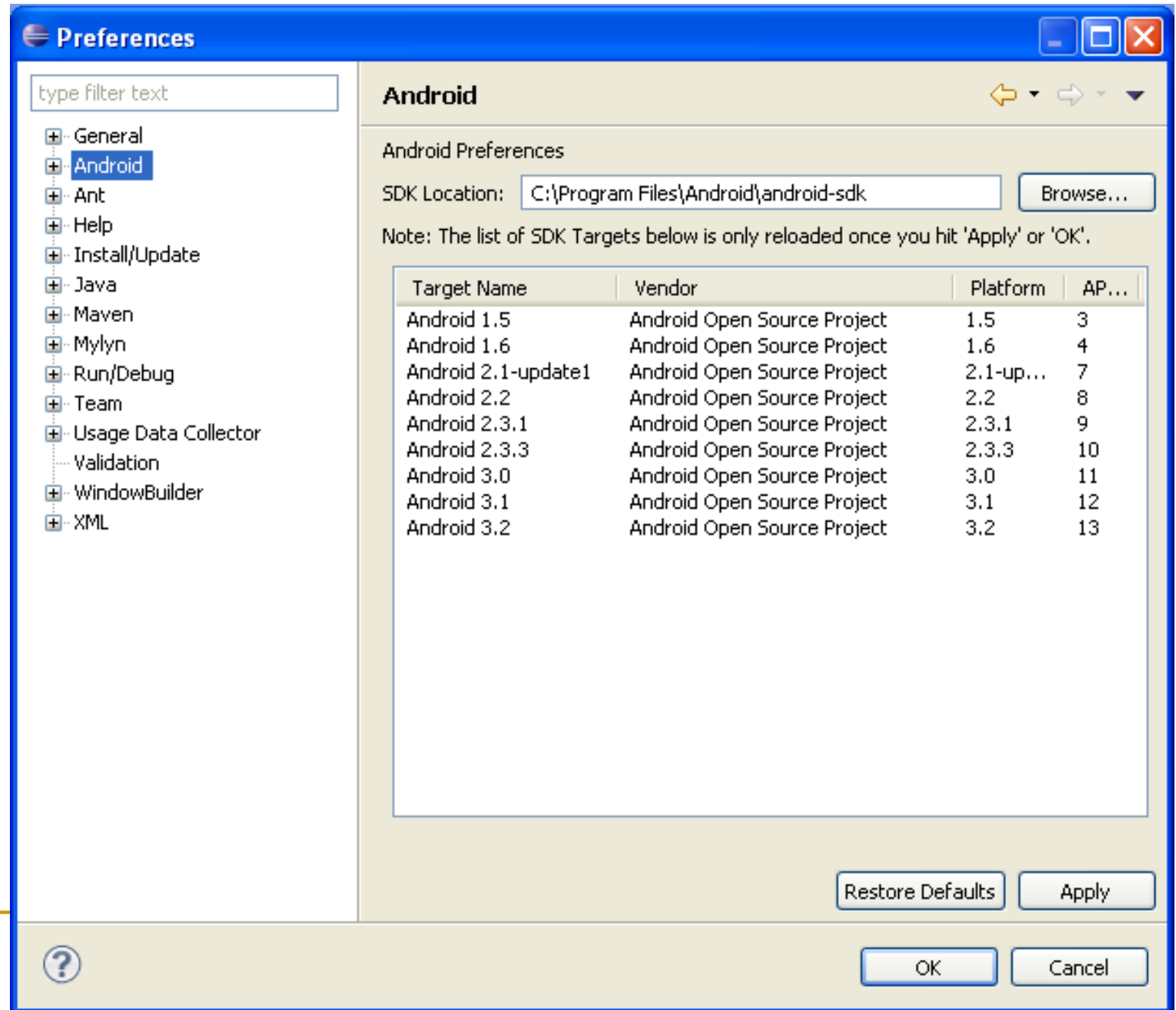
ADT Plugin (2)



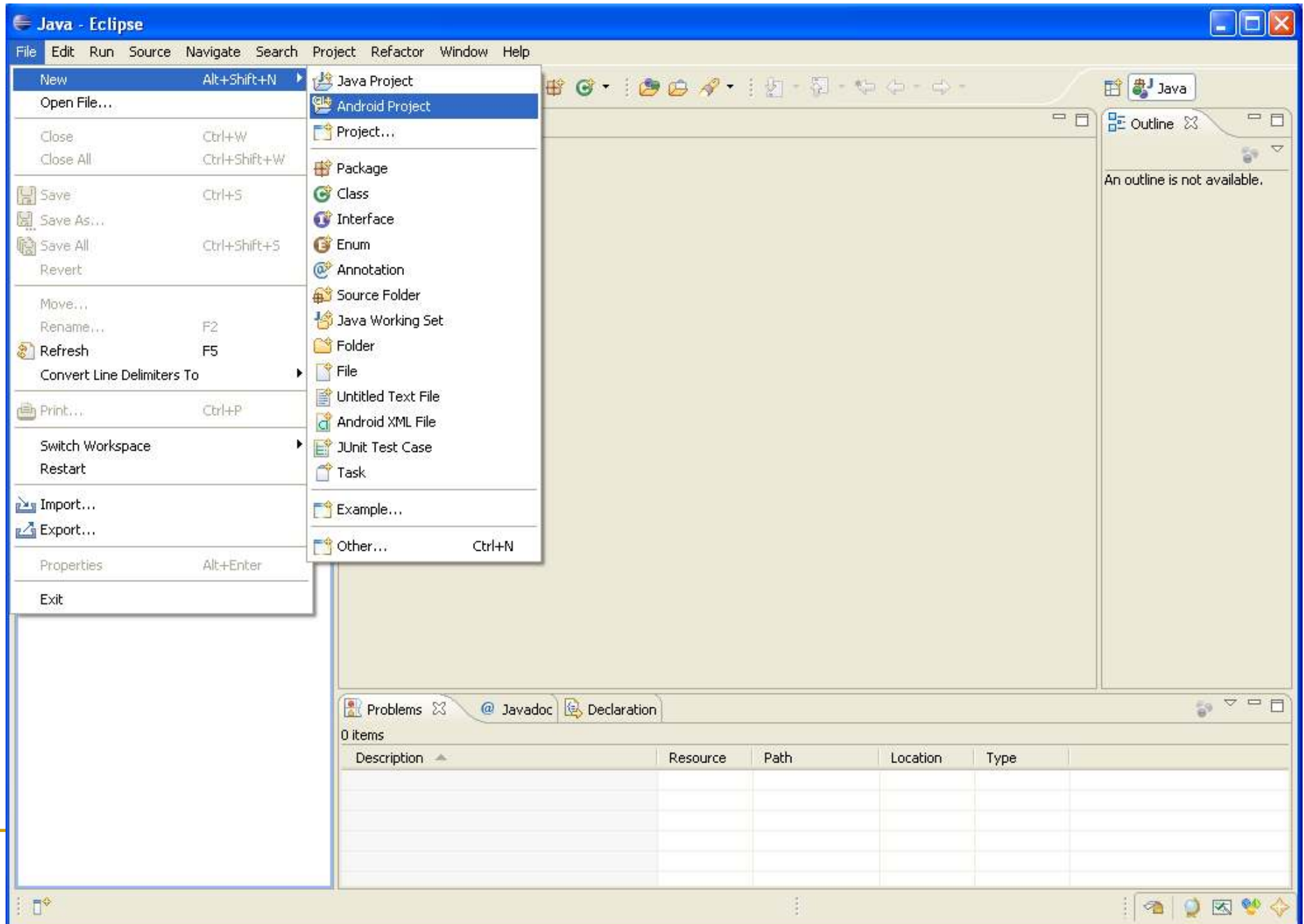
ADT Plugin (3)



ADT Plugin (4)



Creating a Project (1)



Creating a Project (2)

Need
the
items
circled

Then
click
Finish

New Android Project
Creates a new Android Project resource.

Project name:

Contents

- Create new project in workspace
- Create project from existing source
- Use default location

Location:

- Create project from existing sample

Samples:

Build Target

Target Name	Vendor	Platform	API Level
<input type="checkbox"/> Android 1.5	Android Open Source Project	1.5	3
<input type="checkbox"/> Android 1.6	Android Open Source Project	1.6	4
<input type="checkbox"/> Android 2.1-update1	Android Open Source Project	2.1-update1	7
<input checked="" type="checkbox"/> Android 2.2	Android Open Source Project	2.2	8
<input type="checkbox"/> Android 2.3.1	Android Open Source Project	2.3.1	9
<input type="checkbox"/> Android 2.3.3	Android Open Source Project	2.3.3	10
<input type="checkbox"/> Android 3.0	Android Open Source Project	3.0	11
<input type="checkbox"/> Android 3.1	Android Open Source Project	3.1	12
<input type="checkbox"/> Android 3.2	Android Open Source Project	3.2	13

Standard Android platform 3.2

Properties

Application name:

Package name:

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
-

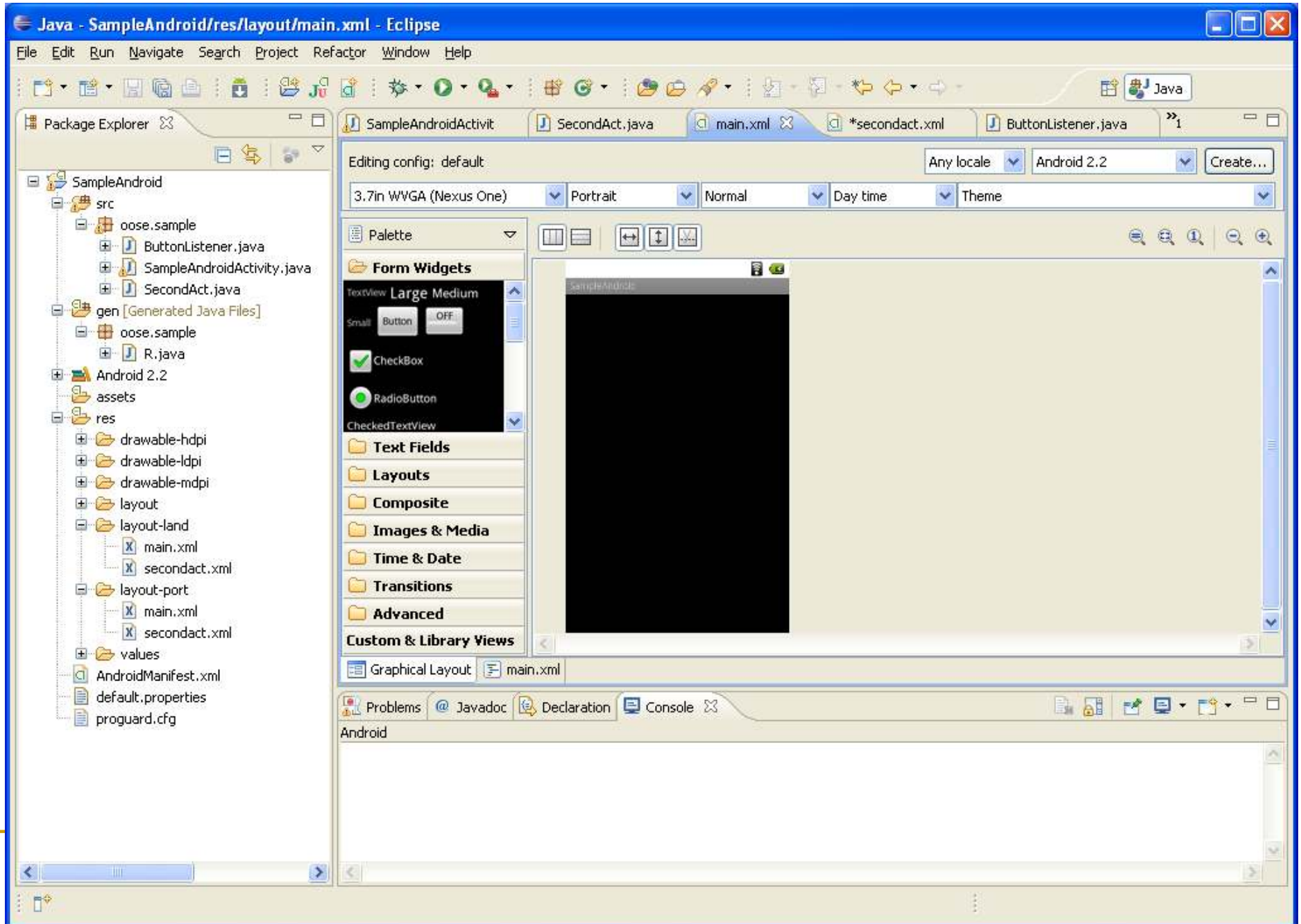
R Class

- 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
-

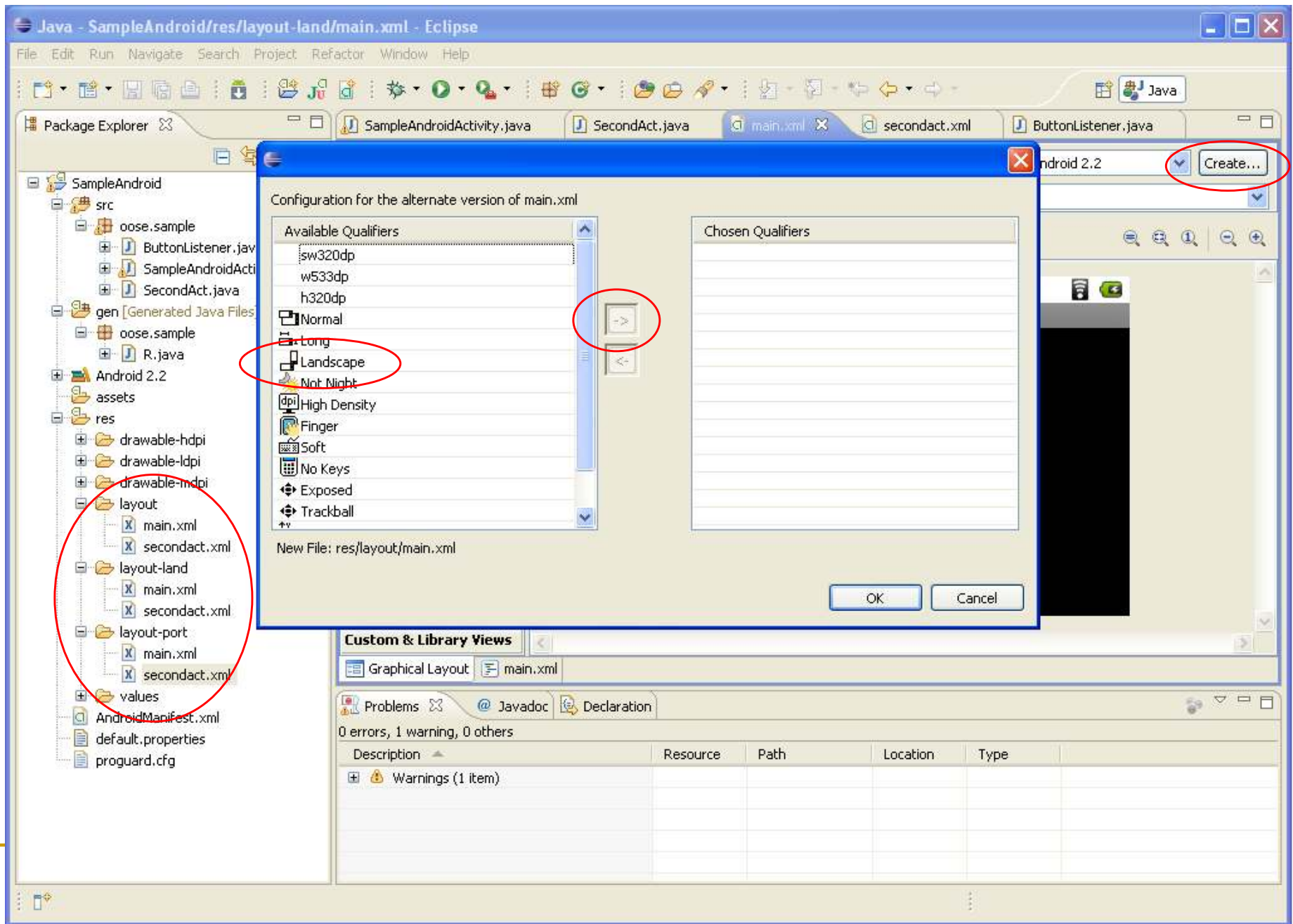
Layouts (2)



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)



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

The screenshot shows the Eclipse IDE interface for editing the AndroidManifest.xml file. The Package Explorer on the left shows the project structure, including the src directory with files like ButtonListener.java, SampleAndroidActivity.java, and SecondAct.java. The main editor displays the Manifest editor with various activity attributes like Label, Icon, Description, Permission, Process, Task affinity, Allow task reparenting, Has code, and Persistent. Below the main editor, the 'Application Nodes' panel shows a tree view of activities, with 'Add...' and 'Remove...' buttons circled in red. The 'Attributes for Activity' panel shows the 'Name*' attribute set to 'SecondAct', also circled in red. The bottom panel shows the Problems view with 0 errors and 1 warning.

Description	Resource	Path	Location	Type
Warnings (1 item)				

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

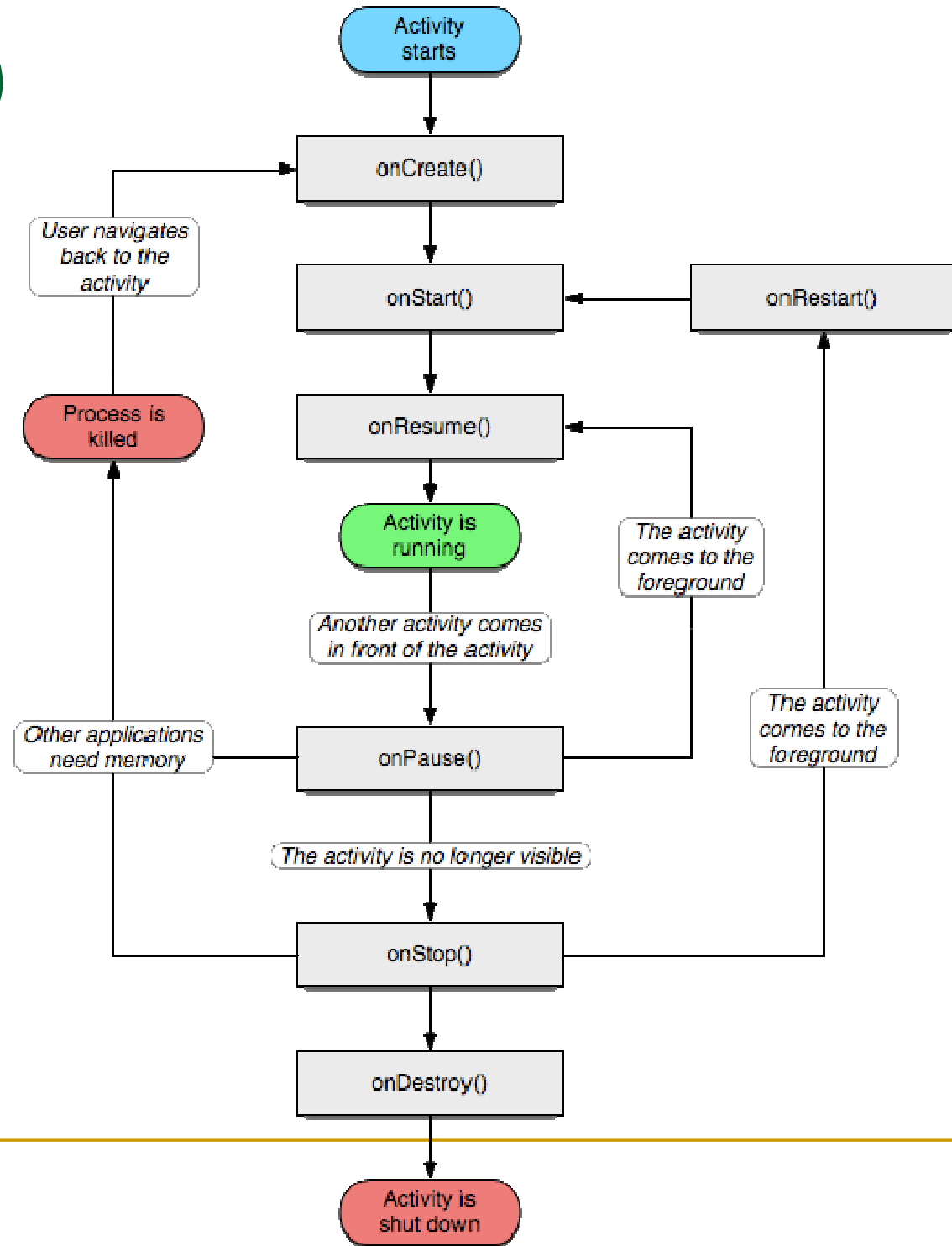
- Android in a nutshell:

- <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
-

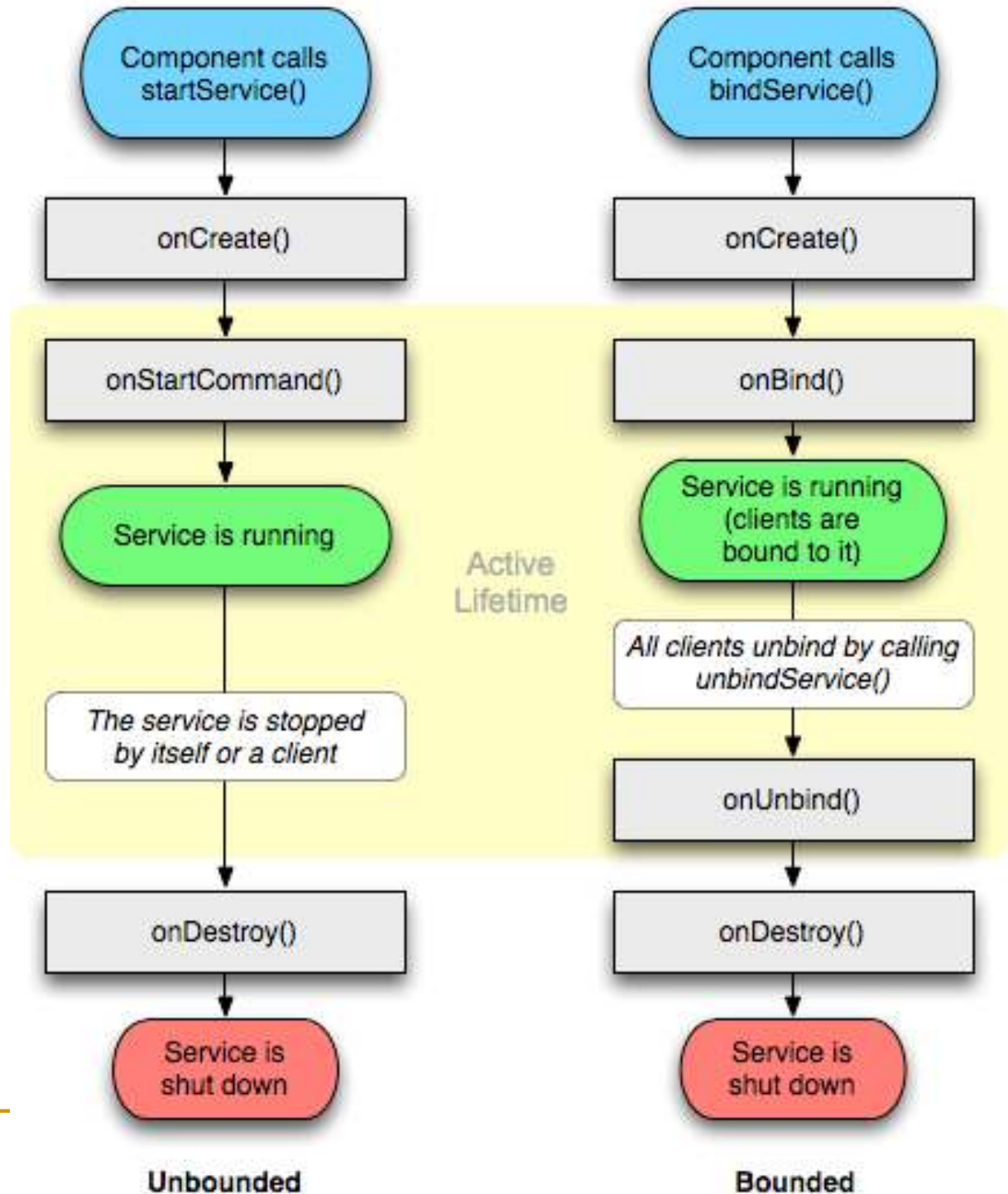
Activities (2)



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
-

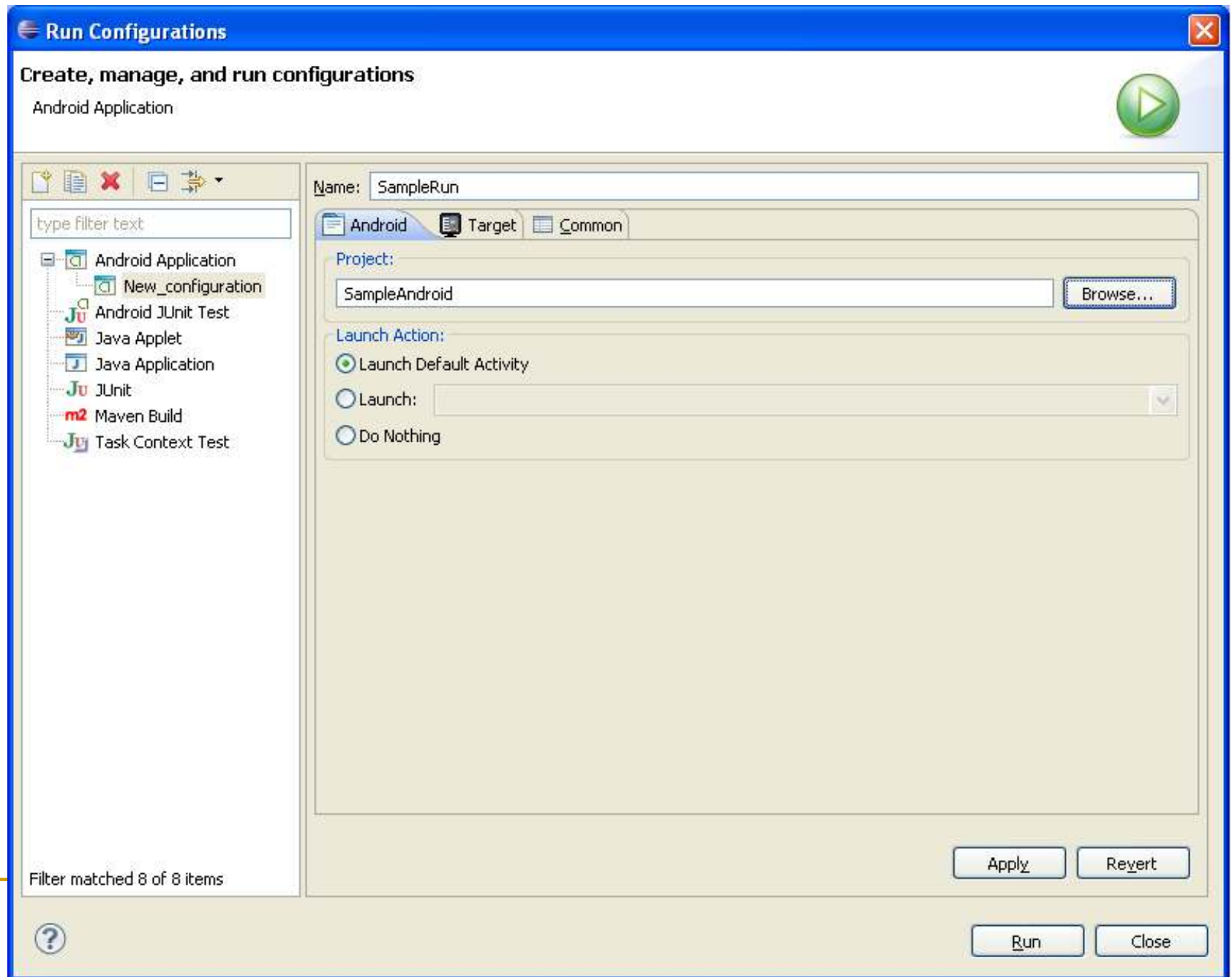
Services (2)



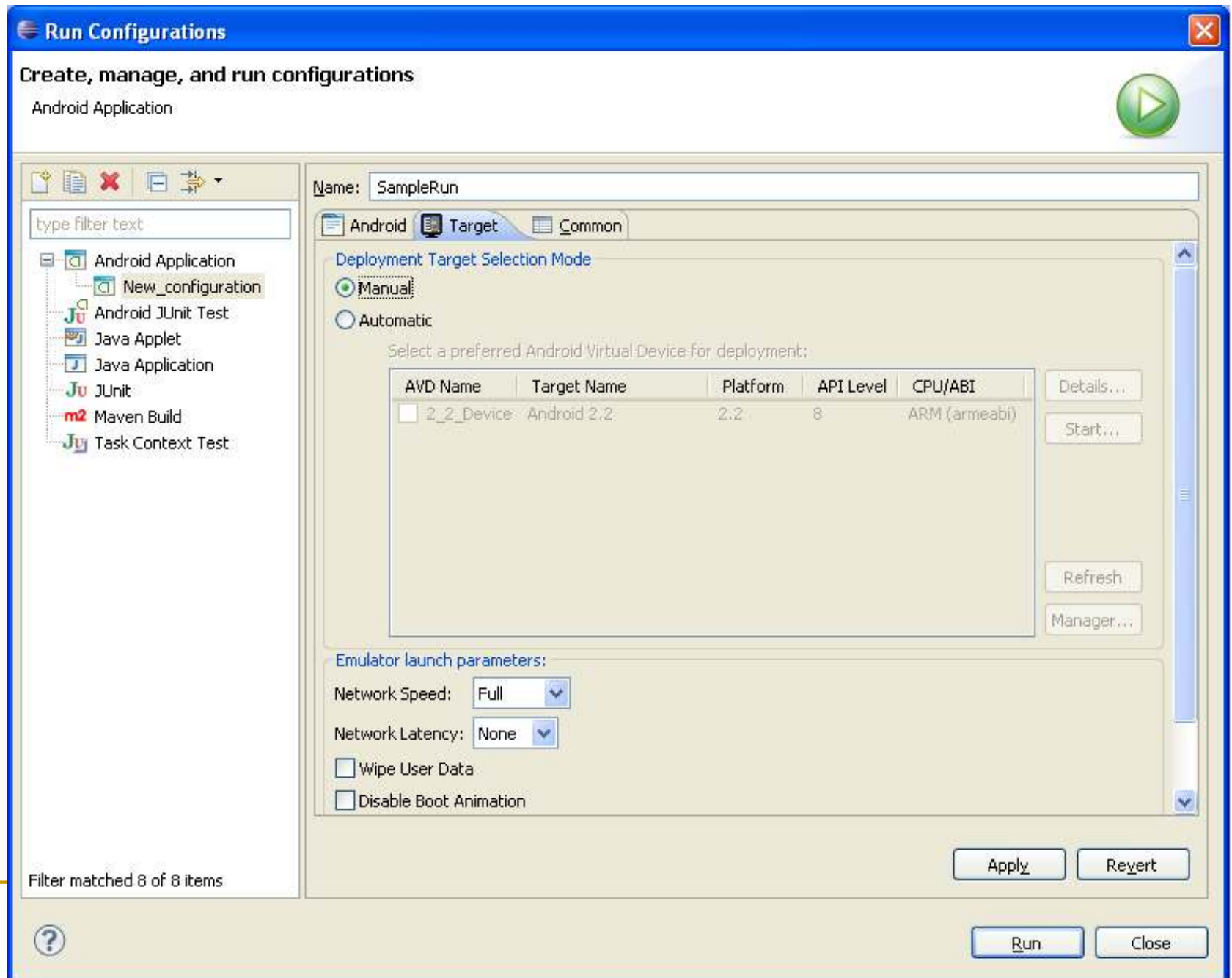
Running in Eclipse (1)

- 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
-

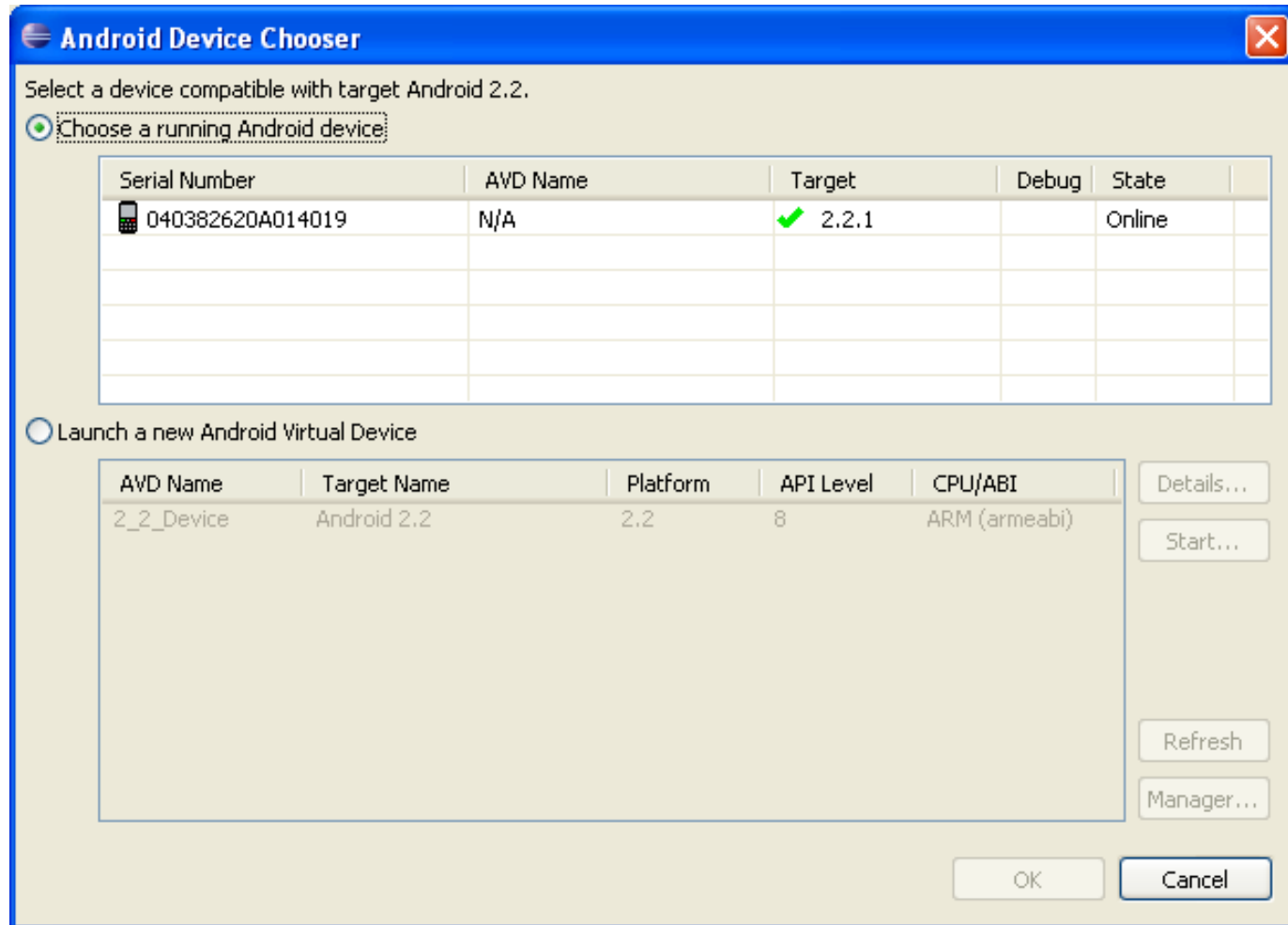
Running in Eclipse (2)



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>

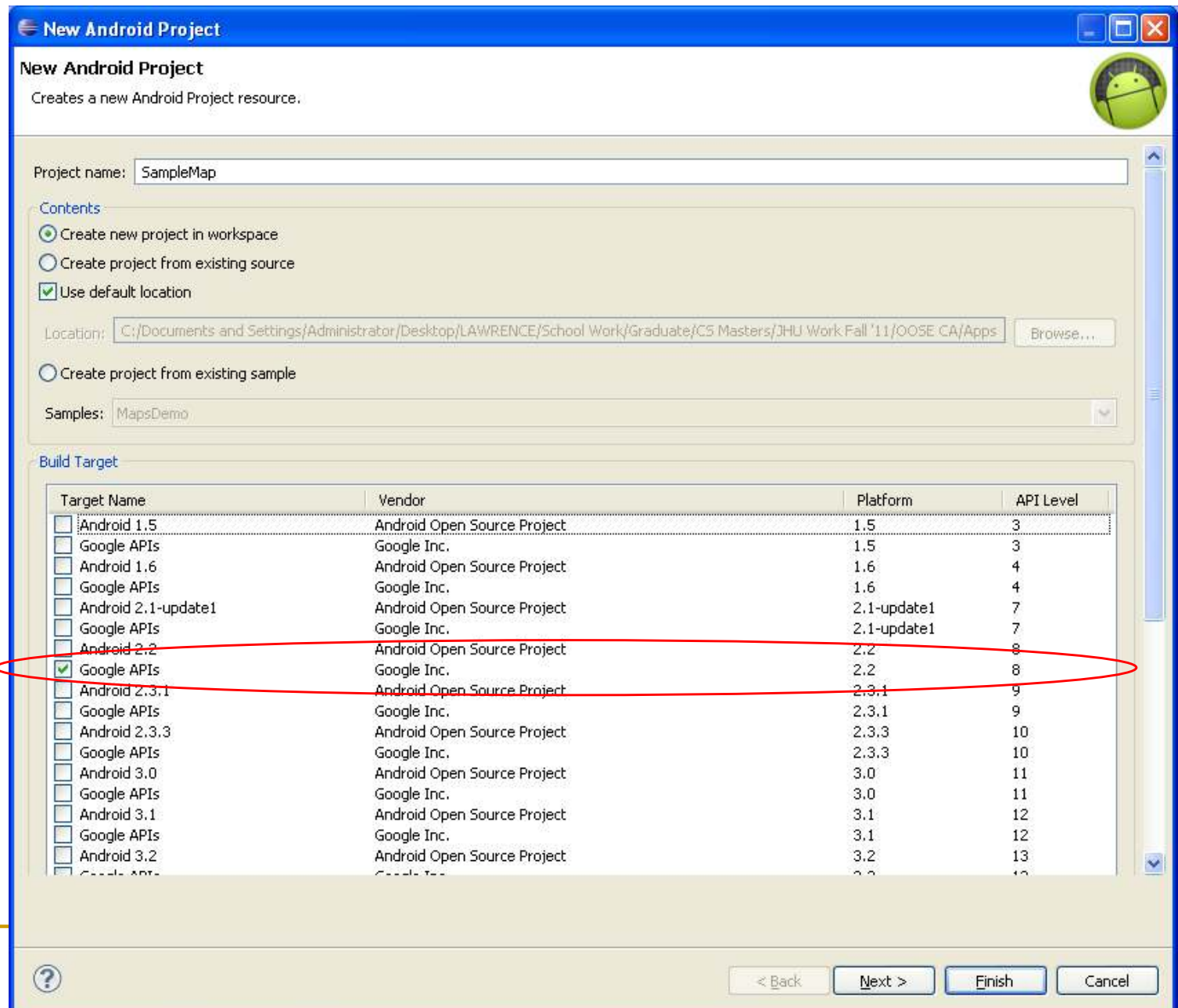
Screen Shots

- 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)

The screenshot shows the Eclipse IDE with the AndroidManifest.xml file open. The Package Explorer on the left shows the project structure, with annotations 1 and 2 pointing to the 'src' and 'res' folders respectively. The main editor shows the Manifest editor with various attributes and the Application Nodes section. The 'Add...' button in the Application Nodes section is circled in red.

Annotation 1 points to the 'src' folder in the Package Explorer.

Annotation 2 points to the 'res' folder in the Package Explorer.

The Manifest editor shows the following attributes:

- Label: @string/app_name
- Icon: @drawable/icon
- Description:
- Permission:
- Process:
- Task affinity:
- Allow task reparenting:
- Has code:
- Persistent:

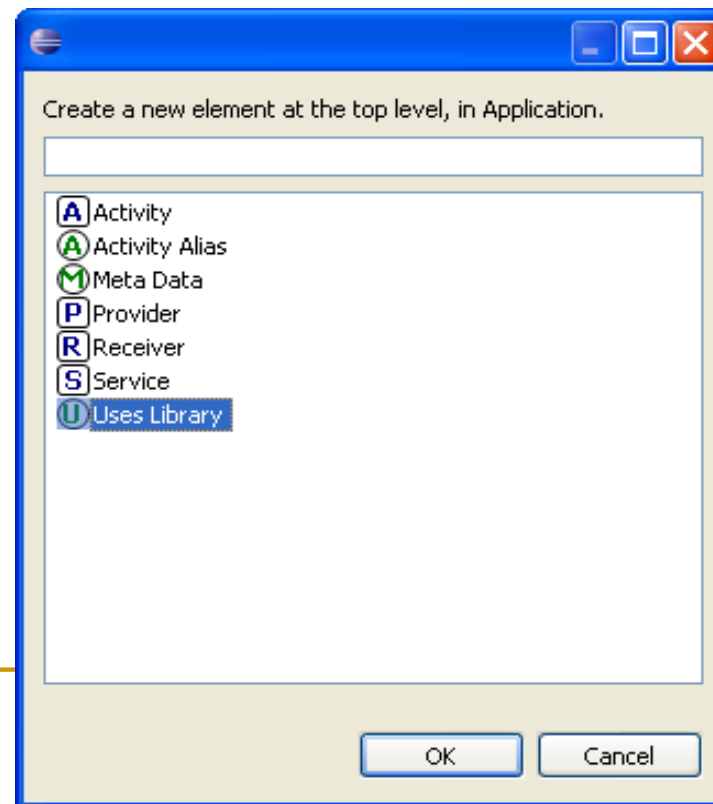
The Application Nodes section shows the following nodes:

- .SampleMapActivity

The 'Add...' button in the Application Nodes section is circled in red.

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)

The screenshot displays the Eclipse IDE interface for editing the `AndroidManifest.xml` file of a project named `SampleMap`. The Package Explorer on the left shows the project structure, including source files, generated files, Google APIs, assets, and resources. The central editor shows the manifest file with various attributes and their values, such as `Label` set to `@string/app_name` and `Icon` set to `@drawable/icon`. The bottom panel shows the `Application Nodes` section with a single node `com.google.android.maps (Uses Library)` selected, and the `Attributes for com.google.android.maps (Uses Library)` section showing the `com.google.android.maps` attribute.

Java - SampleMap/AndroidManifest.xml - Eclipse

File Edit Refactor Run Navigate Search Project Window Help

Package Explorer

- SampleAndroid
- SampleMap
 - src
 - oos.sample.map
 - SampleMapActivity.java
 - gen [Generated Java Files]
 - oos.sample.map
 - R.java
 - Google APIs [Android 2.2]
 - android.jar - C:\Program Files\And
 - maps.jar - C:\Program Files\Androi
 - assets
 - res
 - drawable-hdpi
 - icon.png
 - drawable-ldpi
 - icon.png
 - drawable-mdpi
 - icon.png
 - layout
 - main.xml
 - values
 - strings.xml
 - AndroidManifest.xml
 - default.properties
 - proguard.cfg

SampleMap Manifest

Label Browse... Vm safe mode

Icon Browse... Manage space activity Browse...

Description Browse... Allow clear user data

Permission Test only

Process Browse... Backup agent Browse...

Task affinity Browse... Allow backup

Allow task reparenting Kill after restore

Has code Restore needs application

Persistent Restore any version

Application Nodes Az

com.google.android.maps (Uses Library)

Attributes for com.google.android.maps (Uses Library)

The "uses-libraries" specifies a shared library that this package requires to be linked against.

Name

Required

Manifest Application Permissions Instrumentation AndroidManifest.xml

Problems @ Javadoc Declaration

0 items

Description	Resource	Path	Location	Type

Repository registry initialization

Maps Example (7) – Manifest (5)

The screenshot displays the Eclipse IDE interface for editing an Android Manifest file. The Package Explorer on the left shows the project structure, with the 'AndroidManifest.xml' file selected. The main editor window shows the 'Android Manifest Permissions' dialog, which is used to manage the permissions declared in the manifest. The 'Add...' button is circled in red, and an arrow labeled '2' points to it. The 'Permissions' tab is also circled in red, and an arrow labeled '1' points to it. The bottom editor bar shows the 'Permissions' tab selected, and the 'Problems' view is empty.

2

1

Description	Resource	Path	Location	Type
0 items				

Repository registry initialization

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)

The screenshot shows the Eclipse IDE interface for editing an Android Manifest file. The Package Explorer on the left shows the project structure for 'SampleMap', including source files, generated files, and resources. The main editor displays the 'Android Manifest Permissions' section, where the 'android.permission.INTERNET (Uses Permission)' is listed. The 'Attributes for Uses Permission' section is also visible, showing the 'Name' attribute set to 'android.permission.INTERNET'. The bottom of the IDE shows the 'Problems' view, which is currently empty.

Manifest Application Permissions Instrumentation AndroidManifest.xml

Description	Resource	Path	Location	Type
0 items				

Repository registry initialization

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\\.android\debug.keystore
 - Windows XP: C:\Documents and Settings\\.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>
-

---

# Acknowledgements

- Android Developer's Website
    - Activity and Service life-cycle flow charts
    - Tons of other Android info
  - Google Maps API external library
    - <http://code.google.com/android/add-ons/google-apis/maps-overview.html>
  - MightyPocket
    - <http://www.mightypocket.com/2010/08/android-screenshots-screen-capture-screen-cast/>
  - Numerous Forums & other developer sites, including:
    - <http://www.javacodegeeks.com/2011/02/android-google-maps-tutorial.html>
    - <http://efreedom.com/Question/1-6070968/Google-Maps-API-Directions>
    - <http://www.mail-archive.com/android-developers@googlegroups.com/msg28487.html>
    - <http://android.bigresource.com/> threads
    - <http://groups.google.com/group/android-developers> threads
    - Many <http://stackoverflow.com> threads
    - [http://www.anddev.org/google\\_driving\\_directions\\_-\\_mapview\\_overlaid-t826.html](http://www.anddev.org/google_driving_directions_-_mapview_overlaid-t826.html)
  - Zainan Victor Zhou – for advice and his own tutorial
-