MONASH University

Information Technology

FIT5183: Mobile and Distributed Computing Systems (MDCS)

Lecture 5B
Android Primer

Brief History

- 2005
  - Google acquires startup Android Inc. to start Android platform
  - Work on Dalvik VM begins
  - Dalvik is the process virtual machine (VM) that runs the applications on Android devices
- 2007
  - The OHA (Open Handset Alliance) was established
    - Developing open standards for mobile devices
  - They developed Android
  - Early look at SDK
- 2008
  - SDK 1.0 released in Sep 2008
  - Android released open source (Apache License)
  - Android Dev Phone 1 released which was a version of the HTC Dream

Android SDK Versions

- 2009
  - SDK 1.5 (Cupcake)
  - SDK 1.6 (Donut)
    - Support Wide VGA
  - SDK 2.0/2.0.1/2.1 (Eclair)
    - Revamped UI, browser
- 2010
  - Nexus One released to the public
  - SDK 2.2 (Froyo)
    - Flash support, tethering
  - SDK 2.3 (Gingerbread)
    - UI update, system-wide copy-paste

Versions

- 2011
  - Android 3.0 (Honeycomb) for tablets only
  - Android 3.1 (Honeycomb_MR1) and 3.2 (Honeycomb_MR2)
  - Android 4.0 (Ice Cream Sandwich)
- 2012
  - Android 4.1 API 16 (Jelly Bean), Android 4.2 API 17 (Jelly Bean)
- 2013
  - Android 4.3 API 18 (Jelly Bean), Android 4.4 API 19 (KitKat)
- 2014 and 2015
  - Android 5.0 API 21 (Lollipop), Android 5.1 API 22 (Lollipop)
- Android 6.0 API 23 (Marshmallow)

Android Studio

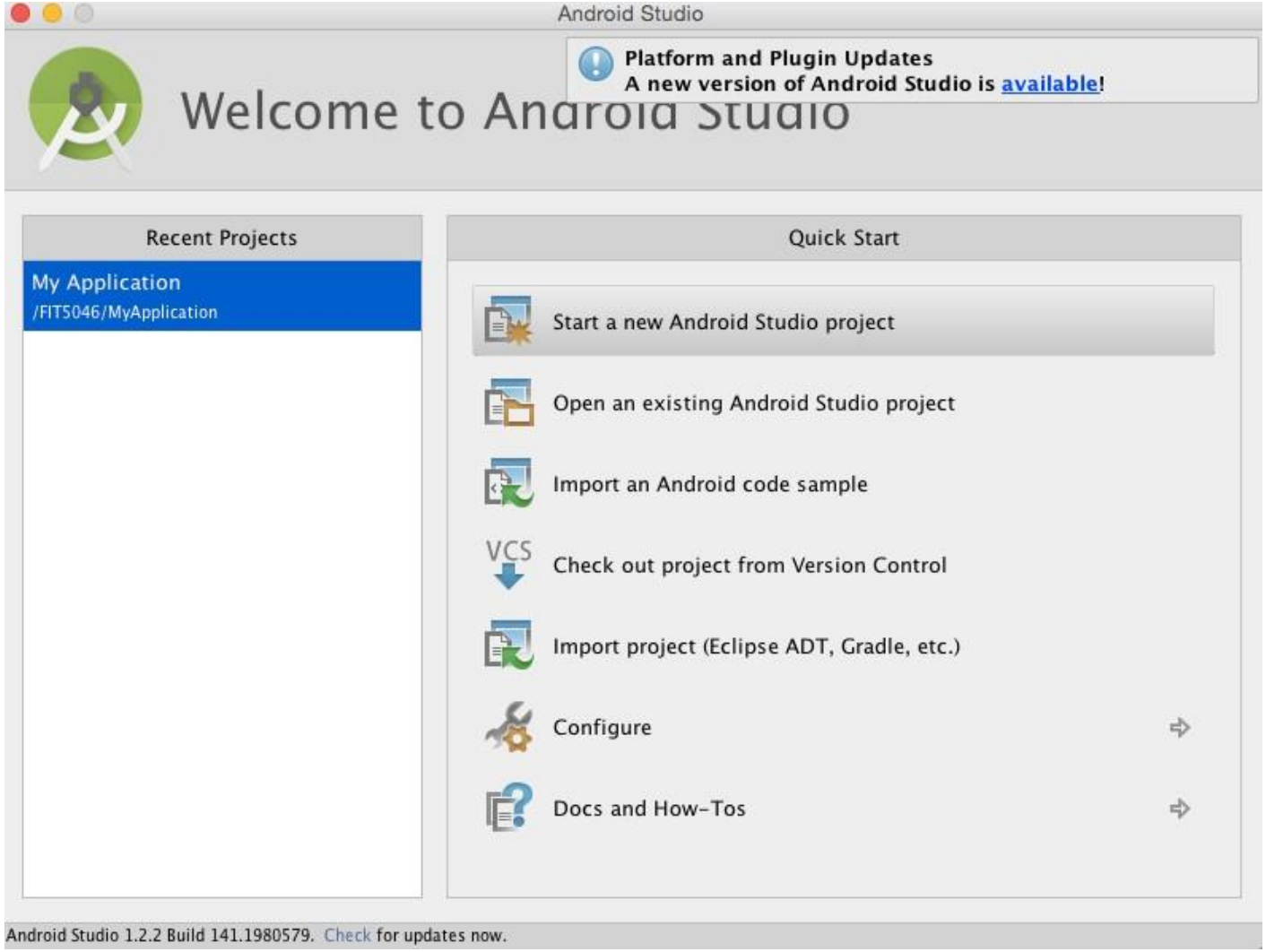
- An IDE for Android application development
  - First stable build released in 2014
- Android 5.0 (Lollipop) Platform
- Based on IntelliJ IDEA (Java IDE)
  - an intelligent code editor for code completion, refactoring, and code analysis
- Emulators for all shapes and sizes
- Enables creating multiple APKs for an Android app with different features using the same project
- Easy GitHub integration and code templates to build common app
- Built-in support for Google Cloud Platform
- Flexible Gradle-based build system
  - allows the way in which projects are built to be configured and managed

Gradle and Android Studio

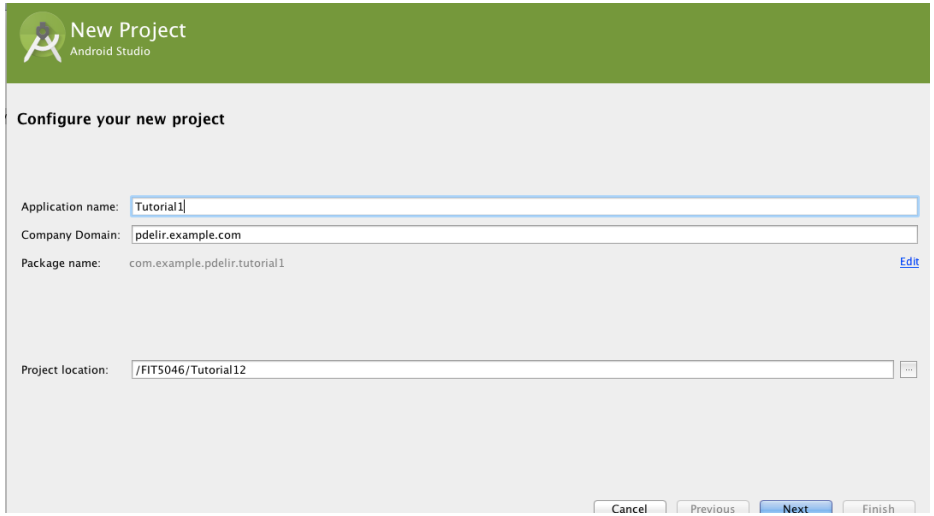
- Provides powerful features to building Android application projects
- A pre-defined set of sensible default configuration settings used unless they are overridden by settings in the build files (by developer to meet other needs)
- To address dependencies (local or remote)
- Local dependencies: A local dependency references an item that is present on the local file system of the computer system on which the build is being performed
  - e.g. a module within an Android Studio project which triggers an intent to load another module in the project.
  - Other examples of dependencies are libraries and JAR files on which the project depends in order to compile and run
- A remote dependency refers to an item that is present on a remote server
  - Handled using another project management tool named Maven
  - E.g. for a remote dependency in a Gradle build file using Maven, the dependency will be downloaded automatically from the server and included in the build process

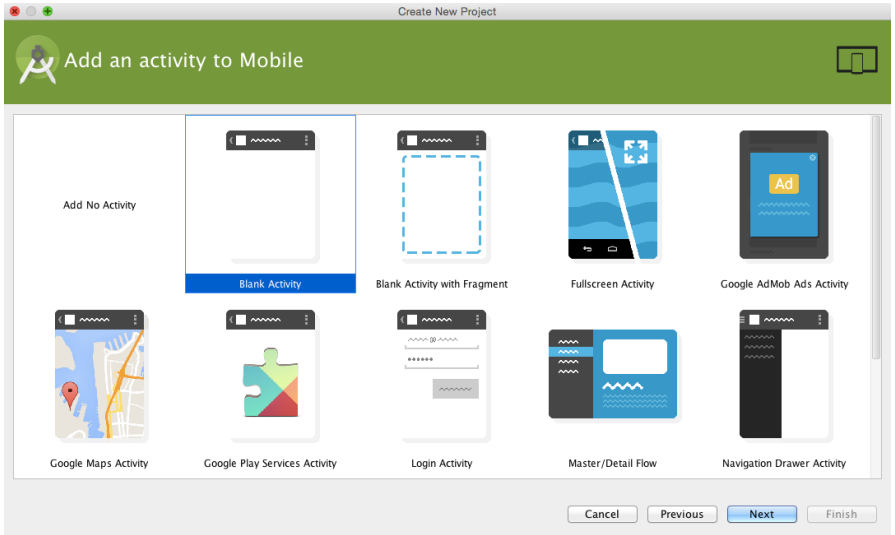
Android Environment and Sample Application

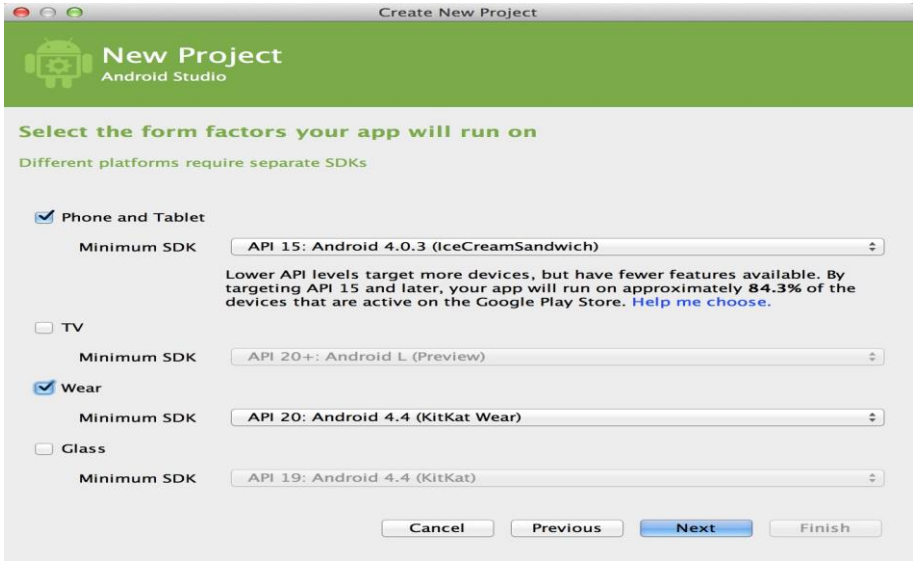
The first screen

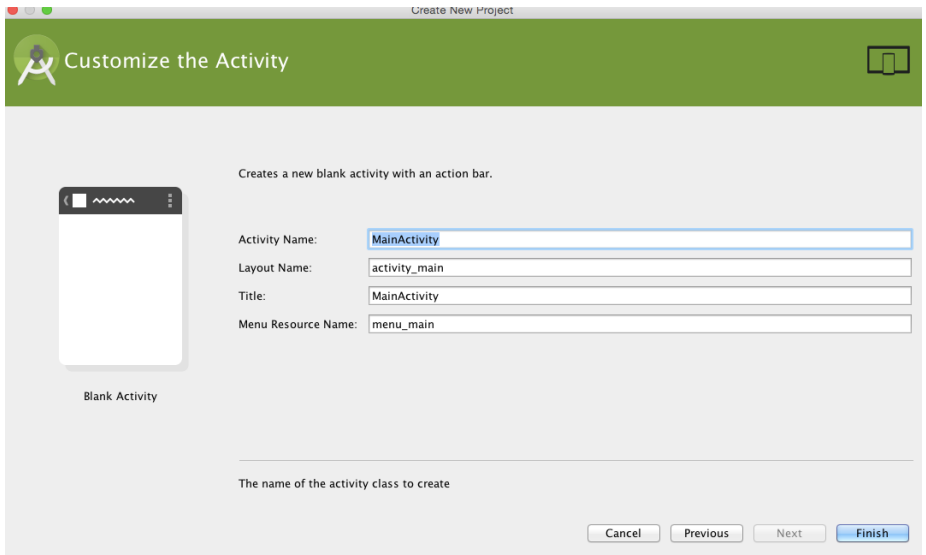


Create A New Android Application Project

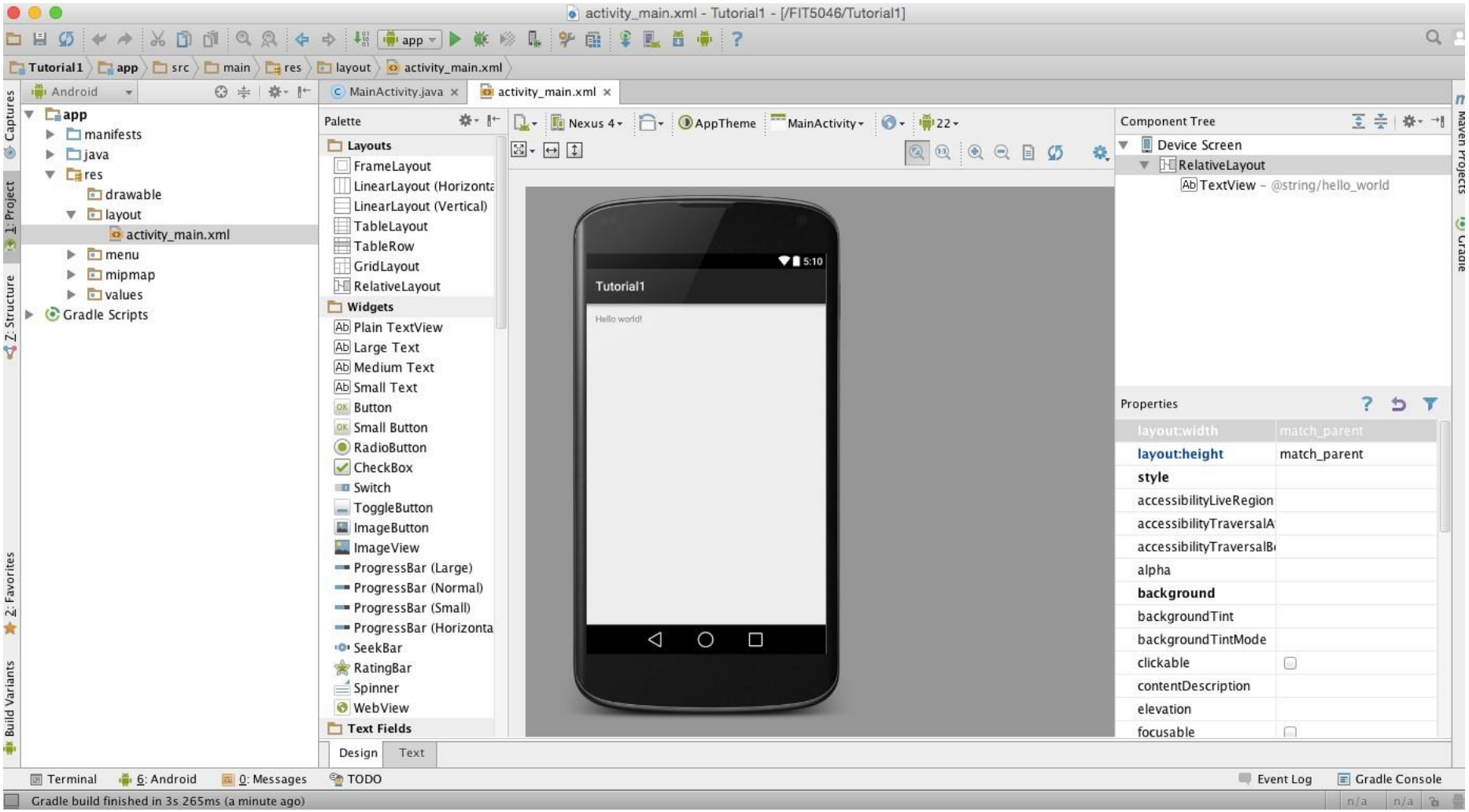




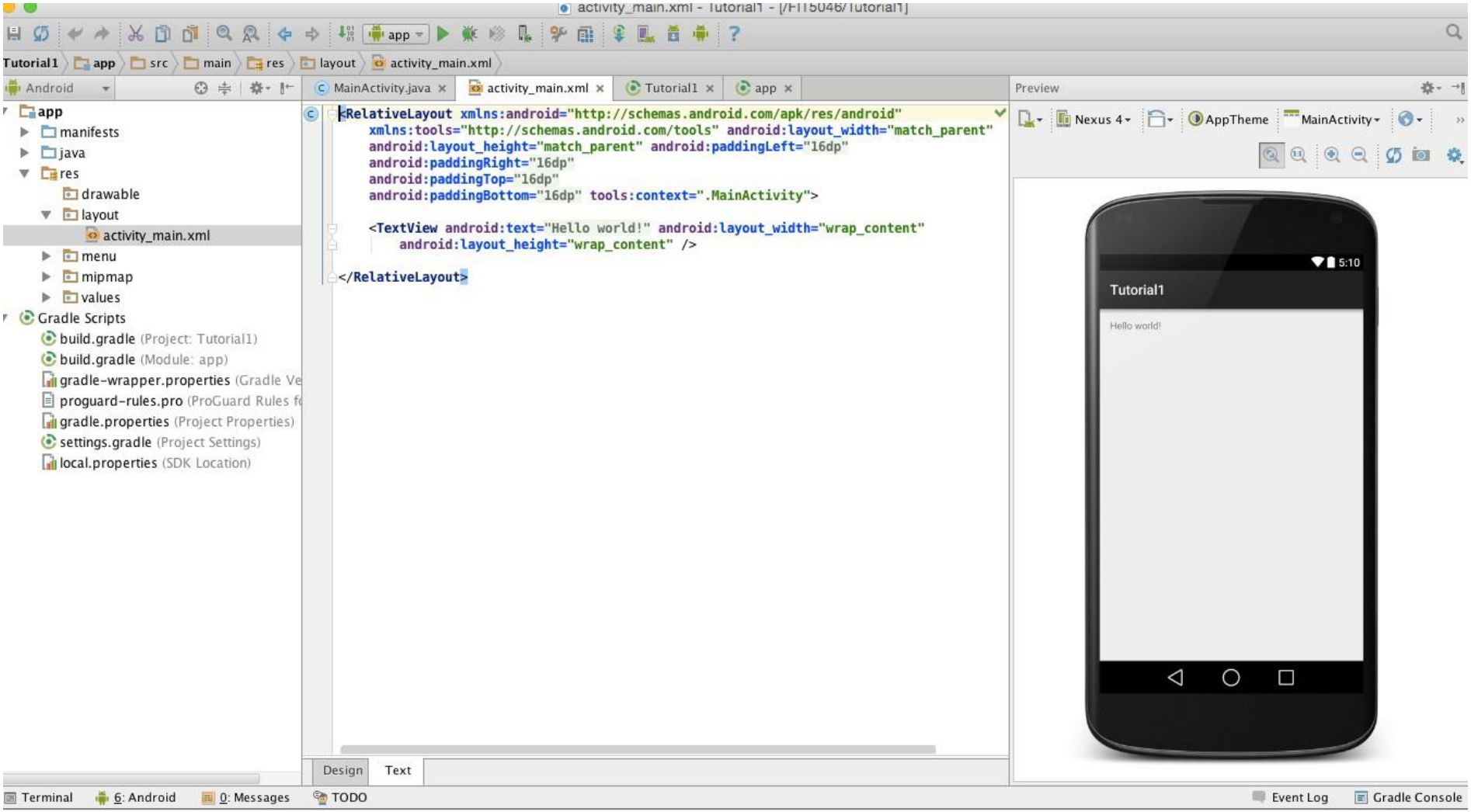




The Environment



Activity_main.xml



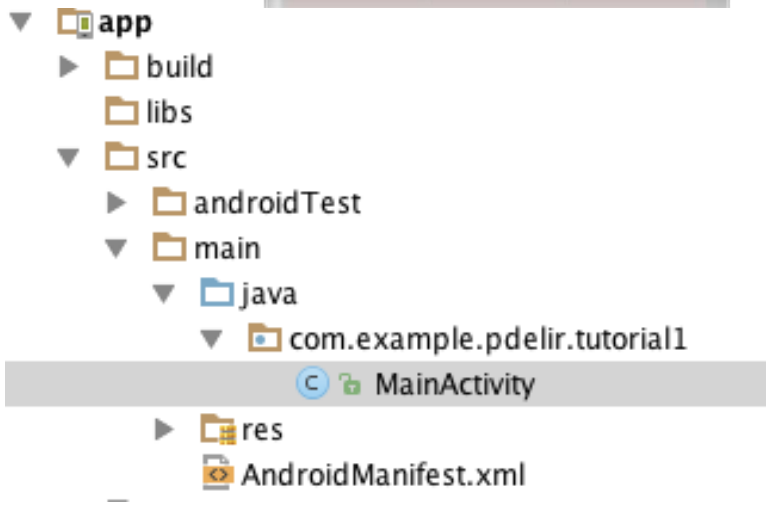
Activities

- Providing users with a screen to interact and do something
- Default activity: ***MainActivity.java***
- The user interface for an activity is provided by a hierarchy of views, which are objects derived from the View class
- An XML layout file saved in your application resources (/res) defines a layout using views
- Default: **activity_main.xml**
- An application can consist of multiple activities, bound to each other but usually one activity is **the "main" activity**
  - Presented to the user when launching the application
  - To create an activity, you must create a subclass of Activity

Activity Code to link Layouts

Previous versions extended: ActionBarActivity Activity

```
package com.example.tutorial1;

import android.support.v7.app.AppCompatActivity;

…

public class MainActivity extends AppCompatActivity {

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

        …

}
```



Important Files and Folders

src/main/java/

- It includes an MainActivity.java class that runs when your app is launched.
- **MainActivity.java**
  - Activity which is started when app executes

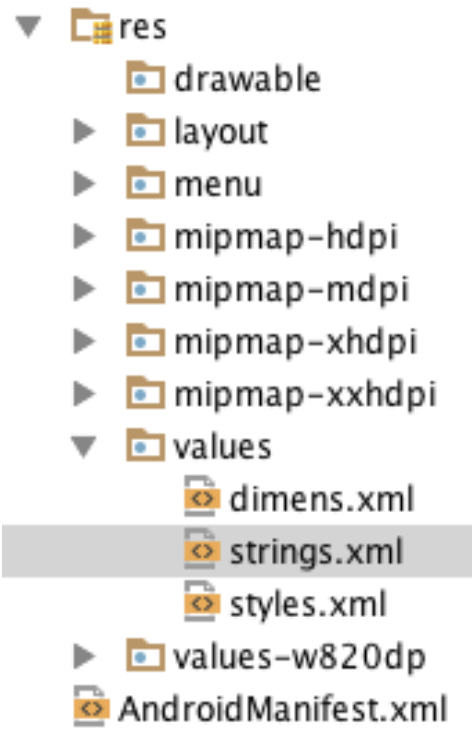
src/main/res/

- Contains several sub-directories for app resources:
- **activity_main.xml**
- Layout is a directory for files that define **your app's user interface**
- **activity_main.xml** defines layout and widgets for the activity
- When you open a layout file, you’re first shown the **Graphical Layout editor**

res/values/strings.xml

- Includes String constants used by app
- String resources allow you to manage all UI text in a single location, which makes it easier to find and update text

```
<resources>
    <string name="app_name">HelloAndroid</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
</resources>
```

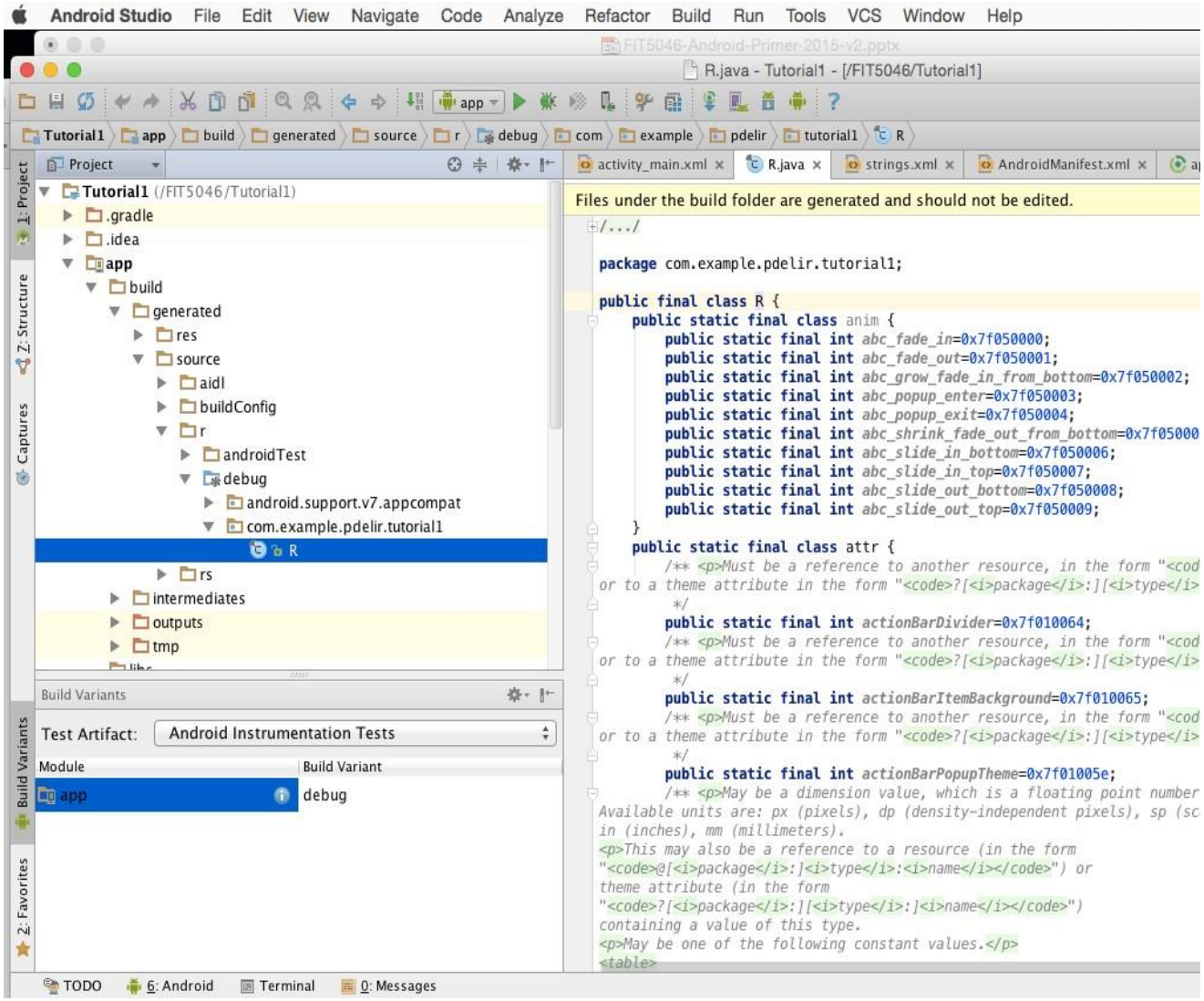


- In the activity_main.xml

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello_world" />
```

R.java

- Auto-generated file with identifiers from xml files, so you can reference your resources using Java code
- Every resource has a corresponding resource object (a unique integer) defined in your project's **R.java** file

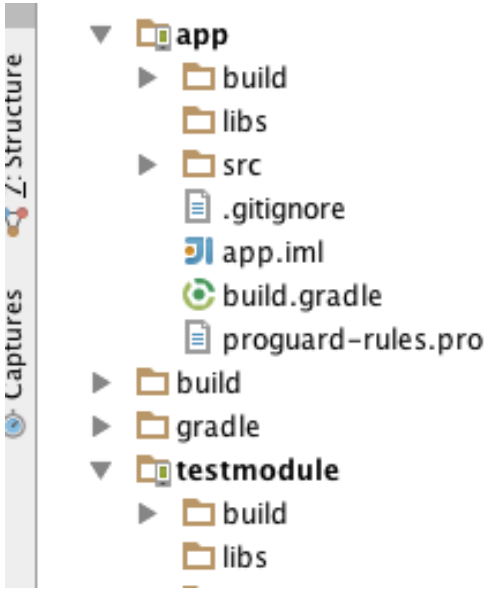


Important Files: AndroidManifest.xml

- Describes the fundamental characteristics of the app and defines their components
- For declaring the required permissions like the access to the internet

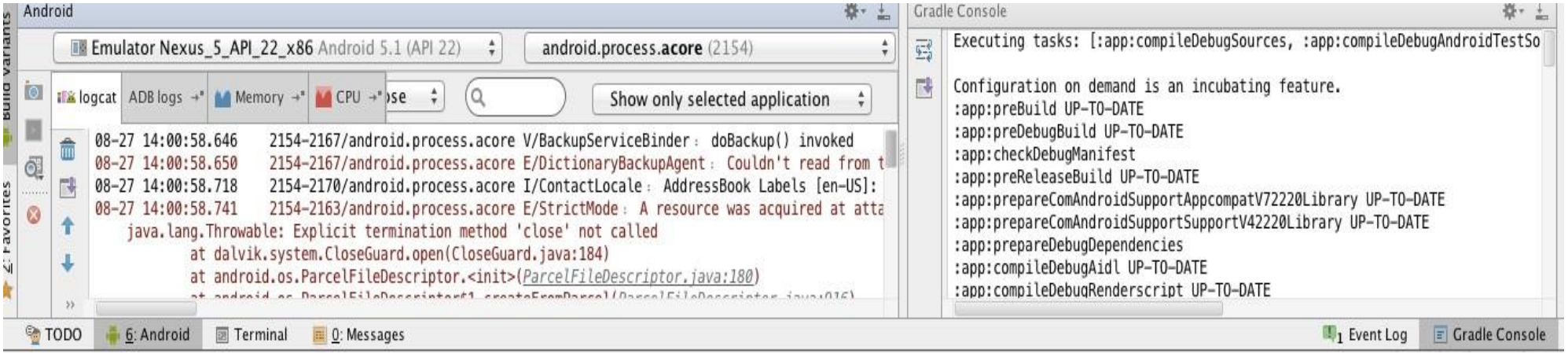
Projects and Modules

- Each project contains one or more different types of modules, such as application modules, library modules, and test modules
- To create a new module, select **File** > **New** > **Module**
- To build the project manually on Android Studio, click **Build** and select **Make Project**



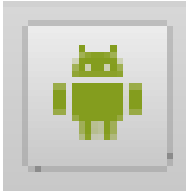
Debugging and Logcat

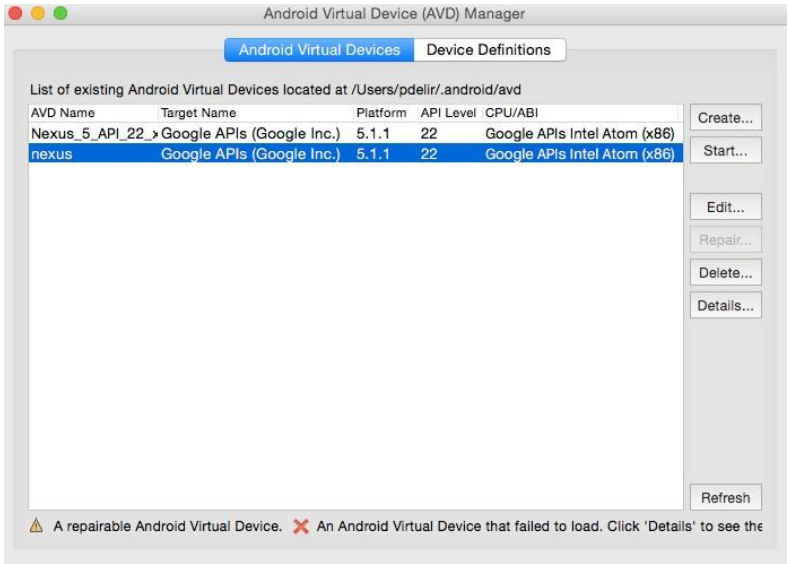
- The Android logging system provides a mechanism for collecting and viewing system debug output.
- Logs from various applications and portions of the system are collected in a series of circular buffers, which then can be viewed and filtered by the logcat. 6: Android Monitor
- More information on Debugging https://developer.android.com/tools/debugging/debugging-studio.html

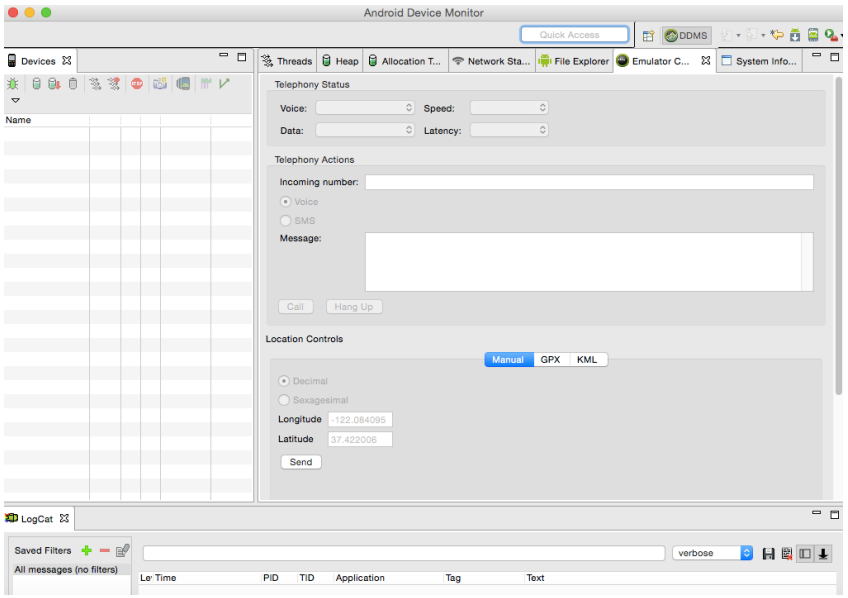


Android Emulator or AVD

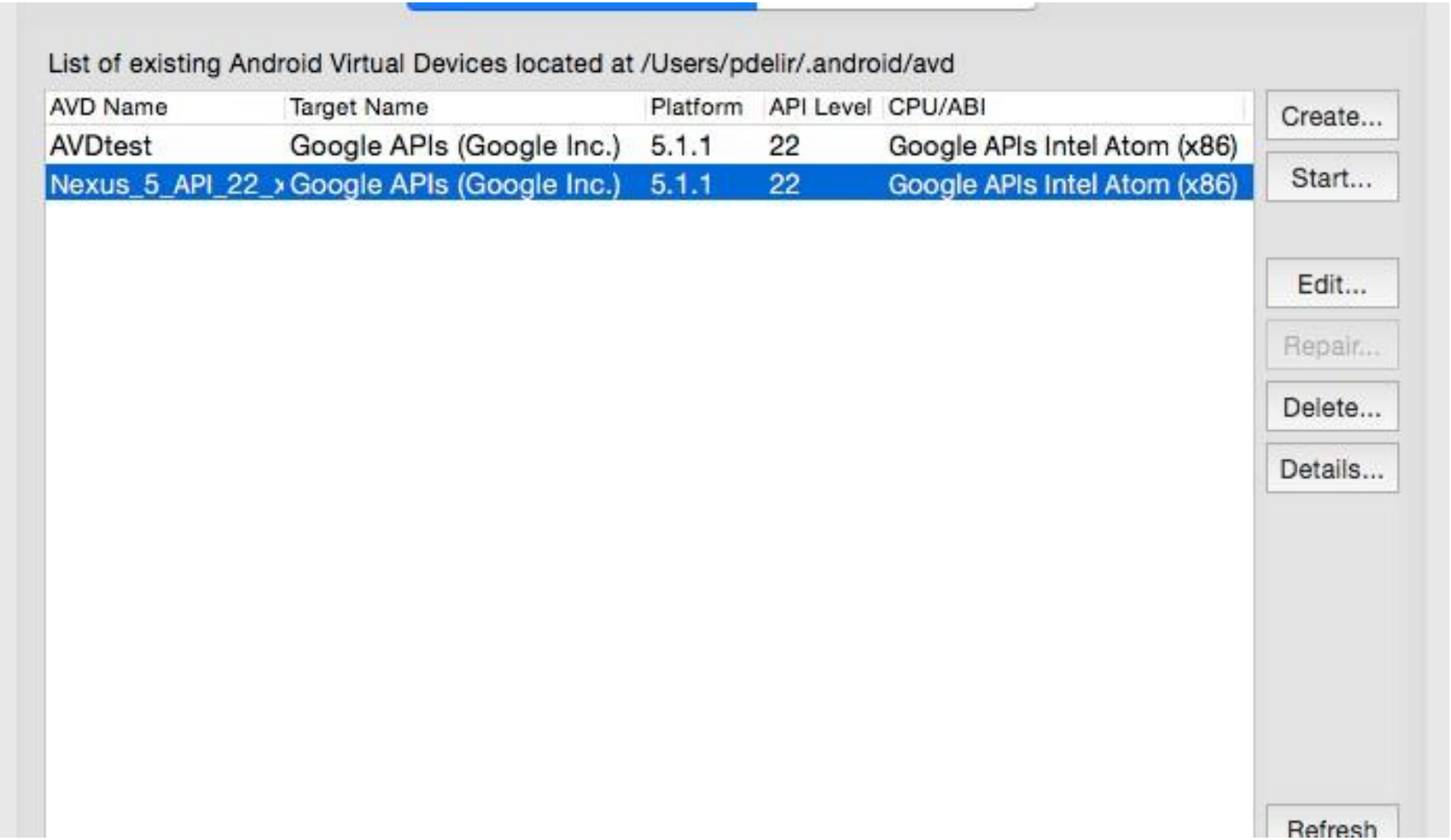
- Emulator is essential to test your application but is not a substitute for a real device
- Emulators are called Android Virtual Devices (AVDs)
- Click on Android Device Monitor
- Then click on Android Virtual Device Manager
- Create a new AVD



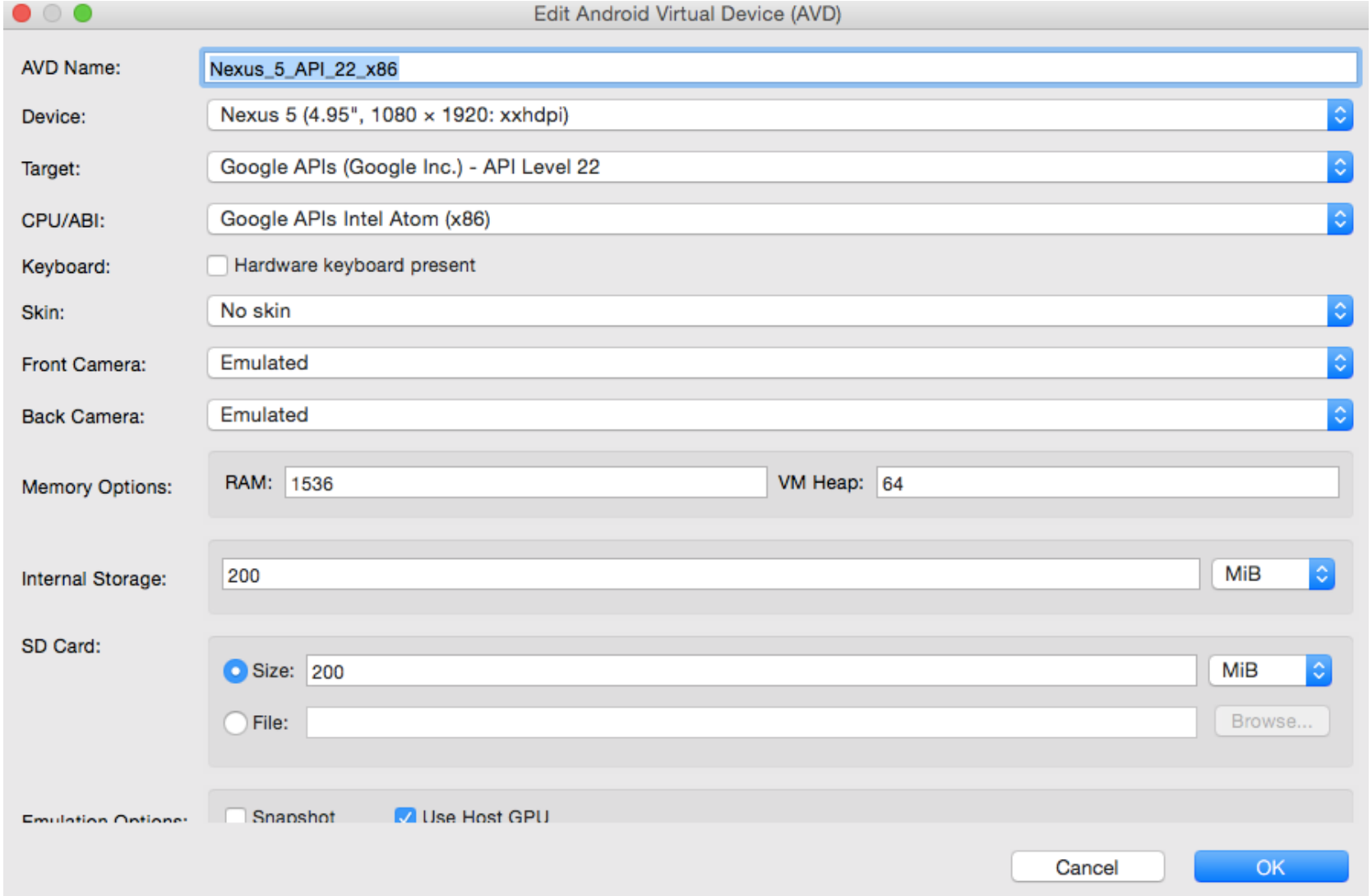




Create, Edit or Start Emulators

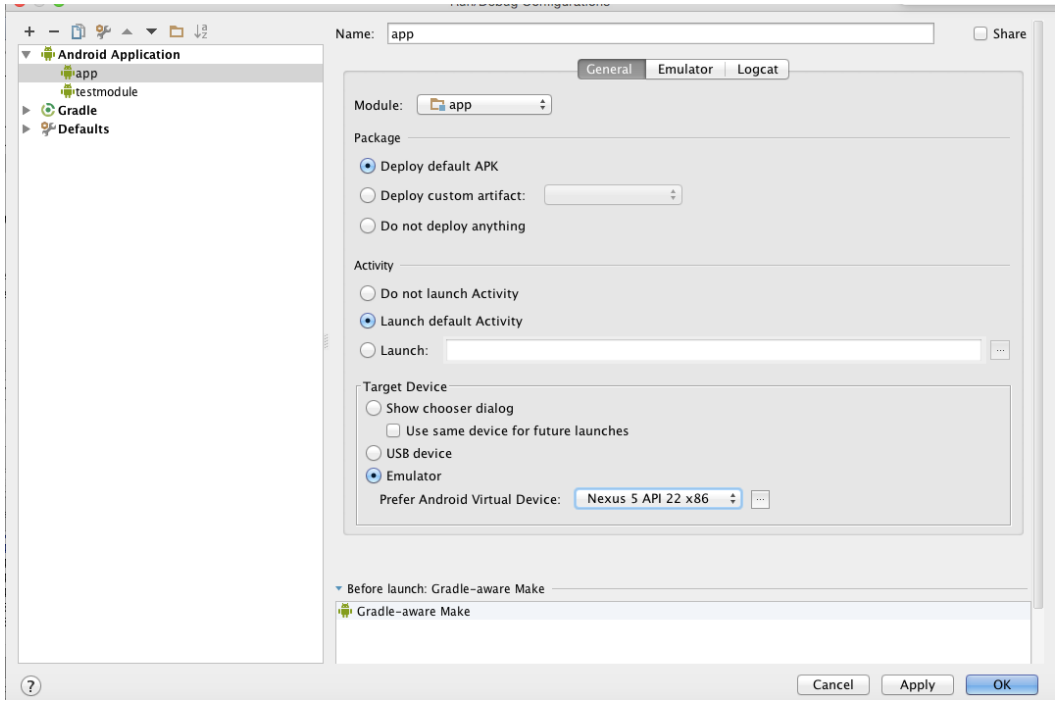


Creating a new AVD

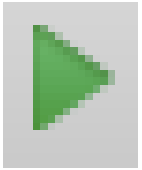


Editing Run Configuration

- When you Run your project, Android Studio automatically creates a default run configuration for the project
- You can however make changes to the default configuration
- The run configuration specifies the module to run, package to deploy, Activity to start, target device, emulator settings, and Logcat options



Android Emulator

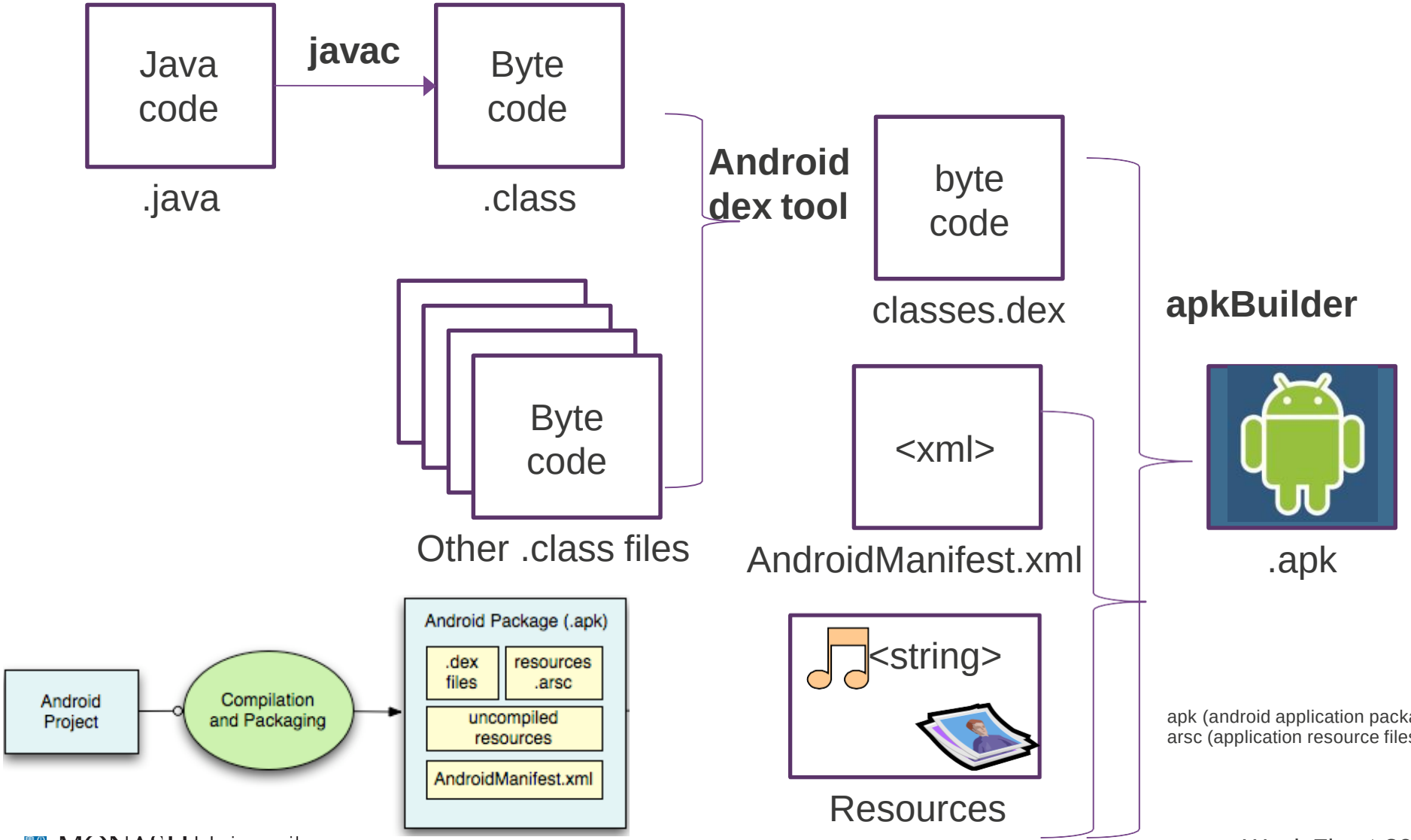


Android Runtime

Consists of

- Core Libraries
- ART (and Dalvik)
  - ART is the successor of Dalvik
  - ART and Dalvik are runtimes running Dex bytecode
- Versions of the platform prior to Android 5.0 use the Dalvik runtime for executing app code
- Android 5.0 and higher uses ART
  - Dalvik limits apps to a single classes.dex bytecode file per APK
  - ART enables using the multidex support library for building apps with multiple Dalvik Executable (DEX) files
  - Apps that reference more than 65536 (65K) methods are required to use multidex configurations

Producing an Android App



apk (android application package)
arsc (application resource files)

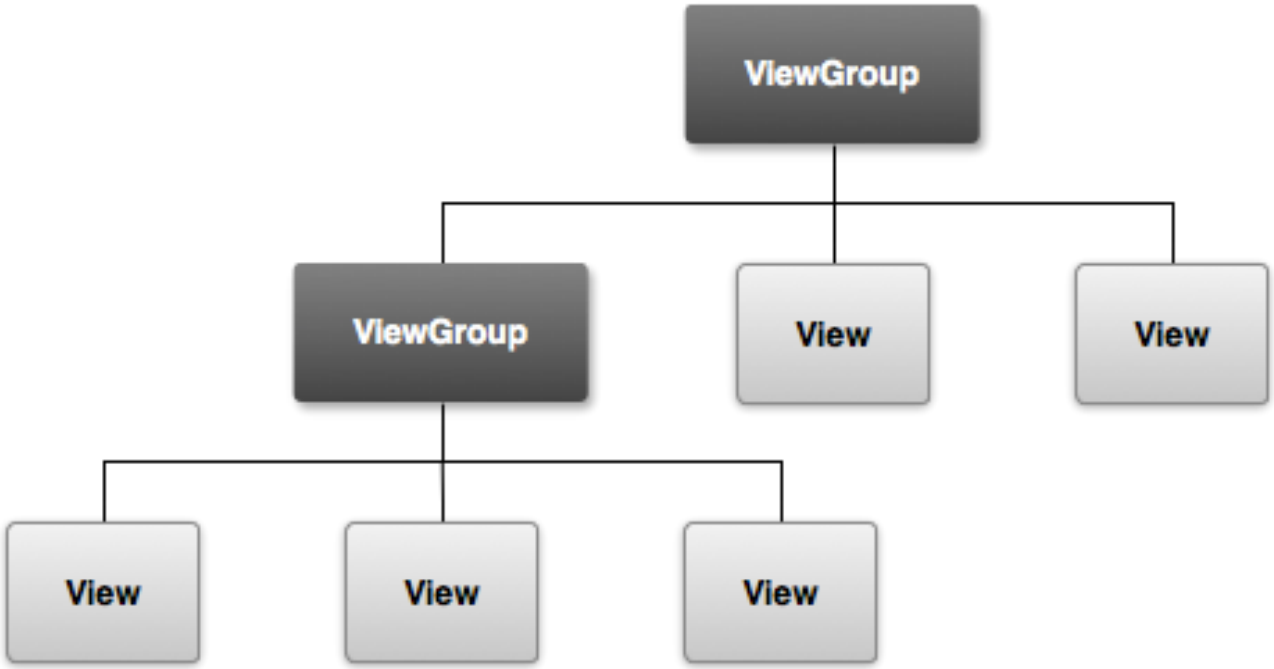
.apk file

- During **the build process**, your Android projects are compiled and packaged into an .apk file
- The .apk file contains all of the information necessary to run your application on a device or emulator
  - Including the application's code (.dex files), resources, assets, and manifest file
- .apk used to distribute and install the application
- Android Studio (new versions) outputs an .apk file automatically to the folder:
  - *projectName/moduleName/build/outputs/apk/...*

Android UI Components

User Interface Layout

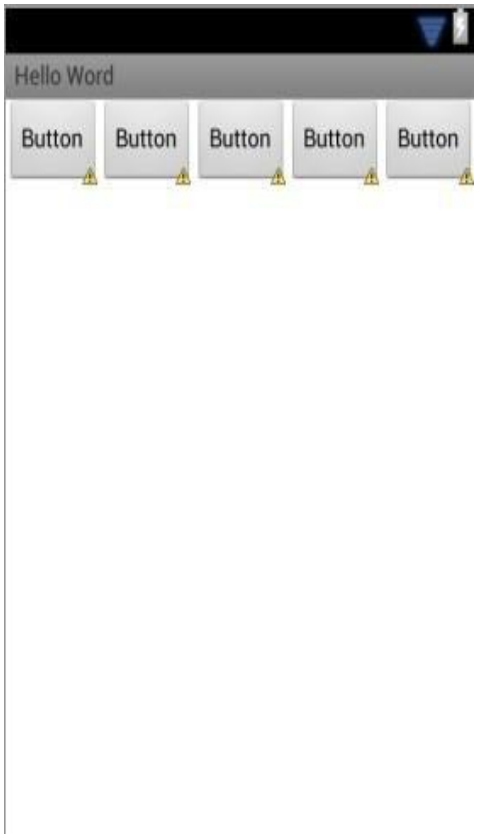
- The user interface for each component of your app is defined using a hierarchy of View and ViewGroup objects
- E.g. a <TextView>element creates a TextView widget in your UI, and a <LinearLayout> element creates a LinearLayout view group.

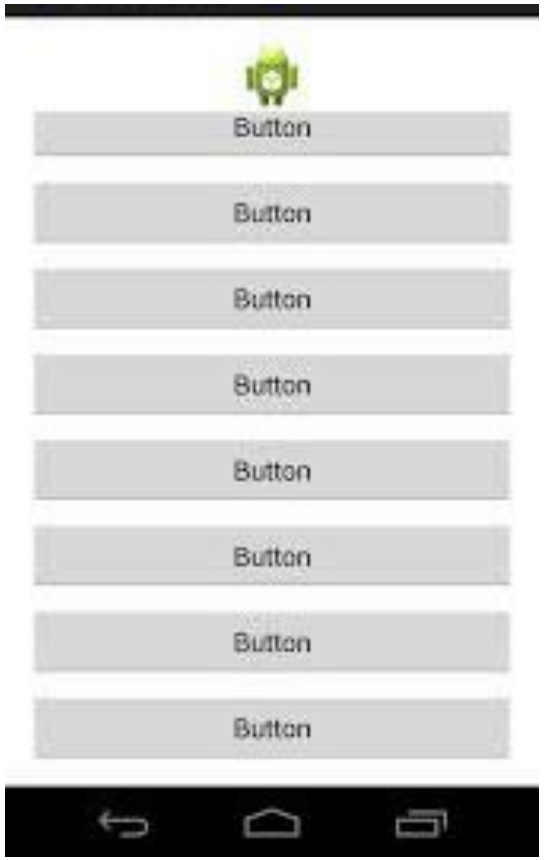


Linear Layout

Vertical or Horizontal

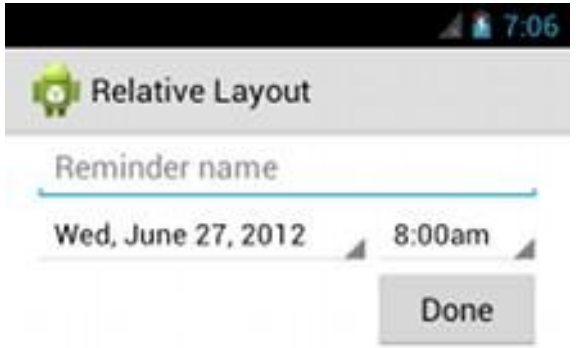
- specify the layout direction with the **android:orientation** attribute





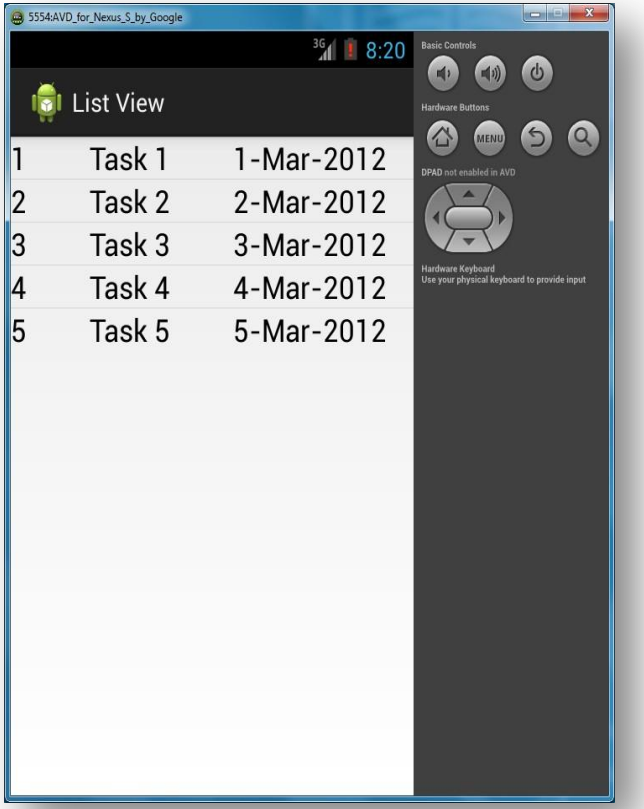
Relative Layout

- A view group that displays child views in relative positions
- The position of each view can be specified as relative to sibling elements
  - E.g. to the left-of or below another view
- or in positions relative to the parent RelativeLayout area
  - E.g. aligned to the bottom, left of centre



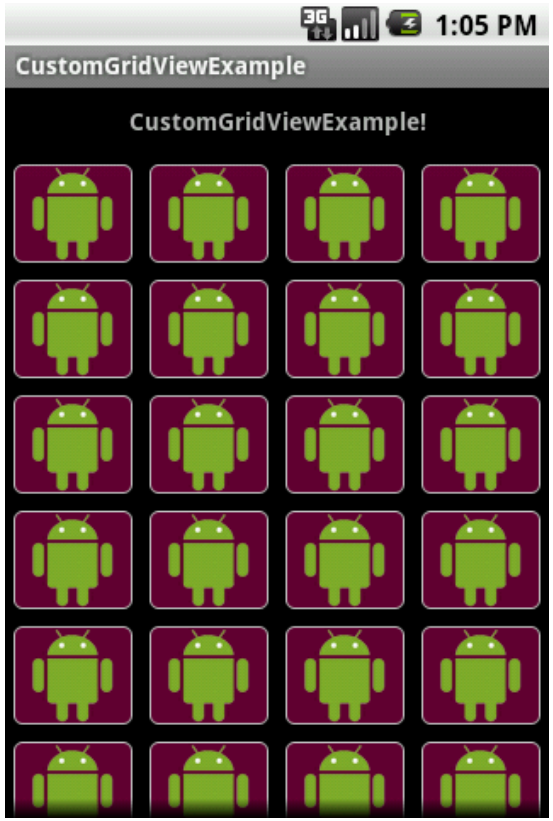
List View

- A view group that displays a list of scrollable items
- The list items are automatically inserted to the list using an Adapter that pulls content from a source such as an array or database query



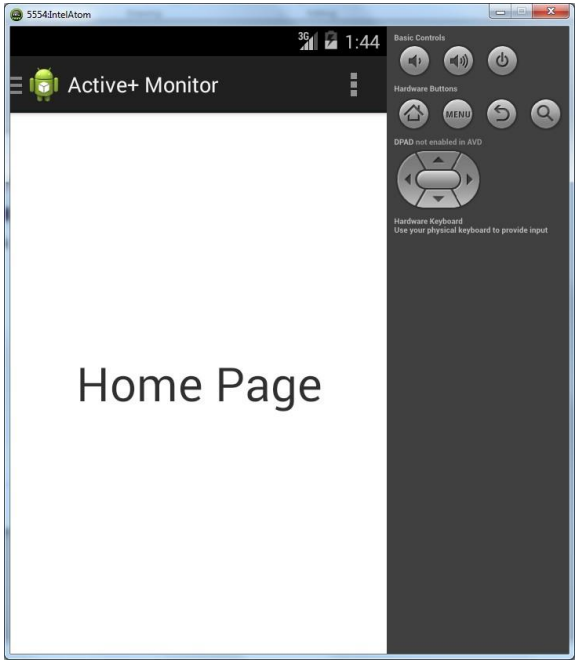
Grid View

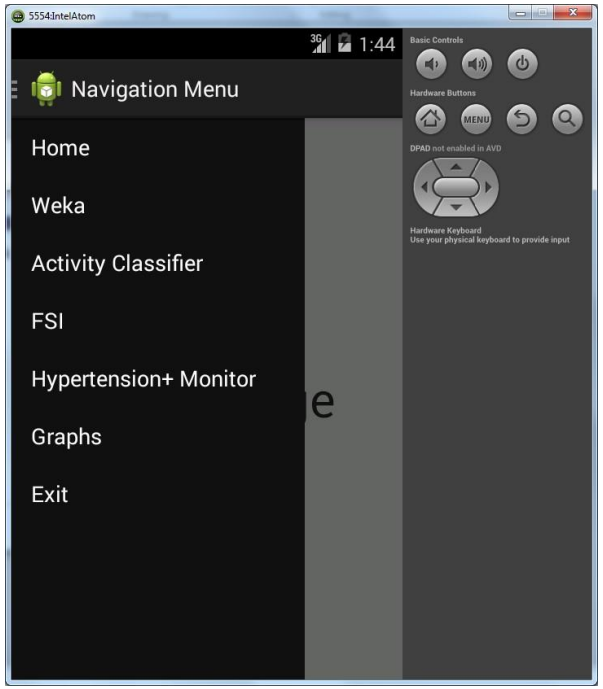
- Displays items in a two-dimensional, scrollable grid. The grid items are automatically inserted to the layout using a ListAdapter



DrawerLayout

- A navigation drawer is a panel that displays the app's main navigation options on the left edge of the screen
- Enables navigation between major modules of the application
- It is hidden most of the time and can be shown by swiping the screen from left edge to right or tapping the app icon on the action bar.





References

- http://developer.android.com/
- To test your application on a real Android device: http://developer.android.com/tools/device.html