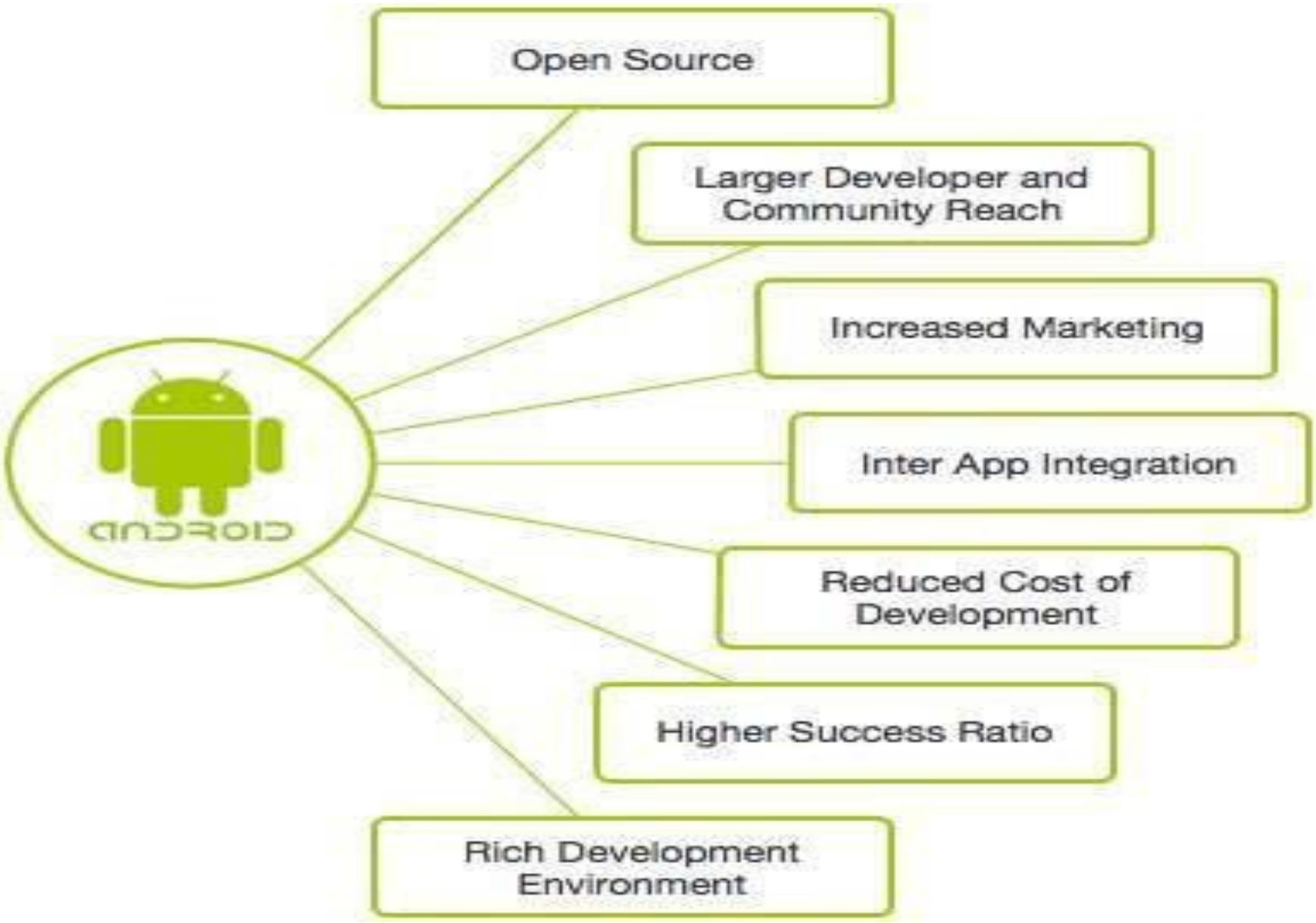
Android

.

- **Android** is a software package and linux based operating system for mobile devices such as tablet computers and smartphones.
- It is developed by Google and later the **OHA** (Open Handset Alliance). Java language is mainly used to write the android code even though other languages can be used.

What is Open Handset Alliance (OHA)

- It's a consortium of 84 companies such as google, samsung, AKM, synaptics, KDDI, Garmin, Teleca, Ebay, Intel etc.
- It was established on 5th November, 2007, led by Google. It is committed to advance open standards, provide services and deploy handsets using the Android Plateform.


Open Source
Larger Developer and Community Reach
Increased Marketing
Inter App Integration
Reduced Cost of Development
Higher Success Ratio
Rich Development Environment
android

ANDROID VERSIONS

Version	Code name	API Level
1.5	Cupcake	3
1.6	Donut	4
2.1	Eclair	7
2.2	Froyo	8
2.3	Gingerbread	9 and 10
3.1 and 3.3	Honeycomb	12 and 13
4.0	Ice Cream Sandwitch	15
4.1, 4.2 and 4.3	Jelly Bean	16, 17 and 18
4.4	KitKat	19
5.0	Lollipop	21
6.0	Marshmallow	23
7.0	Nougat	24-25
8.0	Oreo	26-27

Android Application Development Environment

Operating systems:

- **Microsoft Windows XP or later version.**
- **Mac OS X 10.5.8 or later version with Intel chip.**
- **Linux including GNU C Library 2.7 or later.**

Android application programming:

- **Java JDK5 or JDK6**
- **Android SDK**
- **Eclipse IDE for Java Developers (optional)**
- **Android Development Tools (ADT) Eclipse Plugin (optional)**

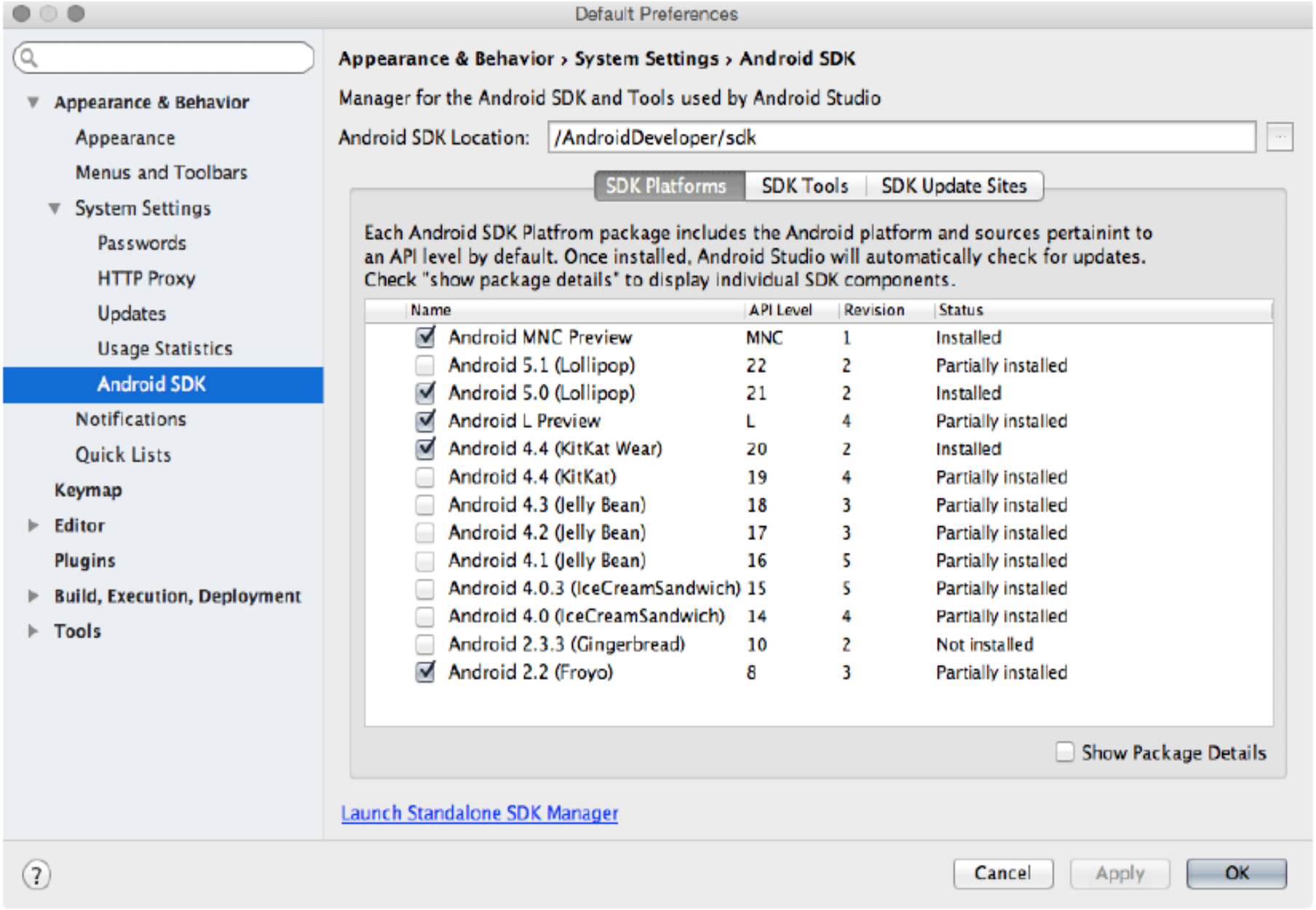
Step 1 – Set up Java Development Kit (JDK)

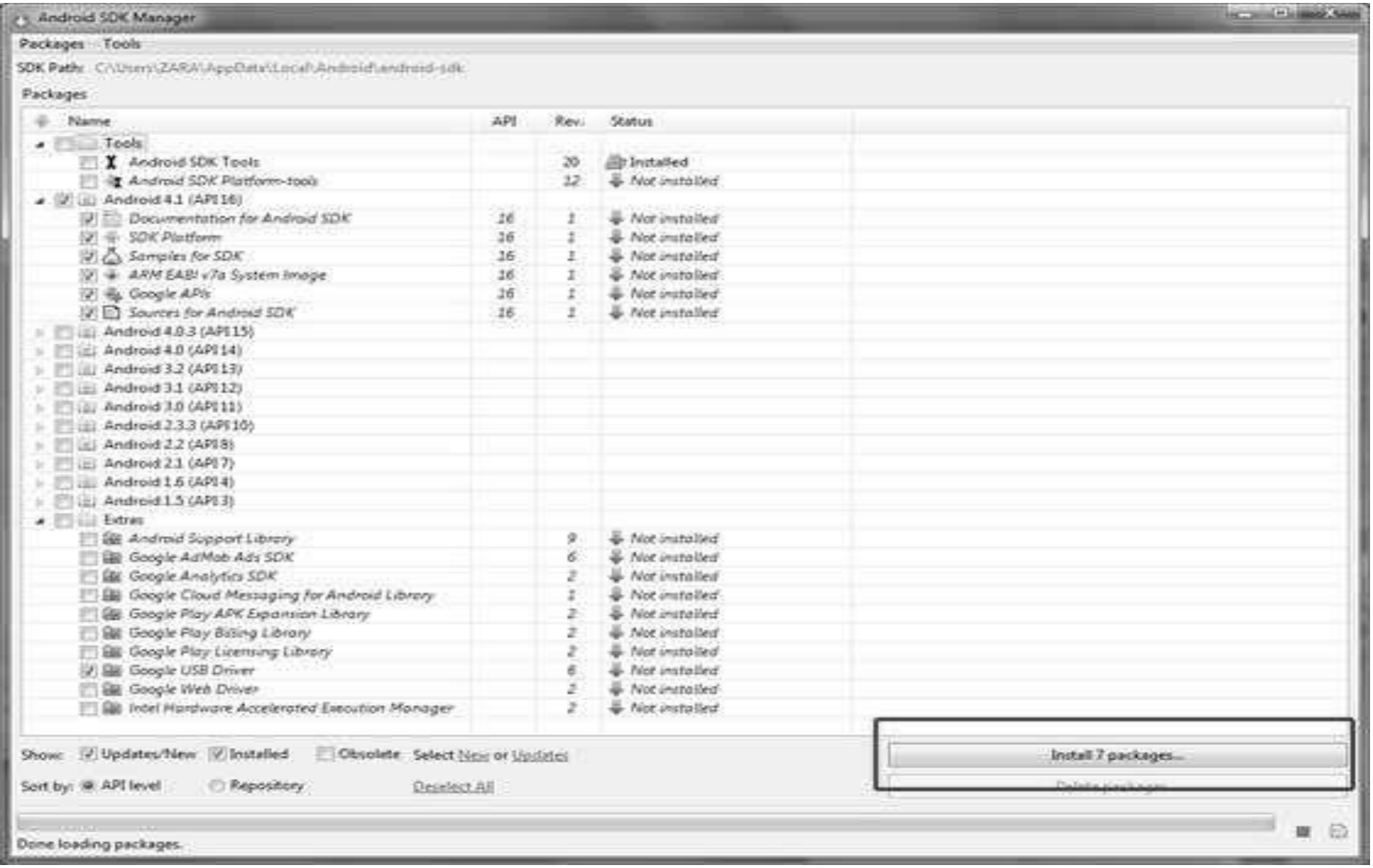
You can download the latest version of Java JDK from **Oracle's Java site**: Java SE Downloads. You will find instructions for installing JDK in downloaded files, follow the given instructions to install and configure the setup. Finally, set PATH and JAVA_HOME environment variables to refer to the directory that

contains **java** and **Javac.**

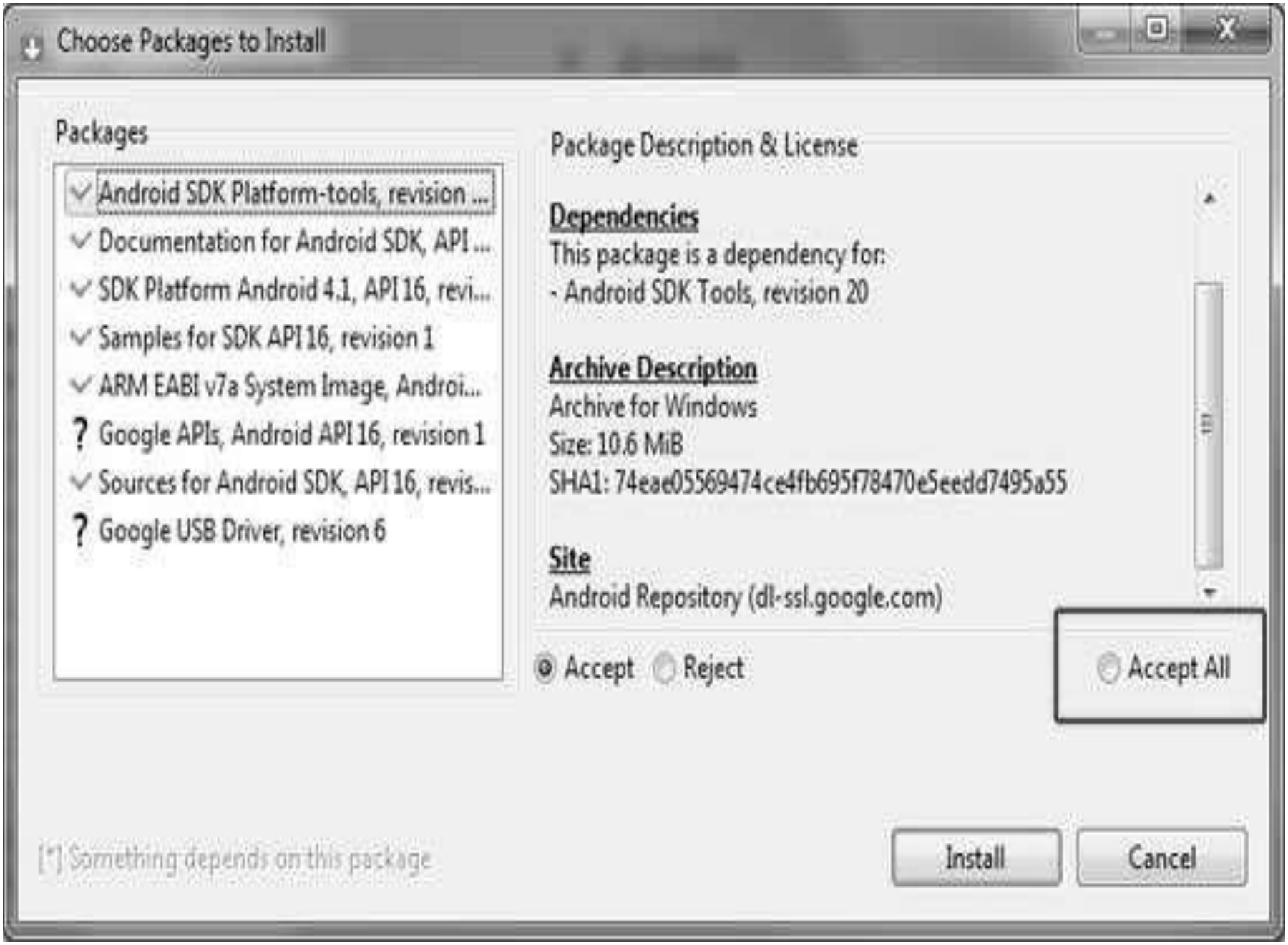
Step 2 - Setup Android SDK

You can download the latest version of Android SDK from **Android's official website**: http://developer.android.com/sdk/index.html. If you are installing SDKon **Windows machine**, then you will find **a*installer_rXX-windows.exe***, so just download and run this exe which will launch ***Android SDK Tool Setup* wizard**


Default Preferences
Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
Passwords
HTTP Proxy
Updates
Usage Statistics
Android SDK
Notifications
Quick Lists
Keymap
Editor
Plugins
Build, Execution, Deployment
Tools
Appearance & Behavior › System Settings › Android SDK
Manager for the Android SDK and Tools used by Android Studio
Android SDK Location: /AndroidDeveloper/sdk
SDK Platforms
SDK Tools
SDK Update Sites
Each Android SDK Platfrom package includes the Android platform and sources pertainint to an API level by default. Once installed, Android Studio will automatically check for updates. Check "show package details" to display individual SDK components.
Name
API Level
Revision
Status
Android MNC Preview MNC 1 Installed
Android 5.1 (Lollipop) 22 2 Partially installed
Android 5.0 (Lollipop) 21 2 Installed
Android L Preview L 4 Partially installed
Android 4.4 (KitKat Wear) 20 2 Installed
Android 4.4 (KitKat) 19 4 Partially installed
Android 4.3 (Jelly Bean) 18 3 Partially installed
Android 4.2 (Jelly Bean) 17 3 Partially installed
Android 4.1 (Jelly Bean) 16 5 Partially installed
Android 4.0.3 (IceCreamSandwich) 15 5 Partially installed
Android 4.0 (IceCreamSandwich) 14 4 Partially installed
Android 2.3.3 (Gingerbread) 10 2 Not installed
Android 2.2 (Froyo) 8 3 Partially installed
Show Package Details
Launch Standalone SDK Manager
Cancel
Apply
OK

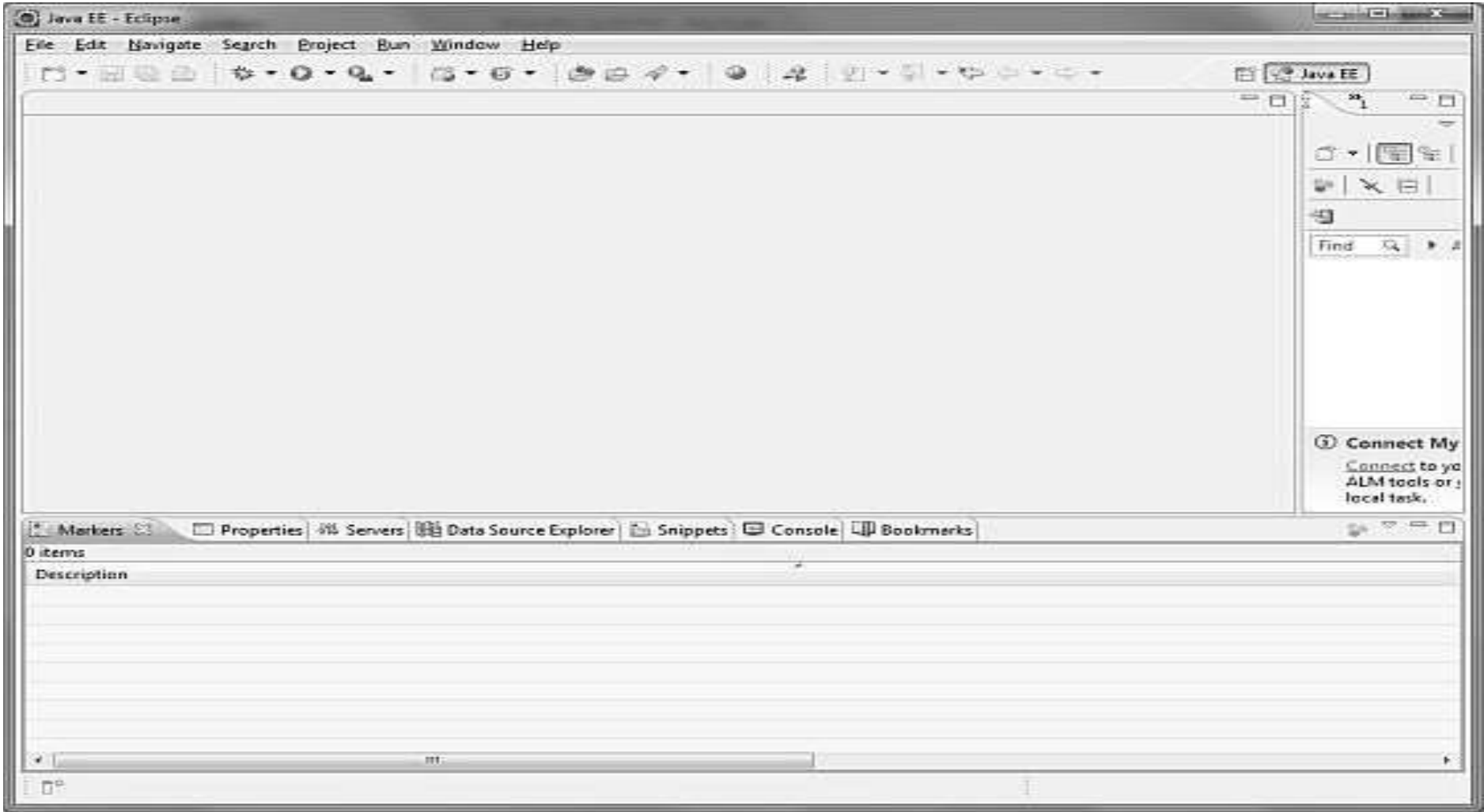


By default it will list down total 7 packages to be installed


Choose Packages to Install
Packages
Android SDK Platform-tools, revision ...
Documentation for Android SDK, API ...
SDK Platform Android 4.1, API 16, revi...
Samples for SDK API 16, revision 1
ARM EABI v7a System Image, Androi...
Google APIs, Android API 16, revision 1
Sources for Android SDK, API 16, revis...
Google USB Driver, revision 6
Package Description & License
Dependencies
This package is a dependency for:
- Android SDK Tools, revision 20
Archive Description
Archive for Windows
Size: 10.6 MiB
SHA1: 74eae05569474ce4fb695f78470e5eedd7495a55
Site
Android Repository (dl-ssl.google.com)
Accept
Reject
Accept All
[*] Something depends on this package
Install
Cancel

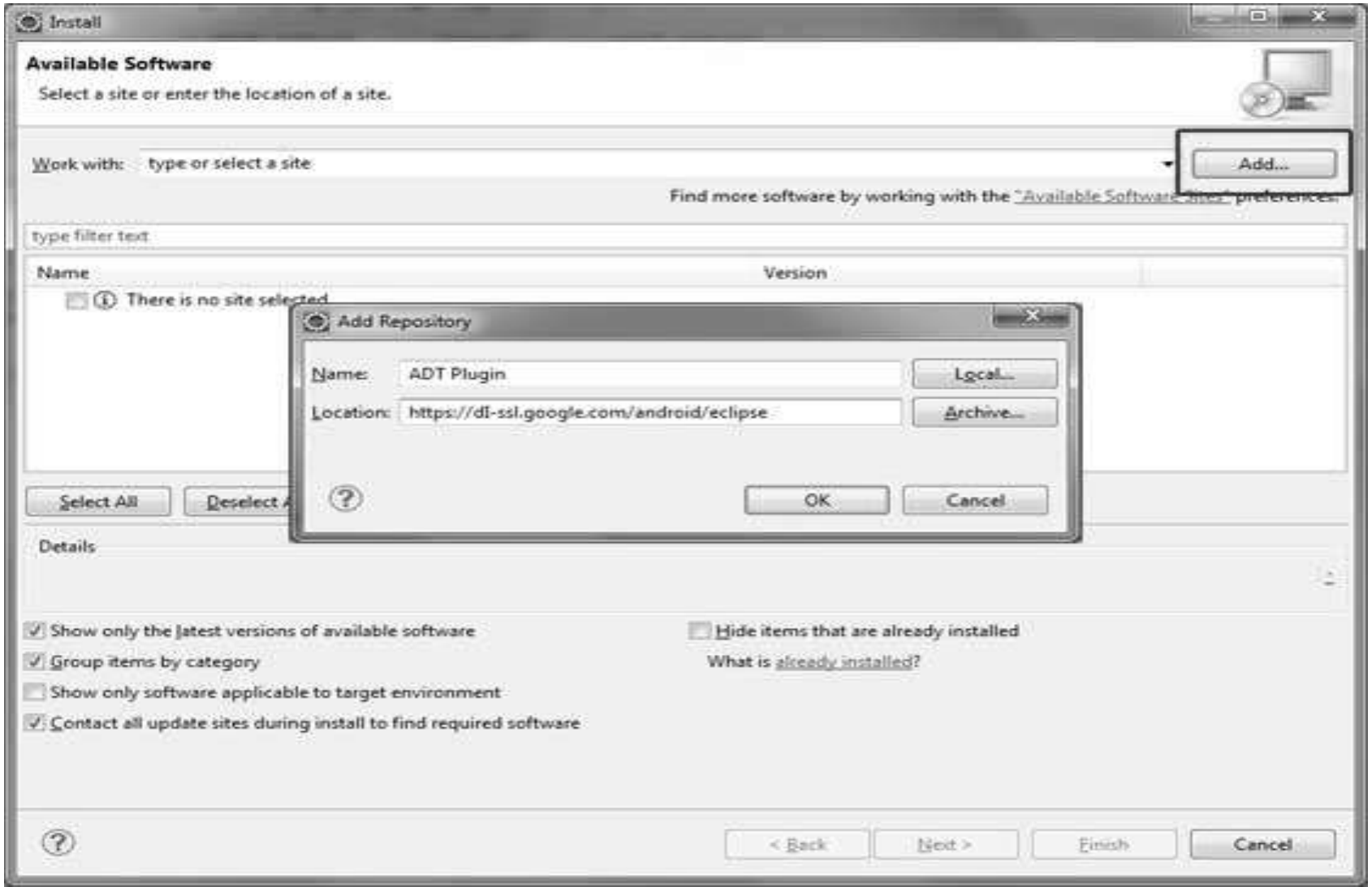
Step 3 - Setup Eclipse IDE

To install Eclipse IDE, download the latest Eclipse binaries from **http://www.eclipse.org/downloads/.** Eclipse can be started by executing the following commands on windows machine, or you can simply double click on eclipse.exe **%C:\eclipse\eclipse.exe**



Step 4 - Setup Android Development Tools (ADT) Plugin

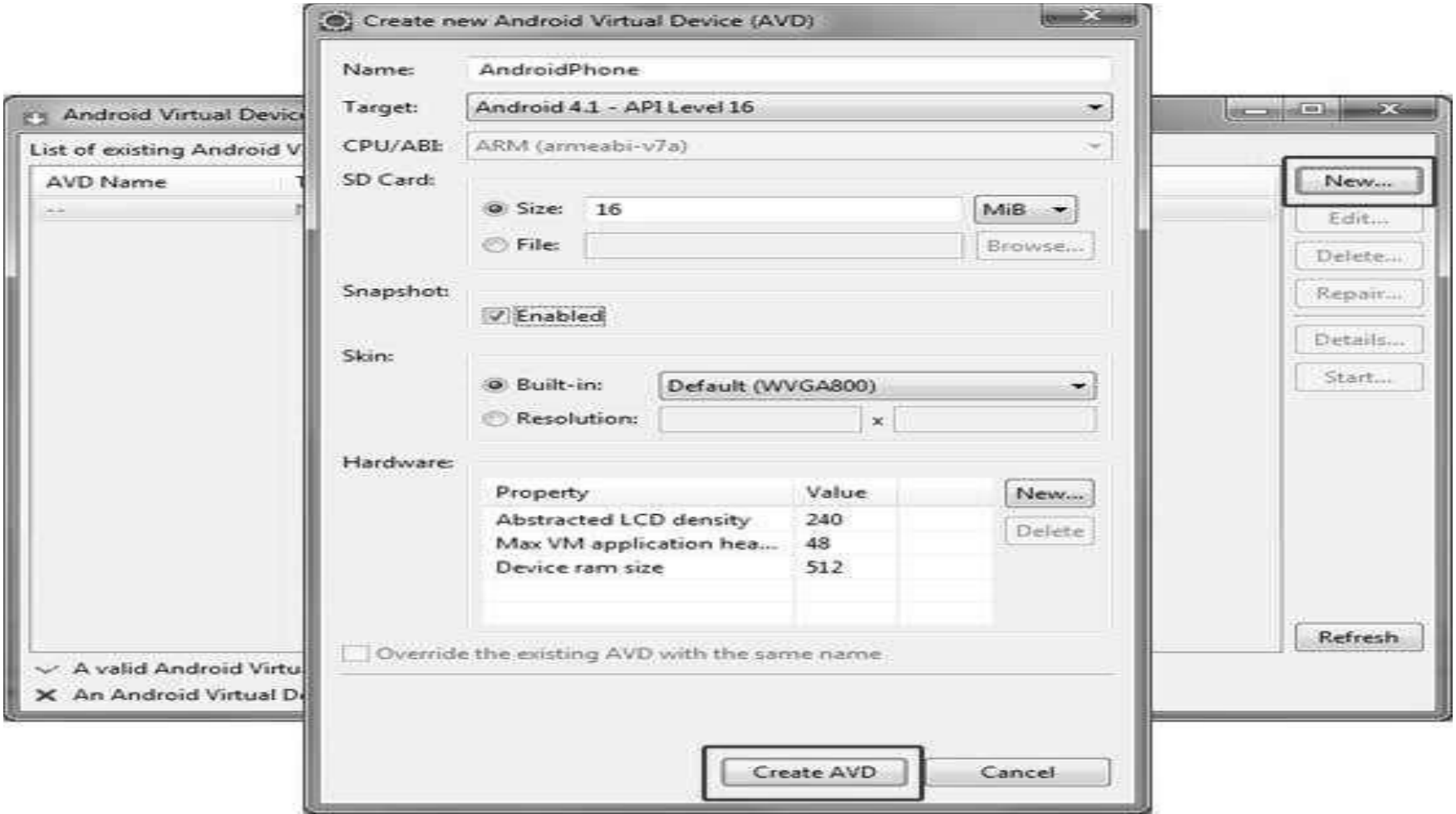
This step will help you in setting Android Development Tool plugin for Eclipse. Let's start with launching Eclipse and then, choose **Help > Software Updates > Install New Software**. This will display the following dialogue box.

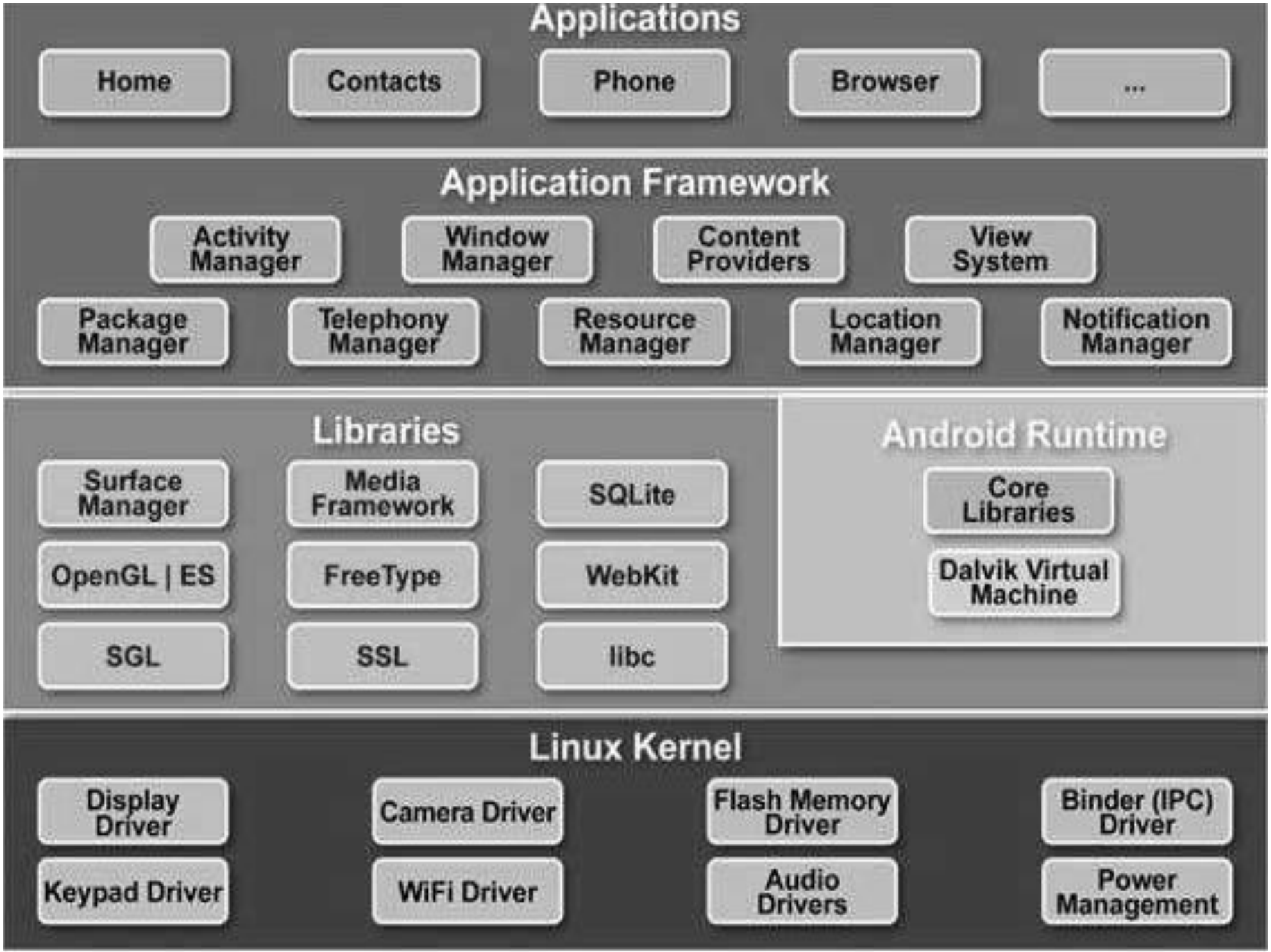


Step 5 - Create Android Virtual Device

To test your Android applications you will need a virtual Android device. So before we start writing our code, let us create an Android virtual device.

Launch Android AVD Manager using Eclipse menu options **Window > AVD Manager>** which will launch Android AVD Manager. Use **New** button to create a new Android Virtual Device




Applications
Home
Contacts
Phone
Browser
...
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

Architecture

- *Android operating system* is a stack of software components which is roughly divided into **five** sections and **four main layers** as shown below in the architecture diagram.
- **Linux kernel**-It contains all the essential hardware drivers like camera, keypad, display etc.
- **Native libraries** such as WebKit, OpenGL, FreeType, SQLite, Media, C runtime library (libc) etc.

  The WebKit library is responsible for browser support,

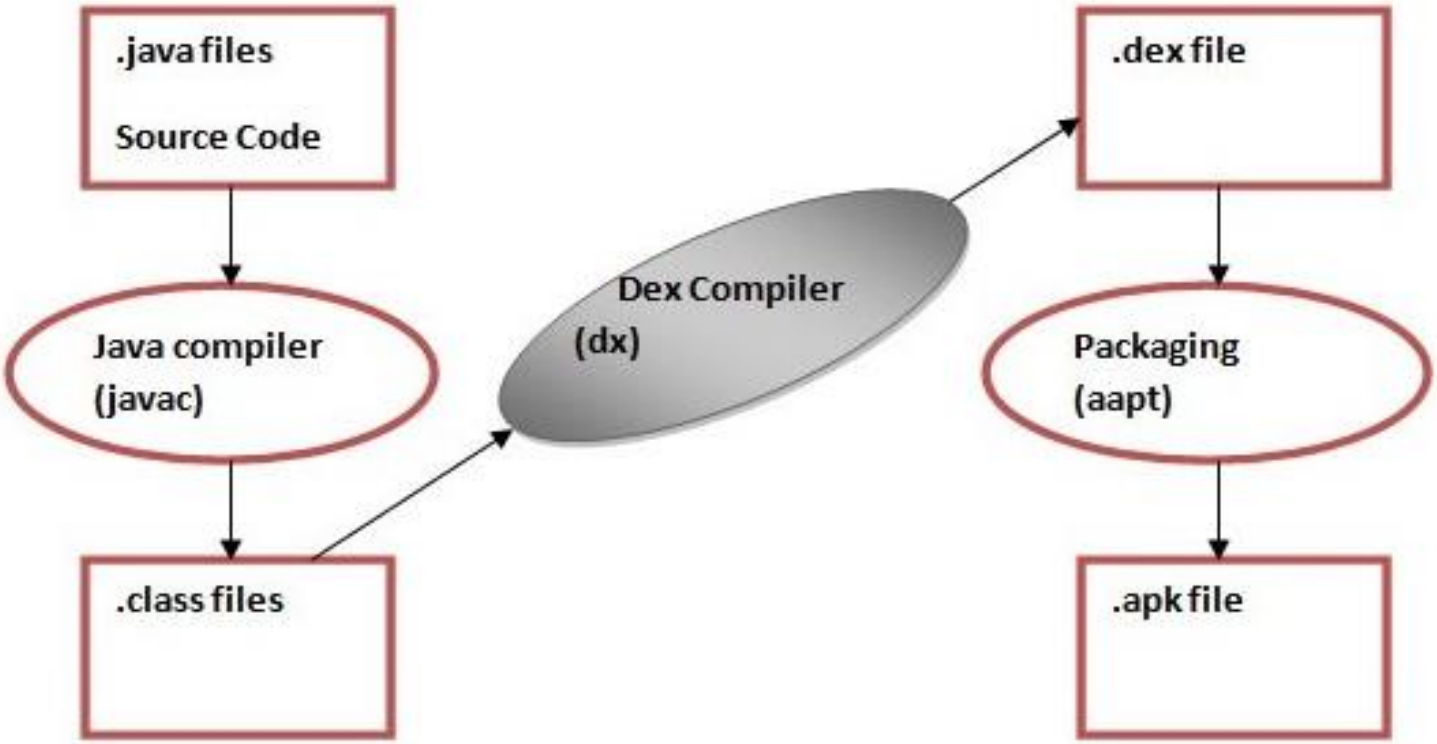
  SQLite is for database,

  FreeType for font support,

  Media for playing and recording audio and video formats.

- **Android run time**: contains the java virtual **machine(Dalvik VM**) which enables every application to run its process, few set of libraries.

Dalvik Virtual Machine | DVM



Dalvik Virtual Machine | DVM

- The **javac tool** compiles the java source file into the class file.
- The **dx tool** takes all the class files of your application and generates a single .dex file. It is a platform-specific tool.
- The **Android Assets Packaging Tool (aapt)** handles the packaging process.

Dalvik Virtual Machine

- The **Dalvik Virtual Machine (DVM)** is an android virtual machine optimized for mobile devices. It optimizes the virtual machine for *memory*, *battery life* and *performance*.
- Dalvik is a name of a town in Iceland. The Dalvik VM was written by Dan Bornstein.
- The Dex compiler converts the class files into the .dex file that run on the Dalvik VM. Multiple class files are converted into one dex file.

Architecture

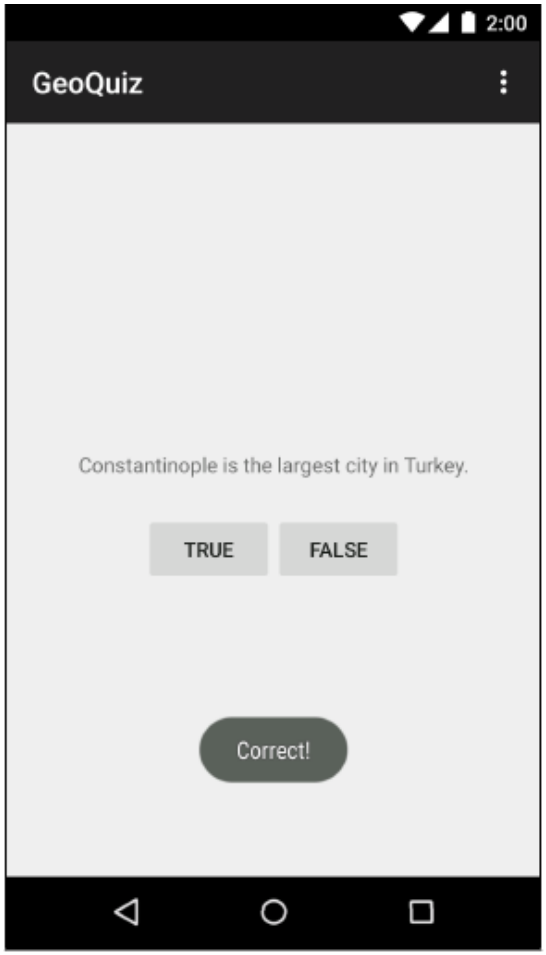
- **Application frame**: provides higher level services to the application.
- services −
  - **Activity Manager** − Controls all aspects of the application lifecycle and activity stack.
  - **Content Providers** − Allows applications to publish and share data with other applications.
  - **Notifications Manager** − Allows applications to display alerts and notifications to the user.
  - **View System** − An extensible set of views used to create application user interfaces.

- **Applications**:We will find all the Android application at the top layer. We will write our application to be installed on this layer only.

Building blocks of Android

Components	Description
Activities	They dictate the UI and handle the user interaction to the smartphone screen
Services	They handle background processing associated with an application.
Broadcast Receivers	They handle communication between Android OS and applications.
Content Providers	They handle data and database management issues.

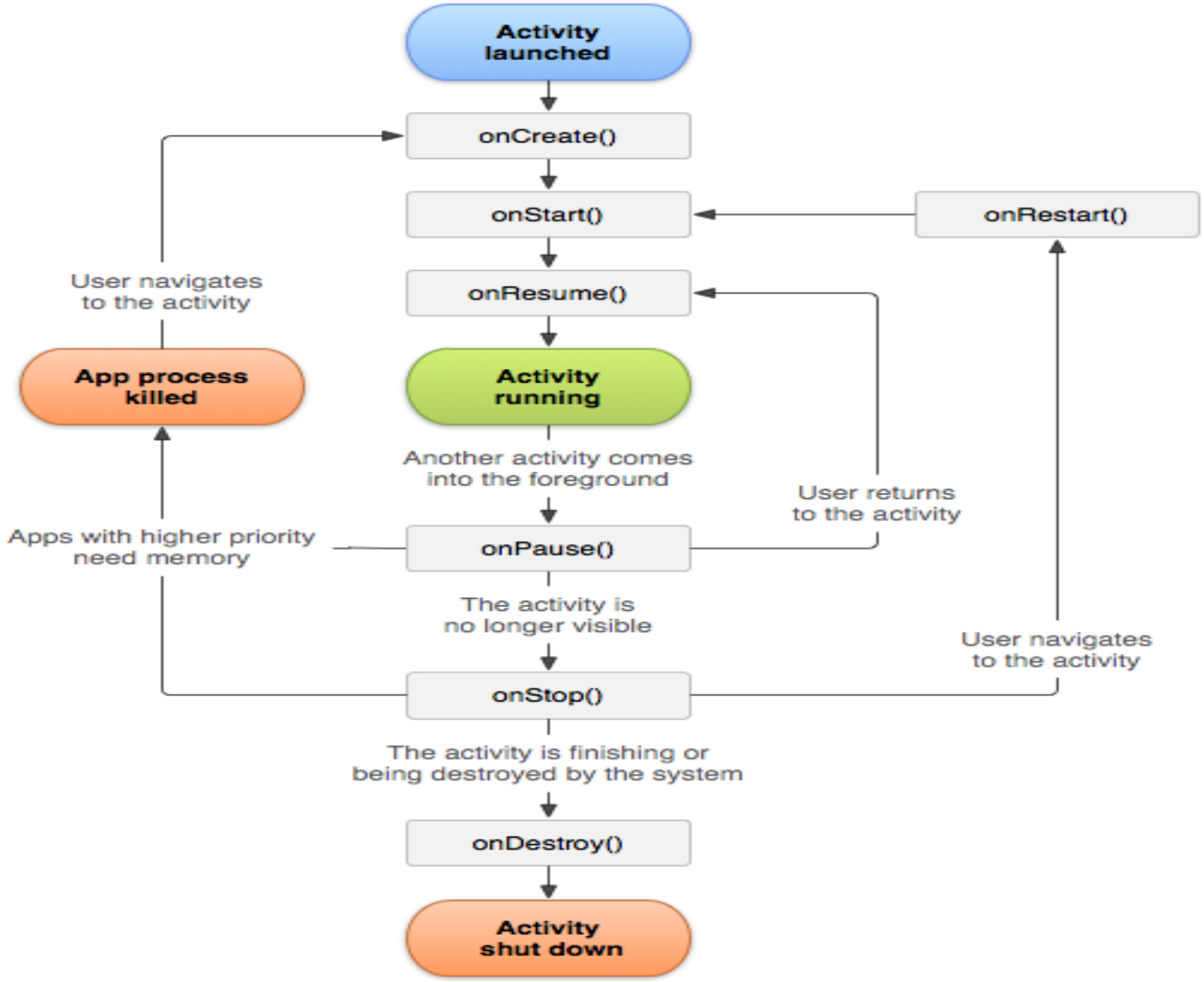
ACTIVITY


2:00
GeoQuiz
Constantinople is the largest city in Turkey.
TRUE
FALSE
Correct!

Building blocks of Android

- **Activities(focussed thing a user can do)-**eg: edit a note, play a game, opening a mail etc. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.
- An activity is implemented as a subclass of **Activity** class as follows −

```
public class MainActivity extends Activity
        {
        }
```


Activity launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

SERVICES

- Android user interface is restricted to perform long running jobs to make user experience smoother. A typical long running tasks can be periodic **downloading of data from internet, saving multiple records into database, perform file I/O, fetching your phone contacts list**, etc. For such long running tasks, Service is the alternative.
- A service is a application component used to perform **long running tasks in background.**
- A service **doesn't have any user interface** and neither can directly communicate to an activity.
- A service can run in the background indefinitely, even if component that started the service is destroyed.
- Usually a service always performs a single operation and stops itself once intended task is complete.

Building blocks of Android-services

- Services(**backgnd process**)-A service is a component that runs in the background to perform long-running operations. For example, a service might play music in the background while the user is in a different application, date update etc.
- A service is implemented as a subclass of **Service** class as follows −

```
public class MyService extends Service
        {
        }
```

Life Cycle of Android Service

- There can be two forms of a service.
- Started
- Bound

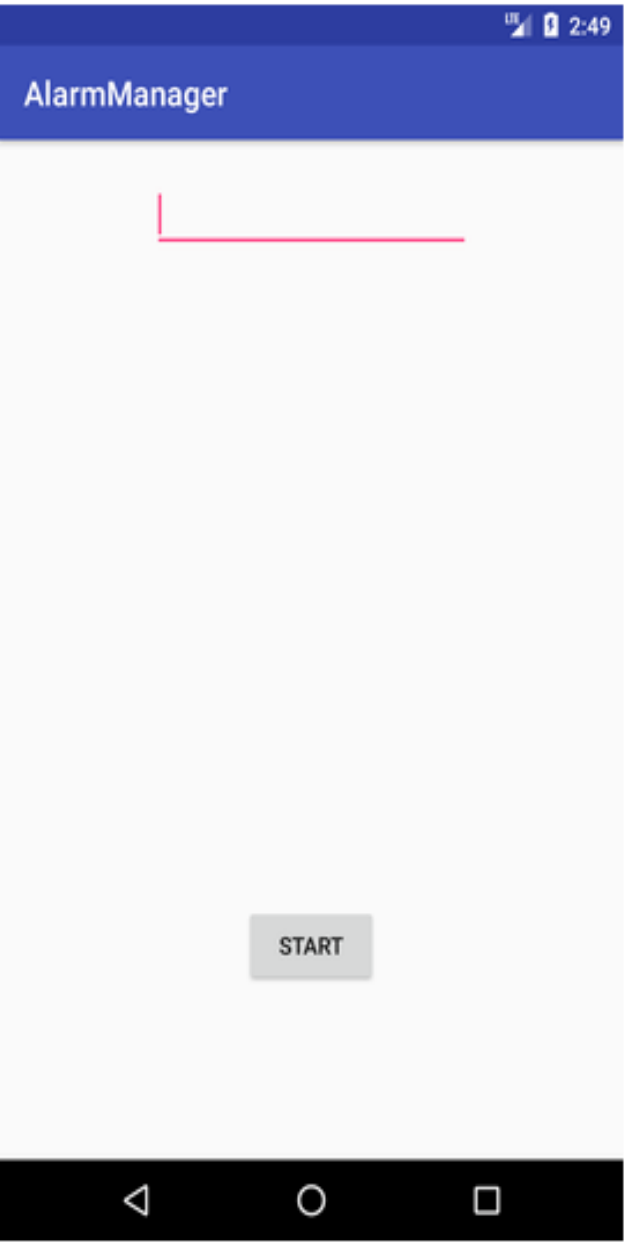
1) Started Service

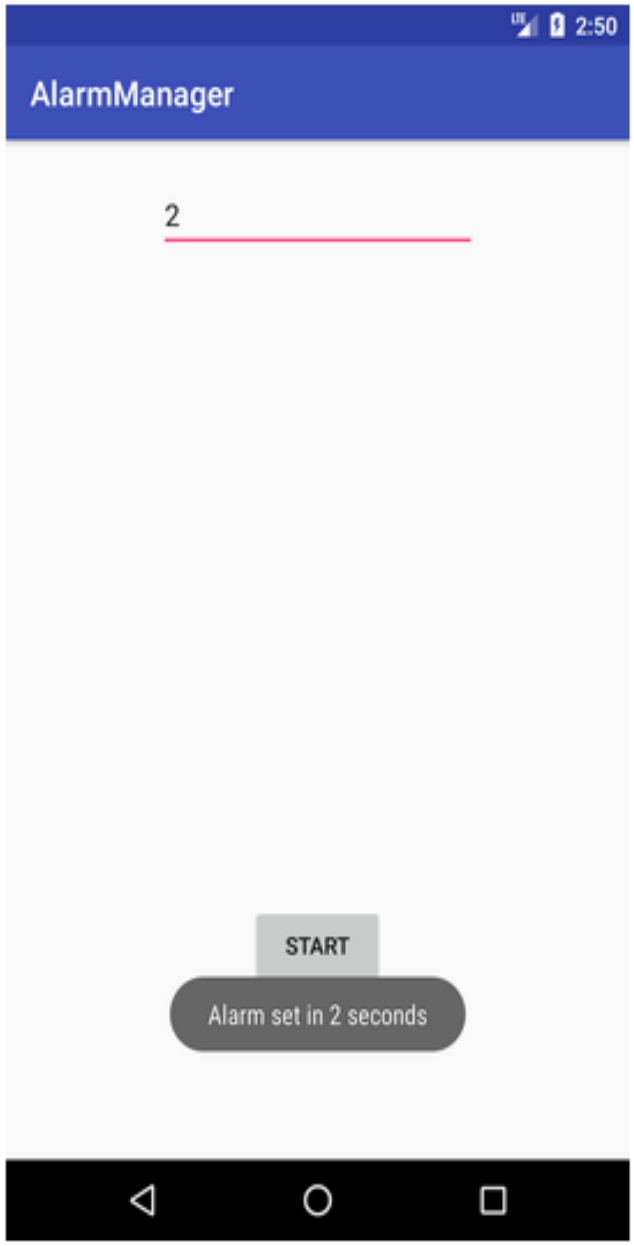
A service is started when component (like activity) calls **startService()** method, now it runs in the background indefinitely. It is stopped by **stopService()** method. The service can stop itself by calling the **stopSelf()** method.

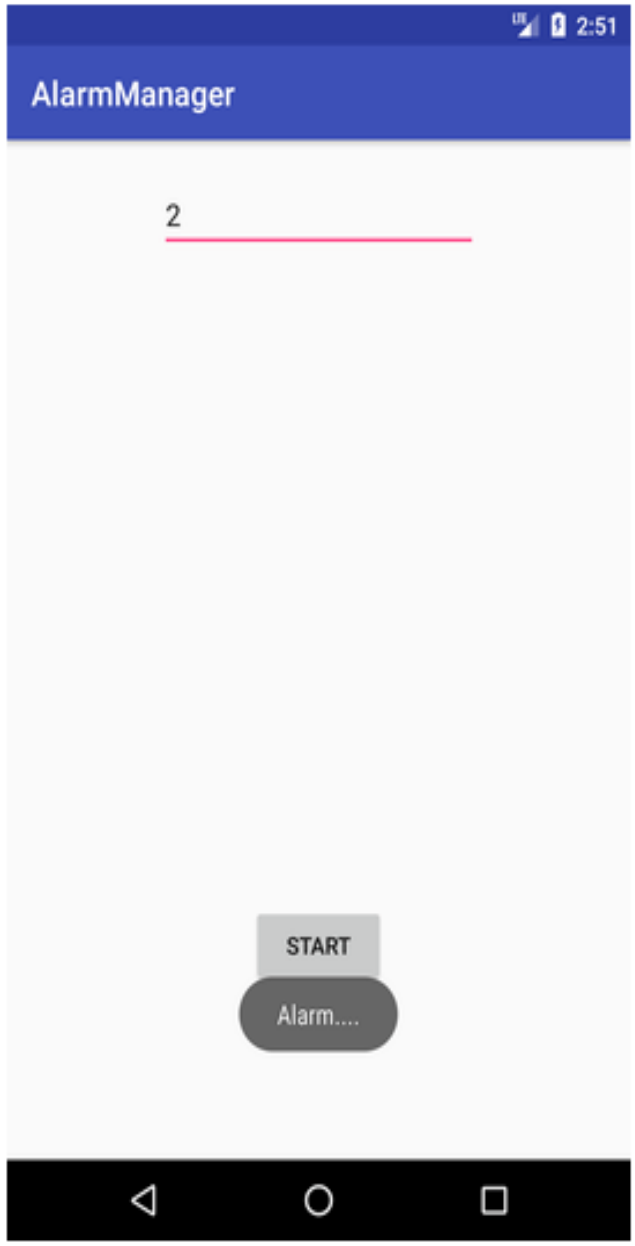
2) Bound Service

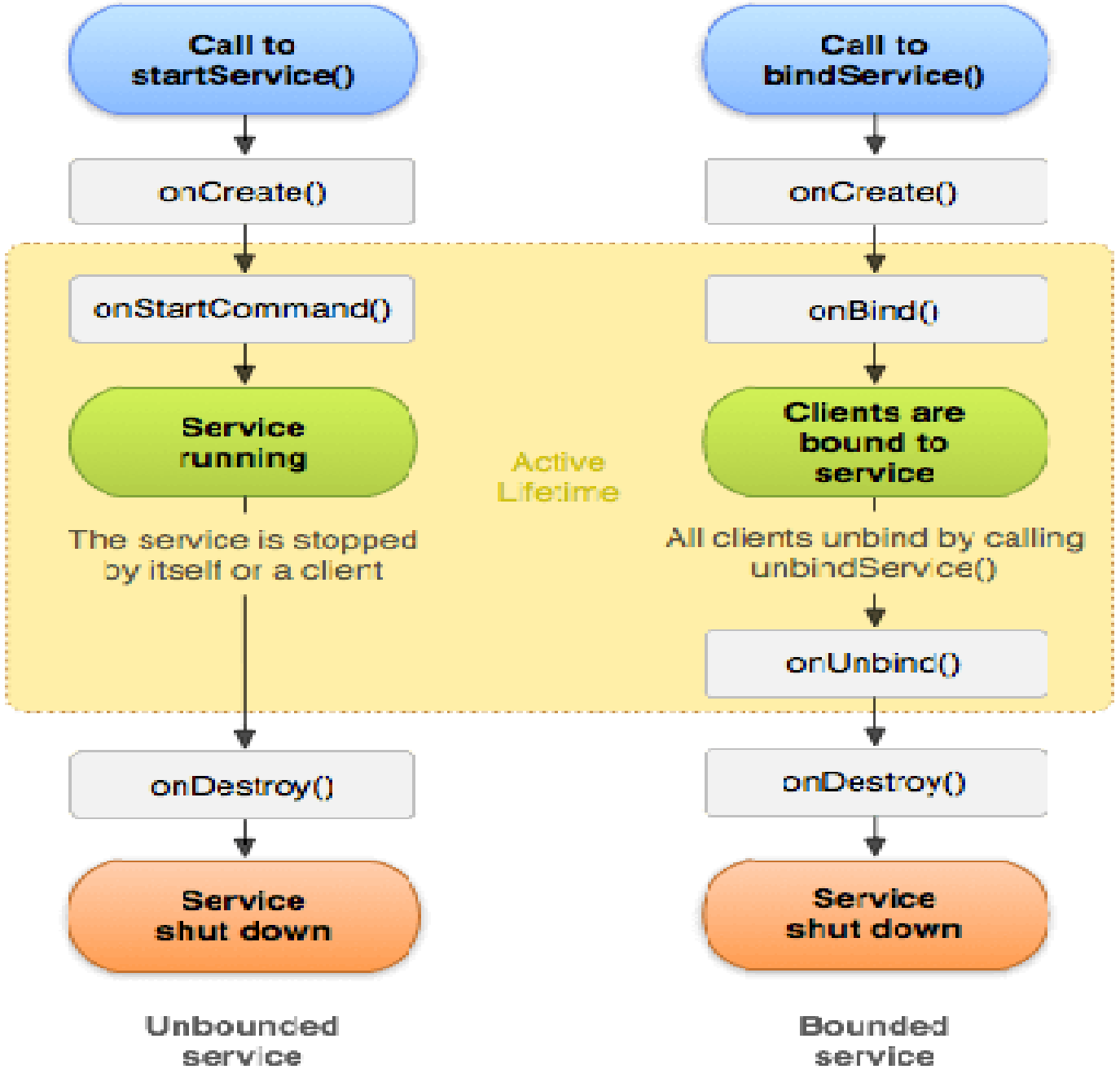
- A service is bound when another component (e.g. client) calls **bindService()** method. The client can unbind the service by calling the **unbindService()** method.
- The service cannot be stopped until all clients unbind the service.

EXAMPLE OF SERVICE








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

BroadCast Receiver

- BroadCast receiver is android components which allows to register for system events
- Different kind of system events which do broadcast
  - Battery low
  - Headset plugged
  - Location change
  - Sms received
  - Call received etc

BroadcastReceiver

- There are following two important steps to make BroadcastReceiver works for the system broadcasted intents −
- Creating the Broadcast Receiver.
- Registering Broadcast Receiver

Building blocks of Android

- Broadcast Receivers **(receiving messages/intents)-**communication between OS and application. eg:trigger alarm upon event.
- A broadcast receiver is implemented as a subclass of BroadcastReceiver class and each message is broadcasted as an ***Intent*** object.

```
        public class MyReceiver  extends  BroadcastReceiver
{
  public void onReceive(context,intent)
                        {
                        }
}
```

- **Context** is an abstract class which allows access to application-specific resources and classes

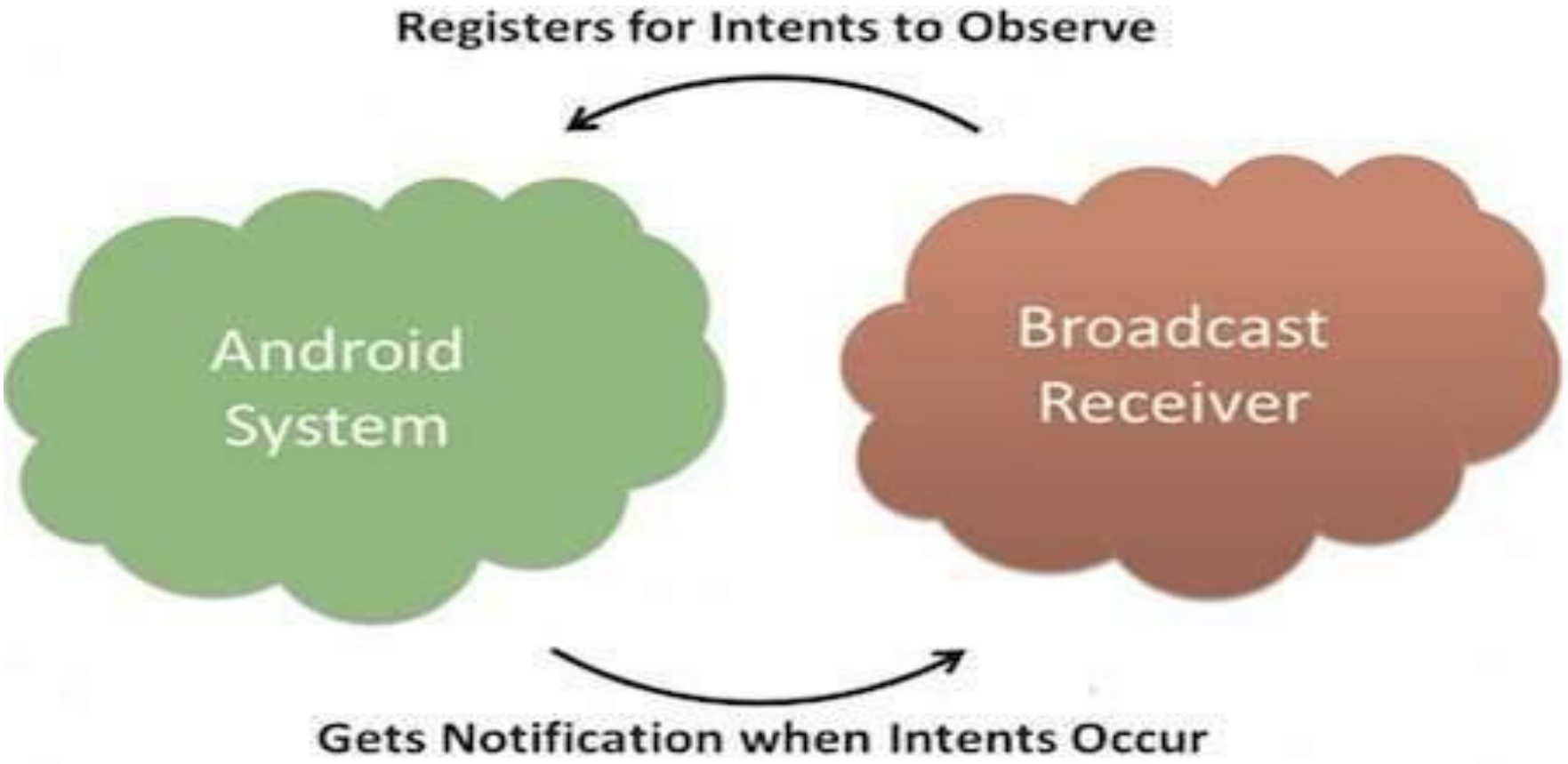
Creating the Broadcast Receiver

- A broadcast receiver is implemented as a subclass of **BroadcastReceiver** class and overriding the onReceive() method where each message is received as a **Intent** object parameter.

```
public class MyReceiver extends BroadcastReceiver
{ @Override public void onReceive(Context context, Intent intent)
 { Toast.makeText(context, "Intent Detected.", Toast.LENGTH_LONG).show();
 }
 }
```

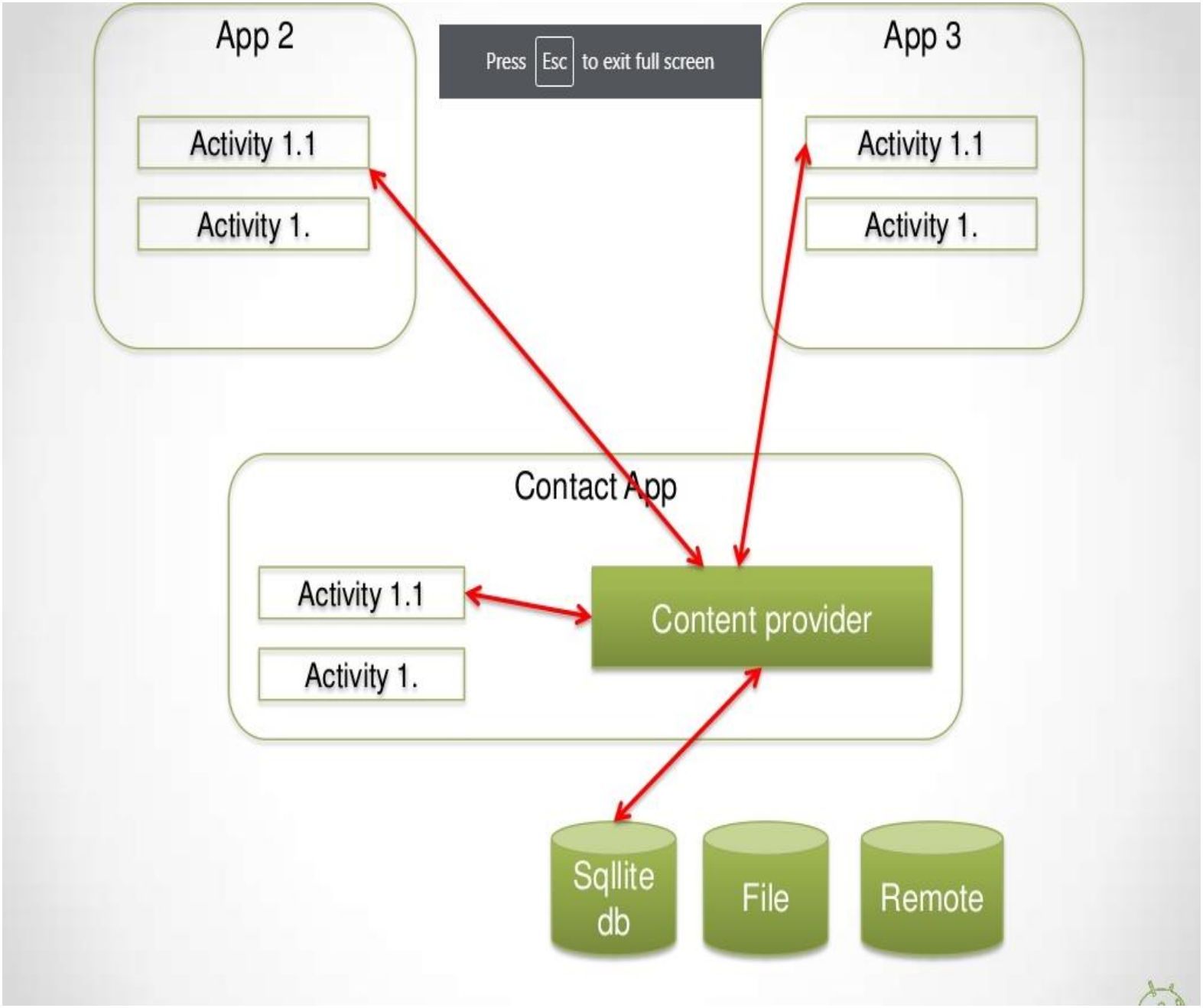
Registering Broadcast Receiver

- An application listens for specific broadcast intents by registering a broadcast receiver in *AndroidManifest.xml* file. Consider we are going to register *MyReceiver* for system generated event ACTION_BOOT_COMPLETED which is fired by the system once the Android system has completed the boot process.



Content Providers

- Helps in managing access to structure set of data
- Content providers are the standard interface that connects data in one process with code running in another process
- Provide a level of abstraction for data store in any format, in the file system, in sqllite or on network
- Android platform provides common data like contacts calendars sms as content providers


App 2
Press Esc to exit full screen
App 3
Activity 1.1
Activity 1.
Activity 1.1
Activity 1.
Contact App
Activity 1.1
Activity 1.
Content provider
Sqllite db
File
Remote

Building blocks of Android

- **Content Providers(store and retrieve data)** They handle data and database management issues.eg: open a phone contact.
- A content provider is implemented as a subclass of **ContentProvider** class and must implement a standard set of APIs that enable other applications to perform transactions.

```
public class MyContentProvider extends ContentProvider
{
    public void onCreate()
        {
        }
}
```

Intents

- Intents are simple message objects which are use to move from 1 activity to another
- Intent defines “intention” of what application want
- Intents can be use to communicate from 1 application to another

Building blocks of Android-Intent

- **Intents** :Messages wiring components together.
- It can be used with **startActivity** to launch an Activity,
- **broadcastIntent** to send it to any interested BroadcastReceivercomponents,and **startService(Intent)** or
- **bindService(Intent, ServiceConnection, int)** to communicate with a background Service.

UI COMPONENTS

- **Views:**UI elements that are drawn on-screen including buttons, lists forms etc. Basic building block of graphical layout.Each view should be described by a View object.

- The View objects are usually called "widgets" and can be one of many subclasses, such as Button or TextView.

- The ViewGroup objects are usually called "layouts"

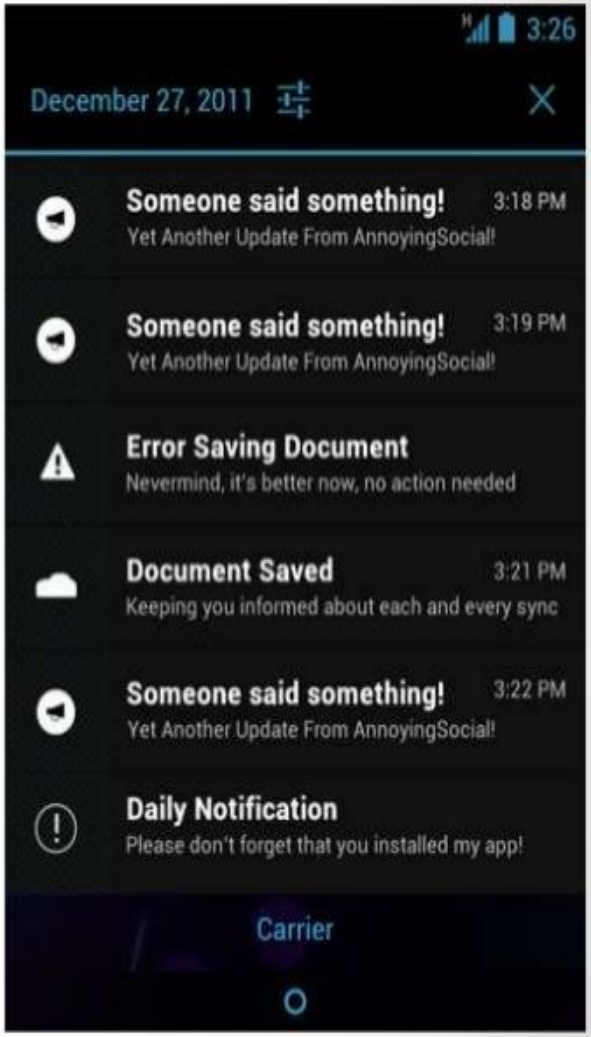
Views

- Views are UI basic building blocks
- Know how to draw themselves
- Respond to events
- Organized as trees to build up GUIs
- Described in XML in layout resources

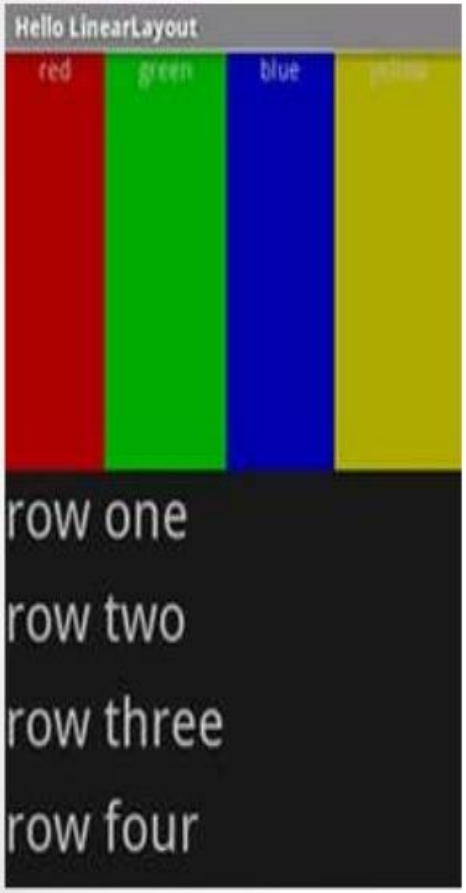

Action Bar
Image Button
Text View
Image View
List View
New Stories 10+
Chandra Patni and 7 other friends like a photo.
Francis Luu added a new photo. — with Adam Mosseri at Facebook HQ.
Like
Comment
Share

NOTIFICATION

- Notifies users about events happening in phone
- It appears as a small icon in notification bar
- User Can interact with this notifications
- Sent via notification manager
- Example SMS, alarm, miss call



Layouts



Linear Layout

Relative Layout

Grid Layout

- To get started, you need to set the notification's content and channel using a NotificationCompat.Builderobject.

```
 NotificationCompat.Builder mBuilder = new
NotificationCompat.Builder(this, CHANNEL_ID)

        .setSmallIcon(R.drawable.notification_icon)
        .setContentTitle(textTitle)
        .setContentText(textContent)
        .setPriority(NotificationCompat.PRIORITY_DEFAULT);
```

- Android manifest file-contains the details of appearance of the first page, theme etc.
- Fragments:Represents a portion of user interface in an Activity.eg:screen can be divided into different portions

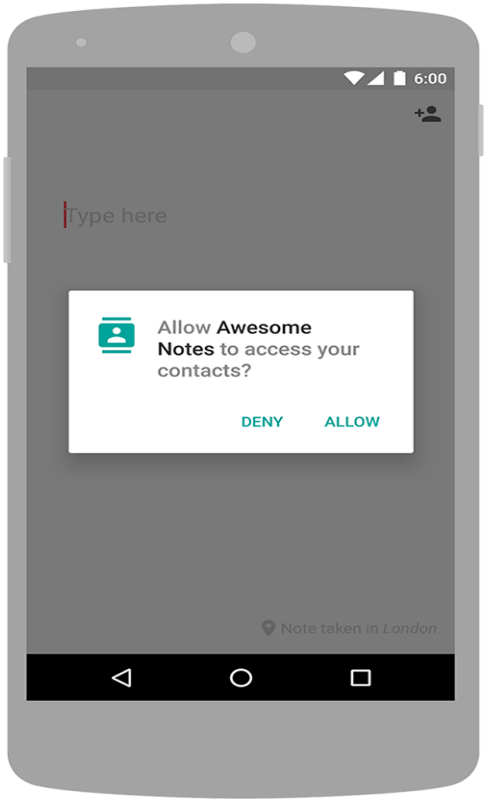
- Resources: directory for UI

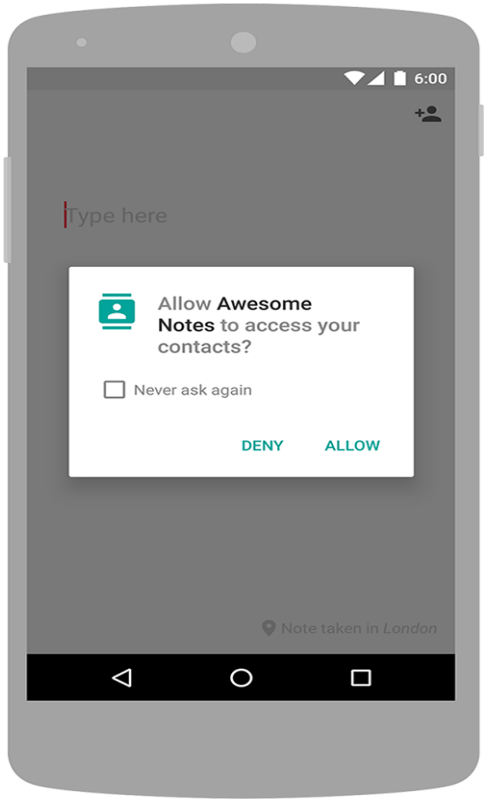
Permissions

- To maintain security for the system and users, Android requires apps to request permission before they can use certain system data and features.
- Depending on how sensitive the area is, the system may grant the permission automatically, or it may ask the user to approve the request.
- <uses-permission> tag shud be included in the manifest file.

- If your app lists *normal* permissions in its manifest (that is, permissions that don't pose much risk to the user's privacy or the device's operation), the system automatically grants those permissions to your app.
- If your app lists *dangerous* permissions in its manifest (that is, permissions that could potentially affect the user's privacy or the device's normal operation), such as the SEND_SMS permission above, the user must explicitly agree to grant those permissions.
-

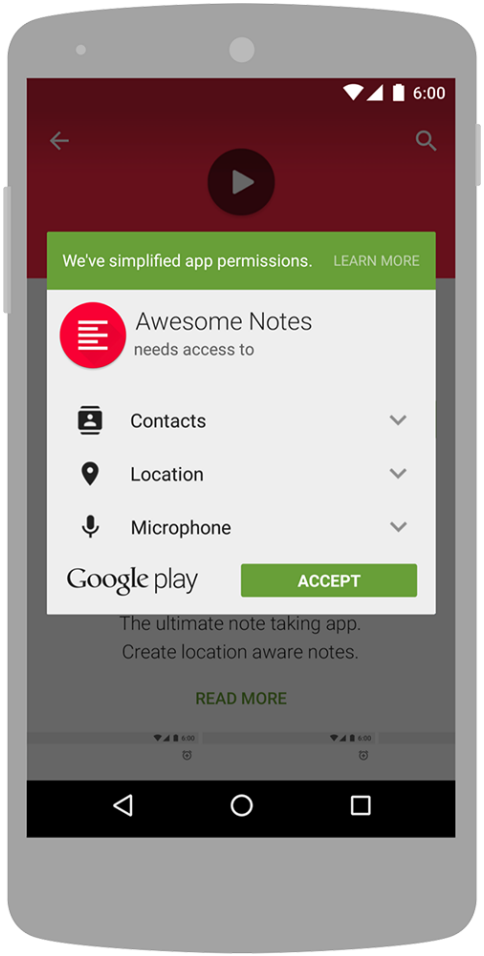
Run time request





If the user denies the permission request, the next time your app requests the permission the dialog will include a checkbox that, when checked, indicates the user does not want to be prompted for the permission again

Install time permission



Example

- <manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.snazzyapp">

- **<uses-permission android:name="android.permission.SEND_SMS"/>**
- **<application>**
- **</application>**
- **</manifest>**

Uses-sdk

- this element is used to specify the API Level, *not* the version number of the SDK (software development kit) or Android platform. The API Level is always a single integer. You cannot derive the API Level from its associated Android version number (for example, it is not the same as the major version or the sum of the major and minor versions).
-

Application programming interface

- API Level is an integer value that uniquely identifies the framework API revision offered by a version of the Android platform.

Emulator-AVD

- An Android Virtual Device (AVD) is a configuration that defines the characteristics of an Android phone, tablet, Wear OS, or Android TV device that you want to simulate in the Android Emulator

SQLite

- SQLite is a opensource SQL database that stores data to a text file on a device. Android comes in with built in SQLite database implementation.
- The main package is android.database.sqlite that contains the classes to manage your own databases
- In order to create a database you just need to call this method openOrCreateDatabase with your database name and mode as a parameter.

SQLiteDatabase mydatabase = openOrCreateDatabase("your database name",MODE_PRIVATE,null);

AndroidManifest.xml file

- The **AndroidManifest.xml file** *contains information of your package*, including components of the application such as activities, services, broadcast receivers, content providers etc.
- It performs some other tasks also:
- It is **responsible to protect the application** to access any protected parts by providing the permissions.
- It also **declares the android api** that the application is going to use.
- It **lists the instrumentation classes**. The instrumentation classes provides profiling and other informations. These informations are removed just before the application is published etc.
- This is the required xml file for all the android application and located inside the root directory.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.javatpoint.hello"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```