RK.UNIVERSITY

One
University
Many
Dreams

# Android Application Development Environment

# Introduction of Android Studio

Android supports java, c++, c# etc. language to develop android applications. Java is the officially supported language for android. All the android examples of this site is developed using Java language and Eclipse IDE.

Here, we are going to tell you, the required softwares to develop android applications using Eclipse IDE. There are two ways to install android.

1. By ADT Bundle
2. By Setup Eclipse Manually

# Introduction of Android Studio

RK UNIVERSITY

SCHOOL OF ENGINEERING RK UNIVERSITY

### 1) By Android Studio

It is the simplest technique to install required software for android application. It includes:

- Eclipse IDE
- Android SDK
- Eclipse Plugin

If you download the Android Studio from android site, you don't need to have eclipse IDE, android SDK and eclipse Plugin because it is already included in Android Studio.

If you have downloaded the Android Studio, unjar it, go to eclipse IDE and start the eclipse by clicking on the eclipse icon. You don't need to do any extra steps here.

### 2) By set up eclipse manually

Visit the next page to learn about setting up android in eclipse manually.

# DVM

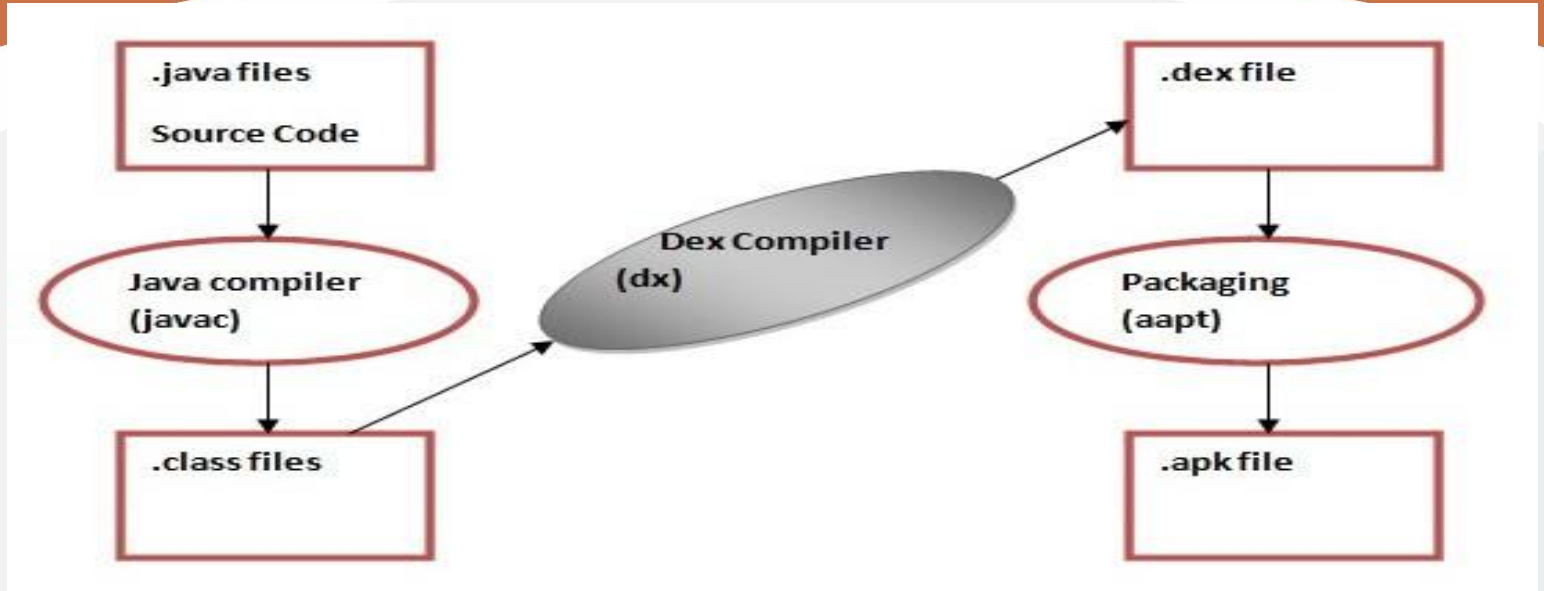


As we know the modern JVM is high performance and provides excellent memory management. But it needs to be optimized for low-powered handheld devices as well.

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. Following is the compiling and packaging process from the source file:

# 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.

# Android SDK



The Android SDK (software development kit) is a set of development tools used to develop applications for Android platform. The Android SDK includes the following:

- Required libraries
- Debugger
- An emulator
- Relevant documentation for the Android application program interfaces (APIs)
- Sample source code

# Android Development Tools

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, 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

 **ADB:** The Android Debug Bridge (adb) is a tool to run commands on a connected Android device

# Android Virtual Device

An Android Virtual Device (AVD) is an emulator configuration that allows developers to test the application by simulating the real device capabilities. We can configure the AVD by specifying the hardware and software options. AVD manager enables an easy way of creating and managing the AVD with its graphical interface. We can create as many AVDs as we need, based on the types of device we want to test for. Below are the steps to create an AVD from AVD manager graphical interface

1) Go to Window ->AVD Manager and select Virtual Devices.
2) Click on New to create a Virtual Device, give it some Name and select Target Android Platform from the drop down list
3) Click “Create AVD”.

RKUNIVERSITY

# Android Directory Structure

SCHOOL OF ENGINEERING RKUNIVERSITY

| Folder Name | Description |
|---|---|
| Src | The 'src' stands for **Source Code.** It contains the Java Source files. |
| Gen | The 'gen' stands for **Generated Java Library.** This library is for Android internal use only. |
| Android 2.2 | The Android Framework Library is stored here. |
| Assets | It is used to store raw asset files. |
| Libs | It contains private libraries. |
| AndroidManifest.xml | This file indicates the Android definition file. This file contains the information about the Android application such as minimum Android version, permission to access Android device capabilities such as Internet access permission, phone permission etc. |

# Android Directory Structure

| Folder Name | Description |
|---|---|
| default.properties | This file contains the project settings, such as build the target. Do not edit this file manually. It should be maintained in a Source Revision Control System. |
| Proguard.cfg | This file defines how ProGuard optimizes and makes your code unclear. |
| MainLayout.xml | This file describes the layout of the page. So all the components such as textboxes, labels, radio buttons, etc. are displaying on the application screen. |
| Activity class | The application occupies the entire device screen which needs at least one class inherits from the Activity class. OnCreate() method initiates the application and loads the layout page. |
| Res | The 'res' stands for Resource file. It can store resource files such as pictures, XML files, etc. It contains some additional folders such as Drawable, Layout and Values.<br><br>anim: It is used for XML files that are compiled into animation objects.<br>color: It is used for XML files that describe colors.<br>drawable: It is used to store various graphics files. In Android project structure, |

# Android Directory Structure

| Folder Name | Description |
|---|---|
| Res | **there are three types of drawable folders,**<br>1. drawable-mdpi 2. drawable-hdpi 3. drawable-ldpi<br><br>The above drawable folders are required in order to adapt to different screen resolutions.<br>**layout:** It is used for placing the XML layout files, which defines how various Android objects such as textbox, buttons, etc. are organized on the screen.<br><br>**menu:** It is used for defining the XML files in the application menu.<br><br>**raw:** The 'raw' stands for **Raw Asset Files.** These files are referenced from the application using a resource identifier in the R class.<br>**For example,** good place for media is MP3 or Ogg files.<br><br>**values:** It is used for XML files which stores various string values, such as titles, labels, etc.<br><br>**xml:** It is used for configuring the application components. |

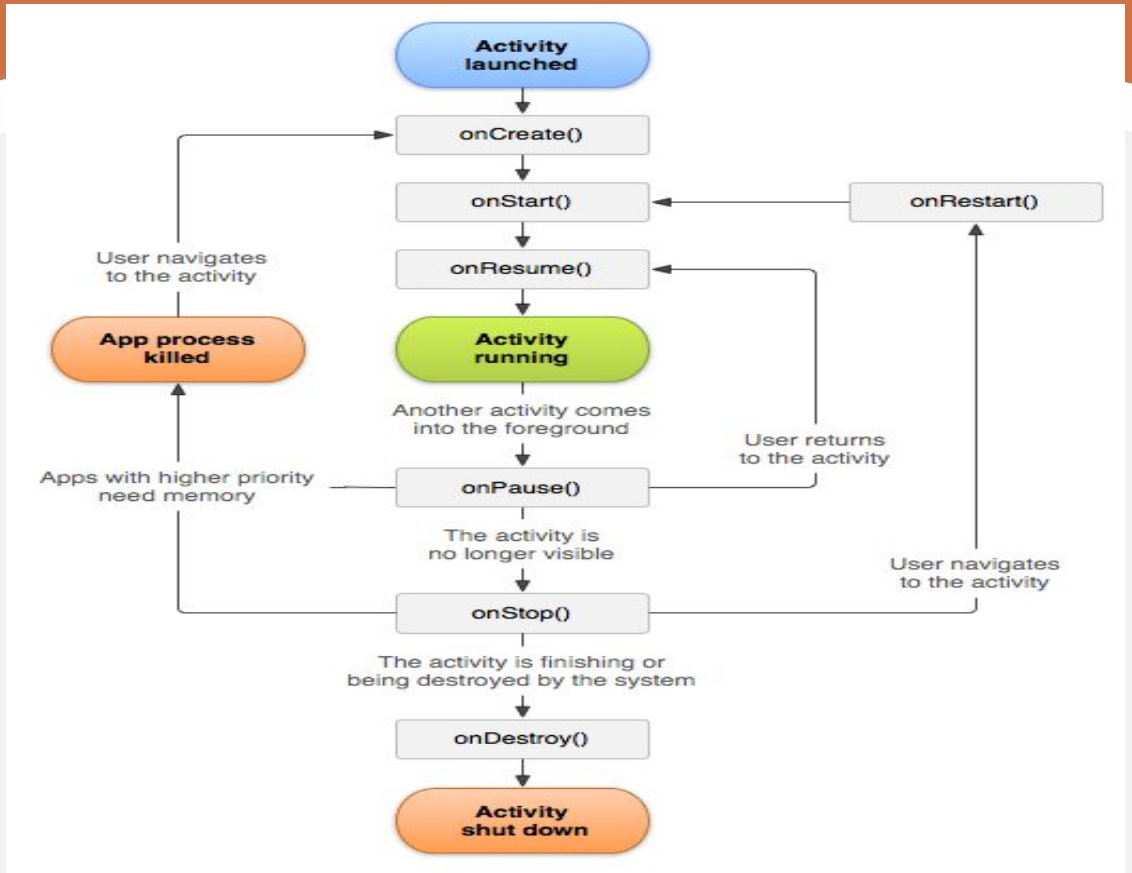
# Activity & Application Life Cycle

An activity is the single screen in android. It is like window or frame of java. By the help of activity, you can place all your UI components or widgets in a single screen. The android activity is the subclass of ContextThemeWrapper class.

**Android activity lifecycle** is controlled by 7 methods of android.app.Activity class. The 7 lifecycle method of activity describes how activity will behave at different states.

| Method | Description |
|---|---|
| onCreate | called when activity is first created. |
| onStart | called when activity is becoming visible to the user. |
| onResume | called when activity will start interacting with the user. |
| onPause | called when activity is not visible to the user. |
| onStop | called when activity is no longer visible to the user. |
| onRestart | called after your activity is stopped, prior to start. |
| onDestroy | called before the activity is destroyed. |

# Activity & Application Life Cycle



# Activity Life Cycle Methods



**Activity Created: onCreate(Bundle savedInstanceState):**

onCreate() method is called when activity gets memory in the OS. **To use create state we need to override onCreate(Bundle savedInstanceState) method.** Now there will be question in mind what is Bundle here, so Bundle is a data repository object that can store any kind of primitive data and this object will be null until some data isn't saved in that.

- When an Activity first call or launched then onCreate(Bundle savedInstanceState) method is responsible to create the activity.
- When ever orientation(i.e. from horizontal to vertical or vertical to horizontal) of activity gets changed or when an Activity gets forcefully terminated by any Operating System then savedInstanceState i.e. object of Bundle Class will save the state of an Activity.
- It is best place to put initialization code.

# Activity Life Cycle Methods



**Activity Started: onStart():**

onStart() method is called just after it's creation. In other case Activity can also be started by calling restart method i.e after activity stop. So this means onStart() gets called by Android OS when user switch between applications. For example, if a user was using Application A and then a notification comes and user clicked on notification and moved to Application B, in this case Application A will be paused. And again if a user again click on app icon of Application A then Application A which was stopped will again gets started.

**Activity Resumed:.onResume():**

Activity resumed is that situation when it is actually visible to user means the data displayed in the activity is visible to user. In lifecycle it always gets called after activity start and in most use case after activity paused (**onPause**).

# Activity Life Cycle Methods



**Activity Paused: onPause():**

Activity is called paused when it's content is not visible to user, in most case onPause() method called by Android OS when user press Home button (Center Button on Device) to make hide.

Activity also gets paused before stop called in case user press the back navigation button. The activity will go in paused state for these reasons also if a notification or some other dialog is overlaying any part (top or bottom) of the activity (screen). Similarly, if the other screen or dialog is transparent then user can see the screen but cannot interact with it. For example, if a call or notification comes in, the user will get the opportunity to take the call or ignore it.

**Activity Stopped: onStop():**

Activity is called stopped when it's not visible to user. Any activity gets stopped in case some other activity takes place of it. For example, if a user was on screen 1 and click on some button and moves to screen 2. In this case Activity displaying content for screen 1 will be stopped.

Every activity gets stopped before destroy in case of when user press back navigation button. So Activity will be in stopped state when hidden or replaced by other activities that have been launched or switched by user. In this case application will not present anything useful to the user directly as it's going to stop.

# Activity Life Cycle Methods

RK.UNIVERSITY

SCHOOL OF ENGINEERING RK.UNIVERSITY

**Activity Restarted: onRestart():**

Activity is called in restart state after stop state. So activity's onRestart() function gets called when user comes on screen or resume the activity which was stopped. In other words, when Operating System starts the activity for the first time onRestart() never gets called. It gets called only in case when activity is resumes after stopped state.

**Activity Destroyed: onDestroy():**

Any activity is known as in destroyed state when it's not in background. There can different cases at what time activity get destroyed.

First is if user pressed the back navigation button then activity will be destroyed after completing the lifecycle of pause and stop.

In case if user press the home button and app moves to background. User is not using it no more and it's being shown in recent apps list. So in this case if system required resources need to use somewhere else then OS can destroy the Activity.

After the Activity is destroyed if user again click the app icon, in this case activity will be recreated and follow the same lifecycle again. Another use case is with Splash Screens if there is call to finish() method from onCreate() of an activity then OS can directly call onDestroy() with calling onPause() and onStop().

# Activity & Application Life Cycle



An activity class loads all the UI component using the XML file available in *res/layout* folder of the project. Following statement loads UI components from *res/layout/activity_main.xml file*:

**setContentView(R.layout.activity_main);**

An application can have one or more activities without any restrictions. Every activity you define for your application must be declared in your *AndroidManifest.xml* file and the main activity for your app must be declared in the manifest with an <intent-filter> that includes the MAIN action and LAUNCHER category as follows:

If either the MAIN action or LAUNCHER category are not declared for one of your activities, then your app icon will not appear in the Home screen's list of apps.

# Activity & Application Life Cycle



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.example.tutorialspoint7.myapplication">
       <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
       android:theme="@style/AppTheme">
       <activity android:name=".MainActivity">
                    <intent-filter>
                      <action android:name="android.intent.action.MAIN" />

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

# Anatomy of Android Application



Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file *AndroidManifest.xml* that describes each component of the application and how they interact. There are following four main components that can be used within an Android application −

## Activities

An activity represents a single screen with a user interface, in-short Activity performs actions on the screen. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. 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

**Ex.:**
```
public class MainActivity extends Activity {
}
```

# Anatomy of Android Application



## Services

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, or it might fetch data over the network without blocking user interaction with an activity. A service is implemented as a subclass of **Service** class as follows −

**Ex.:**
```
public class MyService extends Service {
}
```

## Broadcast Receivers

Broadcast Receivers simply respond to broadcast messages from other applications or from the system. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action. A broadcast receiver is implemented as a subclass of **BroadcastReceiver** class and each message is broadcaster as an **Intent** object.

**Ex.:**
```
public class MyReceiver  extends  BroadcastReceiver {
        public void onReceive(context,intent){}
}
```

# Anatomy of Android Application



## Content Providers

A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the *ContentResolver* class. The data may be stored in the file system, the database or somewhere else entirely.

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.

**Ex.:**

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

# Android Manifest 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.

# Elements Android Manifest File

The elements used in the above xml file are described below.

<manifest>

**manifest** is the root element of the AndroidManifest.xml file. It has **package** attribute that describes the package name of the activity class.

<application>

**application** is the subelement of the manifest. It includes the namespace declaration. This element contains several subelements that declares the application component such as activity etc.

The commonly used attributes are of this element are **icon**, **label**, **theme** etc.

i. **android:icon** represents the icon for all the android application components.

ii. **android:label** works as the default label for all the application components.

iii. **android:theme** represents a common theme for all the android activities.

# Elements Android Manifest File

<activity>
**activity** is the subelement of application and represents an activity that must be defined in the AndroidManifest.xml file. It has many attributes such as label, name, theme, launchMode etc.
**android:label** represents a label i.e. displayed on the screen.
**android:name** represents a name for the activity class. It is required attribute.

<intent-filter>
**intent-filter** is the sub-element of activity that describes the type of intent to which activity, service or broadcast receiver can respond to.

<action>
It adds an action for the intent-filter. The intent-filter must have at least one action element.

<category>
It adds a category name to an intent-filter.

# R.Java File

**Android R.java** is *an auto-generated file by aapt* (Android Asset Packaging Tool) that contains resource IDs for all the resources of res/ directory.
If you create any component in the activity_main.xml file, id for the corresponding component is automatically created in this file. This id can be used in the activity source file to perform any action on the component.

RK.UNIVERSITY

*Thank You*

# ADMISSIONS OPEN

www.rku.ac.in | inquiry@rku.ac.in

+91 97124 89122 | +91 99257 14450

# 50+ PROGRAMS

BAMS | BPHARM | PHARMD | MPHARM | BSC | BSC(MRIT) | MSC | BPT | DMLT
DIPLOMA | BTECH | D2D | MTECH | BCOM | BBA | BBA (APPLIED MANAGEMENT) | MBA | BCA | MCA

**Main Campus** RK University, Bhavnagar HWY, Kasturbadham, Rajkot **City Campus** 2nd Ring Rd, Nr. Kalawad Rd, Mota Mawa, Rajkot

/rkuindia @rkuniversity /school/rkuniversity rku.ac.in/WhatsApp @rkuniversity