15 Marks

1. Android Architecture

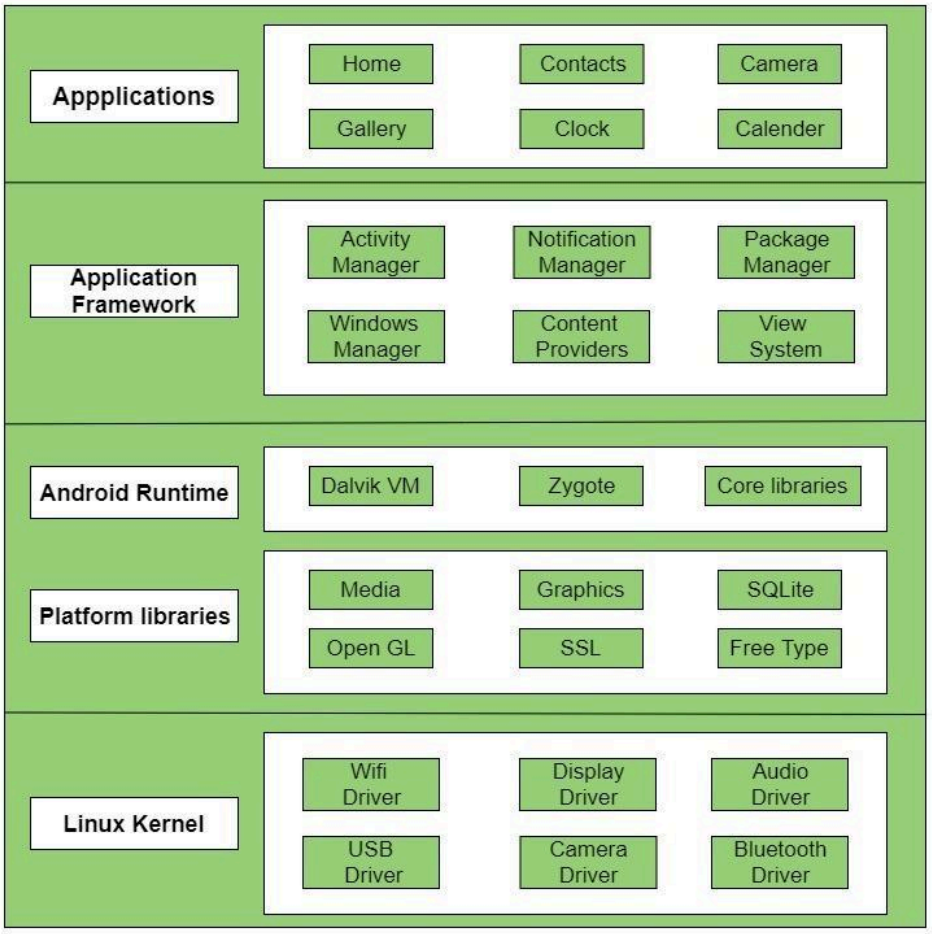
Android architecture contains different number of components to support any android device needs. Android software contains an open-source Linux Kernel having collection of number of C/C++ libraries which are exposed through an application framework services.

Among all the components Linux Kernel provides main functionality of operating system functions to smartphones and Dalvik Virtual Machine (DVM) provide platform for running an android application.s

The main components of android architecture are following:-

- Applications
- Application Framework
- Android Runtime
- Platform Libraries
- Linux Kernel

Pictorial representation of android architecture with several main components and their sub components –



Applications –

Applications is the top layer of android architecture. The pre-installed applications like home, contacts, camera, gallery etc and third party applications downloaded from the play store like chat applications, games etc. will be installed on this layer only.

It runs within the Android run time with the help of the classes and services provided by the application framework.

Application framework –

Application Framework provides several important classes which are used to create an Android application. It provides a generic abstraction for hardware access and also helps in managing the user interface with application resources. Generally, it provides the services with the help of which we can create a particular class and make that class helpful for the Applications creation.

It includes different types of services activity manager, notification manager, view system, package manager etc. which are helpful for the development of our application according to the prerequisite.

Application runtime –

Android Runtime environment is one of the most important part of Android. It contains components like core libraries and the Dalvik virtual machine(DVM). Mainly, it provides the base for the application framework and powers our application with the help of the core libraries.

Like Java Virtual Machine (JVM), **Dalvik Virtual Machine (DVM)** is a register-based virtual machine and specially designed and optimized for android to ensure that a device can run multiple instances efficiently. It depends on the layer Linux kernel for threading and low-level memory management. The core libraries enable us to implement android applications using the standard JAVA or Kotlin programming languages.

Platform libraries –

The Platform Libraries includes various C/C++ core libraries and Java based libraries such as Media, Graphics, Surface Manager, OpenGL etc. to provide a support for android development.

- **Media** library provides support to play and record an audio and video formats.
- **Surface manager** responsible for managing access to the display subsystem.
- **SGL** and **OpenGL** both cross-language, cross-platform application program interface (API) are used for 2D and 3D computer graphics.
- **SQLite** provides database support and **FreeType** provides font support.

- **Web-Kit** This open source web browser engine provides all the functionality to display web content and to simplify page loading.
- **SSL (Secure Sockets Layer)** is security technology to establish an encrypted link between a web server and a web browser.

Linux Kernel –

Linux Kernel is heart of the android architecture. It manages all the available drivers such as display drivers, camera drivers, Bluetooth drivers, audio drivers, memory drivers, etc. which are required during the runtime.

The Linux Kernel will provide an abstraction layer between the device hardware and the other components of android architecture. It is responsible for management of memory, power, devices etc.

The features of Linux kernel are:

- **Security:** The Linux kernel handles the security between the application and the system.
- **Memory Management:** It efficiently handles the memory management thereby providing the freedom to develop our apps.
- **Process Management:** It manages the process well, allocates resources to processes whenever they need them.
- **Network Stack:** It effectively handles the network communication.
- **Driver Model:** It ensures that the application works properly on the device and hardware manufacturers responsible for building their drivers into the Linux build.

2. Android Intent

- An Android application can contain zero or more activities. When your application has more than one activity, you often need to navigate from one to another.
- In Android, you navigate between activities is done through an ***intent.***
- It is the *message* that is passed between components such as activities, content providers, broadcast receivers, services etc.
- It is generally used with startActivity() method to invoke activity, broadcast receivers etc.
- The LabeledIntent is the subclass of android.content.Intent class
- Android intents are mainly used to:
  Start the service
  Launch an activity
  Display a web page
  Display a list of contacts
  Broadcast a message
  Dial a phone call etc.
- **Types of Android Intents:** There are two types of intents in android: implicit and explicit.
  1) Implicit Intent 2) Explicit Intent

Implicit Intent

- It doesn't specify the component.
- In such case, intent provides information of available components provided by the system that is to be invoked.
- **example**

```
Intent intent=new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse("http://www.javatpoint.com"));
startActivity(intent);
```

Explicit Intent

- Explicit Intent specifies the component.
- In such case, intent provides the external class to be invoked.
- **Example**:

```
Intent i = new Intent(getApplicationContext(),ActivityTwo.class);
startActivity(i);
```

Returning Results from an Intent

- The **startActivity()** method invokes another activity but does not return a result to the current activity.
- The information entered by the user in that activity needs to be passed back to the calling activity for further processing.
- If you need to pass data back from an activity, you should instead use the startActivityForResult() method.
- To call an activity and wait for a result to be returned from it, you need to use the startActivityForResult() method
- **Example**:
  startActivityForResult(**new**
  Intent("net.learn2develop.SecondActivity"),request_Code);

Returning Results from an Intent

- In order for an activity to return a value to the calling activity, you use an Intent object to send data back via the **setData()** method:
- **Example**:
  Intent data = **new Intent();**
  EditText txt_username = (EditText) findViewById(R.id.txt_username);
  data.setData(Uri.*parse(*txt_username.getText().toString()));
  setResult(RESULT_OK, data);
  finish();
- The setResult() method sets a result code (either RESULT_OK or RESULT_CANCELLED) and the data (an Intent object) to be returned back to the calling activity.
- The finish() method closes the activity and returns control back to the calling activity.

Passing data using an intent object

- the putExtra() method of an Intent object to add a name/value pair:
- use putExtra() to add new name/value pairs
- **example**:
  Intent i = new
  Intent("net.learn2develop.PassingDataSecondActivity");
  i.putExtra("str1", "This is a string");
  i.putExtra("age1", 25);

Passing data using Bundle object

- The preceding statements add two name/value pairs to the Intent object: one of type string and one of type integer.
- we can also create a **Bundle object** and then attach it using the putExtras() method.
- Think of a Bundle object as a dictionary object — it contains a set of name/ value pairs.
- The following statements create a Bundle object and then add two name/value pairs to it. It is then attached to the Intent object:
- **Example**:

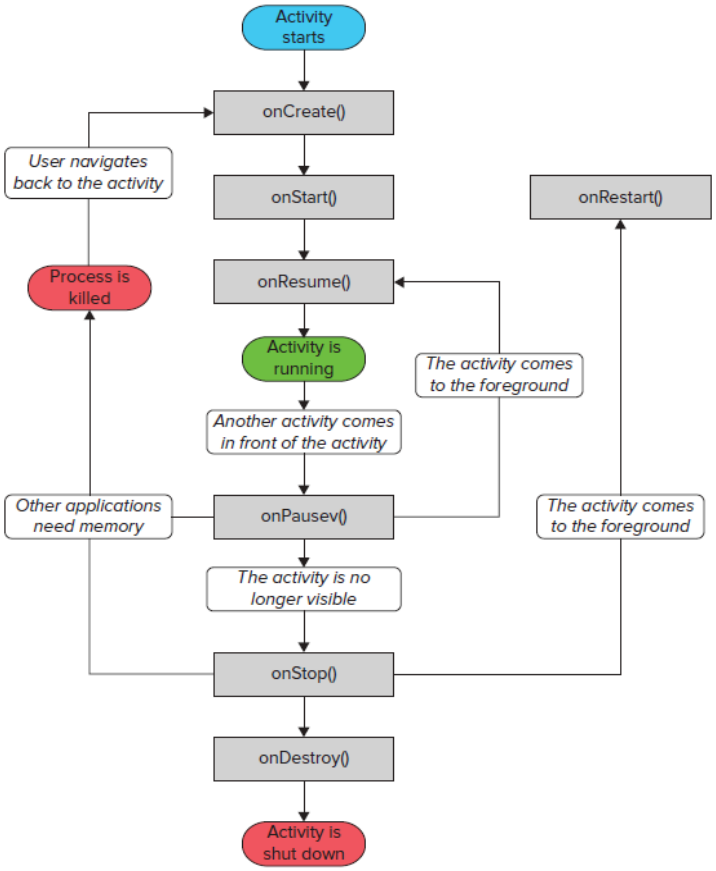
```
Bundle extras = new Bundle();
extras.putString("str2", "This is another string");
extras.putInt("age2", 35);
//---attach the Bundle object to the Intent object---
i.putExtras(extras);
```

3. Life cycle of Android Activity with example program

- In Android, an *activity* is a window that contains the user interface for your application, and users interact directly with the activities of your applications.
- An Android application can contain zero or more activities.
- When your application has more than one activity, you often need to navigate from one to another.
- In Android, you navigate between activities through an intent.
- To create an activity, you create a Java class that extends the Activity base class:

```
package net.learn2develop.Activities;
import android.app.Activity;
import android.os.Bundle;
public class MainActivity extends Activity {
/** Called when the activity is first created. */
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
} }
```

- The Activity base class defines a series of events that governs the life cycle of an activity.
  - onCreate() — Called when the activity is fi rst created
  - onStart() — Called when the activity becomes visible to the user
  - onResume() — Called when the activity starts interacting with the user
  - onPause() — Called when the current activity is being paused and the previous activity is being resumed
  - onStop() — Called when the activity is no longer visible to the user
  - onDestroy() — Called before the activity is destroyed by the system (either manually or by the system) to conserve memory
  - onRestart() — Called when the activity has been stopped and is restarting again




```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="example.javatpoint.com.activitylifecycle.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
```

```
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

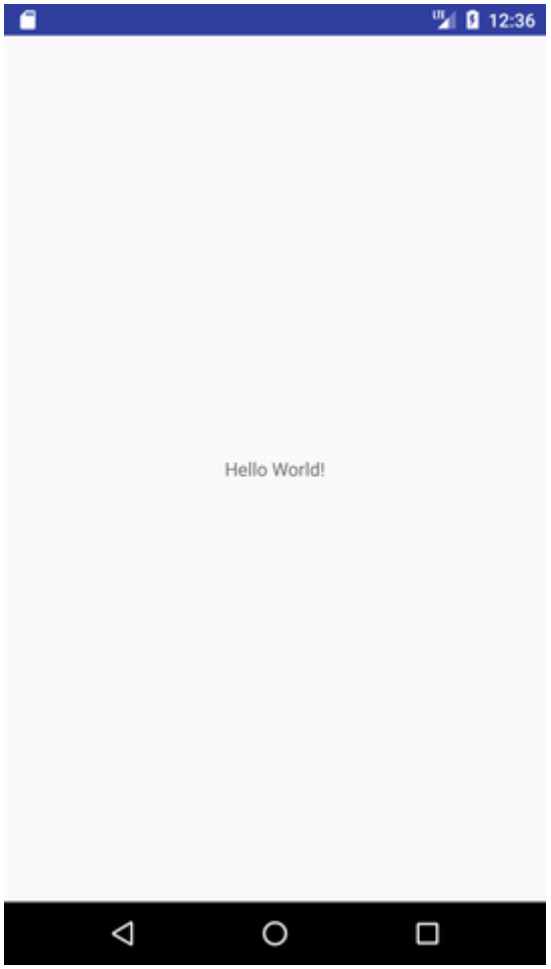
```
package example.javatpoint.com.activitylifecycle;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d("lifecycle","onCreate invoked");
    }
    @Override
    protected void onStart() {
        super.onStart();
        Log.d("lifecycle","onStart invoked");
    }
    @Override
    protected void onResume() {
        super.onResume();
        Log.d("lifecycle","onResume invoked");
    }
    @Override
    protected void onPause() {
        super.onPause();
        Log.d("lifecycle","onPause invoked");
    }
    @Override
    protected void onStop() {
```

```
        super.onStop();
        Log.d("lifecycle","onStop invoked");
    }
    @Override
    protected void onRestart() {
        super.onRestart();
        Log.d("lifecycle","onRestart invoked");
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.d("lifecycle","onDestroy invoked");
    }
}
```



8 Marks

1. Features of Android

- Android Open Source Project
- Android supports different types of connectivity for GSM, CDMA, Wi-Fi, Bluetooth, etc. for telephonic conversation or data transfer.
- It contains multiple APIs to support location-tracking services such as GPS.
- We can manage all data storage related activities by using the file manager.
- It contains a wide range of media supports like AVI, MKV, FLV, MPEG4, etc. to play or record a variety of audio/video.
- It also supports different image formats like JPEG, PNG, GIF, BMP, MP3, etc.
- It supports multimedia hardware control to perform playback or recording using a camera and microphone.
- Android has an integrated open-source WebKit layout based web browser to support User Interface like HTML5, CSS3.
- Android supports multi-tasking means we can run multiple applications at a time and can switch in between them.
- It provides support for virtual reality or 2D/3D Graphics
- Hardware support — Accelerometer Sensor, Camera, Digital Compass, Proximity Sensor, and GPS
    - Multi-touch — Supports multi-touch screens
    - Multi-tasking — Supports multi-tasking applications
    - Flash support — Android 2.3 supports Flash 10.1.
- Tethering — Supports sharing of Internet connections as a wired/wireless hotspot

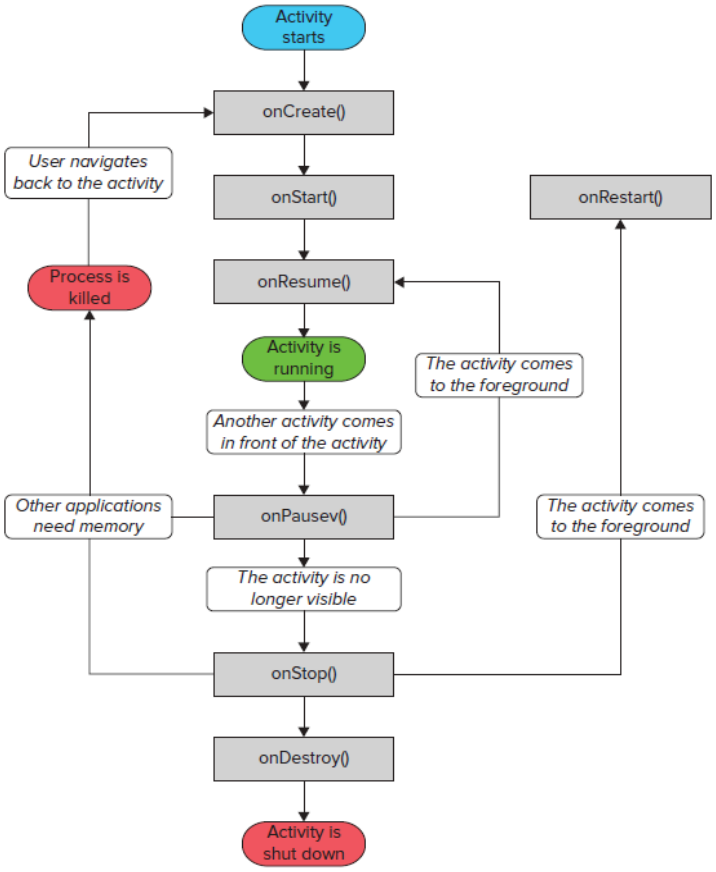
2. ACTIVITIES in Android

- In Android, an *activity* is a window that contains the user interface for your application, and users interact directly with the activities of your applications.
- An Android application can contain zero or more activities.

- When your application has more than one activity, you often need to navigate from one to another.
- In Android, you navigate between activities through an intent.
- To create an activity, you create a Java class that extends the Activity base class:

```
package net.learn2develop.Activities;
import android.app.Activity;
import android.os.Bundle;
public class MainActivity extends Activity {
/** Called when the activity is first created. */
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
} }
```

- The Activity base class defines a series of events that governs the life cycle of an activity.
  - onCreate() — Called when the activity is fi rst created
  - onStart() — Called when the activity becomes visible to the user
  - onResume() — Called when the activity starts interacting with the user
  - onPause() — Called when the current activity is being paused and the previous activity is being resumed
  - onStop() — Called when the activity is no longer visible to the user
  - onDestroy() — Called before the activity is destroyed by the system (either manually or by the system) to conserve memory
  - onRestart() — Called when the activity has been stopped and is restarting again



3.History and Versions in Android

History of Android

1) Initially, Andy Rubin founded Android Incorporation in Palo Alto, California, United States in October, 2003.

2) In 17th August 2005, Google acquired android Incorporation. Since then, it is in the subsidiary of Google Incorporation.

3) The key employees of Android Incorporation are Andy Rubin, Rich Miner, Chris White and Nick Sears.

4) Originally intended for camera but shifted to smart phones later because of low market for camera only.

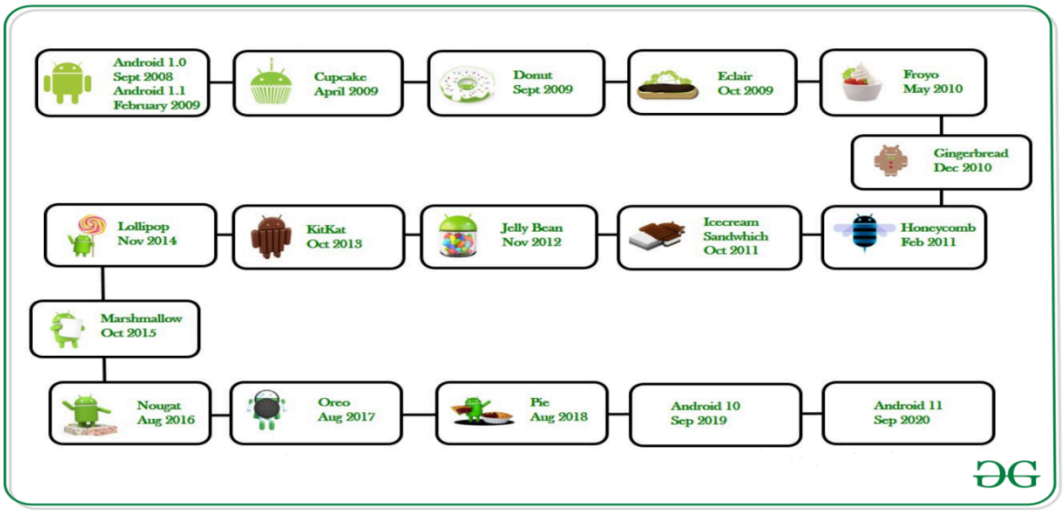
5) Android is the nick name of Andy Rubin given by coworkers because of his love to robots.

6) In 2007, Google announces the development of android OS.

7) In 2008, HTC launched the first android mobile.

Android Versions

Google launched the first version of the Android platform on Nov 5, 2007. Since then, Google released a lot of android versions such as Apple Pie, Banana Bread, Cupcake, Donut, Éclair, Froyo, Gingerbread, Jellybeans, Kitkat, Lollipop, marshmallow, Nougat, Oreo, etc. with extra functionalities and new features.



The following table shows the version details of android which is released by Google from 2007 to date.

Code Name	Version	API level	Release date
Apple Pie	Android 1.0	1	September 23, 2008
Banana Bread	Android 1.1	2	February 9, 2009
Cupcake	Android 1.5	3	April 30, 2009
Donut	Android 1.6	4	September 15, 2009
Eclair	Android 2.0 – 2.1	5-7	October 26, 2009
Froyo	Android 2.2 – 2.2.3	8	May 20, 2010
Gingerbread	Android 2.3 – 2.3.4	9-10	December 6, 2010
Honeycomb	Android 3.0.x – 3.2.x	11 – 13	February 22, 2011
Ice Cream Sandwich	Android 4.0 – 4.0.4	14 – 15	October 18, 2011
Jelly Bean	Android 4.1 – 4.1.2	16 – 18	July 9, 2012
Kitkat	Android 4.4 – 4.4.4	19	July 9, 2012
Lollipop	Android 5.0 – 5.1	21 – 22	October 17, 2014
Marshmallow	Android 6.0 – 6.0.1	23	October 5, 2015
Nougat	Android 7.0 – 7.1	24 – 25	August 22, 2016
Oreo	Android 8.0	26	August 21, 2017
Pie	Android 9.0	27	August 6, 2018
Android Q	Android 10.0	29	September 3, 2019

Android 11	Android 11.0	30	September 8, 2020

4. Setting up Android for Eclipse IDE

How to setup Android for Eclipse IDE

1. Install the JDK
2. Download and install the Eclipse for developing android application
3. Download and Install the android SDK
4. Intall the ADT plugin for eclipse
5. Configure the ADT plugin
6. Create the AVD

1) Install the JDK
For creating android application, JDK must be installed if you are developing the android application with Java language

---

2) Download and install the Eclipse IDE

For developing the android application using eclipse IDE, you need to install the Eclipse

---

3) Download and install the android SDK
The Android SDK contains a debugger, libraries, an emulator, documentation, sample code, and tutorials.

First of all, download the android SDK.

Now double click on the exe file, it will be installed.

---

4) Download the ADT plugin for eclipse

ADT (Android Development Tools) is required for developing the android application in the eclipse IDE. It is the plugin for Eclipse IDE that is designed to provide the integrated environment.

For downloading the ADT, you need to follow these steps:

1) Start the eclipse IDE, then select **Help** > **Install new software...**

2) In the **work with** combo box,
write **https://dl-ssl.google.com/android/eclipse/**
3) **select the checkbox** next to Developer Tools and **click next**

4) You will see, a list of tools to be downloaded here, **click next**

5) **click finish**

6) After completing the installation, restart the eclipse IDE

---

5) Configuring the ADT plugin

After the installing ADT plugin, now tell the eclipse IDE for your android SDK location. To do so:

1. Select the **Window menu > preferences**
2. Now select the android from the left panel. Here you may see a dialog box asking if you want to send the statistics to the google. Click **proceed**.
3. Click on the browse button and locate your SDK directory e.g. my SDK location is C:\Program Files\Android\android-sdk .
4. Click the apply button then OK.

---

6) Create an Android Virtual Device (AVD)

For running the android application in the Android Emulator, you need to create and AVD. For creating the AVD:

1. Select the **Window menu > AVD Manager**
2. Click on the **new** button, to create the AVD
3. Now a dialog appears, write the AVD name e.g. myavd. Now choose the target android version e.g. android2.2.
4. click the **create AVD**

---

7) create and run the simple android example

Visit the next page to create first android application.

5 marks

1. Write brief notes on following :

i)Java JDK ii)Android SDK iii)AVD iv)ADT

- **Java JDK**
  - The Android SDK makes use of the Java SE Development Kit (JDK). Hence, if your computer does not have the JDK installed, you should start off by downloading the JDK
- **The Android SDK**
  - The Android SDK contains a debugger, libraries, an emulator, documentation, sample code, and tutorials.
  - When the SDK Manager is started, it first checks for the packages that are available for installation.
  - The packages contain the documentation and SDK specifi c to each version of the Android OS.
  - They also contain sample code and tools for the various platforms
- **Android Virtual Devices (AVDs)**

- An AVD is an emulator instance that enables you to model an actual device.
- Each AVD consists of a hardware profile, a mapping to a system image, as well as emulated storage, such as a secure digital (SD) card.
- You can create as many AVDs as you want in order to test your applications with several different configurations.

 Android Development Tools (ADT)

- The Android Development Tools (ADT) plug-in for Eclipse is an extension to the Eclipse IDE that supports the creation and debugging of Android applications.
- Create new Android application projects
- Access the tools for accessing your Android emulators and devices
- Compile and debug Android applications
- Export Android applications into Android Packages (APKs)
- Create digital certificates for code-signing your APK

2. Write notes on Toasts Message with example

- A toast provides simple feedback about an operation in a small popup.
- It only fills the amount of space required for the message and the current activity remains visible and interactive.
- Toasts automatically disappear after a timeout
- instantiate a **Toast** object with one of the **makeText()** methods.
- This method takes three parameters:
  - the application **Context**
  - the text message,
  - the duration for the toast.
- It returns a properly initialized Toast object.
- we can display the toast notification with **show()**
- **Example** :

```
Context context = getApplicationContext();
CharSequence text = "Hello toast!";
int duration = Toast.LENGTH_SHORT;
Toast toast = Toast. makeText(context, text, duration);
toast.show();
```

Positioning the Toast

- A standard toast notification appears near the bottom of the screen, centered horizontally.
- You can change this position with the **setGravity(int, int, int**) method.
- This accepts three parameters: a **Gravity** constant, an x-position offset, and a y-position offset.

- **Ex** : toast.setGravity(Gravity.TOP|Gravity.LEFT, 0, 0)

3. With Sample code explain the concept of returning result from intent.
Returning Results from an Intent

- The **startActivity()** method invokes another activity but does not return a result to the current activity.
- The information entered by the user in that activity needs to be passed back to the calling activity for further processing.
- If you need to pass data back from an activity, you should instead use the startActivityForResult() method.
- To call an activity and wait for a result to be returned from it, you need to use the startActivityForResult() method
- **Example**:

```
startActivityForResult(new
Intent(“net.learn2develop.SecondActivity”),request_Code);
```

- In order for an activity to return a value to the calling activity, you use an Intent object to send data back via the **setData()** method:
- **Example**:

```
Intent data = new Intent();
EditText txt_username =
(EditText) findViewById(R.id.txt_username);
data.setData(Uri.parse(
txt_username.getText().toString()));
setResult(RESULT_OK, data);
finish();
```

- The setResult() method sets a result code (either RESULT_OK or RESULT_CANCELLED) and the data (an Intent object) to be returned back to the calling activity.
- The finish() method closes the activity and returns control back to the calling activity.

4. List out the methods used to pass data between intends.
Passing data using an intent object

- the **putExtra**() method of an Intent object to add a name/value pair:
- use putExtra() to add new name/value pairs
- **Example**:

```
Intent i = new Intent(“net.learn2develop.PassingDataSecondActivity”);
i.putExtra(“str1”, “This is a string”);
i.putExtra(“age1”, 25);
```

- The preceding statements add two name/value pairs to the Intent object: one of type string and one of type integer.

Passing data using an intent object

- we can also create a **Bundle object** and then attach it using the putExtras() method.
- Think of a Bundle object as a dictionary object — it contains a set of name/ value pairs.
- The following statements create a Bundle object and then add two name/value pairs to it. It is then attached to the Intent object:
- **Example**:

```
Bundle extras = new Bundle();
extras.putString("str2", "This is another string");
extras.putInt("age2", 35);
//---attach the Bundle object to the Intent object---
i.putExtras(extras);
```