17CAU304A ANDROID PROGRAMMING 3H – 3C

Instruction Hours / week: L: 3 T: 0 P: 0 **Marks:** Int : **40** Ext : **60** Total: **100**

Scope

This is an introductory Android programming course designed to introduce and familiarize participants with programming in the Android environment.

Objectives

- To explain the differences between Android and other mobile development environments.
- teach students to design, create, deploy, and test applications for the Android mobile phone platform.
- introduce students to the most common tools and techniques for writing Android applications.
- to explain how Android applications work, their life cycle, manifest, Intents, and using external resources.
- to teach to access and work with databases under android OS
- design and develop useful Android applications with compelling user. interfaces .by using, extending, and creating your own layouts and Views and using Menus.

Unit-I

Introduction: History of Android, Introduction to Android Operating Systems, Android Development Tools, Android Architecture. (2L)

Unit-II

Overview of object oriented programming using Java: OOPs Concepts: Inheritance, Polymorphism, Interfaces, Abstract class, Threads, Overloading and Overriding, Java Virtual Machine. (4L)

Unit-III

Development Tools: Installing and using Eclipse with ADT plug-in, Installing Virtual machine for Android sandwich/Jelly bean (Emulator), configuring the installed tools, creating a android project– Hello Word, run on emulator, Deploy it on USB-connected Android device. (5L)

Unit-IV

User Interface Architecture: Application context, intents, Activity life cycle, multiple screen size s.(2L) **User Interface Design:** Form widgets, Text Fields, Layouts, Button control, toggle buttons, Spinners(Combo boxes),Images, Menu, Dialog.(2L)

Unit-V

Database: Understanding of SQL database, connecting with the database. (2L)

Suggested readings

1. James C.Sheusi,(2013). *Android application development for Java for Java programmers*, Cengage Learning.

Websites

1. http://www.developer.android.com
2. http://developer.android.com/about/versions/index.html
3. http://developer.android.com/training/basics/firstapp/index.html
4. http://developer.android.com/guide/components/activities.html
5. http://developer.android.com/guide/components/fundamentals.html
6. http://developer.android.com/guide/components/intents-filters.html.
7. http://developer.android.com/training/multiscreen/screensizes.html
8. http://developer.android.com/guide/topics/ui/controls.html
9. http://developer.android.com/guide/topics/ui/declaring-layout.html
10. http://developer.android.com/training/basics/data-storage/databases.html

Enable | Enlighten | Enrich
KARPAGAM UNIVERSITY
(Under Section 3 of UGC Act 1956)

KarpagamAcademy of Higher Education
(Established Under Section 3 of UGC Act 1956)
Eachanari Post, Coimbatore – 641 021. INDIA
Phone : 0422-2611146, 2611082 Fax No : 0422 -2611043

17CAU304A ANDROID PROGRAMMING

LECTURE PLAN

S.No	Lecturer Duration(Hrs)	Topics to be Covered	Support Materials
UNIT I			
1	1	History of Android	W1
2	1	Introduction to Android Operating Systems	W2
3	1	Android Development Tools	T1:7-8
4	1	Android Architecture	W2
5	1	Recapitulation and discussion of Important Questions	
Total No.of Hours planned for Unit-I			**5 Hours**

TEXT BOOK:

T1: James C. Sheusi (2013), Android Application development for java Programmers, Cengage Learning.

WEBSITES

W1:http://www.developer.android.com

W2: www.androidauthority.com

LECTURE PLAN

S.No	Lecturer Duration(Hrs)	Topics to be Covered	Support Materials
		UNIT II	
1	1	Oops Concepts: Inheritance	W3
2	1	Polymorphism	T1:65-66
3	1	Interfaces	T1:66-67
4	1	Abstract class	T1:67-68
5	1	Threads	T1:70-72
6	1	Over loading and over riding	W4
7	1	Java Virtual Machine	W4
8	1	Recapitulation and discussion of Important Questions	
Total No. of Hours planned for Unit II			**8 Hrs**

TEXT BOOK:

T1: James C. Sheusi (2013), Android Application development for java Programmers, Cengage Learning.

WEBSITES

W3:http://developer.android.com/guide/components/activities.html

W4:http://developer.android.com/funcanmentals.html

S.No	Lecturer Duration(Hrs)	Topics to be Covered	Support Materials
		UNIT III	
1	1	Installing and using Eclipse with ADT plug-in	T1:1-4,W4
2	1	Installing Virtual Machine for android sandwich Emulator	T1:4-5,W5
3	1	Installing virtual machine for android Jelly Bean Emulator	T1:5-6,W5
4	1	Configuring the installed Tools	T1:9-10
5	1	Creating a Android project - Hello World	T1:10-11
6	1	Run on Emulator	T1:14-15
7	1	Deploy it on USB-Connected android device	T1:16-18
8	1	Recapitulation and discussion of Important Questions	
Total No. of Hours planned for Unit III			**8 Hrs**

TEXT BOOK:

T1: James C. Sheusi (2013), Android Application development for java Programmers, Cengage Learning.

WEBSITES

W4:http://developer.com/guide/components

W5:http://developer.com/guide/topics

LECTURE PLAN

S.No	Lecturer Duration(Hrs)	Topics to be Covered	Support Materials
		UNIT IV	
1	1	Application context,Intents	T1:29-32
2	1	Activity life cycle	T1:42-43, W6
3	1	multiple screen size	T1:43-45
4	1	User Interface Design : Form widgets	T1:46-47
5	1	Text Fields, Layouts	T1:48-50
6	1	button control, Toggle buttons	T1:65-67, W7
7	1	Spinners (Combo boxes), Images	T1:68-70
8	1	Menu, Dialogue	T1:71-74
9	1	Recapitulation and Discussion of important Questions	
Total No. of Hours planned for Unit IV			**9 Hours**

TEXT BOOK

T1: James C.Sheusi,(2013). *Android application development for Java for Java programmers*, Cengage Learning.

WEBSITES

W6:http://developer.android.com/training

W7:http://developer.android.com/screensize.html

LECTURE PLAN

S.No	Lecturer Duration(Hrs)	Topics to be Covered	Support Materials
		UNIT V	
1	1	Understanding of SQL Database	T1:197-198,W8
2	1	Connecting with the data base	T1:199-200,J1
3	1	Recapitulation and Discussion of important Questions	
4	1	Discussion of Previous ESE papers	
5	1	Discussion of Previous ESE papers	
6	1	Discussion of Previous ESE papers	
Total No. of Hours planned for Unit V			**6 Hours**

TEXT BOOK

T1: James C.Sheusi,(2013). *Android application development for Java for Java programmers*, Cengage Learning.

WEBSITES

W8:http://developer.android.com/training/basics/data.storage/databases.html

JOURNALS:

J1: ”Creating and using Database for Android”, Int. Journal of Database theory and applications, Vol 5, 2012.

ANDROID PROGRAMMING (17CAU304A)

Subject Notes- Unit I

Syllabus:

Introduction: History of Android- Introduction to Android Operating Systems-Android Development Tools- Android Architecture. (2L)

History of Android

The history and versions of android are interesting to know. The code names of android ranges from A to J currently, such as **Aestro**, **Blender**, **Cupcake**, **Donut**, **Eclair**, **Froyo**,**Gingerbread**, **Honeycomb**, **Ice Cream Sandwitch**, **Jelly Bean**, **KitKat** and **Lollipop**. Let's understand the android history in a sequence.

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, Codename and API

Let's see the android versions, codenames and API Level provided by Google.

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

INTRODUCTION TO ANDROID OPERATING SYSTEMS

Android is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touchscreen devices, Google has further developed Android TV for televisions, Android Auto for cars, and Android Wear for

wrist watches, each with a specialized user interface. Variants of Android are also used on notebooks,game consoles, digital cameras, and other electronics.

Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, along with the founding of the Open Handset Alliance – a consortium of hardware, software, and telecommunication companies devoted to advancing open standardsfor mobile devices.

Beginning with the first commercial Android device in September 2008, the operating system has gone through multiple major releases, with the current version being 7.0 "Nougat", released in August 2016. Android applications ("apps") can be downloaded from the Google Play store, which features over 2.7 million apps as of February 2017. Android has been the best-selling OS on tablets since 2013, and runs on the vast majority[a] of smartphones. As of May 2017, Android has two billion monthly active users, and it has the largest installed base of any operating system.

Android's source code is released by Google under an open source license, although most Android devices ultimately ship with a combination of free and open source and proprietary software, including proprietary software required for accessing Google services. Android is popular with technology companies that require a ready-made, low-cost and customizable operating system for high-tech devices. Its open nature has encouraged a large community of developers and enthusiasts to use the open-source code as a foundation for community-driven projects, which deliver updates to older devices, add new features for advanced users or bring Android to devices originally shipped with other operating systems.

The extensive variation of hardware in Android devices causes significant delays for software upgrades, with new versions of the operating system and security patches typically taking months before reaching consumers, or sometimes not at all. The success of Android has made it a target for patent and copyright litigation as part of the so-called "smartphone wars" between technology companies.

ANDROID DEVELOPMENT TOOLS

Android software development is the process by which new applications are created for the Android operating system. Applications are usually developed inJava programming language using the Android software development kit (SDK), but other development environments are also available.

The Android software development kit (SDK) includes a comprehensive set of development tools.[4] These include a debugger, libraries, a handset emulatorbased on QEMU, documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, and Windows

7or later. As of March 2015, the SDK is not available on Android itself, but software development is possible by using specialized Android applications.

Until around the end of 2014, the officially supported integrated development environment (IDE) was Eclipse using the Android Development Tools (ADT) Plugin, though IntelliJ IDEA IDE (all editions) fully supports Android development out of the box,[8] and NetBeans IDE also supports Android development via a plugin.[9] As of 2015, Android Studio,[10] made by Google and powered by IntelliJ, is the official IDE; however, developers are free to use others, but Google made it clear that ADT was officially deprecated since the end of 2015 to focus on Android Studio as the official Android IDE.[11] Additionally, developers may use any text editor to edit Java and XML files, then use command line tools (Java Development Kit and Apache Ant are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s) remotely).

Enhancements to Android's SDK go hand in hand with the overall Android platform development. The SDK also supports older versions of the Android platform in case developers wish to target their applications at older devices. Development tools are downloadable components, so after one has downloaded the latest version and platform, older platforms and tools can also be downloaded for compatibility testing.

Android applications are packaged in .apk format and stored under `/data/app` folder on the Android OS (the folder is accessible only to the root user for security reasons). APK package contains .dex files[14] (compiled byte code files calledDalvik executables), resource files, etc.

Android Debug Bridge

The Android Debug Bridge (ADB) is a toolkit included in the Android SDK package. It consists of both client and server-side programs that communicate with one another. The ADB is typically accessed through the command-line interface,[15]although numerous graphical user interfaces exist to control ADB.

Fastboot.

Fastboot is a diagnostic protocol included with the SDK package used primarily to modify the flash filesystem via a USBconnection from host computer. It requires that the device be started in a boot loader or Secondary Program Loader mode, in which only the most basic hardware initialization is performed. After enabling the protocol on the device itself, it will accept a specific set of commands sent to it via USB using a command line. Some of the most commonly used fastboot commands include:

- flash – rewrites a partition with a binary image stored on the host computer
- erase – erases a specific partition

- reboot – reboots the device into either the main operating system, the system recovery partition or back into its boot loader
- devices – displays a list of all devices (with the serial number) connected to the host computer
- format – formats a specific partition; the file system of the partition must be recognized by the device

ANDROID NDK

Libraries written in C/C++ can be compiled to ARM, MIPS or x86 native code(or their 64-bit variants) and installed using the Android Native Development Kit (NDK). These native libraries can be called from Java code running under the Dalvik VM using the `System.loadLibrary` call, which is part of the standard Android Java classes.

Complete applications can be compiled and installed using traditional development tools.[21] However, according to the Android documentation, NDK should not be used solely because the developer prefers to program in C/C++, as using NDK increases complexity while most applications would not benefit from using it.

The [23]ADB Debugger gives a root shell under the Android Emulator which allows ARM, MIPS or x86 native code to be uploaded and executed. Native code can be compiled using Clang or GCC on a standard PC. Running native code is complicated by Android's use of a non-standard C library (libc, known as Bionic).

It is possible to use the Android Studio with Gradle to develop NDK projects.[26] Other third-party tools allow integrating the NDK into Eclipse[27] and Visual Studio.

Android Open Accessory Development Kit

The Android 3.1 platform (also backported to Android 2.3.4) introduces Android Open Accessory support, which allows external USB hardware (an Android USB accessory) to interact with an Android-powered device in a special "accessory" mode. When an Android-powered device is in accessory mode, the connected accessory acts as the USB host (powers the bus and enumerates devices) and the Android-powered device acts as the USB device. Android USB accessories are specifically designed to attach to Android-powered devices and adhere to a simple protocol (Android accessory protocol) that allows them to detect Android-powered devices that support accessory mode.

Another built-in Android development tool, the Android Device Monitor allows you to monitor your device or virtual device during runtime and get access to information such as how many processes are running on what thread, network stats, the LogCat and more.

ANDROID Architecture

Android is an open source, Linux-based software stack created for a wide array of devices and form factors. The following diagram shows the major components of the Android platform.

The Linux Kernel

The foundation of the Android platform is the Linux kernel. For example, the Android Runtime (ART) relies on the Linux kernel for underlying functionalities such as threading and low-level memory management.

Using a Linux kernel allows Android to take advantage of key security features and allows device manufacturers to develop hardware drivers for a well-known kernel.

Hardware Abstraction Layer (HAL)

The hardware abstraction layer (HAL) provides standard interfaces that expose device hardware capabilities to the higher-level Java API framework. The HAL consists of multiple library modules, each of which implements an interface for a specific type of hardware component, such as the camera or bluetooth module. When a framework API makes a call to access device hardware, the Android system loads the library module for that hardware component.

Android Runtime

For devices running Android version 5.0 (API level 21) or higher, each app runs in its own process and with its own instance of the Android Runtime (ART). ART is written to run multiple virtual machines on low-memory devices by executing DEX files, a bytecode format designed specially for Android that's optimized for minimal memory footprint. Build toolchains, such as Jack, compile Java sources into DEX bytecode, which can run on the Android platform.

Some of the major features of ART include the following:

- Ahead-of-time (AOT) and just-in-time (JIT) compilation
- Optimized garbage collection (GC)
- Better debugging support, including a dedicated sampling profiler, detailed diagnostic exceptions and crash reporting, and the ability to set watchpoints to monitor specific fields

Prior to Android version 5.0 (API level 21), Dalvik was the Android runtime. If your app runs well on ART, then it should work on Dalvik as well, but the reverse may not be true.

Android also includes a set of core runtime libraries that provide most of the functionality of the Java programming language, including some Java 8 language features, that the Java API framework uses.

Native C/C++ Libraries

Many core Android system components and services, such as ART and HAL, are built from native code that require native libraries written in C and C++. The Android platform provides Java framework APIs to expose the functionality of some of these native libraries to apps. For example, you can access OpenGL ES through the Android framework's Java OpenGL API to add support for drawing and manipulating 2D and 3D graphics in your app.

If you are developing an app that requires C or C++ code, you can use the Android NDK to access some of thesenative platform libraries directly from your native code.

Java API Framework

The entire feature-set of the Android OS is available to you through APIs written in the Java language. These APIs form the building blocks you need to create Android apps by simplifying the reuse of core, modular system components and services, which include the following:

- A rich and extensible View System you can use to build an app's UI, including lists, grids, text boxes, buttons, and even an embeddable web browser
- A Resource Manager, providing access to non-code resources such as localized strings, graphics, and layout files
- A Notification Manager that enables all apps to display custom alerts in the status bar

- An Activity Manager that manages the lifecycle of apps and provides a common navigation back stack
- Content Providers that enable apps to access data from other apps, such as the Contacts app, or to share their own data

Developers have full access to the same framework APIs that Android system apps use.

System Apps

Android comes with a set of core apps for email, SMS messaging, calendars, internet browsing, contacts, and more. Apps included with the platform have no special status among the apps the user chooses to install. So a third-party app can become the user's default web browser, SSMS messenger, or even the default keyboard (some exceptions apply, such as the system's Settings app).

The system apps function both as apps for users and to provide key capabilities that developers can access from their own app. For example, if your app would like to deliver an SMS message, you don't need to build that functionality yourself—you can instead invoke whichever SMS app is already installed to deliver a message to the recipient you specify.

PART-B

Two mark Questions

1. What is an Android Operating System?
2. What is object oriented programming?
3. State IDE.
4. Define CheckBox.
5. What is meant by database?

PART-C

8 Mark Questions

1. Explain the history of Android.
2. Discuss the architecture of Android.
3. Explain Android Development Tools (ADT).
4. Discuss Android SDK.
5. Discuss Android operating system.
6. Explain the architecture of Android.
7. Explain the version, code name and API level of android.
8. Explain Android open accessory development kit.
9. Explain Linux kernel.
10. Discuss Java API framework.

QUESTIONS	OPT1	OPT2	OPT3	OPT4	OPT5	OPT6	ANSWER
Who invented Android programming_____	Andy Rubin	Gondy Rubin	Cloud John	Cloudy William			**Andy Rubin**
Android Incorporation was founded in_____	2004	2003	2002	2007			**2003**
SDK refers to _____.	System Development Kit	Software Development Kit	Soft Door Kit	Some Distributed Kit			**Software Development Kit**
JDK refers to _____.	Jova Developer Kit	Jas Developer Kit	Java Developer Kit	Jade Developer Kit			**Java Developer Kit**
JVM stands for _____.	Java Very Machine	Java Vat Machine	Java Virtual Mechanic	Java Virtual Machine			**Java Virtual Machine**
Android incorporation is now controlled by _____.	Gugle	Microsoft	Oracle	Google			**Google**
Eclipse is used to execute _____ programs.	Java and C	Java and Oracle	Java and Android	Java and VB.Net			**Java and Android**
ADT stands for _____.	Android Design Tool	Android Development Tool	Abstract Design Tool	Abstract Development Tool			**Android Development Tool**
Which year Google acquired Android Incorporation?	2004	2003	2005	2006			**2003**
Which company first launched Android Mobile?	HTC	STC	YTC	MTC			**HTC**
Android version 1.5 is called as _____	CupCake	CupBun	Cloud Ice	Cloudy Coffee			**CupCake**
Donut is the _____Android version.	1.5	1.7	1.6	1.8			**1.6**
Android version 1.5 is called as _____	Exclarie	CupBun	Choclate	Eclair			**Eclair**
Froyo is the _____ Android version.	2.4	2.2	2.3	2.6			**2.2**
Android version 2.3 is called as _____	GingerBake	GingerTea	Gingercoffee	Gingerbread			**Gingerbread**
Android version 3.1 and 3.3 are called as _____	HONYWELL	Honeycomb	HoneyDates	Honeybottle			**Honeycomb**
Android version 4.0 is called as _____.	Icecream	Vannila Ice	Ice cream Sandwitch	Icebar			**Ice cream Sandwitch**

Android version 4.1,4,2 and 4.3 are called as _____.	Jellyfish	Jelly Bean	Jellyice	Jellysugar			**Jelly Bean**
Android version 4.4 is called as _____.	Kitkat	kitkut	Katkit	KitKowt			**kitkat**
Android version 5.0 is called as _____.	Lolliice	Lollipop	Lollirose	Lollistick			**Lollipop**
Android is working based on _____.	Linux Kernel	Windows Kernel	Unix Kernel	Mac Kernel			**Linux Kernel**
Android version 7.0 is called as _____.	Bugat	Nougat	Chickloll i pop	Soya Ball			**Nougat**
Android version 7.0 is released in the year_____.	2014	2015	2016	2017			**2016**
ADB refers to _____.	Android Design Bridge	Android Development Bridge	Abstract Design Bar	Android Debug Bridge			**Android Debug Bridge**
_____ is a dignostic protocol.	fastbot	fastboot	bootfeet	slowboot			**fastboot**
_____ is an example for fastboot command.	flash	slash	lash	mash			**flash**
NDK stands for _____.	Native Developer Kit	Native Development Kit	Native Dummy Kit	Native Design Kit			**Native Development Kit**
ndk libraries are written in _____ language.	C/Pascal	Cobal	c/c++	c and C#			**c/c++**
Android is an _____ software.	open source	close	free	licensed			**open source**
ART refers to _____.	Android Rough Tme	Ant Rrun Time	Android Run Time	Android Rug Time			**Android Run Time**
HAL stands for _____.	Hard Abstraction Layer	Hardware Abstraction Layer	Honey Abstrsct Layer	Hot Absolute Layer			**Hardware Abstraction Layer**
HAL will interact with hardware like _____	whitetooth	blueray	yellowtooth	bluetooth			**bluetooth**
JIT Compilaion is _____	Just-In-Terms	Just-In-Time	Just-In-Tat	Just-In-Temp			**Just-In-Time**
GC stands for _____.	Garbage Collection	Gondy Collection	Gas Collectio	Google Collection			**Garbage Collection**
Java openGL is used for _____.	designing software	developing coding	doing testing	drawing 2D and 3D graphics			**drawing 2D and 3D graphics**

Android design code is done in _____.	html	mml	XML	WML			**XML**
Android Event driven coding is done in _____.	Java	c	c#	asp.net			**Java**
System Apps comes with a set of core apps for _____.	Playstore	Playstation	calendar, SMS and Email	playground			**calendar,SMS and Email**
Android Virtual Machine is _____.	Dolvik	Dalvik	Damvik	Dasvik			**Dalvik**
Android supports all _____.	C++ API	C API	C# API	Java API			**Java API**
Android activity is written in _____ Coding.	C#	C++	JAVA	asp.net			**JAVA**
There are _____ types of layout in Android.	3	4	2	1			**2**
Android apps are stored in _____ format.	API	AXE	APK	AXP			**APK**
_____ is one of the founders of Android.	Rich Miner	Rich Major	Bill Gats	Steve Jobs			**Rich Miner**
The nick name of Andy Rubin is _____.	Gondroid	Axdroid	Astroid	Android			**Android**
Android OS is used in _____ nowadays.	TV and Smartwatches	gas stove	Washing machine	Air coolers			**TV and Smartwatches**
The success of Android leads to increase _____ market.	TV	Electronics	Smart phones	telephones			**Smart phones**
Android _____ is used to run Android Coding in computers.	simulator	Developer	Emulator	Calculator			**Emulator**
ADB consists of Android _____ programs	only client side	only server side	both client and server side	windows side			**both client and server side**
_____ protocol detects Android Powered devices.	Android Accessory	Android Soft	Android hard	Android bean			**Android Accessory**

Subject Notes- Unit II

Unit-II Syllabus

Overview of object oriented programming using Java: OOPs Concepts: Inheritance-Polymorphism-Interfaces-Abstract class-Threads-Overloading and Overriding-Java Virtual Machine. (4L)

Overview of object oriented programming using Java

Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

Inheritance

Inheritance is one of the feature of Object-Oriented Programming (**OOPs**). Inheritance allows a class to use the properties and methods of another class. In other words, the derived class inherits the states and behaviors from the base class. The derived class is also called subclass and the base class is also known as super-class. The derived class can add its own additional variables and methods. These additional variable and methods differentiates the derived class from the base class.

Inheritance is a **compile-time** mechanism. A super-class can have any number of subclasses. But a subclass can have only one superclass. This is because Java does not support multiple inheritance.

Benefits of inheritance

- For Method Overriding (so runtime polymorphism can be achieved).
- For Code Reusability.

Syntax of Java Inheritance

```
class Subclass-name extends Superclass-name
{
 //methods and fields
}
```

The **extends keyword** indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality.

In the terminology of Java, a class which is inherited is called parent or super class and the new class is called child or subclass.

```
class Employee
   {
     float salary=40000;
   }
class Programmer extends Employee
      {
int bonus=10000;
public static void main(String args[])
          {
  Programmer p=new Programmer();
  System.out.println("Programmer salary is:"+p.salary);
   System.out.println("Bonus of Programmer is:"+p.bonus);
          }
      }
```

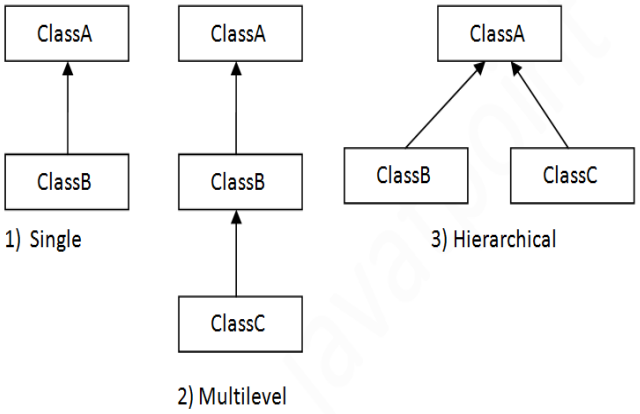
Output:

```
Programmer salary is:40000.0
Bonus of Programmer is:10000
```

Types of inheritance in java

On the basis of class, there can be three types of inheritance in java: single, multilevel and hierarchical.

In java programming, multiple and hybrid inheritance is supported through interface only. We will learn about interfaces later.



Single Inheritance Example

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void play(){System.out.println("playing...");}
}
class TestInheritance{
public static void main(String args[]){
Dog d=new Dog();
d.play();
d.eat();
}}
```

Output:

```
playing...

barking...
```

Multilevel Inheritance Example

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class BabyDog extends Dog{
void weep(){System.out.println("weeping...");}
}
class TestInheritance2{
```

```
public static void main(String args[]){
BabyDog d=new BabyDog();
d.weep();
d.bark();
d.eat();
}}
```

Output:

```
weeping...
barking...
eating...
```

Hierarchical Inheritance Example

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class Cat extends Animal{
void sleep(){System.out.println("sleeping...");}
}
class TestInheritance3{
public static void main(String args[]){
Cat c=new Cat();
c.sleep();
c.eat();
```

Output:

```
sleeping...
eating...
```

Polymorphism

Polymorphism in java is a concept by which we can perform a single action by different ways. Polymorphism is derived from 2 greek words: poly and morphs. The word "poly" means many and "morphs" means forms. So polymorphism means many forms.

There are two types of polymorphism in java: compile time polymorphism and runtime polymorphism. We can perform polymorphism in java by method overloading and method overriding.

Following concepts demonstrate different types of polymorphism in java.

1) **Method Overloading**

2) **Method Overriding**

Method Overloading:

In Java, it is possible to define two or more methods of same name in a class, provided that there argument list or parameters are different. This concept is known as Method Overloading.

Example:

```
class Overload
{
   void demo (int a)
   {
     System.out.println ("a: " + a);
   }
   void demo (int a, int b)
   {
     System.out.println ("a and b: " + a + "," + b);
   }
   double demo(double a) {
     System.out.println("double a: " + a);
     return a*a;
   }
}
class MethodOverloading
{
   public static void main (String args [])
   {
      Overload Obj = new Overload();
      double result;
      Obj .demo(10);
      Obj .demo(10, 20);
      result = Obj .demo(5.5);
      System.out.println("O/P : " + result);
   }
}
```

Output:

```
a: 10
a and b: 10,20
double a: 5.5
O/P : 30.25
```

Method Overriding

Child class has the same method as of base class. In such cases child class overrides the parent class method without even touching the source code of the base class. This feature is known as method overriding.
Example:

```
public class BaseClass
{
   public void methodToOverride() //Base class method
   {
       System.out.println ("I'm the method of BaseClass");
   }
}
public class DerivedClass extends BaseClass
{
   public void methodToOverride() //Derived Class method
   {
       System.out.println ("I'm the method of DerivedClass");
   }
}

public class TestMethod
{
    public static void main (String args []) {
       // BaseClass reference and object
       BaseClass obj1 = new BaseClass();
       // BaseClass reference but DerivedClass object
       BaseClass obj2 = new DerivedClass();
       // Calls the method from BaseClass class
       obj1.methodToOverride();
       //Calls the method from DerivedClass class
       obj2.methodToOverride();
    }
}
```

Output:

```
I'm the method of BaseClass
I'm the method of DerivedClass
```

Interface

Java interfaces are like Java classes but they contain only static final constants and declaration of methods. Methods are not defined and classes which implements an interface must define the body of method(s) of interface(s). Final constants can't be modified once they are initialized; final, interface, extend and implements are Java keywords.

Declaration of interface:

```
interface InterfaceName {
  // constants declaration
  // methods declaration
}
```

```
interface Info {
  static final String language = "Java";
  public void display();
}

class Simple implements Info {
  public static void main(String []args) {
    Simple obj = new Simple();
    obj.display();
  }

  // Defining method declared in interface

  public void display() {
    System.out.println(language + " is awesome");
  }
}
```

Output:

Java is awesome

Abstraction in Java

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

Another way, it shows only important things to the user and hides the internal details for example sending sms, you just type the text and send the message. You don't know the internal processing about the message delivery.

Abstract class in Java

A class that is declared as abstract is known as **abstract class**. It needs to be extended and its method implemented. It cannot be instantiated.

Example abstract class

```
abstract class A{}
```

abstract method

A method that is declared as abstract and does not have implementation is known as abstract method.

Example abstract method

abstract void printStatus();//no body and abstract

```
abstract class Bike{
  abstract void run();
}
class Honda4 extends Bike{
void run(){System.out.println("running safely..");}
public static void main(String args[]){
 Bike obj = new Honda4();
 obj.run();
}
}
```

Output:

running safely..

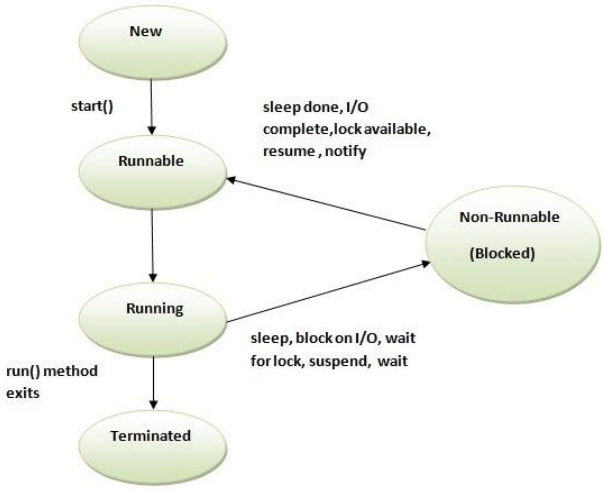
Threads

Thread is basically a lightweight sub-process, a smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.

Life cycle of a Thread (Thread States)

The life cycle of the thread in java is controlled by JVM. The java thread states are as follows:

1. New
2. Runnable
3. Running
4. Non-Runnable (Blocked)
5. Terminated



1) New

The thread is in new state if you create an instance of Thread class but before the invocation of start() method.

2) Runnable

The thread is in runnable state after invocation of start() method, but the thread scheduler has not selected it to be the running thread.

3) Running

The thread is in running state if the thread scheduler has selected it.

4) Non-Runnable (Blocked)

This is the state when the thread is still alive, but is currently not eligible to run.

5) Terminated

A thread is in terminated or dead state when its run() method exits.

Java Thread Example by extending Thread class

```
class Multi extends Thread{

public void run(){

System.out.println("thread is running...");

}
```

```
public static void main(String args[]){

Multi t1=new Multi();

t1.start();

 }

}
```

Output:
thread is running...

```
class Multi3 implements Runnable{

public void run(){

System.out.println("thread is running...");

}

  public static void main(String args[]){

Multi3 m1=new Multi3();

Thread t1 =new Thread(m1);

t1.start();

 }

}
```

Output:
thread is running...

Priority of a Thread

Each thread have a priority. Priorities are represented by a number between 1 and 10. In most cases, thread schedular schedules the threads according to their priority.

1. public static int MIN_PRIORITY
2. public static int NORM_PRIORITY
3. public static int MAX_PRIORITY

Default priority of a thread is 5 (NORM_PRIORITY). The value of MIN_PRIORITY is 1 and the value of MAX_PRIORITY is 10.

Example of priority of a Thread:

```
class TestMultiPriority1 extends Thread{

 public void run(){

  System.out.println("running thread name is:"+Thread.currentThread().getName());

System.out.println("running thread priority is:"+Thread.currentThread().getPriority());

   }

 public static void main(String args[]){

TestMultiPriority1 m1=new TestMultiPriority1();

 TestMultiPriority1 m2=new TestMultiPriority1();

 m1.setPriority(Thread.MIN_PRIORITY);

 m2.setPriority(Thread.MAX_PRIORITY);

 m1.start();

 m2.start();

   }

}
```

Output:

```
running thread name is:Thread-0
running thread priority is:10
running thread name is:Thread-1
running thread priority is:1
```

Java virtual machine

A **Java virtual machine** (**JVM**) is an abstract computing machine that enables a computer to run a Java program. There are three notions of the JVM: specification, implementation, and instance. The specification is a document that formally describes what is required of a JVM implementation. Having a single specification ensures all implementations are interoperable. A

JVM implementation is a computer program that meets the requirements of the JVM specification. An instance of a JVM is an implementation running in a process that executes a computer program compiled into Java bytecode.

Java Runtime Environment (**JRE**) is a software package that contains what is required to run a Java program. It includes a Java Virtual Machine implementation together with an implementation of the Java Class Library. The Oracle Corporation, which owns the Java trademark, distributes a Java Runtime environment with their Java Virtual Machine called HotSpot.

Java Development Kit (**JDK**) is a superset of a JRE and contains tools for Java programmers, e.g. a javac compiler. The Java Development Kit is provided free of charge either by Oracle Corporation directly, or by the OpenJDK open source project, which is governed by Oracle.

JVM specification

The Java virtual machine is an abstract (virtual) computer defined by a specification. This specification omits implementation details that are not essential to ensure interoperability: the memory layout of run-time data areas, the garbage-collection algorithm used, and any internal optimization of the Java virtual machine instructions (their translation into machine code). The main reason for this omission is to not unnecessarily constrain implementers. Any Java application can be run only inside some concrete implementation of the abstract specification of the Java virtual machine.[1]

Starting with Java Platform, Standard Edition (J2SE) 5.0, changes to the JVM specification have been developed under the Java Community Process as JSR 924. As of 2006, changes to specification to support changes proposed to the class file format (JSR 202) are being done as a maintenance release of JSR 924. The specification for the JVM was published as the blue book, The preface states:

We intend that this specification should sufficiently document the Java Virtual Machine to make possible compatible clean-room implementations. Oracle provides tests that verify the proper operation of implementations of the Java Virtual Machine.

One of Oracle's JVMs is named HotSpot, the other, inherited from BEA Systems is JRockit. Clean-room Java implementations include Kaffe and IBM J9. Oracle owns the Java trademark and may allow its use to certify implementation suites as fully compatible with Oracle's specification.

Class loader

Main article: Java Class loader

One of the organizational units of JVM byte code is a class. A class loader implementation must be able to recognize and load anything that conforms to the Java class file format. Any

implementation is free to recognize other binary forms besides class files, but it must recognize class files.

The class loader performs three basic activities in this strict order:

1. Loading: finds and imports the binary data for a type
2. Linking: performs verification, preparation, and (optionally) resolution
    - Verification: ensures the correctness of the imported type
    - Preparation: allocates memory for class variables and initializing the memory to default values
    - Resolution: transforms symbolic references from the type into direct references.
3. Initialization: invokes Java code that initializes class variables to their proper starting values.

JVM languages

Main article: List of JVM languages

A JVM language is any language with functionality that can be expressed in terms of a valid class file which can be hosted by the Java Virtual Machine. A class file contains Java Virtual Machine instructions (Java byte code) and a symbol table, as well as other ancillary information. The class file format is the hardware- and operating system-independent binary format used to represent compiled classes and interfaces.

There are several JVM languages, both old languages ported to JVM and completely new languages. JRuby and Jython are perhaps the most well-known ports of existing languages, i.e. Ruby and Python respectively. Of the new languages that have been created from scratch to compile to Java bytecode, Clojure, Groovy and Scala may be the most popular ones. A notable feature with the JVM languages is that they are compatible with each other, so that, for example, Scala libraries can be used with Java programs and vice versa.

Java 7 JVM implements JSR 292: Supporting Dynamically Typed Languages on the Java Platform, a new feature which supports dynamically typed languages in the JVM. This feature is developed within the Da Vinci Machine project whose mission is to extend the JVM so that it supports languages other than Java

JVM in the web browser

Since the very early stages of the design process, Java (and JVM) has been marketed as a web technology for creating Rich Internet Applications.

PART-B
Two mark Questions

1. Define software development kit.
2. What is the concept of inheritance?
3. Define Eclipse.
4. What is the function of TableLayout?
5. Define query.

PART-C
Eight Mark Questions

1.Discuss the concepts of OOPs in Java.
2.Explain the concept of inheritance and its types.
3. Explain multilevel inheritance in Java with suitable program.
4.Discuss the concept of method overloading in java.
5. Explain the concept of Method overriding with an example Java program.
6. Discuss Abstract class with an example Java program.
7.Explain interface in Java with an example Java program.
8. Discuss the Multi threading concept with an example Java program.
9.Explain various life cycles and priorities of thread.
10. Discuss Java virtual machine.

Subject Notes- Unit II

Unit-II Syllabus

Overview of object oriented programming using Java: OOPs Concepts: Inheritance-Polymorphism-Interfaces-Abstract class-Threads-Overloading and Overriding-Java Virtual Machine. (4L)

Overview of object oriented programming using Java

Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

Inheritance

Inheritance is one of the feature of Object-Oriented Programming (**OOPs**). Inheritance allows a class to use the properties and methods of another class. In other words, the derived class inherits the states and behaviors from the base class. The derived class is also called subclass and the base class is also known as super-class. The derived class can add its own additional variables and methods. These additional variable and methods differentiates the derived class from the base class.

Inheritance is a **compile-time** mechanism. A super-class can have any number of subclasses. But a subclass can have only one superclass. This is because Java does not support multiple inheritance.

Benefits of inheritance

- For Method Overriding (so runtime polymorphism can be achieved).
- For Code Reusability.

Syntax of Java Inheritance

```
class Subclass-name extends Superclass-name
{
 //methods and fields
}
```

The **extends keyword** indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality.

In the terminology of Java, a class which is inherited is called parent or super class and the new class is called child or subclass.

```
class Employee
   {
     float salary=40000;
   }
class Programmer extends Employee
      {
int bonus=10000;
public static void main(String args[])
          {
  Programmer p=new Programmer();
  System.out.println("Programmer salary is:"+p.salary);
   System.out.println("Bonus of Programmer is:"+p.bonus);
          }
      }
```

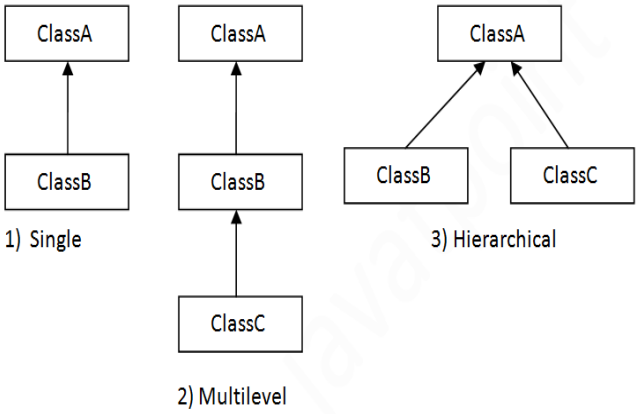
Output:

```
Programmer salary is:40000.0
Bonus of Programmer is:10000
```

Types of inheritance in java

On the basis of class, there can be three types of inheritance in java: single, multilevel and hierarchical.

In java programming, multiple and hybrid inheritance is supported through interface only. We will learn about interfaces later.



Single Inheritance Example

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void play(){System.out.println("playing...");}
}
class TestInheritance{
public static void main(String args[]){
Dog d=new Dog();
d.play();
d.eat();
}}
```

Output:

```
playing...

barking...
```

Multilevel Inheritance Example

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class BabyDog extends Dog{
void weep(){System.out.println("weeping...");}
}
class TestInheritance2{
```

```
public static void main(String args[]){
BabyDog d=new BabyDog();
d.weep();
d.bark();
d.eat();
}}
```

Output:

```
weeping...
barking...
eating...
```

Hierarchical Inheritance Example

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class Cat extends Animal{
void sleep(){System.out.println("sleeping...");}
}
class TestInheritance3{
public static void main(String args[]){
Cat c=new Cat();
c.sleep();
c.eat();
```

Output:

```
sleeping...
eating...
```

Polymorphism

Polymorphism in java is a concept by which we can perform a single action by different ways. Polymorphism is derived from 2 greek words: poly and morphs. The word "poly" means many and "morphs" means forms. So polymorphism means many forms.

There are two types of polymorphism in java: compile time polymorphism and runtime polymorphism. We can perform polymorphism in java by method overloading and method overriding.

Following concepts demonstrate different types of polymorphism in java.

1) **Method Overloading**

2) **Method Overriding**

Method Overloading:

In Java, it is possible to define two or more methods of same name in a class, provided that there argument list or parameters are different. This concept is known as Method Overloading.

Example:

```
class Overload
{
   void demo (int a)
   {
     System.out.println ("a: " + a);
   }
   void demo (int a, int b)
   {
     System.out.println ("a and b: " + a + "," + b);
   }
   double demo(double a) {
     System.out.println("double a: " + a);
     return a*a;
   }
}
class MethodOverloading
{
   public static void main (String args [])
   {
      Overload Obj = new Overload();
      double result;
      Obj .demo(10);
      Obj .demo(10, 20);
      result = Obj .demo(5.5);
      System.out.println("O/P : " + result);
   }
}
```

Output:

```
a: 10
a and b: 10,20
double a: 5.5
O/P : 30.25
```

Method Overriding

Child class has the same method as of base class. In such cases child class overrides the parent class method without even touching the source code of the base class. This feature is known as method overriding.
Example:

```
public class BaseClass
{
   public void methodToOverride() //Base class method
   {
       System.out.println ("I'm the method of BaseClass");
   }
}
public class DerivedClass extends BaseClass
{
   public void methodToOverride() //Derived Class method
   {
       System.out.println ("I'm the method of DerivedClass");
   }
}

public class TestMethod
{
    public static void main (String args []) {
       // BaseClass reference and object
       BaseClass obj1 = new BaseClass();
       // BaseClass reference but DerivedClass object
       BaseClass obj2 = new DerivedClass();
       // Calls the method from BaseClass class
       obj1.methodToOverride();
       //Calls the method from DerivedClass class
       obj2.methodToOverride();
    }
}
```

Output:

```
I'm the method of BaseClass
I'm the method of DerivedClass
```

Interface

Java interfaces are like Java classes but they contain only static final constants and declaration of methods. Methods are not defined and classes which implements an interface must define the body of method(s) of interface(s). Final constants can't be modified once they are initialized; final, interface, extend and implements are Java keywords.

Declaration of interface:

```
interface InterfaceName {
  // constants declaration
  // methods declaration
}
```

```
interface Info {
  static final String language = "Java";
  public void display();
}

class Simple implements Info {
  public static void main(String []args) {
    Simple obj = new Simple();
    obj.display();
  }

  // Defining method declared in interface

  public void display() {
    System.out.println(language + " is awesome");
  }
}
```

Output:

Java is awesome

Abstraction in Java

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

Another way, it shows only important things to the user and hides the internal details for example sending sms, you just type the text and send the message. You don't know the internal processing about the message delivery.

Abstract class in Java

A class that is declared as abstract is known as **abstract class**. It needs to be extended and its method implemented. It cannot be instantiated.

Example abstract class

```
abstract class A{}
```

abstract method

A method that is declared as abstract and does not have implementation is known as abstract method.

Example abstract method

abstract void printStatus();//no body and abstract

```
abstract class Bike{
  abstract void run();
}
class Honda4 extends Bike{
void run(){System.out.println("running safely..");}
public static void main(String args[]){
 Bike obj = new Honda4();
 obj.run();
}
}
```

Output:

running safely..

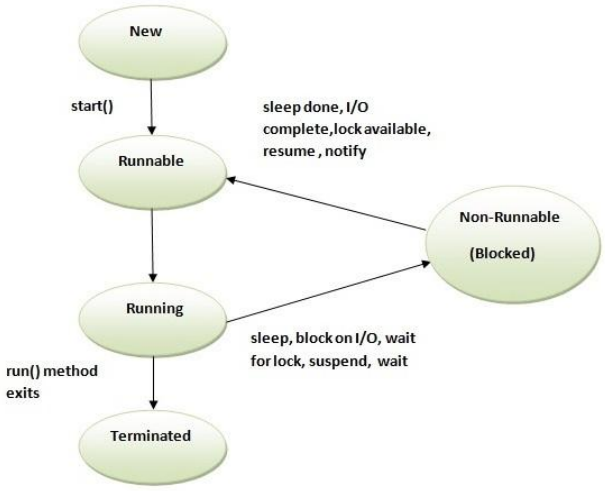
Threads

Thread is basically a lightweight sub-process, a smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.

Life cycle of a Thread (Thread States)

The life cycle of the thread in java is controlled by JVM. The java thread states are as follows:

1. New
2. Runnable
3. Running
4. Non-Runnable (Blocked)
5. Terminated



1) New

The thread is in new state if you create an instance of Thread class but before the invocation of start() method.

2) Runnable

The thread is in runnable state after invocation of start() method, but the thread scheduler has not selected it to be the running thread.

3) Running

The thread is in running state if the thread scheduler has selected it.

4) Non-Runnable (Blocked)

This is the state when the thread is still alive, but is currently not eligible to run.

5) Terminated

A thread is in terminated or dead state when its run() method exits.

Java Thread Example by extending Thread class

```
class Multi extends Thread{

public void run(){

System.out.println("thread is running...");

}
```

```
public static void main(String args[]){

Multi t1=new Multi();

t1.start();

 }

}
```

Output:
thread is running...

```
class Multi3 implements Runnable{

public void run(){

System.out.println("thread is running...");

}

  public static void main(String args[]){

Multi3 m1=new Multi3();

Thread t1 =new Thread(m1);

t1.start();

 }

}
```

Output:
thread is running...

Priority of a Thread

Each thread have a priority. Priorities are represented by a number between 1 and 10. In most cases, thread schedular schedules the threads according to their priority.

1. public static int MIN_PRIORITY
2. public static int NORM_PRIORITY
3. public static int MAX_PRIORITY

Default priority of a thread is 5 (NORM_PRIORITY). The value of MIN_PRIORITY is 1 and the value of MAX_PRIORITY is 10.

Example of priority of a Thread:

```
class TestMultiPriority1 extends Thread{

 public void run(){

  System.out.println("running thread name is:"+Thread.currentThread().getName());

System.out.println("running thread priority is:"+Thread.currentThread().getPriority());

   }

 public static void main(String args[]){

TestMultiPriority1 m1=new TestMultiPriority1();

 TestMultiPriority1 m2=new TestMultiPriority1();

 m1.setPriority(Thread.MIN_PRIORITY);

 m2.setPriority(Thread.MAX_PRIORITY);

 m1.start();

 m2.start();

   }

}
```

Output:

```
running thread name is:Thread-0
running thread priority is:10
running thread name is:Thread-1
running thread priority is:1
```

Java virtual machine

A **Java virtual machine** (**JVM**) is an abstract computing machine that enables a computer to run a Java program. There are three notions of the JVM: specification, implementation, and instance. The specification is a document that formally describes what is required of a JVM implementation. Having a single specification ensures all implementations are interoperable. A

JVM implementation is a computer program that meets the requirements of the JVM specification. An instance of a JVM is an implementation running in a process that executes a computer program compiled into Java bytecode.

Java Runtime Environment (**JRE**) is a software package that contains what is required to run a Java program. It includes a Java Virtual Machine implementation together with an implementation of the Java Class Library. The Oracle Corporation, which owns the Java trademark, distributes a Java Runtime environment with their Java Virtual Machine called HotSpot.

Java Development Kit (**JDK**) is a superset of a JRE and contains tools for Java programmers, e.g. a javac compiler. The Java Development Kit is provided free of charge either by Oracle Corporation directly, or by the OpenJDK open source project, which is governed by Oracle.

JVM specification

The Java virtual machine is an abstract (virtual) computer defined by a specification. This specification omits implementation details that are not essential to ensure interoperability: the memory layout of run-time data areas, the garbage-collection algorithm used, and any internal optimization of the Java virtual machine instructions (their translation into machine code). The main reason for this omission is to not unnecessarily constrain implementers. Any Java application can be run only inside some concrete implementation of the abstract specification of the Java virtual machine.[1]

Starting with Java Platform, Standard Edition (J2SE) 5.0, changes to the JVM specification have been developed under the Java Community Process as JSR 924. As of 2006, changes to specification to support changes proposed to the class file format (JSR 202) are being done as a maintenance release of JSR 924. The specification for the JVM was published as the blue book, The preface states:

We intend that this specification should sufficiently document the Java Virtual Machine to make possible compatible clean-room implementations. Oracle provides tests that verify the proper operation of implementations of the Java Virtual Machine.

One of Oracle's JVMs is named HotSpot, the other, inherited from BEA Systems is JRockit. Clean-room Java implementations include Kaffe and IBM J9. Oracle owns the Java trademark and may allow its use to certify implementation suites as fully compatible with Oracle's specification.

Class loader

Main article: Java Class loader

One of the organizational units of JVM byte code is a class. A class loader implementation must be able to recognize and load anything that conforms to the Java class file format. Any

implementation is free to recognize other binary forms besides class files, but it must recognize class files.

The class loader performs three basic activities in this strict order:

1. Loading: finds and imports the binary data for a type
2. Linking: performs verification, preparation, and (optionally) resolution
    - Verification: ensures the correctness of the imported type
    - Preparation: allocates memory for class variables and initializing the memory to default values
    - Resolution: transforms symbolic references from the type into direct references.
3. Initialization: invokes Java code that initializes class variables to their proper starting values.

JVM languages

Main article: List of JVM languages

A JVM language is any language with functionality that can be expressed in terms of a valid class file which can be hosted by the Java Virtual Machine. A class file contains Java Virtual Machine instructions (Java byte code) and a symbol table, as well as other ancillary information. The class file format is the hardware- and operating system-independent binary format used to represent compiled classes and interfaces.

There are several JVM languages, both old languages ported to JVM and completely new languages. JRuby and Jython are perhaps the most well-known ports of existing languages, i.e. Ruby and Python respectively. Of the new languages that have been created from scratch to compile to Java bytecode, Clojure, Groovy and Scala may be the most popular ones. A notable feature with the JVM languages is that they are compatible with each other, so that, for example, Scala libraries can be used with Java programs and vice versa.

Java 7 JVM implements JSR 292: Supporting Dynamically Typed Languages on the Java Platform, a new feature which supports dynamically typed languages in the JVM. This feature is developed within the Da Vinci Machine project whose mission is to extend the JVM so that it supports languages other than Java

JVM in the web browser

Since the very early stages of the design process, Java (and JVM) has been marketed as a web technology for creating Rich Internet Applications.

PART-B
Two mark Questions

1. Define software development kit.
2. What is the concept of inheritance?
3. Define Eclipse.
4. What is the function of TableLayout?
5. Define query.

PART-C
Eight Mark Questions

1.Discuss the concepts of OOPs in Java.
2.Explain the concept of inheritance and its types.
3. Explain multilevel inheritance in Java with suitable program.
4.Discuss the concept of method overloading in java.
5. Explain the concept of Method overriding with an example Java program.
6. Discuss Abstract class with an example Java program.
7.Explain interface in Java with an example Java program.
8. Discuss the Multi threading concept with an example Java program.
9.Explain various life cycles and priorities of thread.
10. Discuss Java virtual machine.

UNIT III SUBJECT NOTES

UNIT –III Syllabus
Development Tools: Installing and using Eclipse with ADT plug-in- Installing Virtual machine for Android sandwich/Jelly bean (Emulator)-configuring the installed tools- creating a android- project– Hello Word, run on emulator-Deploy it on USB-connected Android device. (5L)

Installing and using Eclipse with ADT Plugin

Installing the Eclipse Plugin

Android offers a custom plugin for the Eclipse IDE, called Android Development Tools (ADT). This plugin provides a powerful, integrated environment in which to develop Android apps. It extends the capabilities of Eclipse to let you quickly set up new Android projects, build an app UI, debug your app, and export signed (or unsigned) app packages (APKs) for distribution.

Download the ADT Plugin

---

1. Start Eclipse, then select **Help** > **Install New Software**.
2. Click **Add**, in the top-right corner.
3. In the Add Repository dialog that appears, enter "ADT Plugin" for the Name and the following URL for the Location:

   https://dl-ssl.google.com/android/eclipse/

4. Click **OK**.

   If you have trouble acquiring the plugin, try using "http" in the Location URL, instead of "https" (https is preferred for security reasons).

5. In the Available Software dialog, select the checkbox next to Developer Tools and click **Next**.
6. In the next window, you'll see a list of the tools to be downloaded. Click **Next**.
7. Read and accept the license agreements, then click **Finish**.

   If you get a security warning saying that the authenticity or validity of the software can't be established, click **OK**.

8. When the installation completes, restart Eclipse.

Configure the ADT Plugin

Once Eclipse restarts, you must specify the location of your Android SDK directory:

1. In the "Welcome to Android Development" window that appears, select **Use existing SDKs**.
2. Browse and select the location of the Android SDK directory you recently downloaded and unpacked.
3. Click **Next**.

Your Eclipse IDE is now set up to develop Android apps, but you need to add the latest SDK platform tools and an Android platform to your environment. To get these packages for your SDK, continue to Adding Platforms and Packages.

Troubleshooting Installation

If you are having trouble downloading the ADT plugin after following the steps above, here are some suggestions:

- If Eclipse can not find the remote update site containing the ADT plugin, try changing the remote site URL to use http, rather than https. That is, set the Location for the remote site to:

  http://dl-ssl.google.com/android/eclipse/

- If you are behind a firewall (such as a corporate firewall), make sure that you have properly configured your proxy settings in Eclipse. In Eclipse, you can configure proxy information from the main Eclipse menu in **Window** (on Mac OS X, **Eclipse**) > **Preferences** > **General** > **Network Connections**.

If you are still unable to use Eclipse to download the ADT plugin as a remote update site, you can download the ADT zip file to your local machine and manually install it:

1. Download the ADT Plugin zip file (do not unpack it):

Package	Size	MD5 Checksum
ADT-21.1.0.zip	13564671 bytes	f1ae183891229784bb9c33bcc9c5ef1e

2. Start Eclipse, then select **Help** > **Install New Software**.
3. Click **Add**, in the top-right corner.
4. In the Add Repository dialog, click **Archive**.

5. Select the downloaded ADT-21.1.0.zip file and click **OK**.
6. Enter "ADT Plugin" for the name and click **OK**.
7. In the Available Software dialog, select the checkbox next to Developer Tools and click **Next**.
8. In the next window, you'll see a list of the tools to be downloaded. Click **Next**.
9. Read and accept the license agreements, then click **Finish**.

   If you get a security warning saying that the authenticity or validity of the software can't be established, click **OK**.

10. When the installation completes, restart Eclipse.

Installing virtual machine for Android sandwich

Android is Linux based open source operating system, especially designed for touch based smart phones and tablets and is one of the most widely used operating system by Mobile phone and tablets manufacturers. As Android OS only supports ARM architecture based hardware so you can't run it on x86 architecture i.e. Computer or laptop. In order to run it on x86 architecture, you need to have an Android OS which supports x86 architecture luckily Android x86 project provides it for various testing purposes and you can install Android OS along with your Windows Vista, 7 & 8 operating system.

Steps for installing Android OS Ice Cream Sandwich on Virtual PC

As I am using Microsoft Windows 8 OS thus I have mentioned the steps for installing Android OS on Virtual PC along with Windows 8(learn how to install Windows 8 on virtual PC) but these steps are very much applicable to Windows Vista and 7.

- First download and install Oracle VM VirtualBox from this link (http://www.oracle.com/technetwork/server-storage/virtualbox/downloads/index.html). If you already have it installed then upgrade it to the latest version

- Now visit android-x86.org site and head over to download page. If your system name is listed then download the Android OS ISO image from the respective link else download the ISO image especially created for x86 architecture based hardware which will work on every system (from here https://docs.google.com/open?id=0B4GbJReHMmu_amMzQzJoNGw3WFU). Also if you experience internet connectivity and audio problem with ISO image downloaded for respective system then download the generic ISO image from link given above

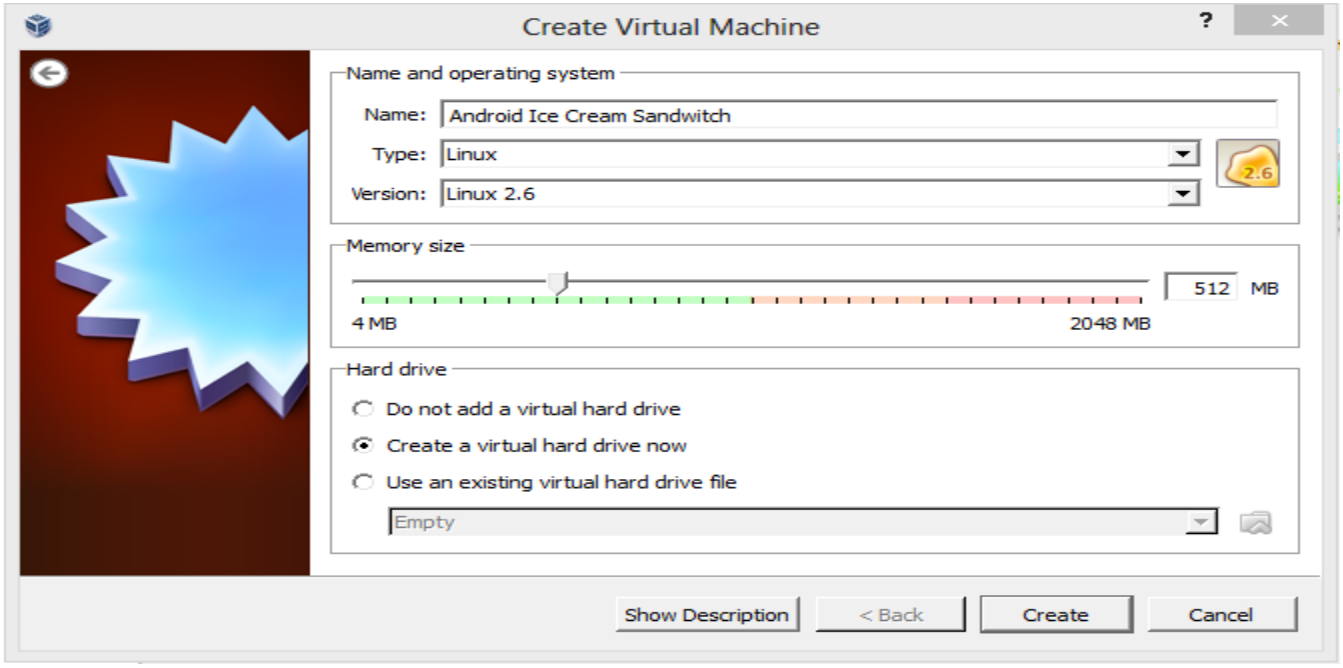
- Now open Oracle VM VirtualBox and press CTRL + N for creating a new Virtual Machine and also click the Hide Description button to make visible the hidden Hard drive options

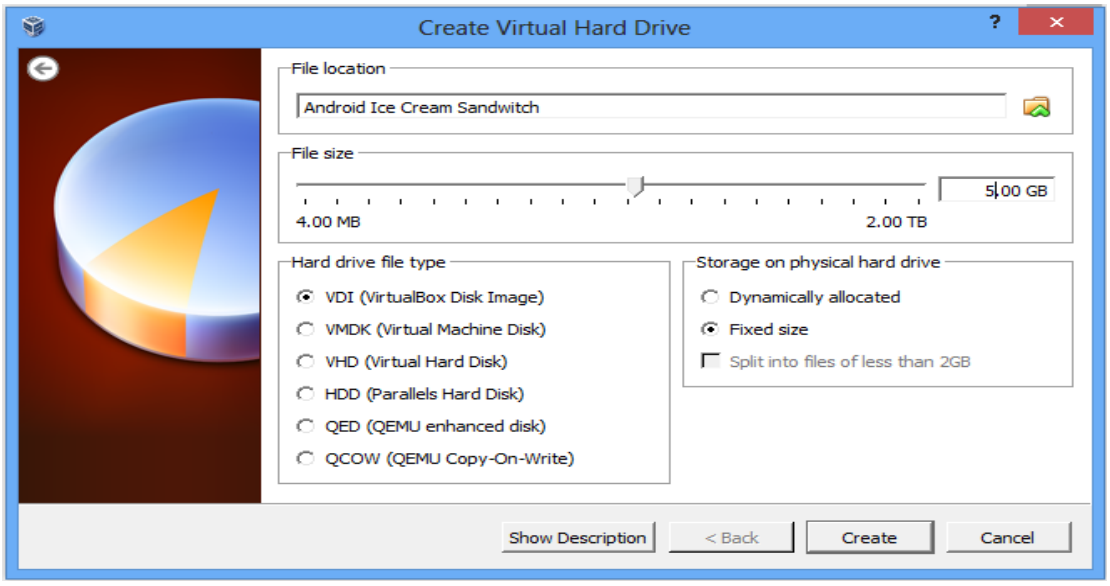
- In name box write Android ICS or it could be anything which helps you to recognize it

easily, in Type box select Linux option and in Version choose Linux 2.6 respectively

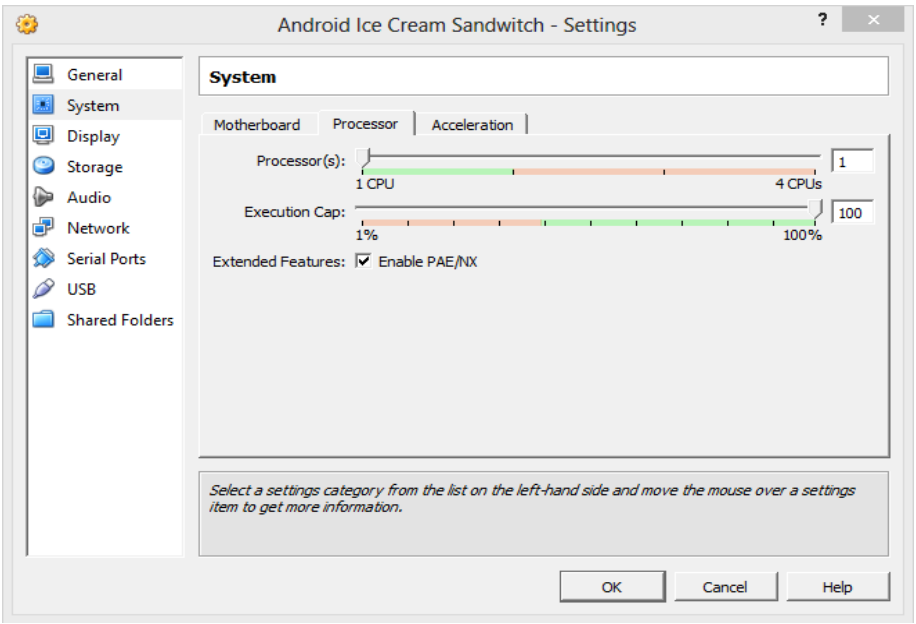
- Under Memory size option increase the slider to 512 MB for better performance though you can assign higher or lower memory than this

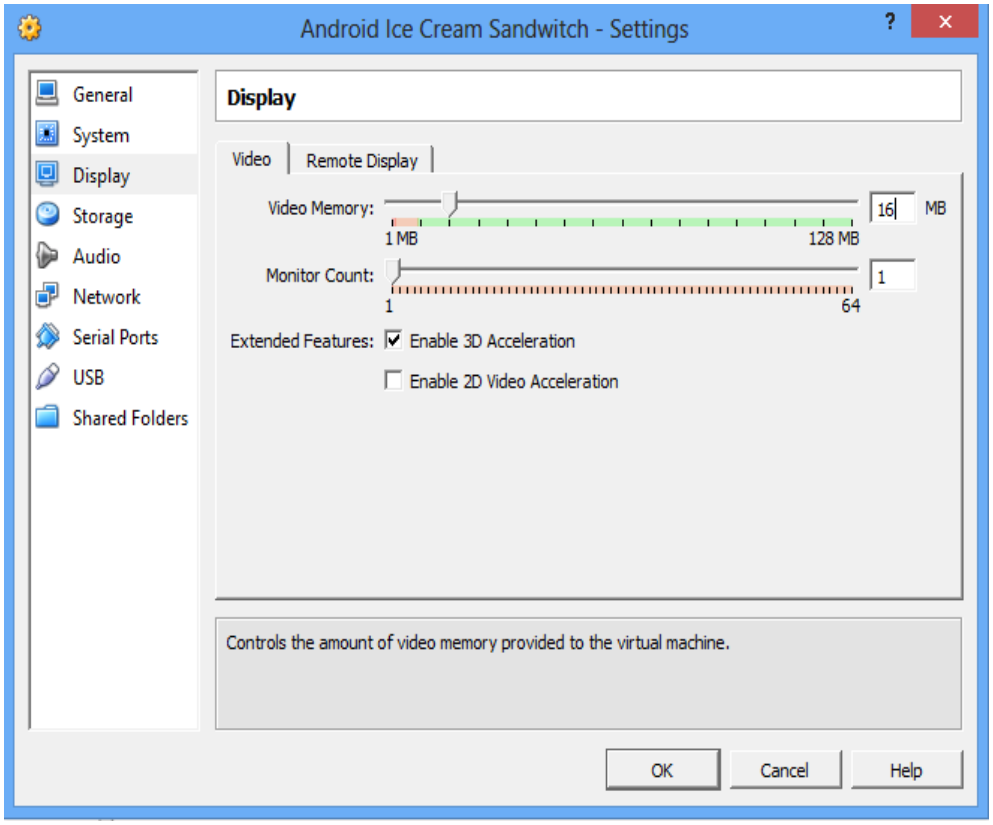
- In Hard drive option make sure "Create a virtual hard drive now" is selected, Click Create button





This will create the virtual machine named Android ICS. Now you have to modify few options to optimize it for better performance. Open settings Window, navigate to System tab then to Processor tab and tick the check box against the "Enable PAE/NX" option. Now navigate to display tab and increase the video memory size to more than 10 MB and enable the 3D acceleration under extended feature.





On Android –x86 Installation Window, select the "Installation – Install Android –x86 to harddisk option". It will initiate the process of installing Android OS

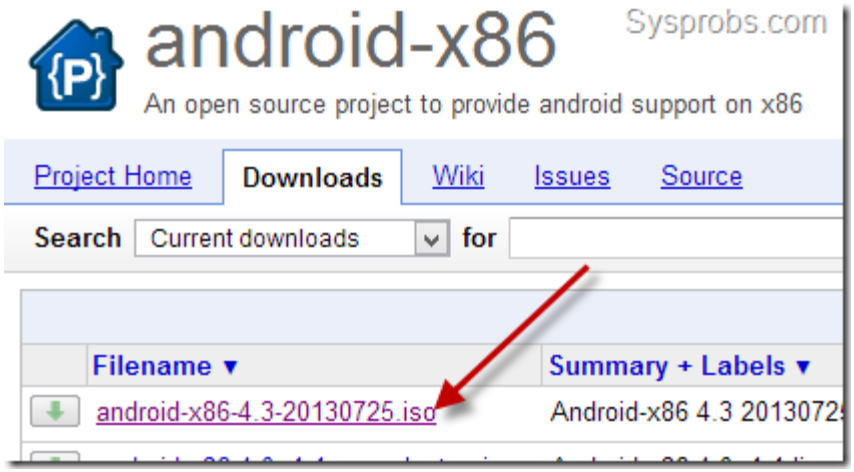
- On Choose Partition Window, choose "Create/Modify Partitions" option and click OK button. It will open up cfdisk utility. Under cfdisk utility choose the options as below
1) Select NEW option
2) Select PRIMARY option
3) On next screen hit enter again to accept the default partition size
4) Select BOOTABLE option then WRITE option
5) Type yes when prompted to write the partition table to disk
6) Now QUIT the cfdisk utility
7) On next screen, select the "sda1 Linux VBOX HARDDISK" option and click OK button
8) Under Choose file system, select the "ext3" option. On the next screen, press YES option to format sda to ext3 file system
9) Press YES option when prompted to install boot loader GRUB and install/system directory as read-write
10) If you wish you can create a fake SD card by selecting the "Create a fake SD card" option else select the Reboot option.
11) You are Done with installation of Android Ice Cream Sandwich OS on Virtual PC.

Installing virtual machine for Android Jellybean

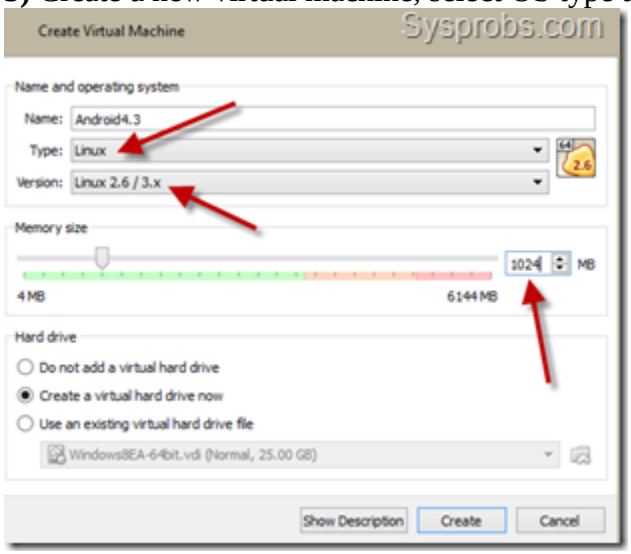
Steps to Install Jelly Bean Android With VirtualBox

1) Make sure you have the latest VirtualBox on your PC.

2) Download Android 4.3 ISO from Google site here.



3) Create a new virtual machine, select OS type as **Linux** as below.

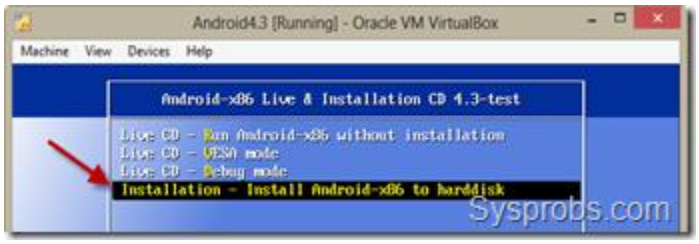


Set the RAM size to more than 512MB. I configured 1GB. Create a new hard disk.

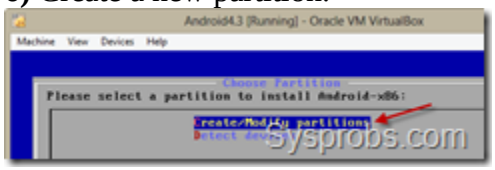
4) Go to the settings of virtual machine and edit the storage settings. We need to browse and mount the ISO file which was downloaded from Google site, to IDE controller of CD/DVD drive.

The CD/DVD drive should appear as shown above after loading the ISO.

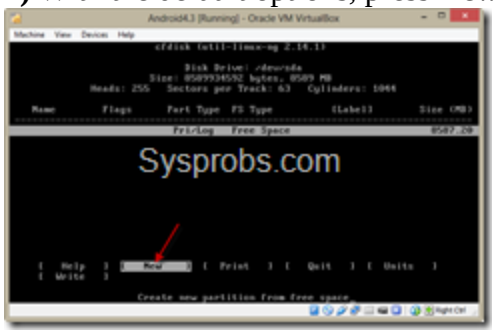
5) Power on VM which will boot from attached ISO. Select the installation option as below.



6) Create a new partition.

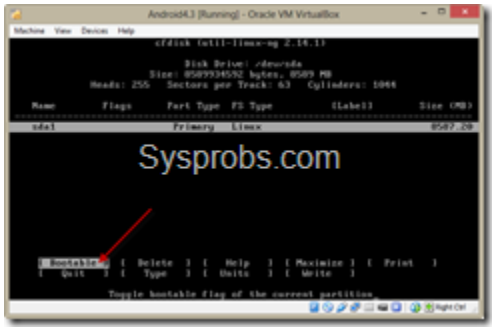


7) With the default options, press **New**.

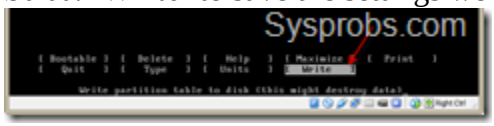


Make it as **'Primary**' in next screen and press Enter to allocate full size for the partition.

8.) The partition should be bootable, select **'Bootable'** in next screen.



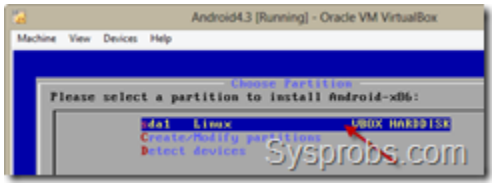
Select '**Write'** to save the settings we did earlier on the partition.



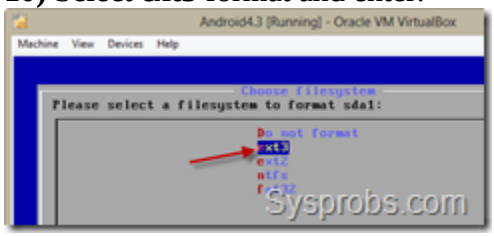
To confirm type **'yes'** and press enter.

Quit from the next screen.

9) Once you have come out of partition creation tool, you can chose the newly created partition to start the installation on VirtualBox.



10) Select **ext3** format and enter.



Press **'Yes'** to format the partition. Also select **'Yes'** to install **boot loader GRUB**. Again **'Yes'** to install **/System directory as read-write** in next screen.

Installation process will start.

11) We have successfully install Android 4.3 on Windows 8 with VirtualBox. Reboot the virtual machine to use.

Note – Remove the ISO file from CD/DVD drive before booting, otherwise it will again boot from ISO and start installation process.

12) Once virtual machine is booted, it is better to disable mouse integration with VM. So, it will be easy to access and use mouse inside Android OS.

13) Network worked directly in bridge mode inside virtual machine. Performance of graphics is not up the the standard. Do not think to play Android games inside this virtual machine, it will not work. But still it is worth to install and play around with it without having a real phone or tablet device.

Here are some of the screenshots taken from Jelly Bean virtual machine in Windows 8.

Creating a Simple Hello World Android Project

Creating a Simple Hello World Android Project

To create a simple Hello World Android project can be done either with Eclipse or Android Studio. Here I am going to explain how it can be created by using Android Studio 0.8.0.

Android Studio:

Studio can be downloaded from the below link.

http://tools.android.com/download/studio/beta

Pre-requisite:

Ensure appropriate JDK version is installed.

Download appropriate Android SDK based on the version we are developing.

https://www.codeproject.com/KB/android/803646/SDKManager.png

Create new project

First step load Android Studio. Click on the New project…

https://www.codeproject.com/KB/android/803646/NewProject.png

Configure the New Project

Enter the application and company domain and select the project location as shown below and click on Next button.

https://www.codeproject.com/KB/Android/803646/Configure.png

Select form factor

Select the appropriate minimum version of android we are going to target as shown in the list as below

https://www.codeproject.com/KB/android/803646/formfactor.png

Select the Activity

Select the template need as pre requirement. I have selected the blank activity.

https://www.codeproject.com/KB/Android/803646/Activity.png

The class will be created based on the Activity Name entered.

https://www.codeproject.com/KB/Android/803646/ActivityName.png

Click on the finish button. The project gets created and will be shown as below

https://www.codeproject.com/KB/Android/803646/FinishNavigation.png

Files / Components

Important files and directory of Android project to be known and their purpose

1. src - This contains the .java source files for your project. By default, it includes an MainActivity.java source file having an activity class that runs when your app is launched using the app icon.
2. generated - This contains the .R file, a compiler-generated file that references all the resources found in your project. You should not modify this file
3. bin - This folder contains the Android package files .apk built by the ADT during the build process and everything else needed to run an Android application.
4. res/drawable-hdpi - This is a directory for drawable objects that are designed for high-density screens.
5. res/layout - This is a directory for files that define your app's user interface.
6. res/values - This is a directory for other various XML files that contain a collection of resources, such as strings and colors definitions.
7. AndroidManifest.xml - This is the manifest file which describes the fundamental characteristics of the app and defines each of its components.

https://www.codeproject.com/KB/Android/803646/filecomponents.png

String file

The strings.xml file is located in the res/values folder and it contains all the text that your application uses. For example, the names of buttons, labels, default text, and similar types of strings go into this file. This file is responsible for their textual content. For example, a default strings file will look like as following file

R file

The gen/myapps.helloworld/R.java file is the glue between the activity Java files like Main.java and the resources like strings.xml. It is an automatically generated file and you should not modify the content of the R.java file. Following is a sample of R.java filehttps://www.codeproject.com/KB/Android/803646/rfile.png

Layout File

The activity_main.xml is a layout file available in res/layout directory, that is referenced by your application when building its interface. You will modify this file very frequently

to change the layout of your application. For your "Hello World!" application, this file will have following content related to default layout

Text Mode

https://www.codeproject.com/KB/Android/803646/Layouttext.png

Layout Design

https://www.codeproject.com/KB/Android/803646/layoutdesign.png

Running app on Emulator

Emulator takes more time to load so before running app we should start emulator. Emulator can be started from SDK manager tools Manage AVDs.

https://www.codeproject.com/KB/Android/803646/SDKManager.png**AVD Manager**

Create the AVD and click on the start to run the emulator

https://www.codeproject.com/KB/Android/803646/AVDmanagers.png

Once emulator is started it will get loaded by creating an AVD Manager

https://www.codeproject.com/KB/Android/803646/Emulator.png

Things to do on Mobile Device

Pre-requisite

Generate a signed APK from the Android Studio under Build / generate signed APK.

Steps

1. Go to settings on Mobile Device
2. Tap on applications or Developer options
3. If it is applications options on mobile device follow below steps
    a. Put a check for Unknown Sources (to allow installation of non-Market applications)
    b. Tap on Development (to set options for application development)
4. Check on USB debugging
5. Plug the USB cable to computer.
6. Go the platform-tools under studio directory and run the following comment
    a. adb install app-release.apk. App installs
    b. On success full install you can run the app on mobile.

PART-B
Two mark Questions

1. Give short note on ADT.
2. Define Method overloading.
3. What is meant by emulator?
4. What is meant by a widget?
5. What is the function of Edit Text field?

PART-C
EIGHT MARK QUESTIONS

1.Explain the installation procedure of Eclipse with ADT plug-in. .
2.Explain the steps of installing Virtual machine for Android sandwich.
3.Explain the steps of installing Virtual machine for Android Jelly bean.
4.Discuss configuring the installed tools of Android.
5.Explain the steps to create the Android project-Hello World.
6.Discuss all the toggle button and spinner controls in Android.
7.Explain emulator for android.
8.Explain how to create an android project?
9.Explain the installation of virtual machine.
10.Discuss the steps to run an android project in an emulator.

QUESTIONS	OPT1	OPT2	OPT3	OPT4	OPT5	OPT6	ANSWER
What is Pending Intent in android?	It is a kind of an intent	It is used to pass the data between activities	It will fire at a future point of time	None of the Above			It will fire at a future point of time
What is the life cycle of services in android?	onCreate()−>onStartCommand()−>onDestory()	onRecieve()	final()	Service life cycle is same as activity life cycle.			onCreate()−>onStartCommand()−>onDestory()
How many threads are there in asyncTask in android?	Only one	Two	AsyncTask doesn't have tread	None of the Above			Only one
How to store heavy structured data in android?	Shared Preferences	Cursor	SQlite database	Not possible			SQlite database
What is singleton class in android?	A class that can create only one object	Anonymous class	Java class	Manifest file			A class that can create only one object
What is ADB in android?	Image tool	Development tool	Android Debug Bridge	None of the above.			Android Debug Bridge
What is an HTTP client class in android?	httprequest(get/post) and returns response from the server	Cookies management	Authentication management	None of the above			httprequest(get/post) and returns response from the server

What is fragment life cycle in android?	onReceive()	onCreate()	onAttach()->onCreate() –> onCreateView() –> onActivityCreated() –> onStart() –> onResume()	None of the above			onAttach()->onCreate() –> onCreateView() –> onActivityCreated() –> onStart() –> onResume()
What is the purpose of super.onCreate() in android?	To create an activity	To create a graphical window for subclass	It allows the developers to write the program	None of the above			To create a graphical window for subclass
What is off-line synchronization in android?	Synchronization with internet	Background synchronization	Synchronization without internet	None of the above			Synchronization without internet
_____specifies how child Views are positioned.	android:layout_weight	android:layout_gravity	android:layout_width	android : layout_x			android:layout_gravity
_____Layout is a view group that aligns all children in a single direction, vertically or horizontally.	Relative	Table	Linear	Frame			Linear
_____specifies how much of the extra space in the layout should be allocated to the View.	android:layout_gravity	android : layout_x	android:layout_weight	android:layout_width			android:layout_weight
Which are the screen sizes in Android?	small	normal	large	a & b & c			a & b & c
You can shut down an activity by calling its _____ method	onDestory()	finishActivity()	a & b	finish()			finish()
What is off-line synchronization in android?	Synchronization with internet	Background synchronization	Synchronization without internet	None of the above			Synchronization without internet
_____Layout is a view group that displays child views in relative positions.	Table	Relative	Frame	Linear			Relative

What is fragment life cycle in android?	onReceive()	onCreate()	onAttach()->onCreate() –> onCreateView() –> onActivityCreated() –> onStart() –> onResume()	None of the above			onAttach()->onCreate() –> onCreateView() –> onActivityCreated() –> onStart() –> onResume()
Which component is not activated by an Intent?	activity	services	contentProvider	broadcastReceiver			contentProvider
What are the indirect Direct subclasses of Activity?	launcherActivity	preferenceActivity	tabActivity	a & b & c			a & b & c
Characteristics of the Loaders?	they are available to every Activity and Fragment.	they provide asynchronous loading of data	they monitor the source of their data and deliver new results when the content changes	all of the above			all of the above
Parent class of Service?	Object	Context	ContextWrapper	ContextThemeWrapper			ContextWrapper
_____Layout is a view that groups views into rows and columns.	Relative	Frame	Table	Linear			Table
What are the indirect Direct subclasses of Services?	recognitionService	remoteViewsService	spellCheckerService	inputMethodService			inputMethodService
What is the life cycle of services in android?	onCreate()–>onStartCommand()–>onDestory()	onRecieve()	final()	Service life cycle is same as activity life cycle.			onCreate()–>onStartCommand()–>onDestory()

If your service is private to your own application and runs in the same process as the client (which is common), you should create your interface by extending the _____class?	messenger	binder	AIDL	AISL			binder
If you need your interface to work across different processes, you can create an interface for the service with a _____?	Binder	Messenger	AIDL	b or c			b or c
_____ is a drop-down list that allows users to select one value from a set.	Spinner	Check box	Drop down list box	Dialog box			Spinner
_____Layout enables you to specify the exact location of its children.	Linear	Absolute	Relative	Frame			Absolute
Once installed on a device, each Android application lives in_____?	device memory	external memory	security sandbox	a & b			security sandbox
What are the Direct subclasses of Activity?	ListActivity	ActivityGroup	Fragment Activity	All of the above			All of the above
When contentProvider would be activated?	using Intent	using SQLite	using ContentResolver	usingOracle			using ContentResolver
Difference between android api and google api?	The google API includes Google Maps and other Google-specific libraries. The Android one only includes core android libraries	The google API one only includes core android libraries. The Android includes Google Maps and other Google-specific libraries	Both a&b	No differences			The google API includes Google Maps and other Google-specific libraries. The Android one only includes core android libraries
The XML file that contains all the text that your application uses.	stack.xml	text.xml	strings.xml	string.java			strings.xml

_____Layout is a placeholder on screen that you can use to display a single view.	Linear	Absolute	Frame	Relative			Frame
How is a simulator different from an emulator?	Emulators are only used to play old SNES games, simulators are used for software development	The emulator is shipped with the Android SDK and third party simulators are not	The emulator can virtualize sensors and other hardware features, while the simulator cannot	The emulator imitates the machine executing the binary code, rather than simulating the behaviour of the code at a higher level			
Which piece of code used in Android is not open source?	Keypad driver	WiFi-driver	Audio driver	Power management			WiFi-driver
How many ways to start services?	started	bound	a & b	messenger			a & b
When the activity is not in focus, but still visible on the screen it is in?	running state	stopped state	paused state	destroyed state			paused state
What are the indirect Direct subclasses of Activity?	launcher Activity	preferenceActivity	tabActivity	a & b & c			a & b & c
The XML file that contains all the text that your application uses.	stack.xml	text.xml	strings.xml	string.java			strings.xml
Which among these are NOT a part of Android's native libraries?	Webkit	Dalvik	SQLite	OpenGL			Dalvik
What was the main reason for replacing the Java VM with the Dalvik VM when the project began?	There was not enough memory capability	Java virtual machine was not free	Java VM was too complicated to configure	Java VM ran too slow			Java virtual machine was not free

Definition of Loader?	loaders make it easy to synchronously load data in an activity or fragment	loaders make it easy to asynchronously load data in an activity or fragment.	loaders does not make it easy to asynchronously load data in an activity or fragment	Loaders are adequately load data in the forms			loaders make it easy to asynchronously load data in an activity or fragment.
_____Layout is a placeholder on screen that you can use to display a single view.	Linear	Absolute	Frame	Relative			Frame
How many ways to start services?	started	bound	a & b	messenger			a & b
Which one is NOT related to fragment class?	dialogFragment	listFragment	preferenceFragment	cursorFragment			cursorFragment
What is the difference between Activity context and Application Context?	The Activity instance is tied to the lifecycle of an Activity. while the application instance is tied to the lifecycle of the application	The Activity instance is tied to the lifecycle of the application, while the application instance is tied to the lifecycle of an Activity	The Activity instance is tied to the lifecycle of the Activity, while the application instance is tied to the lifecycle of an application	Both are same			The Activity instance is tied to the lifecycle of an Activity. while the application instance is tied to the lifecycle of the application
_____ is a ViewGroup that displays items in a two-dimensional, scrollable grid.	Grid View	Frame	List View	Linear			Grid View
_____Layout is a view group that aligns all children in a single direction, vertically or horizontally.	Relative	Table	Linear	Frame			Linear
What year was the Open Handset Alliance announced?	2005	2006	2007	2008			2007

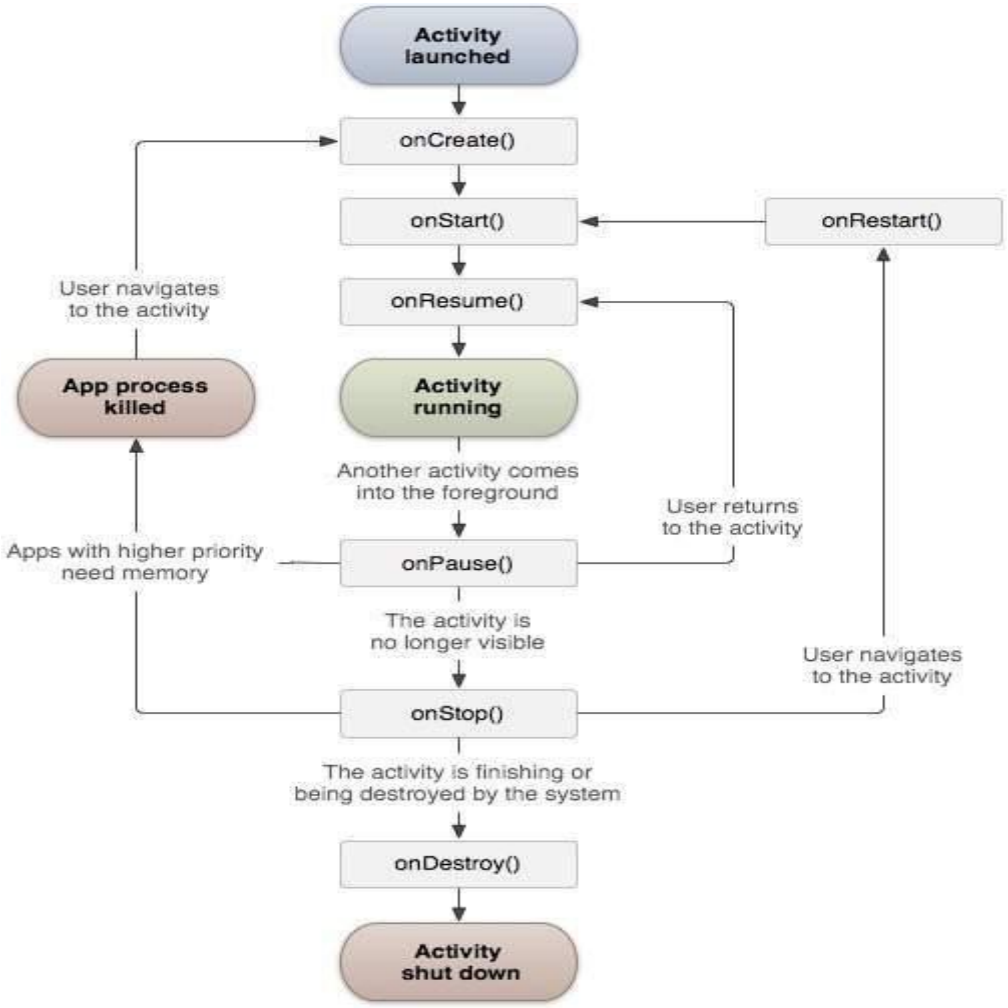
Unit-4 Notes

Syllabus

User Interface Architecture: Application context-intents-Activity life cycle-multiple screen sizes.(2L) User Interface Design: Form widgets-Text Fields-Layouts-Button control-toggle buttons-Spinners(Combo boxes)-Images-Menu-Dialog.(2L

Activity life cycle

In C, C++ or Java programming language, program starts from **main()** function. Very similar way, Android system initiates its program with in an **Activity** starting with a call on onCreate() callback method. There is a sequence of callback methods that start up an activity and a sequence of callback methods that tear down an activity as shown in the below Activity life cycle diagram:



The Activity class defines the following call backs i.e. events. You don't need to implement all the callbacks methods. However, it's important that you understand each one and implement those that ensure your app behaves the way users expect.

Sr.No	Callback & Description
1	**onCreate()** This is the first callback and called when the activity is first created.
2	**onStart()** This callback is called when the activity becomes visible to the user.
3	**onResume()** This is called when the user starts interacting with the application.
4	**onPause()** The paused activity does not receive user input and cannot execute any code and called when the current activity is being paused and the previous activity is being resumed.
5	**onStop()** This callback is called when the activity is no longer visible.
6	**onDestroy()** This callback is called before the activity is destroyed by the system.
7	**onRestart()** This callback is called when the activity restarts after stopping it.

Multiple screen size

Android runs on a variety of devices that offer different screen sizes and densities. For applications, the Android system provides a consistent development environment across devices and handles most of the work to adjust each application's user interface to the screen on which it is displayed. At the same time, the system provides APIs that allow you to control your application's UI for specific screen sizes and densities, in order to optimize your UI design for different screen configurations. For example, you might want a UI for tablets that's different from the UI for handsets.

Although the system performs scaling and resizing to make your application work on different screens, you should make the effort to optimize your application for different screen sizes and densities. In doing so, you maximize the user experience for all devices

and your users believe that your application was actually designed for *their* devices—rather than simply stretched to fit the screen on their devices.

By following the practices described in this document, you can create an application that displays properly and provides an optimized user experience on all supported screen configurations, using a single .apk file.

Overview of Screens Support

This section provides an overview of Android's support for multiple screens, including: an introduction to the terms and concepts used in this document and in the API, a summary of the screen configurations that the system supports, and an overview of the API and underlying screen-compatibility features.

Terms and concepts

Screen size

Actual physical size, measured as the screen's diagonal.

For simplicity, Android groups all actual screen sizes into four generalized sizes: small, normal, large, and extra-large.

Screen density

The quantity of pixels within a physical area of the screen; usually referred to as dpi (dots per inch). For example, a "low" density screen has fewer pixels within a given physical area, compared to a "normal" or "high" density screen.

For simplicity, Android groups all actual screen densities into six generalized densities: low, medium, high, extra-high, extra-extra-high, and extra-extra-extra-high.

Orientation

The orientation of the screen from the user's point of view. This is either landscape or portrait, meaning that the screen's aspect ratio is either wide or tall, respectively. Be aware that not only do different devices operate in different orientations by default, but the orientation can change at runtime when the user rotates the device.

Resolution

The total number of physical pixels on a screen. When adding support for multiple screens, applications do not work directly with resolution; applications should be concerned only with screen size and density, as specified by the generalized size and density groups.

Density-independent pixel (dp)

A virtual pixel unit that you should use when defining UI layout, to express layout dimensions or position in a density-independent way.

Range of screens supported

Android provides support for multiple screen sizes and densities, reflecting the many different screen configurations that a device may have. You can use features of the Android system to optimize your application's user interface for each screen configuration and ensure that your application not only renders properly, but provides the best user experience possible on each screen.

To simplify the way that you design your user interfaces for multiple screens, Android divides the range of actual screen sizes and densities into:

- A set of four generalized **sizes**: *small*, *normal*, *large*, and *xlarge*

  Note: Beginning with Android 3.2 (API level 13), these size groups are deprecated in favor of a new technique for managing screen sizes based on the available screen width. If you're developing for Android 3.2 and greater, see Declaring Tablet Layouts for Android 3.2 for more information.

- A set of six generalized **densities**:
  - *ldpi* (low) ~120dpi
  - *mdpi* (medium) ~160dpi
  - *hdpi* (high) ~240dpi
  - *xhdpi* (extra-high) ~320dpi
  - *xxhdpi* (extra-extra-high) ~480dpi
  - *xxxhdpi* (extra-extra-extra-high) ~640dpi

Each generalized size and density spans a range of actual screen sizes and densities. For example, two devices that both report a screen size of *normal* might have actual screen sizes and aspect ratios that are slightly different when measured by hand. Similarly, two devices that report a screen density of *hdpi* might have real pixel densities that are slightly different. Android makes these differences abstract to applications, so you can provide UI designed for the generalized sizes and densities and let the system handle any final adjustments as necessary. Figure 1 illustrates how different sizes and densities are roughly categorized into the different size and density groups.

Figure 1. Illustration of how Android roughly maps actual sizes and densities to generalized sizes and densities (figures are not exact).

As you design your UI for different screen sizes, you'll discover that each design requires a minimum amount of space. So, each generalized screen size above has an associated minimum resolution that's defined by the system. These minimum sizes are in "dp" units—the same units you should use when defining your layouts—which allows the system to avoid worrying about changes in screen density.

- *xlarge* screens are at least 960dp x 720dp

- *large* screens are at least 640dp x 480dp
- *normal* screens are at least 470dp x 320dp
- *small* screens are at least 426dp x 320dp

Note: These minimum screen sizes were not as well defined prior to Android 3.0, so you may encounter some devices that are mis-classified between normal and large. These are also based on the physical resolution of the screen, so may vary across devices—for example a 1024x720 tablet with a system bar actually has a bit less space available to the application due to it being used by the system bar.

To optimize your application's UI for the different screen sizes and densities, you can provide alternative resources for any of the generalized sizes and densities. Typically, you should provide alternative layouts for some of the different screen sizes and alternative bitmap images for different screen densities. At runtime, the system uses the appropriate resources for your application, based on the generalized size or density of the current device screen.

You do not need to provide alternative resources for every combination of screen size and density. The system provides robust compatibility features that can handle most of the work of rendering your application on any device screen, provided that you've implemented your UI using techniques that allow it to gracefully resize (as described in the Best Practices, below).

Density independence

Your application achieves "density independence" when it preserves the physical size (from the user's point of view) of user interface elements when displayed on screens with different densities.

Maintaining density independence is important because, without it, a UI element (such as a button) appears physically larger on a low-density screen and smaller on a high-density screen. Such density-related size changes can cause problems in your application layout and usability. Figures 2 and 3 show the difference between an application when it does not provide density independence and when it does, respectively.

Example application without support for different densities, as shown on low, medium, and high-density screens.

Example application with good support for different densities (it's density independent), as shown on low, medium, and high density screens.

The Android system helps your application achieve density independence in two ways:

- The system scales dp units as appropriate for the current screen density
- The system scales drawable resources to the appropriate size, based on the current screen density, if necessary

In most cases, you can ensure density independence in your application simply by specifying all layout dimension values in density-independent pixels (dp units) or with "wrap_content", as appropriate. The system then scales bitmap drawables as appropriate in order to display at the appropriate size, based on the appropriate scaling factor for the current screen's density.

However, bitmap scaling can result in blurry or pixelated bitmaps, which you might notice in the above screenshots. To avoid these artifacts, you should provide alternative bitmap resources for different densities. For example, you should provide higher-resolution bitmaps for high-density screens and the system will use those instead of resizing the bitmap designed for medium-density screens.

Intents

An Android **Intent** is an abstract description of an operation to be performed. It can be used with **startActivity** to launch an Activity, **broadcastIntent** to send it to any interested BroadcastReceiver components, and **startService(Intent)** or **bindService(Intent, ServiceConnection, int)** to communicate with a background Service.

For example, let's assume that you have an Activity that needs to launch an email client and sends an email using your Android device. For this purpose, your Activity would send an ACTION_SEND along with appropriate **chooser**, to the Android Intent Resolver. The specified chooser gives the proper interface for the user to pick how to send your email data.

```
Intent email = new Intent(Intent.ACTION_SEND, Uri.parse("mailto:"));
email.putExtra(Intent.EXTRA_EMAIL, recipients);
email.putExtra(Intent.EXTRA_SUBJECT, subject.getText().toString());
email.putExtra(Intent.EXTRA_TEXT, body.getText().toString());
startActivity(Intent.createChooser(email, "Choose an email client from..."));
```

Above syntax is calling startActivity method to start an email activity and result should be as shown below −

There are separate mechanisms for delivering intents to each type of component − activities, services, and broadcast receivers.

Sr.No	Method & Description
1	**Context.startActivity()** The Intent object is passed to this method to launch a new activity or get an existing activity to do something new.
2	**Context.startService()** The Intent object is passed to this method to initiate a service or deliver new instructions to an ongoing service.
3	**Context.sendBroadcast()**

	The Intent object is passed to this method to deliver the message to all interested broadcast receivers.

Intent Objects

Android Intent 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 **dictionary meaning** of intent is *intention or purpose*. So, it can be described as the intention to do action.

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.

Action

This is mandatory part of the Intent object and is a string naming the action to be performed — or, in the case of broadcast intents, the action that took place and is being reported. The action largely determines how the rest of the intent object is structured . The Intent class defines a number of action constants corresponding to different intents. Here is a list of Android Intent Standard Actions

The action in an Intent object can be set by the setAction() method and read by getAction().

Data

Adds a data specification to an intent filter. The specification can be just a data type (the mimeType attribute), just a URI, or both a data type and a URI. A URI is specified by separate attributes for each of its parts −

These attributes that specify the URL format are optional, but also mutually dependent −

- If a scheme is not specified for the intent filter, all the other URI attributes are ignored.

- If a host is not specified for the filter, the port attribute and all the path attributes are ignored.

The setData() method specifies data only as a URI, setType() specifies it only as a MIME type, and setDataAndType() specifies it as both a URI and a MIME type. The URI is read by getData() and the type by getType().

Some examples of action/data pairs are –

Sr.No.	Action/Data Pair & Description
1	**ACTION_VIEW content://contacts/people/1** Display information about the person whose identifier is "1".
2	**ACTION_DIAL content://contacts/people/1** Display the phone dialer with the person filled in.
3	**ACTION_VIEW tel:123** Display the phone dialer with the given number filled in.
4	**ACTION_DIAL tel:123** Display the phone dialer with the given number filled in.
5	**ACTION_EDIT content://contacts/people/1** Edit information about the person whose identifier is "1".
6	**ACTION_VIEW content://contacts/people/** Display a list of people, which the user can browse through.
7	**ACTION_SET_WALLPAPER** Show settings for choosing wallpaper

User Interface Design: Form widgets

There are given a lot of **android widgets** with simplified examples such as Button, EditText, AutoCompleteTextView, ToggleButton, DatePicker, TimePicker, ProgressBar etc.

Android widgets are easy to learn. The widely used android widgets with examples are given below:

Android Button

Let's learn how to perform event handling on button click.

Android Toast

Displays information for the short duration of time.

Custom Toast

We are able to customize the toast, such as we can display image on the toast

ToggleButton

It has two states ON/OFF.

CheckBox

Let's see the application of simple food ordering.

AlertDialog

AlertDialog displays a alert dialog containing the message with OK and Cancel buttons.

Spinner

Spinner displays the multiple options, but only one can be selected at a time.

AutoCompleteTextView

Let's see the simple example of AutoCompleteTextView.

RatingBar

RatingBar displays the rating bar.

DatePicker

Datepicker displays the datepicker dialog that can be used to pick the date.

TimePicker

TimePicker displays the timepicker dialog that can be used to pick the time.

ProgressBar

ProgressBar displays progress task.

Button control

A Button is a Push-button which can be pressed, or clicked, by the user to perform an action.

Button Attributes

Following are the important attributes related to Button control. You can check Android official documentation for complete list of attributes and related methods which you can use to change these attributes are run time.

Inherited from **android.widget.TextView** Class −

Sr.No	Attribute & Description
1	**android:autoText** If set, specifies that this TextView has a textual input method and automatically corrects some common spelling errors.
2	**android:drawableBottom** This is the drawable to be drawn below the text.
3	**android:drawableRight** This is the drawable to be drawn to the right of the text.
4	**android:editable** If set, specifies that this TextView has an input method.
5	**android:text** This is the Text to display.

Android Button represents a push-button. The android.widget.Button is subclass of TextView class and CompoundButton is the subclass of Button class.

There are different types of buttons in android such as RadioButton, ToggleButton, CompoundButton etc.

Here, we are going to create two textfields and one button for sum of two numbers. If user clicks button, sum of two input values is displayed on the Toast.

Drag the component or write the code for UI in activity_main.xml

First of all, drag 2 textfields from the Text Fields palette and one button from the Form Widgets palette as shown in the following figure.

The generated code for the ui components will be like this:

File: activity_main.xml

```
<RelativeLayout xmlns:androclass="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity" >

   <EditText
      android:id="@+id/editText1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="24dp"
      android:ems="10" />

   <EditText
      android:id="@+id/editText2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignLeft="@+id/editText1"
      android:layout_below="@+id/editText1"
      android:layout_marginTop="34dp"
      android:ems="10" >
```

Layouts

We have different layouts which are subclasses of ViewGroup class and a typical layout defines the visual structure for an Android user interface and can be created either at run time using **View/ViewGroup** objects or you can declare your layout using simple XML file **main_layout.xml** which is located in the res/layout folder of your project.

A layout may contain any type of widgets such as buttons, labels, textboxes, and so on. Following is a simple example of XML file having LinearLayout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/androic
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

   <TextView android:id="@+id/text"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="This is a TextView" />

   <Button android:id="@+id/button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="This is a Button" />

   <!-- More GUI components go here  -->

</LinearLayout>
```

Android Layout Types

There are number of Layouts provided by Android which you will use in almost all the Android applications to provide different view, look and feel.

Sr.No	Layout & Description
1	**Linear Layout** LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally.
2	**Relative Layout** RelativeLayout is a view group that displays child views in relative positions.
3	**Table Layout** TableLayout is a view that groups views into rows and columns.
4	**Absolute Layout** AbsoluteLayout enables you to specify the exact location of its children.
5	**Frame Layout** The FrameLayout is a placeholder on screen that you can use to display a single view.
6	**List View** ListView is a view group that displays a list of scrollable items.

7	**Grid View** GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid.

Layout Attributes

Each layout has a set of attributes which define the visual properties of that layout. There are few common attributes among all the layouts and their are other attributes which are specific to that layout. Following are common attributes and will be applied to all the layouts:

Sr.No	Attribute & Description
1	**android:id** This is the ID which uniquely identifies the view.
2	**android:layout_width** This is the width of the layout.
3	**android:layout_height** This is the height of the layout
4	**android:layout_marginTop** This is the extra space on the top side of the layout.
5	**android:layout_marginBottom** This is the extra space on the bottom side of the layout.
6	**android:layout_marginLeft** This is the extra space on the left side of the layout.
7	**android:layout_marginRight** This is the extra space on the right side of the layout.

Toggle button

A ToggleButton displays checked/unchecked states as a button. It is basically an on/off button with a light indicator.

Toggle Button

Android Toggle Button can be used to display checked/unchecked (On/Off) state on the button.

It is beneficial if user have to change the setting between two states. It can be used to On/Off Sound, Wifi, Bluetooth etc.

Since Android 4.0, there is another type of toggle button called *switch* that provides slider control.

Android ToggleButton and Switch both are the subclasses of CompoundButton class.

Android ToggleButton class

ToggleButton class provides the facility of creating the toggle button.

XML Attributes of ToggleButton class

The 3 XML attributes of ToggleButton class.

XML Attribute	Description
android:disabledAlpha	The alpha to apply to the indicator when disabled.
android:textOff	The text for the button when it is not checked.
android:textOn	The text for the button when it is checked.

Methods of ToggleButton class

The widely used methods of ToggleButton class are given below.

Method	Description
CharSequence getTextOff()	Returns the text when button is not in the checked state.
CharSequence getTextOn()	Returns the text for when button is in the checked

	state.
void setChecked(boolean checked)	Changes the checked state of this button.

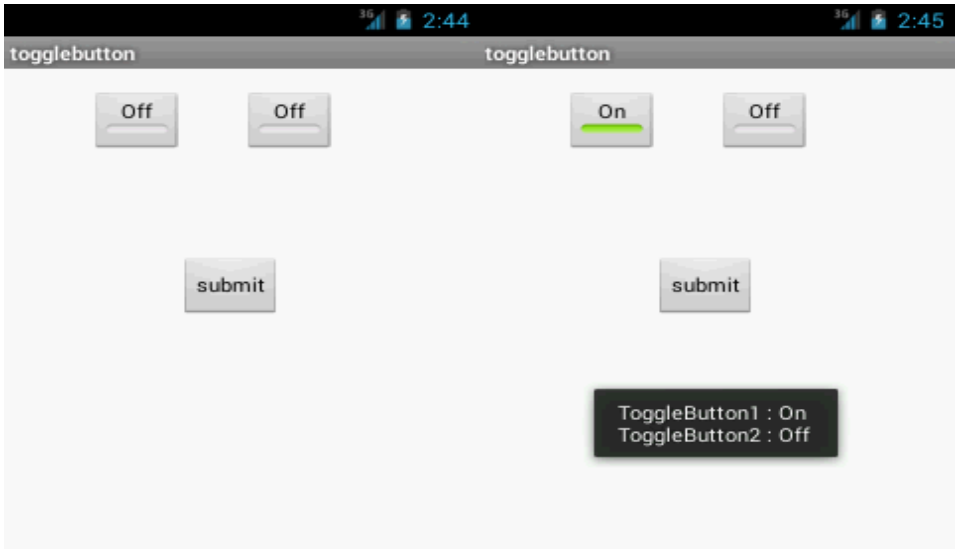
File: activity_main.xml

```
<RelativeLayout xmlns:androclass="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity" >

  <ToggleButton
    android:id="@+id/toggleButton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:layout_marginLeft="60dp"
    android:layout_marginTop="18dp"
    android:text="ToggleButton1"
    android:textOff="Off"
    android:textOn="On" />
```

File: MainActivity.java

```
package com.example.togglebutton;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;
import android.widget.ToggleButton;

public class MainActivity extends Activity {
   private ToggleButton toggleButton1, toggleButton2;
   private Button buttonSubmit;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      addListenerOnButtonClick();
   }
```

```
public void addListenerOnButtonClick(){
   //Getting the ToggleButton and Button instance from the layout xml file
   toggleButton1=(ToggleButton)findViewById(R.id.toggleButton1);
   toggleButton2=(ToggleButton)findViewById(R.id.toggleButton2);
   buttonSubmit=(Button)findViewById(R.id.button1);

   //Performing action on button click
   buttonSubmit.setOnClickListener(new OnClickListener(){

      @Override
      public void onClick(View view) {
         StringBuilder result = new StringBuilder();
           result.append("ToggleButton1 : ").append(toggleButton1.getText());
           result.append("\nToggleButton2 : ").append(toggleButton2.getText());
         //Displaying the message in toast
         Toast.makeText(getApplicationContext(), result.toString(),Toast.LENGTH_LONG).show();
      }

   });
```



Android Spinner (Combo boxes)

Android Spinner is like the combox box of AWT or Swing. It can be used to display the multiple options to the user in which only one item can be selected by the user.

Android spinner is like the drop down menu with multiple values from which the end user can select only one value.

Android spinner is associated with AdapterView. So you need to use one of the adapter classes with spinner.

Android Spinner class is the subclass of AsbSpinner class.

Android Spinner Example

In this example, we are going to display the country list. You need to use **ArrayAdapter** class to store the country list.

Let's see the simple example of spinner in android.

activity_main.xml

Drag the Spinner from the pallete, now the activity_main.xml file will like this:

File: activity_main.xml

```
<RelativeLayout xmlns:androclass="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity" >

   <Spinner
      android:id="@+id/spinner1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="83dp" />

</RelativeLayout>
```

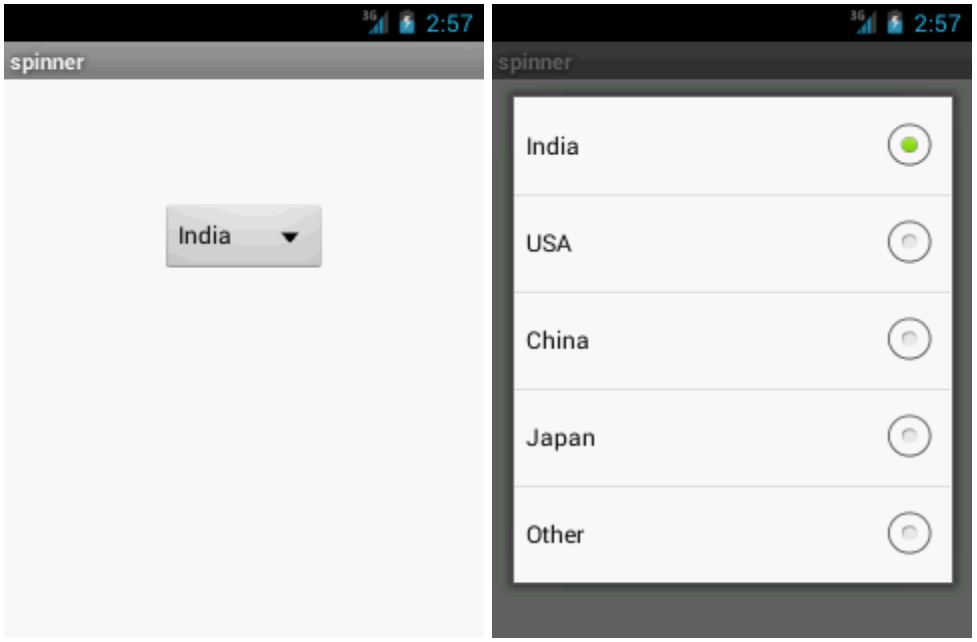
Activity class

Let's write the code to display item on the spinner and perform event handling.

File: MainActivity.java

```
package com.example.spinner;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends Activity implements
AdapterView.OnItemSelectedListener {

   String[] country = { "India", "USA", "China", "Japan", "Other", };

   @Override
   protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //Getting the instance of Spinner and applying OnItemSelectedListener on it
    Spinner spin = (Spinner) findViewById(R.id.spinner1);
    spin.setOnItemSelectedListener(this);

    //Creating the ArrayAdapter instance having the country list
    ArrayAdapter aa = new ArrayAdapter(this,android.R.layout.simple_spinner_item,country);
    aa.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    //Setting the ArrayAdapter data on the Spinner
    spin.setAdapter(aa);
}


//Performing action onItemSelected and onNothing selected
@Override
public void onItemSelected(AdapterView<?> arg0, View arg1, int position,long id) {
    Toast.makeText(getApplicationContext(),country[position] ,Toast.LENGTH_LONG).show();
}
```



Images

Android provides many views which we can use to define a user interface for our apps. Amongst these it provides a large number to display information and take input from the user, these include text and image views.

Android provides views which can be used to display images from various sources and provide transitions between them. Some of these views are the ImageView and the ImageSwitcher. These views provide a high level of functionality to display images in a user interface so that we can concentrate on the images we want to display rather than taking care of rendering.

Nested classes

class	Image.Plane A single color plane of image data.

Public methods

abstract void	close() Free up this frame for reuse.
Rect	getCropRect() Get the crop rectangle associated with this frame.
abstract int	getFormat() Get the format for this image.
abstract int	getHeight() The height of the image in pixels.
abstract Plane[]	getPlanes() Get the array of pixel planes for this Image.
abstract long	getTimestamp() Get the timestamp associated with this frame.
abstract int	getWidth() The width of the image in pixels.
void	setCropRect(Rect cropRect) Set the crop rectangle associated with this frame.
void	setTimestamp(long timestamp) Set the timestamp associated with this frame.

```
public class ImageGalleryActivity extends Activity {

   private Integer images[] = {R.drawable.pic1, R.drawable.pic2, R.drawable.pic3};

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

   private void addImagesToThegallery() {
      LinearLayout imageGallery = (LinearLayout) findViewById(R.id.imageGallery);
      for (Integer image : images) {
         imageGallery.addView(getImageView(image));
      }
   }

   private View getImageView(Integer image) {
      ImageView imageView = new ImageView(getApplicationContext());
      LinearLayout.LayoutParams lp = new
LinearLayout.LayoutParams(LinearLayout.LayoutParams.WRAP_CONTENT,
LinearLayout.LayoutParams.WRAP_CONTENT);
      lp.setMargins(0, 0, 10, 0);
      imageView.setLayoutParams(lp);
      imageView.setImageResource(image);
      return imageView;
   }
}
```

Menus

Menus are a common user interface component in many types of applications. To provide a familiar and consistent user experience, you should use the Menu APIs to present user actions and other options in your activities.

Beginning with Android 3.0 (API level 11), Android-powered devices are no longer required to provide a dedicated *Menu* button. With this change, Android apps should migrate away from a dependence on the traditional 6-item menu panel and instead provide an app bar to present common user actions.

Although the design and user experience for some menu items have changed, the semantics to define a set of actions and options is still based on the Menu APIs. This guide shows how to create the three fundamental types of menus or action presentations on all versions of Android.

Options menu and app bar

The options menu is the primary collection of menu items for an activity. It's where you should place actions that have a global impact on the app, such as "Search," "Compose email," and "Settings."

Context menu and contextual action mode

A context menu is a floating menu that appears when the user performs a long-click on an element. It provides actions that affect the selected content or context frame.

Popup menu

A popup menu displays a list of items in a vertical list that's anchored to the view that invoked the menu.

Android Option Menus are the primary menus of android. They can be used for settings, search, delete item etc.

Here, we are going to see two examples of option menus. First, the simple option menus and second, options menus with images.

Here, we are inflating the menu by calling the **inflate()** method of **MenuInflater** class. To perform event handling on menu items, you need to override **onOptionsItemSelected()** method of Activity class.

Android Option Menu Example

Let's see how to create menu in android. Let's see the simple option menu example that contains three menu items.

activity_main.xml

We have only one textview in this file.

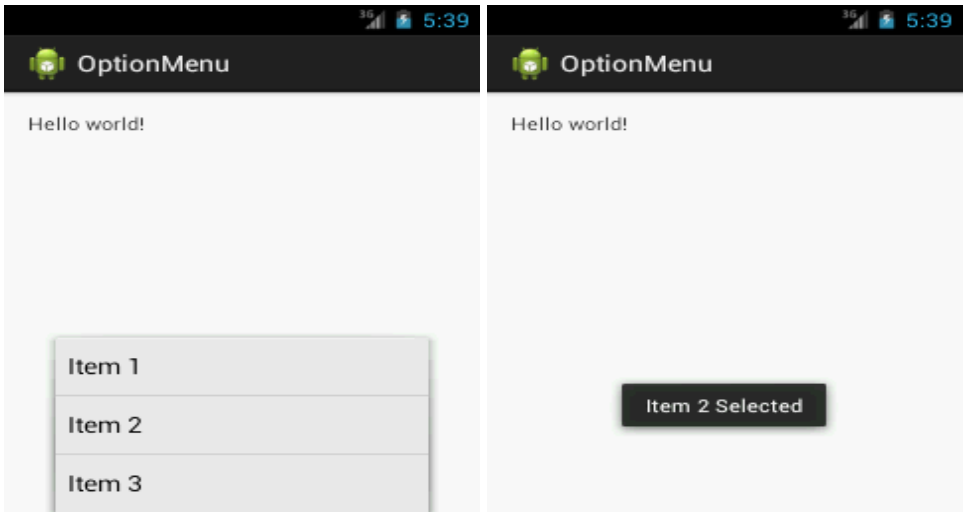
File: activity_main.xml

menu_main.xml

It contains three items as show below. It is created automatically inside the res/menu directory.

File: menu_main.xml

```
<menu xmlns:androclass="http://schemas.android.com/apk/res/android" >
   <item  android:id="@+id/item1"
      android:title="Item 1"/>
   <item  android:id="@+id/item2"
      android:title="Item 2"/>
   <item  android:id="@+id/item3"
      android:title="Item 3"/>
</menu>
```



Dialog

A Dialog is small window that prompts the user to a decision or enter additional information. A dialog does not fill the screen and is normally used for modal events that require users to take an action before they can proceed.

In order to make an alert dialog, you need to make an object of AlertDialogBuilder which an inner class of AlertDialog. Its syntax is given below

AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);

Now you have to set the positive (yes) or negative (no) button using the object of the AlertDialogBuilder class. Its syntax is

alertDialogBuilder.setPositiveButton(CharSequence text,
  DialogInterface.OnClickListener listener)
alertDialogBuilder.setNegativeButton(CharSequence text,
  DialogInterface.OnClickListener listener)

Apart from this , you can use other functions provided by the builder class to customize the alert dialog. These are listed below

Sr.No	Method type & description
1	**setIcon(Drawable icon)** This method set the icon of the alert dialog box.
2	**setCancelable(boolean cancel able)** This method sets the property that the dialog can be cancelled or not
3	**setMessage(CharSequence message)** This method sets the message to be displayed in the alert dialog
4	**setMultiChoiceItems(CharSequence[] items, boolean[] checkedItems, DialogInterface.OnMultiChoiceClickListener listener)** This method sets list of items to be displayed in the dialog as the content. The selected option will be notified by the listener
5	**setOnCancelListener(DialogInterface.OnCancelListener onCancelListener)** This method Sets the callback that will be called if the dialog is cancelled.
6	**setTitle(CharSequence title)** This method set the title to be appear in the dialog

Application context

It is an instance which can be accessed in an activity via getApplicationContext(). This context is tied to the lifecycle of an application. The application context can be used where you need a context whose lifecycle is separate from the current context or when you are passing a context beyond the scope of an activity.

We generally call context when we need to get information about different parts of our application like Activities, Applications etc.

Some operations(things where assistant is needed) where context is involved:

1. Loading common resources
2. Creating dynamic views
3. Displaying Toast messages
4. Launching Activities etc.

Different ways of getting context:

- getContext()
- getBaseContext()
- getApplicationContext()
- this

Need of Context :

The documentation says that every view needs the context to access the right resources (e.g. the theme, strings etc.).

1.Because the resources must be accessible while the view is being constructed (the constructor will need some resources to fully initialise the view).

2.This allows the flexibility of using a context that is different from the one of the current activity (imagine a view that uses some other string resources and not the ones from the current activity).

3.The designers of the Android SDK seem to have chosen that the context must be set only once and then stay the same throughout the lifetime of the view.

PART-B
Two mark Questions

1. What is meant by Android Debug Bridge?
2. Define Method overriding.
3. Define API
4. List out the image formats supported by Android?
5. Write the function of OnClickListener?

PART-C
EIGHT MARK QUESTIONS

1.Explain Application context with suitable program.
2.Discuss the user interface design of Android.
3.Discuss TextField in Android with suitable example.
4.Discuss Layouts and Button control of Android.
5.Explain Activity Life cycle in User Interface Architecture
6.Discuss in detail (i) Form Widgets (ii) Images
7.Discuss the user interface architecture of Android.
8.Explain Menu and Dialog controls of Android.
9.Explain the user interface design of Android.
10.Discuss Application context and intents of Android user interface architecture.

QUESTIONS	OPT1	OPT2	OPT3	OPT4	OPT5	OPT6	ANSWER
While developing Android applications, developers can test their apps on…	Emulator included in Android SDK	Physical Android phone	Third-party Emulators (Youwave, etc.)	All these options work			All these options work
How is a simulator different from an emulator?	Emulators are only used to play old SNES games, simulators are used for software development	The emulator is shipped with the Android SDK and third party simulators are not	The emulator can virtualize sensors and other hardware features, while the simulator cannot	The emulator imitates the machine executing the binary code, rather than simulating the behaviour of the code at a higher level			The emulator imitates the machine executing the binary code, rather than simulating the behaviour of the code at a higher level
The Emulator is identical to running a real phone EXCEPT when emulating/simulating what?	Telephony	Applications	Sensors	The emulator can emulate/simulate all aspects of a smart phone			Sensors
Which of these are not one of the three main components of the APK?	Dalvik Executable	Resources	Native Libraries	Webkit			Webkit
Which are the screen sizes in Android?	small	normal	large	a & b & c			a & b & c
Parent class of Activity?	object	Context	activityGroup	contextThemeWrapper			contextThemeWrapper

What file is responsible for glueing everything together, explaining what the application consists of, what its main building blocks are, ext.…?	Layout file	Strings XML	R file	Manifest file			Manifest file
Parent class of Service?	Object	Context	Context Wrapper	ContextThemeWrapper			ContextWrapper
If the UI begins to behave sluggishly or crash while making network calls, this is likely due to…	Network latency	Hardware malfunctions	Virus on the Server	Activity manager contains too much.			Network latency
Android tries hard to _____low-level components, such as the software stack, with interfaces so that vendor-specific code can be managed easily.	confound	abstract	modularize	compound			abstract
Creating a UI (User Interface) in Android requires careful use of…	Java and SQL	XML and Java	XML and C++	Dream weaver			XML and Java
Which are the screen densities in Android?	low density	medium density	extra high density	all of the above			all of the above
Dialog classes in android?	AlertDialog	Progress Dialog	DatePickerDialog	all the above classes			all the above classes
What is the name of the program that converts Java byte code into Dalvik byte code?	Android Interpretive Compiler (AIC)	Dalvik Converter	Dex compiler	Mobile Interpretive Compiler (MIC)			Dex compiler
Which of the following should be used to save the unsaved data and release resources being used by an Android application?	Activity.onStop()	Activity.onPause()	Activity.onDestroy()	Activity.onShutdown()			Activity.onDestroy()

What is the purpose of the ContentProvider class?	To play rich media content files	To create and publish rich media files	To share data between Android applications	To access the global information about an application environment			To share data between Android applications
Layouts in android?	Frame Layout	Relative Layout	Linear Layout	All of the above			All of the above
How many ways to start services?	started	bound	a & b	messenger			a & b
Broadcast receivers are Android's implementation of a system-wide publish/subscribe mechanism, or more precisely, what design pattern?	Observer	Mediator	Command	Facade			Observer
Which of the following would you have to include in your project to use the APIs and classes required to access the camera on the mobile device?	Import android drivers	Import android hardware camera	Import android camera	Import android util			Import android hardware camera
Android tries hard to _____low-level components, such as the software stack, with interfaces so that vendor-specific code can be managed easily.	confound	abstract	modularize	compound			abstract
Immediate base class for activity and services	CONTEXT	APPLICATIONCONTEXT	CONTEXTAPP	ONCREATE			CONTEXT
Which of the following fields of the Message class should be used to store custom message codes about the Message?	tag	what	arg1	userData			what
Which of the following can you use to display a progress bar in an Android application?	Progress Bar	Progress Dialog	Progress View	Both a&b			Both a&b

Which of the following is/are appropriate for saving the state of an Android application?	Activity.onFreeze()	Activity.onPause()	Activity.onStop()	Activity.onDestroy()			Activity.onPause()
The R file is a(an) generated file	Automatically	Manually	Emulated	Backup automatically			Automatically
Which of the following can you use to add items to the screen menu?	Activity.onCreateOptionsMenu	Activity.onCreate	Activity.onPrepareOptionsMenu	Both a&b			Both a&b
Which of the following are valid features that you can request using requestWindowFeature?	FEATURE_NO_TITLE	FEATURE_NO_ICON	FEATURE_RIGHT_ICON	Both a& c			Both a& c
What is “Android-Positron”?	A command line tool to create Android project files	A framework to create unit tests for Android projects	A resource editor to create user interface for Android applications	A tool to generate Android byte code from .class files			A framework to create unit tests for Android projects
Which answer is not part of the design philosophy talked about in chapter five?	Always whole and complete	Small increments	large increments	Refactoring code			large increments
What is “Android-dx”?	A command line tool to create Android project files	A framework to create unit tests for Android projects	A tool to generate Android byte code from .class files	A resource editor to create user interface for Android applications			A tool to generate Android byte code from .class files
Which of the following is the parent class for the main application class in an Android application that has a user interface?	MIDLet	Android App	Activity	AppLet			Activity
Which of the following are classes that can be used to handle the Bluetooth functionality on a device?	Adapter	Manager	Matcher	BluetoothAdapter			BluetoothAdapter

Which of the following function calls can be used to start a Service from your Android application?	bindService	startService	runService	Both a&b			Both a&b
Which of the following are UI elements that you can use in a window in an Android application?	TextBox	TextView	EditText	Both b&c			Both b&c
Which of the following can be accomplished by using the TelephoneNumberUtil class?	Save a phone number to the contacts in the phone device	Retrieve a phone number from the contacts in the phone device	Delete a phone number from the contacts in the phone device	Format an international telephone number			Format an international telephone number
What does the .apk extension stand for?	Application Package	Application Program Kit	Android Proprietary Kit	Android Package			Application Package
Which of the following can be used to bind data from an SQL database to a ListView in an Android application?	SimpleCursor	SimpleCursorAdapter	SimpleAdapter	SQLiteCursor			SimpleCursorAdapter
Which of the following would you have to include in your project to use the SimpleAdapter class?	import android.content	import android.widget	import android.database	import android.database.sqlite			import android.widget
What is a key difference with the distribution of apps for Android based devices than other mobile device platform applications?	Applications are distributed by Apple App Store only	Applications are distributed by multiple vendors with different policies on applications	Applications are distributed by multiple vendors with the exact same policies on applications	Applications are distributed by the Android Market only			Applications are distributed by multiple vendors with different policies on applications
Android is based on Linux for the following reason	Security	Portability	Networking	All of these			All of these

Unit-5 Notes

Syllabus:
Database: Understanding of SQL database - connecting with the database.

SQLite Database

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.

SQLite supports all the relational database features. In order to access this database, you don't need to establish any kind of connections for it like JDBC,ODBC e.t.c

Database - Package

The main package is android.database.sqlite that contains the classes to manage your own databases

Database - Creation

In order to create a database you just need to call this method openOrCreateDatabase with your database name and mode as a parameter. It returns an instance of SQLite database which you have to receive in your own object.Its syntax is given below

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

Apart from this , there are other functions available in the database package , that does this job. They are listed below

S.No	Method & Description
1	**openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags, DatabaseErrorHandler errorHandler)** This method only opens the existing database with the appropriate flag mode. The common flags mode could be OPEN_READWRITE OPEN_READONLY
2	**openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags)** It is similar to the above method as it also opens the existing database but it does not define any handler to handle the errors of databases
3	**openOrCreateDatabase(String path, SQLiteDatabase.CursorFactory factory)** It not only opens but create the database if it not exists. This method is equivalent to openDatabase method.

	openOrCreateDatabase(File file, SQLiteDatabase.CursorFactory factory)
4	This method is similar to above method but it takes the File object as a path rather then a string. It is equivalent to file.getPath()

Database - Insertion

we can create table or insert data into table using execSQL method defined in SQLiteDatabase class. Its syntax is given below

```
mydatabase.execSQL("CREATE TABLE IF NOT EXISTS TutorialsPoint(Username
VARCHAR,Password VARCHAR);");
mydatabase.execSQL("INSERT INTO TutorialsPoint
VALUES('admin','admin');");
```

This will insert some values into our table in our database. Another method that also does the same job but take some additional parameter is given below

Sr.No	Method & Description
	execSQL(String sql, Object[] bindArgs)
1	This method not only insert data , but also used to update or modify already existing data in database using bind arguments

Database - Fetching

We can retrieve anything from database using an object of the Cursor class. We will call a method of this class called rawQuery and it will return a resultset with the cursor pointing to the table. We can move the cursor forward and retrieve the data.

```
Cursor resultSet = mydatbase.rawQuery("Select * from
TutorialsPoint",null);
resultSet.moveToFirst();
String username = resultSet.getString(0);
String password = resultSet.getString(1);
```

There are other functions available in the Cursor class that allows us to effectively retrieve the data. That includes

Sr.No	Method & Description
	getColumnCount()
1	This method return the total number of columns of the table.
	getColumnIndex(String columnName)
2	This method returns the index number of a column by specifying the name of the

	column	
	getColumnName(int columnIndex)	
3		
	This method returns the name of the column by specifying the index of the column	
	getColumnNames()	
4		
	This method returns the array of all the column names of the table.	
	getCount()	
5		
	This method returns the total number of rows in the cursor	
	getPosition()	
6		
	This method returns the current position of the cursor in the table	
	isClosed()	
7		
	This method returns true if the cursor is closed and return false otherwise	

Database - Helper class

For managing all the operations related to the database , an helper class has been given and is called SQLiteOpenHelper. It automatically manages the creation and update of the database. Its syntax is given below

```
public class DBHelper extends SQLiteOpenHelper {
   public DBHelper(){
      super(context,DATABASE_NAME,null,1);
   }
   public void onCreate(SQLiteDatabase db) {}
   public void onUpgrade(SQLiteDatabase database, int oldVersion, int
newVersion) {}
}
```

Example of android SQLite database

Let's see the simple example of android sqlite database.

File: Contact.java

```
package com.example.sqlite;
public class Contact {
  int _id;
   String _name;
   String _phone_number;
    public Contact(){   }
   public Contact(int id, String name, String _phone_number){
      this._id = id;
      this._name = name;
      this._phone_number = _phone_number;
   }

   public Contact(String name, String _phone_number){
      this._name = name;
      this._phone_number = _phone_number;
   }
   public int getID(){
      return this._id;
   }


   public void setID(int id){
      this._id = id;
   }

   public String getName(){
      return this._name;
   }

   public void setName(String name){
      this._name = name;
   }

   public String getPhoneNumber(){
      return this._phone_number;
   }

   public void setPhoneNumber(String phone_number){
      this._phone_number = phone_number;
   }
}
```

Connecting with the database

MYSQL is used as a database at the webserver and PHP is used to fetch data from the database. Our application will communicate with the PHP page with necessary parameters and PHP will contact MYSQL database and will fetch the result and return the results to us.

PHP - MYSQL

Creating Database

MYSQL database can be created easily using this simple script. The **CREATE DATABASE** statement creates the database.

```
<?php
  $con=mysqli_connect("example.com","username","password");
  $sql="CREATE DATABASE my_db";
  if (mysqli_query($con,$sql)) {
    echo "Database my_db created successfully";
  }
?>
```

Creating Tables

Once database is created, its time to create some tables in the database. The **CREATE TABLE** statement creates the database.

```
<?php
  $con=mysqli_connect("example.com","username","password","my_db");
  $sql="CREATE TABLE table1(Username CHAR(30),Password CHAR(30),Role
CHAR(30))";
  if (mysqli_query($con,$sql)) {
    echo "Table have been created successfully";
  }
?>
```

Inserting Values in tables

When the database and tables are created. Now its time to insert some data into the tables. The Insert Into statement creates the database.

```
<?php
  $con=mysqli_connect("example.com","username","password","my_db");
  $sql="INSERT INTO table1 (FirstName, LastName, Age) VALUES ('admin',
'admin','adminstrator')";
  if (mysqli_query($con,$sql)) {
```

```
    echo "Values have been inserted successfully";
  }
?>
```

Android - Connecting MYSQL

Connecting Via Get Method

There are two ways to connect to MYSQL via PHP page. The first one is called Get method. We will use HttpGet and HttpClient class to connect. Their syntax is given below −

```
URL url = new URL(link);
HttpClient client = new DefaultHttpClient();
HttpGet request = new HttpGet();
request.setURI(new URI(link));
```

After that you need to call **execute** method of HttpClient class and receive it in a HttpResponse object. After that you need to open streams to receive the data.

```
HttpResponse response = client.execute(request);
BufferedReader in = new BufferedReader
(new InputStreamReader(response.getEntity().getContent()));
```

Connecting Via Post Method

In the Post method, the URLEncoder,URLConnection class will be used. The urlencoder will encode the information of the passing variables. It's syntax is given below −

```
URL url = new URL(link);
String data  = URLEncoder.encode("username", "UTF-8")
+ "=" + URLEncoder.encode(username, "UTF-8");
data += "&" + URLEncoder.encode("password", "UTF-8")
+ "=" + URLEncoder.encode(password, "UTF-8");
URLConnection conn = url.openConnection();
```

The last thing you need to do is to write this data to the link. After writing, you need to open stream to receive the responded data.

```
 OutputStreamWriter wr = new  OutputStreamWriter(conn.getOutputStream());
wr.write( data );
BufferedReader reader = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
```

PART-B
Two mark Questions

1. What do you mean by Fastboot?
2. Define thread in java.
3. What is the purpose of xml files in android project?
4. How to run an android project?
5. Write the use of update( ) method in Android.

PART-C
EIGHT MARK QUESTIONS

1.Discuss SQLite database management system.
2. Explain the packages to be implemented in database.
3.Explain working with Data tables using SQLite.
4.Explain SQLite DBMS.
5.Discuss about the queries in SQLite.
6.Explain how to work with Data Tables Using SQLite.
7. Discuss SQLite database.
8.Discuss SQLite data tables.
9.Explain SQLite DBMS.
10.Explain how to connect with database using Android coding.

QUESTIONS	OPT1	OPT2	OPT3	OPT4	OPT5	OPT6	ANSWER
Android is licensed under which open source licensing license?	Gnu's GPL	OSS	Apache/ MIT	Sourceforge			Apache/MIT
An activity can be thought of as corresponding to what?	A Java project	A Java class	A method call	An object field			A Java class
Intents	are messages that are sent among major building blocks	trigger activities to being, services to start or stop, or broadcast	are asynchronous	all of those			all of those
The android OS comes with many useful system services, which include processes you can easily ask for things such as your..	All of these and more	Location	Sensor Readings	WiFi? Hot Spots			All of these and more
Which of the following is the most "resource hungry" part of dealing with Activities on Android?	Closing an app	Suspending an app	Opening a new app	Restoring the most recent app			Opening a new app
Android Applications must be signed	After they are installed	Before they are installed	Never	Within two weeks of installation			Before they are installed
Which of the following would you have to include in your project to use the SimpleAdapter class?	import android.content	import android.widget	import android.database	import android.database.sqlite			import android.widget
What operating system is used as the base of the Android stack?	Linux	Windows	Java	XML			Linux
What runs in the background and doesn't have any UI components?	Intents	Content Providers	Services	Applications			Services

Although most people's first thought when they think of Android is Google, Android is not actually owned by Google. Who owns the Android platform?	Oracle Technology	Dalvik	Open Handset Alliance	The above statement is and Android is owned by Google			Open Handset Alliance
Broadcast receivers are Android's implementation of a system-wide publish/subscribe mechanism, or more precisely, what design pattern?	Observer	Mediator	Command	Facade			Observer
What does the Gargenta mean in his Design Philosophy when he says that the project will, "Always be whole and complete"?	He means that when we finish the entire project we will have a working application, even though there will be points along the way when we will stop and the application will not run	He means that the program must always be able to compile	He means that we will work on the program by adding self-contained chunks to it so ,Each additional chunk simply adds a new functionality to the application	Not known			He means that we will work on the program by adding self-contained chunks to it so ,Each additional chunk simply adds a new functionality to the application
When did Google purchase Android?	2007	2005	2008	2010			2005
Intents	are messages that are sent among major building blocks	trigger activities to being, services to start or stop, or broadcast	are asynchronous	all of those			all of those

As an Android programmer, what version of Android should you use as your minimum development target?	Versions 1.6 or 2.0	Versions 1.0 or 1.1	Versions 1.2 or 1.3	Versions 2.3 or 3.0			Versions 1.6 or 2.0
To create an emulator, you need an AVD. What does it stand for?	Android Virtual Display	Android Virtual Device	Active Virtual Device	Application Virtual Display			Android Virtual Device
What part of the Android platform is open source?	low-level Linux modules	all of these answers #The entire stack is an open source platform	native libraries	application framework			all of these answers #The entire stack is an open source platform
What year was development on the Dalvik virtual machine started?	2003	2005	2007	2006			2005
What is an Activity?	A single screen the user sees on the device at one time	message sent among the major building blocks	A component that runs in the background without any interface	Context referring to the application environment			A single screen the user sees on the device at one time
Android releases since 1.5 have been given nicknames derived how?	Adjective and strange animal	Food	Something that starts w/ ‘A’ -> Something that starts w/ ‘B’…	American states			Food
Which of the following are not a component of an APK file?	Resources	All of these are components of the APK	Native Libraries	Dalvik executable			All of these are components of the APK

Why the so few users are left with versions 1.0 and 1.1?	The first phones were released with version 1.5	1.0 and 1.1 had security holes that forced carriers to recall phones using them	1.0 and 1.1 are just number designations for the version Apple's iPhone is running	Everyone with 1.0 and 1.1 were upgraded to 1.5 over the air automatically			Everyone with 1.0 and 1.1 were upgraded to 1.5 over the air automatically
Android Applications must be signed	After they are installed	Before they are installed	Never	Within two weeks of installation			Before they are installed
What built-in database is Android shipped with?	SQLite	MySQL	Apache	Oracle			SQLite
What year was development on the Dalvik virtual machine started?	2003	2005	2007	2006			2005
What is an Activity?	A single screen the user sees on the device at one time	message sent among the major building blocks	A component that runs in the background without any interface	Context referring to the application environment			A single screen the user sees on the device at one time
As an Android programmer, what version of Android should you use as your minimum development target?	Versions 1.6 or 2.0	Versions 1.0 or 1.1	Versions 1.2 or 1.3	Versions 2.3 or 3.0			Versions 1.6 or 2.0
How does Google check for malicious software in the Android Market?	Every new app is scanned by a virus scanner	Users report malicious software to Google	Google employees verify each new app	A separate company monitors the Android Market for Google			Users report malicious software to Google
What does the .apk extension stand for?	Application Package	Application Program Kit	Android Proprietary Kit	Android Package			Application Package

The _____ file specifies the layout of your screen?	Layout file	Manifest file	Strings XML	R file			Layout file
What is contained within the manifest xml file?	The permissions the app requires	The list of strings used in the app	The source code	All other choices			The permissions the app requires
The emulated device for android	Runs the same code base as the actual device, all the way down to the machine layer	Is more of a simulator, and acts as a virtual machine for the Android device	Runs the same code base as the actual device, however at a higher level	An imaginary machine built on the hopes and dreams of baby elephants			Runs the same code base as the actual device, all the way down to the machine layer
Status data will be exposed to the rest of the Android system via:	Intents	A content provider	Network receivers	Altering permissions			A content provider
Which one is not a nickname of a version of Andriod?	cupcake	Gingerbread	Honeycomb	Muffin			Muffin
Intents	are messages that are sent among major building blocks	trigger activities to being, services to start or stop, or broadcast	are asynchronous	all of those			all of those
Which of the following is NOT a state in the lifecycle of a service?	Starting	Running	Destroyed	Paused			Paused
What is contained within the Layout xml file?	Orientations and layouts that specify what the display looks like	The permissions required by the app	The strings used in the app	The code which is compiled to run the app			Orientations and layouts that specify what the display looks like

How does Google check for malicious software in the Android Market?	Every new app is scanned by a virus scanner	Users report malicious software to Google	Google employees verify each new app	A separate company monitors the Android Market for Google			Users report malicious software to Google
When developing for the Android OS, Java byte code is compiled into what?	Java source code	Dalvik application code	Dalvik byte code	C source code			Dalvik byte code
What is the driving force behind an Android application and that ultimately gets converted into a Dalvik executable?	Java source code.	R-file.	The emulator	The SDK			Java source code.
What is a funny fact about the start of Android?	It was orginaly going to be called UFO	The first version of Android was released without an actual phone on the market	Androids main purpose was to unlock your car door when you left the keys inside of it	Was going to be a closed source application to make more money for its company			The first version of Android was released without an actual phone on the market
What was Google's main business motivation for supporting Android?	To level the playing field for mobile devices	To directly compete with the iPhone	To corner the mobile device application market for licensing purposes	To allow them to advertise more			To allow them to advertise more
Which Android version had the greatest share of the market as of January 2011?	1.1	1.5	2.3	3.4			1.5
When an activity doesn't exist in memory it is in	Starting state	Running state	Loading state	Inexistent state			Starting state
Which one is not a nickname of a version of Andriod?	cupcake	Gingerbread	Honeycomb	Muffin			Muffin

Intents	are messages that are sent among major building blocks	trigger activities to being, services to start or stop, or broadcast	are asynchronous	all of those			all of those
_____specifies how much of the extra space in the layout should be allocated to the View.	android:layout_gravity	android : layout_x	android:layout_weight	android:layout_width			android:layout_weight
Which are the screen sizes in Android?	small	normal	large	a & b & c			a & b & c
You can shut down an activity by calling its _____ method	onDestory()	finishActivity()	a & b	finish()			finish()
What is off-line synchronization in android?	Synchronization with internet	Background synchronization	Synchronization without internet	None of the above			Synchronization without internet
_____Layout is a view group that displays child views in relative positions.	Table	Relative	Frame	Linear			Relative
Which of the following would you have to include in your project to use the SimpleAdapter class?	import android.content	import android.widget	import android.database	import android.database.sqlite			import android.widget
What is a key difference with the distribution of apps for Android based devices than other mobile device platform applications?	Applications are distributed by Apple App Store only	Applications are distributed by multiple vendors with different policies on applications	Applications are distributed by multiple vendors with the exact same policies on applications	Applications are distributed by the Android Market only			Applications are distributed by multiple vendors with different policies on applications
Android is based on Linux for the following reason	Security	Portability	Networking	All of these			All of these
Android is licensed under which open source licensing license?	Gnu's GPL	OSS	Apache/ MIT	Sourceforge			Apache/MIT

An activity can be thought of as corresponding to what?	A Java project	A Java class	A method call	An object field			A Java class
Intents	are messages that are sent among major building blocks	trigger activities to being, services to start or stop, or broadcast	are asynchronous	all of those			all of those
The android OS comes with many useful system services, which include processes you can easily ask for things such as your..	All of these and more	Location	Sensor Readings	WiFi? Hot Spots			All of these and more
What year was development on the Dalvik virtual machine started?	2003	2005	2007	2006			2005
What is an Activity?	A single screen the user sees on the device at one time	message sent among the major building blocks	A component that runs in the background without any interface	Context referring to the application environment			A single screen the user sees on the device at one time
Android releases since 1.5 have been given nicknames derived how?	Adjective and strange animal	Food	Something that starts w/ ‘A’ -> Something that starts w/ ‘B’…	American states			Food
Which of the following are not a component of an APK file?	Resources	All of these are components of the APK	Native Libraries	Dalvik executable			All of these are components of the APK

Why the so few users are left with versions 1.0 and 1.1?	The first phones were released with version 1.5	1.0 and 1.1 had security holes that forced carriers to recall phones using them	1.0 and 1.1 are just number designations for the version Apple's iPhone is running	Everyone with 1.0 and 1.1 were upgraded to 1.5 over the air automatically			Everyone with 1.0 and 1.1 were upgraded to 1.5 over the air automatically
Android Applications must be signed	After they are installed	Before they are installed	Never	Within two weeks of installation			Before they are installed
What built-in database is Android shipped with?	SQLite	MySQL	Apache	Oracle			SQLite
What year was development on the Dalvik virtual machine started?	2003	2005	2007	2006			2005
What is an Activity?	A single screen the user sees on the device at one time	message sent among the major building blocks	A component that runs in the background without any interface	Context referring to the application environment			A single screen the user sees on the device at one time
As an Android programmer, what version of Android should you use as your minimum development target?	Versions 1.6 or 2.0	Versions 1.0 or 1.1	Versions 1.2 or 1.3	Versions 2.3 or 3.0			Versions 1.6 or 2.0
How does Google check for malicious software in the Android Market?	Every new app is scanned by a virus scanner	Users report malicious software to Google	Google employees verify each new app	A separate company monitors the Android Market for Google			Users report malicious software to Google
What does the .apk extension stand for?	Application Package	Application Program Kit	Android Proprietary Kit	Android Package			Application Package

Register Number_____

[16CAU304A]

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.

BCA DEGREE EXAMINATION, NOVEMBER 2017
(For the candidates admitted from 2016 onwards)
Third Semester
ANDROID PROGRAMMING

Maximum : 60 Marks **Duration: 3 Hours**

PART-A (20 X 1 = 20 Marks)
(Online Examination)

PART-B (5 X 2 = 10 Marks)
(Answer ALL the Questions)

21. What is an Android Operating System?
22. What is object oriented programming?
23. State IDE.
24. Define CheckBox.
25. What is meant by database?

PART-C (5 X 6 = 30 Marks)
(Answer ALL the Questions)

26. a) Explain the history of Android.

[OR]

b) Discuss the architecture of Android.

27. a) Discuss the concepts of OOPs in Java.

[OR]

b) Explain the concept of inheritance and its types.

28. a) Explain the installation procedure of Eclipse with ADT plug-in.

[OR]

b) Explain the steps of installing Virtual machine for Android sandwich.

29. a) Explain Application context with suitable program.

[OR]

b)Discuss the user interface design of Android.

30. a) Discuss SQLite database management system.

[OR]

b) Explain the packages to be implemented in database.

Reg No. ……..……..

[17CAU304A]

Karpagam Academy Of Higher Education
(Established Under Section 3 of UGC Act 1956)
COIMBATORE – 641 021

BCA Degree Examination
(For the candidates admitted from 2017 onwards)
Third Semester
First Internal Exam July 2017
ANDROID PROGRAMMING

Duration: 2 Hrs **Maximum Marks: 50 Marks**
Date & Session: **Class: II BCA**

Part - A (20 X 1 = 20 Marks)
(Answer all the Questions)

SECTION A – (20 X 1 = 20 Marks)
ANSWER ALL THE QUESTIONS

1.Who invented Android programming_____
a. **Andy Rubin** b. Dennis Ritchie c. James gosling d. Martin

2. Android Incorporation was founded in_____
a. 2005 b. **2003** c. 2008 d. 2001

3. SDK refers to_____
a. System development kit b. **Software Development Kit**
c. Software Designing Kernal d. Software Development Kernal

4. ADT stands for_____
a. Activex Designing tool b. Activex Development tool
c**. Android Development tool** d. Android Designing tool

5. Expand IDE_____
a. Internet Designing environment b. **Integrated development environment**
c. Integrated Designing environment d. Intermediate development environment

6. Android byte code is called as_____
a. Source file b.Object file
c. **Dalvik code** d. jdk file

7. ADB represents_____
a**. Android debug bridge** b. Android designing bridge
c. Android development bridge d. Android dalvik bridge

8. Android NDK is a _____
a. Nation Development kit b. **Native Development kit**
c. Nation Development kernel d. Native Designing kit

9. HAL provides_____
a. **Hardware capabilities** b. Software capabilities
c. Hardware capacities d. System capabilities

10. OpenGL API is used to support_____
a. **2D and 3D graphics** b. CUI
c. JIT compiler d. Dalvik code

11. Expand API_____
a. **Android Programming interface** b. Application Programming interface
c. Android Programming interactive d. Active Programming interface

12. A notification manager is used to display_____
a. Interface b. **Alerts**
c. Webpage d. Applications

13. The version of Lollipop Android OS is _____
a. 4.1 b. 4.4
c. **5.0** d. 5.1

14. Inheritance is used for_____
a. creating objects b. creating class
c. **code reusability** d. interface

15. The base class is also called as_____
a. **parent class** b.child class
c.user class d. interface class

16. Java does not support_____
a.single inheritance b. hybrid inheritance
c. **multiple inheritance** d. Multilevel inheritance

17. Method overloading is an example for_____
a. interface b.objects
c.polymorphism d. class

18. _____ keyword is used in inheritance to access another class
a**. extends** b.inherits
c. derives d. generates

19. Interface is used in Java to support_____
a.single inheritance b. hybrid inheritance
c**. multiple inheritance** d. Multilevel inheritance

20. Which of the following is not OOPS concept?
a. object b. class
c. inheritance d. **double**

SECTION – B (3 X 2 =6 Marks)
ANSWER ALL THE QUESTIONS

21. List any four Android operating systems.
1.lolipop 2,Kitkat, 3.Gingerbread 4.Jellybean

22. What is meant by SDK?
A software developer's kit (**SDK**) is a set of programs used by a computer programmer to write application programs. Typically, an **SDK** includes a visual screen builder, an editor, a compiler, a linker, and sometimes other facilities.

23. Define inheritance.
Inheritance is an Object oriented programming concept which inherits the properties of one class to another class. It supports code reusability.

SECTION – C (3 X 8 =24 Marks)
ANSWER ALL THE QUESTIONS

24. (a) Discuss the history of Android.

The history and versions of android are interesting to know. The code names of android ranges from A to J currently, such as **Aestro**, **Blender**, **Cupcake**, **Donut**, **Eclair**, **Froyo**,**Gingerbread**, **Honeycomb**, **Ice Cream Sandwitch**, **Jelly Bean**, **KitKat** and **Lollipop**. Let's understand the android history in a sequence.

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, Codename and API

[OR]

(b) Explain the Android operating System.

Android is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touchscreen devices, Google has further developed Android TV for televisions, Android Auto for cars, and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on notebooks,game consoles, digital cameras, and other electronics.

Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, along with the founding of the Open Handset Alliance – a consortium of hardware, software, and telecommunication companies devoted to advancing open standardsfor mobile devices.

Beginning with the first commercial Android device in September 2008, the operating system has gone through multiple major releases, with the current version being 7.0 "Nougat", released in August 2016. Android applications ("apps") can be downloaded from the Google Play store, which features over 2.7 million apps as of February 2017. Android has been the best-selling OS on tablets since 2013, and runs on the vast majority[a] of smartphones. As of May 2017, Android has two billion monthly active users, and it has the largest installed base of any operating system.

25. (a) Explain Android Development Tools .

Android software development is the process by which new applications are created for the Android operating system. Applications are usually developed inJava programming language using the Android software development kit (SDK), but other development environments are also available.

The Android software development kit (SDK) includes a comprehensive set of development tools.[4] These include a debugger, libraries, a handset emulatorbased on QEMU, documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, and Windows 7or later. As of March 2015, the SDK is not available on Android itself, but software development is possible by using specialized Android applications.

Until around the end of 2014, the officially supported integrated development environment (IDE) was Eclipse using the Android Development Tools (ADT) Plugin, though IntelliJ IDEA IDE (all editions) fully supports Android development out of the box,[8] and NetBeans IDE also supports Android development via a plugin.[9] As of 2015, Android Studio,[10] made by Google and powered by IntelliJ, is the official IDE; however, developers are free to use others, but Google made it clear that ADT was officially deprecated since the end of 2015 to focus on Android Studio as the official Android IDE.[11] Additionally, developers may use any text editor to edit Java and XML files, then use command line tools (Java Development Kit and Apache Ant are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s) remotely).

[OR]

(b) Explain the architecture of Android.

Android is an open source, Linux-based software stack created for a wide array of devices and form factors. The following diagram shows the major components of the Android platform.

The Linux Kernel

The foundation of the Android platform is the Linux kernel. For example, the Android Runtime (ART) relies on the Linux kernel for underlying functionalities such as threading and low-level memory management.
Using a Linux kernel allows Android to take advantage of key security features and allows device manufacturers to develop hardware drivers for a well-known kernel.

Hardware Abstraction Layer (HAL)

The hardware abstraction layer (HAL) provides standard interfaces that expose device hardware capabilities to the higher-level Java API framework. The HAL consists of multiple library modules, each of which implements an interface for a specific type of hardware component, such as the camera or bluetooth module. When a framework API makes a call to access device hardware, the Android system loads the library module for that hardware component.

Android Runtime

For devices running Android version 5.0 (API level 21) or higher, each app runs in its own process and with its own instance of the Android Runtime (ART). ART is written to run multiple virtual

machines on low-memory devices by executing DEX files, a bytecode format designed specially for Android that's optimized for minimal memory footprint. Build toolchains, such as Jack, compile Java sources into DEX bytecode, which can run on the Android platform.

26. (a) Discuss the concept of inheritance in Java.

Inheritance is one of the feature of Object-Oriented Programming (**OOPs**). Inheritance allows a class to use the properties and methods of another class. In other words, the derived class inherits the states and behaviors from the base class. The derived class is also called subclass and the base class is also known as super-class. The derived class can add its own additional variables and methods. These additional variable and methods differentiates the derived class from the base class.

Inheritance is a **compile-time** mechanism. A super-class can have any number of subclasses. But a subclass can have only one superclass. This is because Java does not support multiple inheritance.

Benefits of inheritance

- For Method Overriding (so runtime polymorphism can be achieved).
- For Code Reusability.

Syntax of Java Inheritance

```
class Subclass-name extends Superclass-name
{
 //methods and fields
}
```

The **extends keyword** indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality.

In the terminology of Java, a class which is inherited is called parent or super class and the new class is called child or subclass.

[OR]

(b) Explain the concept of polymorphism in Java.

Polymorphism in java is a concept by which we can perform a single action by different ways. Polymorphism is derived from 2 greek words: poly and morphs. The word "poly" means many and "morphs" means forms. So polymorphism means many forms.

There are two types of polymorphism in java: compile time polymorphism and runtime polymorphism. We can perform polymorphism in java by method overloading and method overriding.

Following concepts demonstrate different types of polymorphism in java.

1) **Method Overloading**
2) **Method Overriding**

Method Overloading:

In Java, it is possible to define two or more methods of same name in a class, provided that there argument list or parameters are different. This concept is known as Method Overloading.

Example:

```
class Overload{
   void demo (int a)    {
      System.out.println ("a: " + a);    }
   void demo (int a, int b)    {
      System.out.println ("a and b: " + a + "," + b);    }
   double demo(double a) {
      System.out.println("double a: " + a);
      return a*a;    }}
class MethodOverloading{
   public static void main (String args [])    {
       Overload Obj = new Overload();
       double result;
```

```
        Obj .demo(10);
        Obj .demo(10, 20);
        result = Obj .demo(5.5);
        System.out.println("O/P : " + result);
    }
}
```

Register Number_____
[17CAU304A]

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Established Under Section 3 of UGC Act 1956)
Coimbatore-641021
BCA Degree Examination
(For the candidates admitted from 2017 onwards)
Third Semester
Second Internal Examination
ANDROID PROGRAMMING

Duration: 2 Hrs **Maximum Marks: 50 Marks**
Date & Session: **Class: II BCA**

SECTION A – (20 X 1 = 20 Marks)
ANSWER ALL THE QUESTIONS

1.Expand IDE_____
a. Internet Designing environment b**. Integrated development environment**
c. Integrated Designing environment d. Intermediate development environment

2. Android byte code is called as_____
a. Source file b.Object file
c. **Dalvik code** d. jdk file

3. ADB represents_____
a. **Android debug bridge** b. Android designing bridge
c. Android development bridge d. Android dalvik bridge

4. Android NDK is a _____
a. Nation Development kit b. **Native Development kit**
c. Nation Development kernel d. Native Designing kit

5._____keyword is used in inheritance to access another class
a. **extends** b.inherits
c. derives d. generates

6. Interface is used in java to support_____
a.single inheritance b. hybrid inheritance
c. **multiple inheritance** d. Multilevel inheritance

7. Which of the following is not an Oops concept?
a. object b. class
c. inheritance d**. thread**

8. The properties of one class can be acquired by another class is called _____
a. Object b**. Inheritance** c. Class d. Data abstraction

9. Eclipse allows the creation of new Workspace _____
a**. directories** b. subprograms c.Modules d.path

10. The API level of Android 2.0 is _____
a. 1 b. 2 c**.** 3 d.**5**

11. _____ is a related set of classes.
a. **package** b. class c.directory d.sub class

12. The first listing is a subdirectory called . _____
a. application b. **src** c. dest source

13. CRUD stands for _____.
a. **Create, Read, Update, and Delete** b. Create, Redo, Update, and Delete
c. Create, Result, Update, and Delete d. Create, Recursion, Update, and Delete

14. Check boxes and radio buttons are _____
a.**mutually exclusive** b. mutually non-exclusive
c. mutually inclusive d. mutually non-inclusive
15. The _____ restricts the user to a list of valid, correctly spelled entries.
a. checkbox b. textbox c. **spinner** d. radio button
16. The _____ is used to display date.
a. **DatePicker** b. DateDay c. DateFunction d. DateTime
17. _____ method is used to match the tags.
a. getNamed() b. **getName()** c. putName() d. putNames()
18. _____ is used to close the database.
a. **void close()** b. void exit() c.void start() d.void remove()
19.A _____ is used to make the source code more readable.
a. constant b.identifier c**.symbolic constant** d. variable
20. ScrollView is used for _____
a. creating tables b. inserting time c. **screen layout** d.closing

SECTION – B (3 X 2 =6 Marks)
ANSWER ALL THE QUESTIONS

21. **What is the function of JRE?**

The Java Runtime Environment (JRE) is a set of software tools for development of Java applications. It combines the Java Virtual Machine (JVM), platform core classes and supporting libraries.JRE is part of the Java Development Kit (JDK), but can be downloaded separately. JRE was originally developed by Sun Microsystems Inc.,

22. Distinguish between overloading and overriding.

Overloading occurs when two or more methods in one class have the same method name but different parameters. **Overriding** means having two methods with the same method name and parameters (i.e., method signature). One of the methods is in the parent class and the other is in the child class

23. Define Emulator of Android.

An Android emulator is an Android Virtual Device (AVD) that represents a specific Android device. You can use an Android emulator as a target platform to run and **test** your Android applications on your PC.

SECTION – C (3 X 8 =24 Marks)
ANSWER ALL THE QUESTIONS

24. **(a) Discuss Java Virtual Machine.**

The Java virtual machine is an abstract (virtual) computer defined by a specification. This specification omits implementation details that are not essential to ensure interoperability: the memory layout of run-time data areas, the garbage-collection algorithm used, and any internal optimization of the Java virtual machine instructions (their translation into machine code). The main reason for this omission is to not unnecessarily constrain implementers. Any Java application can be run only inside some concrete implementation of the abstract specification of the Java virtual machine.[1]

Starting with Java Platform, Standard Edition (J2SE) 5.0, changes to the JVM specification have been developed under the Java Community Process as JSR 924. As of 2006, changes to specification to support changes proposed to the class file format (JSR 202) are being done as a maintenance release of JSR 924. The specification for the JVM was published as the blue book, The preface states:

We intend that this specification should sufficiently document the Java Virtual Machine to make possible compatible clean-room implementations. Oracle provides tests that verify the proper operation of implementations of the Java Virtual Machine.

One of Oracle's JVMs is named HotSpot, the other, inherited from BEA Systems is JRockit. Clean-room Java implementations include Kaffe and IBM J9. Oracle owns the Java trademark and may allow its use to certify implementation suites as fully compatible with Oracle's specification.

[OR]

(b) **Explain about JVM languages.**

A JVM language is any language with functionality that can be expressed in terms of a valid class file which can be hosted by the Java Virtual Machine. A class file contains Java Virtual Machine instructions (Java byte code) and a symbol table, as well as other ancillary information. The class file format is the hardware- and operating system-independent binary format used to represent compiled classes and interfaces.

There are several JVM languages, both old languages ported to JVM and completely new languages. JRuby and Jython are perhaps the most well-known ports of existing languages, i.e. Ruby and Python respectively. Of the new languages that have been created from scratch to compile to Java bytecode, Clojure, Groovy and Scala may be the most popular ones. A notable feature with the JVM languages is that they are compatible with each other, so that, for example, Scala libraries can be used with Java programs and vice versa.

Java 7 JVM implements JSR 292: Supporting Dynamically Typed Languages on the Java Platform, a new feature which supports dynamically typed languages in the JVM. This feature is developed within the Da Vinci Machine project whose mission is to extend the JVM so that it supports languages other than Java

25. **(a) Explain about the installation of Eclipse plugin.**

Android offers a custom plugin for the Eclipse IDE, called Android Development Tools (ADT). This plugin provides a powerful, integrated environment in which to develop Android apps. It extends the capabilities of Eclipse to let you quickly set up new Android projects, build an app UI, debug your app, and export signed (or unsigned) app packages (APKs) for distribution.

1. Start Eclipse, then select **Help** > **Install New Software**.
2. Click **Add**, in the top-right corner.
3. In the Add Repository dialog that appears, enter "ADT Plugin" for the Name and the following URL for the Location:
   https://dl-ssl.google.com/android/eclipse/
4. Click **OK**.
   If you have trouble acquiring the plugin, try using "http" in the Location URL, instead of "https" (https is preferred for security reasons).
5. In the Available Software dialog, select the checkbox next to Developer Tools and click **Next**.
6. In the next window, you'll see a list of the tools to be downloaded. Click **Next**.
7. Read and accept the license agreements, then click **Finish**.
   If you get a security warning saying that the authenticity or validity of the software can't be established, click **OK**.
8. When the installation completes, restart Eclipse.

[OR]

(b) Explain the virtual machine for Android sandwich.

Android is Linux based open source operating system, especially designed for touch based smart phones and tablets and is one of the most widely used operating system by Mobile phone and tablets manufacturers. As Android OS only supports ARM architecture based hardware so you can't run it on x86 architecture i.e. Computer or laptop. In order to run it on x86 architecture, you need to have an Android OS which supports x86 architecture luckily Android x86 project provides it for various testing purposes and you can install Android OS along with your Windows Vista, 7 & 8 operating system.

Steps for installing Android OS Ice Cream Sandwich on Virtual PC

As I am using Microsoft Windows 8 OS thus I have mentioned the steps for installing Android OS on Virtual PC along with Windows 8(learn how to install Windows 8 on virtual PC) but these steps are very much applicable to Windows Vista and 7.

- First download and install Oracle VM VirtualBox from this link (http://www.oracle.com/technetwork/server-storage/virtualbox/downloads/index.html). If you already have it installed then upgrade it to the latest version

- Now visit android-x86.org site and head over to download page. If your system name is listed then download the Android OS ISO image from the respective link else download the ISO image especially created for x86 architecture based hardware which will work on every system (from here https://docs.google.com/open?id=0B4GbJReHMmu_amMzQzJoNGw3WFU). Also if you experience internet connectivity and audio problem with ISO image downloaded for respective system then download the generic ISO image from link given above

- Now open Oracle VM VirtualBox and press CTRL + N for creating a new Virtual Machine and also click the Hide Description button to make visible the hidden Hard drive options

26. **(a) Discuss the installation of virtual machine for Jellybean.**

1) Make sure you have the latest VirtualBox on your PC.
2) Download Android 4.3 ISO from Google site here.
3) Create a new virtual machine, select OS type as **Linux** as below.
Set the RAM size to more than 512MB. I configured 1GB. Create a new hard disk.
4) Go to the settings of virtual machine and edit the storage settings. We need to browse and mount the ISO file which was downloaded from Google site, to IDE controller of CD/DVD drive.
The CD/DVD drive should appear as shown above after loading the ISO.
5) Power on VM which will boot from attached ISO. Select the installation option as below.
6) Create a new partition.
7) With the default options, press **New**.
Make it as **'Primary**' in next screen and press Enter to allocate full size for the partition.
8.) The partition should be bootable, select **'Bootable'** in next screen.
Select '**Write'** to save the settings we did earlier on the partition.
To confirm type **'yes'** and press enter.
Quit from the next screen.
9) Once you have come out of partition creation tool, you can chose the newly created partition to start the installation on VirtualBox.
10) Select **ext3** format and enter.
Press **'Yes'** to format the partition. Also select **'Yes'** to install **boot loader GRUB**. Again **'Yes'** to install **/System directory as read-write** in next screen.
Installation process will start.

[OR]

(b) How to create a simple Hello World Android project?

Studio can be downloaded from the below link.

http://tools.android.com/download/studio/beta

Pre-requisite:

Ensure appropriate JDK version is installed.

Download appropriate Android SDK based on the version we are developing.

https://www.codeproject.com/KB/android/803646/SDKManager.png

Create new project

First step load Android Studio. Click on the New project…

https://www.codeproject.com/KB/android/803646/NewProject.png

Configure the New Project

Enter the application and company domain and select the project location as shown below and click on Next button.

https://www.codeproject.com/KB/Android/803646/Configure.png

Select form factor

Select the appropriate minimum version of android we are going to target as shown in the list as below

https://www.codeproject.com/KB/android/803646/formfactor.png

Select the Activity

Select the template need as pre requirement. I have selected the blank activity.

https://www.codeproject.com/KB/Android/803646/Activity.png

The class will be created based on the Activity Name entered.

https://www.codeproject.com/KB/Android/803646/ActivityName.png

Click on the finish button. The project gets created and will be shown as below

https://www.codeproject.com/KB/Android/803646/FinishNavigation.png