PART-B ESSAY QUESTIONS WITH SOLUTIONS

1.1 INTRODUCTION TO ANDROID OPERATING SYSTEM

1.1.1 Android OS Design and Features

Q11. Write about android operating system.

Answer : Model Paper-I, Q2(a)

Android Operating System

Android is an operating system developed for mobiles which depends on Linux Kernel. Initially, it was developed by Android Inc. Later Google purchased Android and made it an open and free operating system by releasing its code under the open source Apache license. Anyone can use Android just by downloading its source code and can develop the OS by adding some user-defined code. Because of this development, Android became very attractive to many vendors. An android comprises of apache server provisions (i.e., www.apache.org) which are the codes for HTTP methods like GET, POST, PUT, DELETE, HEAD, OPTIONS and TRACE.

Android provides a unique and different approach to develop an application. This is the main advantage of android. The devices that uses Android Operating System can run any number of applications. The execution of managed codes is performed under the management of common language runtime virtual machine. In simple terms, the android OS make use of Java libraries from Google and then android Java codes are executed in a specific VM implementation developed for mobile devices. Moreover, it provides a software development kit that makes easy for the developers to start with new applications. In today's world, android applications are playing a major role.

Versions of Android

Since its first release, Android has undergone many changes. Different versions of Android with their code names is listed below,

(i) Version 1.1 released on 9th February 2009.
(ii) Version 1.5 released on 30th April 2009, Codename: Cupcake.
(iii) Version 1.6 released on 15th September 2009, Codename: Donut.
(iv) Version 2.0/2.1 released on 26th October 2009, Codenmae: Eclair.
(v) version 2.2 released on 20th May 2010, Codename: Froyo.
(vi) Version 2.3 released on 6th December 2010, Codename: Gingerbread.
(vii) Version 3.0/3.1/3.2 released on 22nd February 2011, Codename: Honeycomb.
(viii) Version 4.0 released on 19th October 2011, Codename: Ice-cream Sandwitch.
(ix) Version 4.1 released on 9th July 2012, codename : Jelly Bean
(x) Verseion 4.4 released on 31st October 2013, codename : Kitkat.
(xi) Version 5.0 released on 12th November 2014, codename : Lollipop.
(xii) Version 6.0 released on 5th October 2015, codename : Marshmallow.
(xiii) Version 7.0 released or 22nd August 2016, codename : Nougat.
(xiv) Version 8.0/8.1 released on 21st August, 2017/5th December 2017, codename: Oreo
(xv) Version 9.0 released on 6th August 2018, codename : Pie.
(xvi) Version 10.0 released on 3rd September 2019, codenmae : Android 10.
(xvii) Version 11.0 released on 8th September, 2020, codename : Android 11.

Android 3.0 provides support for wide screen devices and has undergone many changes. They are,

- It supports multicore processors.
- It provides new user interface for tablets.
- It supports multi-tasking.
- It supports new features for web browser like private browsing, form autofill, bookmark synchronization etc.
- It provides support for 3D desktop with added features.

The applications developed for Android 3.0 will not run on devices running previous versions. An Android application can be used by all versions of devices, only after changing the code according to the specific version.

Q12. Write the uses of android OS.

Answer :

Uses of Android OS

An android OS offers a variety of uses to a user. Some of them are as follows,

1. **General Uses**

(a) It allows a users to share SMS and MMS among themselves.

(b) It facilitates for search through voice i.e., if the user wants to search for something then that information can be obtained by simply speaking it out.

(c) It extends it support for multi-touch.

(d) It consists of the multitasking feature.

(e) It supports a wide range of languages.

2. **Connectivity**

(a) **Bluetooth**

This feature allows two users to share certain documents and media among themselves provided that the users are at a close proximity to each other.

(b) **Tethering**

This feature of an android OS enables it to function as a wired/wireless Wi-Fi hotspot using which the nearby devices can access the internet (or) utilize its services.

3. **Media**

(a) **Streaming Media Support**

This feature enables the users to live stream the videos.

(b) **Media Support**

It supports a variety of audio, video and image formats like JPEG, PNG, MP3, MPEG etc.

(c) **External Storage**

This enables the insertion of external SD card through which the storage capacity will be slightly increased.

4. **Hardware Support**

Android OS extends its support for cameras, touch screens, GPS, 3D graphics, gaming controls etc.

5. **Other Uses**

The other uses of android OS include support for a variety of apps empowering the user. Also, it has the ability to support any size of the smart phone i.e., it supports all sorts of handset layouts.

Q13. List out the features of android.

Answer :

Features of Android

Model Paper-II, Q2(a)

Features of android are as follows,

1. **Storage**

Android uses SQLite for storing the data. SQLite is a lightweight relational database.

2. **Connectivity**

Android supports various connections such as,

- GSM/Edge
- Bluetooth
- CDMA
- UMTS
- EV-DO
- WiMAX
- WiFi
- LTE.

3. **Multi-touch**

   It provides multi-touch screens.

4. **Messaging**

   Android supports SMS for text messaging and MMS for video messaging.

5. **Flash**

   Android provides support for flash by using flash 10.1 for version 2.3.

6. **Browser**

   Android supports Chrome's V8 Javascript engine.

7. **Multi-tasking**

   Android also provides support for multi-tasking applications.

8. **Tethering**

   Android supports Internet sharing using wired/wireless hotspot connections.

9. **Other Supportable Features**

   In addition, Android supports Camera, GPS, Digital compass, Proximity sensor and Accelerometer sensor.

Q14. Explain the architecture of android OS with a neat block diagram.

Answer :

Architecture of Android

The android OS is divided into four layers which together contain five sections. The architecture of android is as g below,

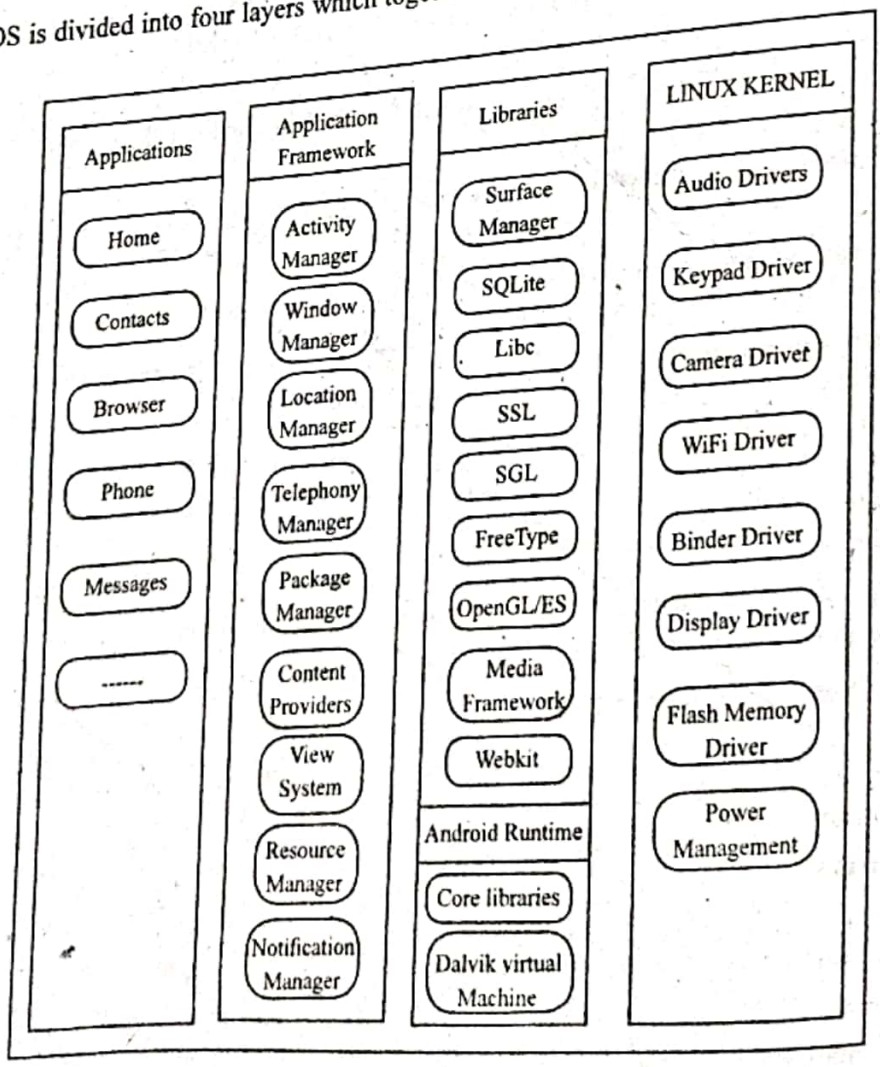


Figure: Architecture of Android

The four layers of android OS are as follows,

1. **Applications Layer**

This is the first layer of Android OS which contains all the applications related to Android device. These applications may be inbuilt applications or the new applications which the user can download and install from Android Market.

2. **Application Framework Layer**

This layer comprises of different capabilities of Android. These capabilities can be used by application developers to design their applications.

3. **Libraries Layer**

This layer contains the complete code of the features of an Android. For example, the library Webkit contains the code related to the web browser feature.

4. **Android Runtime Layer**

This is another section incorporated in the libraries layer. Android Runtime section contains the following,

(a) **Core Libraries**

A collection of core libraries are present in Android Runtime which are used by the developers to develop the Android apps in Java.

(b) **Dalvik Virtual Machine**

Dalvik is a virtual machine used by Android, to ensure that all the Android applications should be used in its own process along with its own virtual machine.

5. **Linux Kernel Layer**

This layer contains all the low-level device drivers required to run the hardware components of Android.

Q15. Briefly explain the features of different layers of Android OS.

Answer : Model Paper-III, Q2(a)

The features of different layers of android OS are as follows,

(i) **Features of Applications Layer**

The applications that are developed by third party users or developers will be installed on this layer. It contains all the components that are required for building an android OS. The following are the building blocks of android,

(a) **Activities**

This component manages the activities of user interface and its interaction with the android device.

(b) **Services**

This component manages the background processing with respect to the application.

(c) **Broadcast Receivers**

This component manages interaction between android OS and other applications.

(d) **Content Providers**

This component manages task related to data and database management.

(ii) **Features of Application Framework**

Application framework is responsible for handling the basic functions of android device like resource management, voice call management and so on. The following are the essential components of application framework,

(a) **Activity Manager**

This component handles the entire activity life cycle of applications.

(b) **Content Providers**

This component manages content sharing between two applications.

(c) **Telephony Manager**

This component handles all voice calls.

(d) **Location Manager**

This component manages the locations that are received via GPS or cell tower.

(e) **Resource Manager**

This component handles various kinds of resources used in android app.

(iii) **Features of Libraries Layer**

This layer is present on the top of Linux kernel. It contains various libraries that help in proper working of android OS. The following are some of the libraries present in libraries layer,

(a) **SQLite**

It helps in accessing the content generated by content providers and contains SQLite database management classes.

(b) **SSL**

It provides internet security.

(c) **Open GL**

It helps in rendering 2D or 3D graphics content to the screen.

(d) **Media Framework**

It provides various media codes that enable the recording and playback of various media formats.

(e) **Webkit**

It is used for presenting internet data HTML data.

(iv) **Features of Android Runtime Layer**

This layer contains most essential component of android known as Dalvik virtual machine (DVM). DVM is designed and optimized for android only. It uses core functions of linux like memory management and multithreading. It allows every android app to run their own process.

(v) **Features of Linux Kernel Layer**

This layer is the bottom layer of android OS. It supports basic functions like process management, memory management and device management. Moreover, it contains list of device drivers which make the task of interfacing the android with peripheral devices easier.

1.1.2 Android Development Framework, SDK Features

Q16. Explain about android development framework.

Answer :

Model Paper-I, Q2(b)

Android application are usually written by using Java as programming language and executed by means of custom VM called Dalvik. The Android application runs in a separate process in Dalvik instance by waiving off from responsibility for memory and process management to Android run time that stops and even kills the process as required for managing the resources. Android SDK includes everything required for beginning the development, testing and debugging the Android applications.

For remaining answer refer Unit-I, Q18.

The Android software stack is a linux kernel and collection of C/C++ libraries that are exposed through application framework which provides services and management of runtime and applications. It contains elements such as linux kernel, libraries, Android run time, application frame work and Application layer.

The application framework provides classes which are used for creating android applications. It even provides generic abstraction for hardware access and then manages user interface as well as application resources.

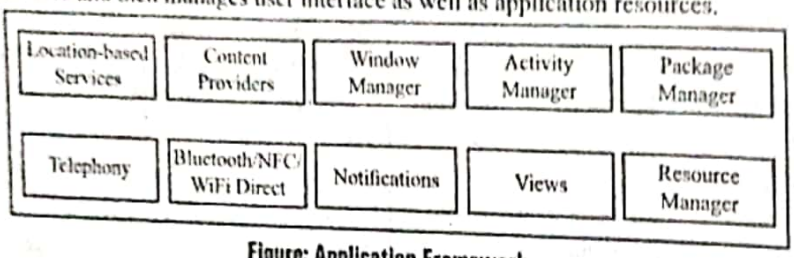


Figure: Application Framework

The application services of the above framework are as follows,

1. **Activity Management and Fragment Manager**
   It controls the lifecycle of activities and fragments along with the management of activity stack.
2. **Notification Manager**
   It provides consistent and non-instrusive mechanism for the purpose of signaling to the users.
3. **Views**
   It is used to develop user interfaces for activist and fragments.
4. **Resource Manager**
   It allows non-code resources like strings and graphics that must be externalized.
5. **Content Providers**
   It allows the applications to share data.
6. **Intents**
   It provides mechanisms to transfer data in between applications and components.

Q17. List out the features of SDK.

Answer :

For answer refer Unit-I, Q15.

1.1.3 Installing and Running Applications on Android Studio, Creating AVDs

Q18. What is the process of installing android SDK? Explain about android platform SDK starter package.

Answer :

Installation of Android SDK

Model Paper-II, Q2(b)

Android applications development needs installation of the following four applications.

(i) Java Development Kit (JDK)

JDK is downloaded from the below link,

http://oracle.com/java/technologies/javase_downloads.html

(ii) Eclipse IDE

Eclipse IDE is downloaded from the below link,

http://www.eclipse.org/downloads/

(iii) **Android Platform SDK Starter Package**

The android platform SDK starter package is downloaded from the below link,

http://developer.android.com/sdk/index.html

(iv) **Android Development Tools (ADT) Plug-in**

ADT Plug-in is downloaded from the below link,

http://developer.android.com/sdk/eclipse-adt.html

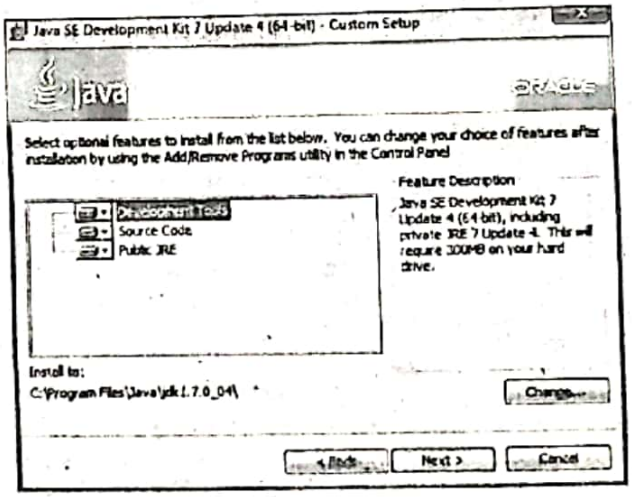
The project templates and eclipse tools are available in plug-in used to create and manage the android project.

The android SDK can be installed in two ways i.e., by the installation of SDK and by the installation of android platforms and other components.

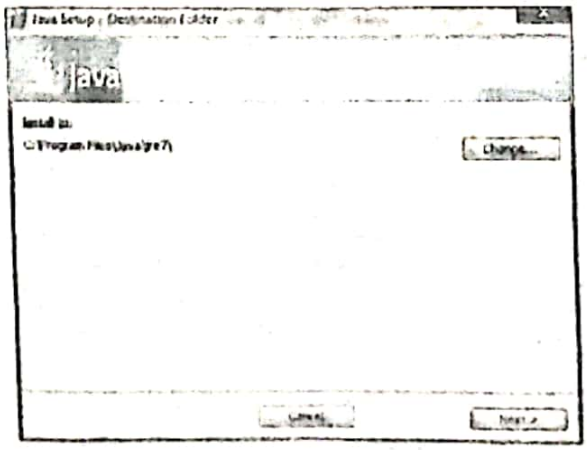
Android Platform SDK Starter Package

Android SDK contains core SDK tools responsible for downloading the remaining SDK components. When android SDK tools are installed then the necessary android platform tools and other components are required to be installed for developing the android applications. Hence, the link for downloading the SDK package is http://developer.android.com/sdk/index.html. To install windows download the .exe file and double click on installer r/8 windows.exe to start the process of installation. The window of android SDK manager gets opened and the first screen appears on the screen is welcome screen. Click on the next button to go to the next screen. The android SDK verifies whether the Java SE Development Kit (JDK) is available or not in the system to perform the operations.

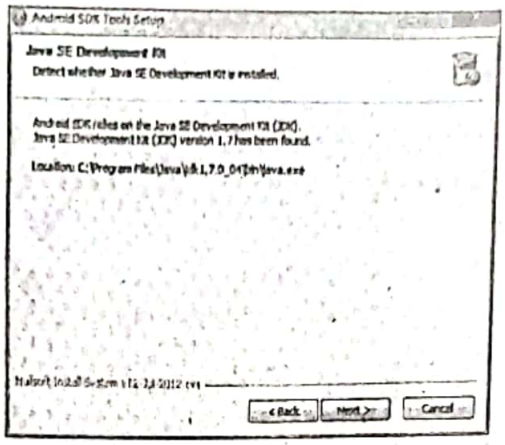
If JDK is not available then it has to be downloaded from the link java.oracle.com. The Java setup wizard welcome screen is the first screen that appears on the window. Upon clicking the next button, custom setup dialog box appears on the screen. It shows optional JDK features required for installation.



In the above figure, development tools, source code and public JRE are the optional features which can be selected to install them and save in the default drive i.e., c:\program files\java\jdk1.7.0-04\. The location can be changed by selecting change button present in the dialog box. Then, the "Next>" button must be selected to continue the installation process. The dialog box of destination folder appears on the screen. It installs the Java Runtime Environment (JRE) which is shown in below figure,



In the above figure, the default location of JRE is shown i.e., c:\program files\Java\jre7. The "change..." button is used for changing the location required. The next button available on the above dialog box can be selected to continue the process. All the java files are installed in the default location. When the installation process is successfully completed, a dialog box is shown to confirm. Select the "Finish" button to exit the wizard. Once installation of Java is completed, the android SDK Tools setup wizard is resumed by itself as shown in below figure,

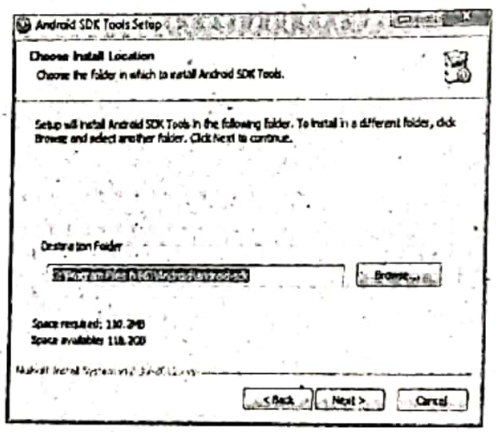


In the above figure, if the "Next>" button is selected it displays a dialog box that contains two options for selecting the users i.e.,

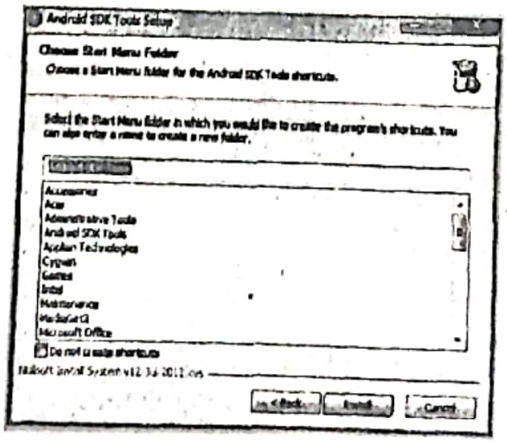
(a) Install for anyone using this computer

(b) Install just for me.

Select the first option and then click on the "Next>" button. The dialog box appeared will ask for the location to save the installation of android SDK tools shown in below figure,

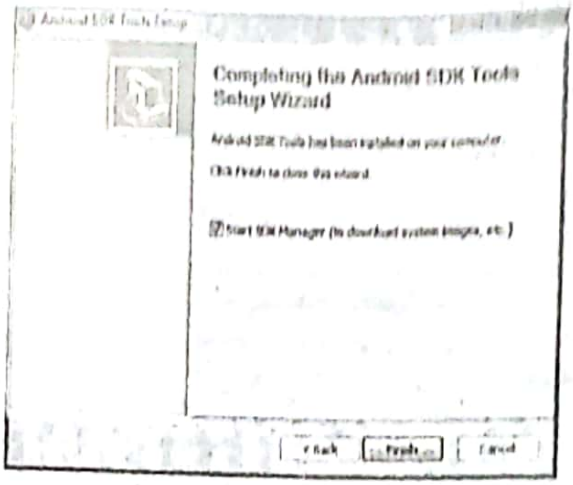


In the above figure, the destination folder stores the default location of installation of android SDK tools is c:\program files(x86)\Android/Android-SDK. The location can be changed by selecting the "Browse..." button. The "Next>" button can be selected to continue the process that leads to the new dialog box that specifies the start menu folder to store the shortcuts of a program as shown in below figure,



In the above figure, the default folder name is set as Android SDK tools. If any user does not want to create a start menu folder then "do not create shortcuts" check box can be selected. For example, a start menu folder is created with its default folder name. To begin the installation process select "Install" button.

Now, select the "Next" button followed by the "Finish" button to complete the installation process.



In the above figure, the start SDK manager checkbox is selected by default before the "Finish" button is clicked. Android SDK manager is one of the tools present in the android SDK tools package is launched with this.

Q19. How android platforms and other components are added to android SDK?

Answer :

Adding Android Platforms and Other Components to Andorid SDK

The android SDK manager shows a set of packages and their installation status. If any of the package is not installed then it can be installed by selecting the respective checkbox. The android SDK manager checks the Android 4.1(API 16) and the google USB driver package by default.

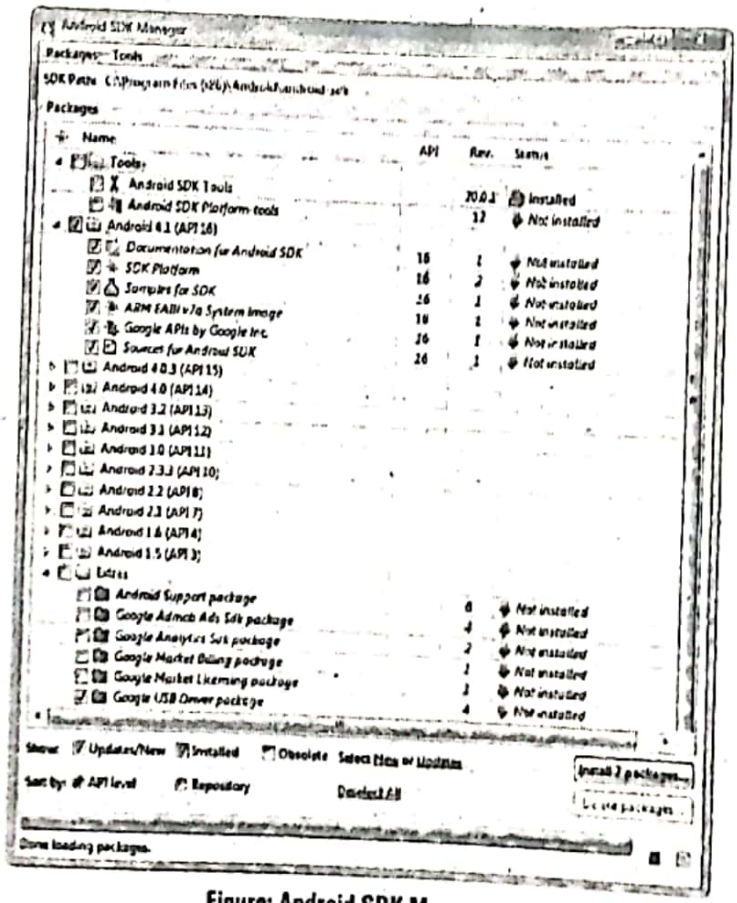


Figure: Android SDK Manager

In the above figure, select or de-select the required packages for installation. In Extras field, select all the packages and click on Install 7 packages button to start the process of installation. The next dialog box shows the set of packages selected for installation along with packages description and license described terms. Now, select the 'Accept all' option followed by "Install" button to start the installation process.

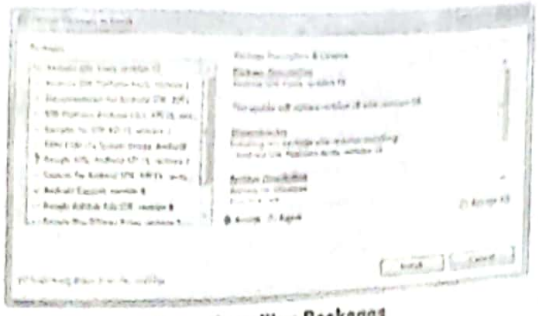


Figure: Installing Packages

The status of downloading and installation is shown in the android SDK manager log window along with the android SDK platform tools that are downloaded and installed.

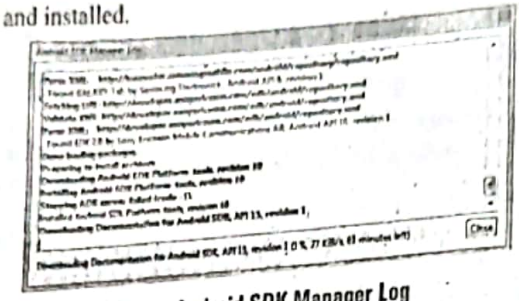


Figure: Android SDK Manager Log

In the above figure, the "Close" button is selected to go to the next dialog box. This will show ADB Restart window that shows information related to the updates and restating the ADB. Now, select the Yes button to restart the ADB.

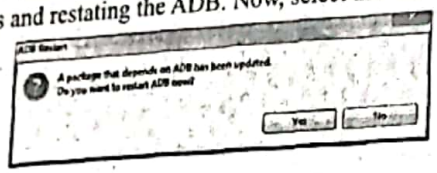


Figure: ADB Restart Window

Android Debug Bridge (ADB) is one of the command-line tool mainly used for communicating with the android emulator and android devices.

The dialog box Android SDK Manager ensures the installation of Android SDK platform tools android 4.1 (API 16) and all its components.

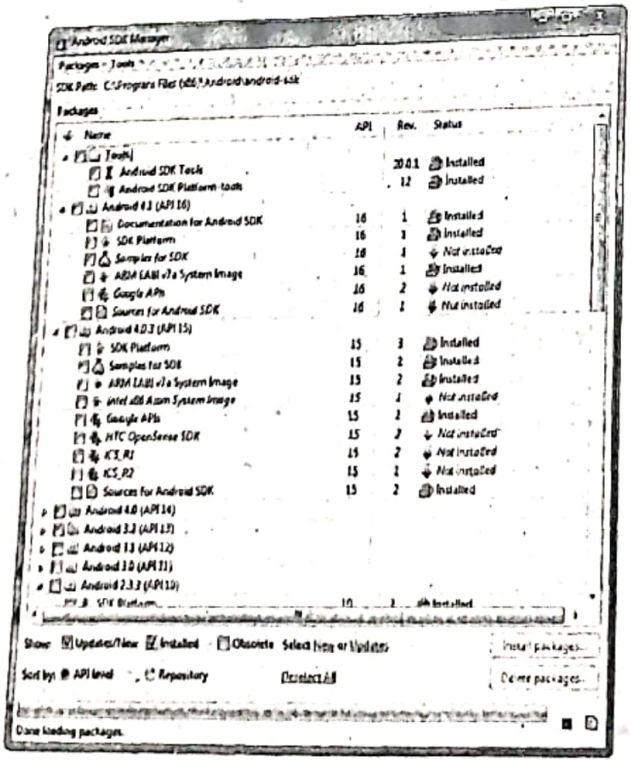


Figure: Android SDK Manager

Q10. Discuss about the installation of eclipse IDE and ADT plug-in in an android application.

Answer :

An android application is the collection of various small components including the Java files, XML resource and layout files, manifest files etc. They need more time to be created manually. The following two applications are used for creating the components.

(i) **Eclipse IDE**

Eclipse IDE makes the creation of Java applications easier. A complete platform that develops the java application along with the support of compilation, debugging and testing are provided by IDE.

(ii) **Android Development Tools (ADT) Plug-in**

It is a plug-in that is added to the Eclipse IDE. It creates the required android files automatically.

Installation of Eclipse

Eclipse IDE is considered as a multi-language software development platform that is used to develop the java applications. Plug-ins are added to this Eclipse IDE to expand its features for developing the applications in other languages.

1. Eclipse IDE is downloaded from the link http:. www.eclipse.org/downloads/. Recommended options for java developers are eclipse classic and eclipse IDE.
2. Run the Eclipse.exe file to launch the eclipse.
3. The Eclipse IDE starts by showing its logo followed by workspace launcher dialog box.

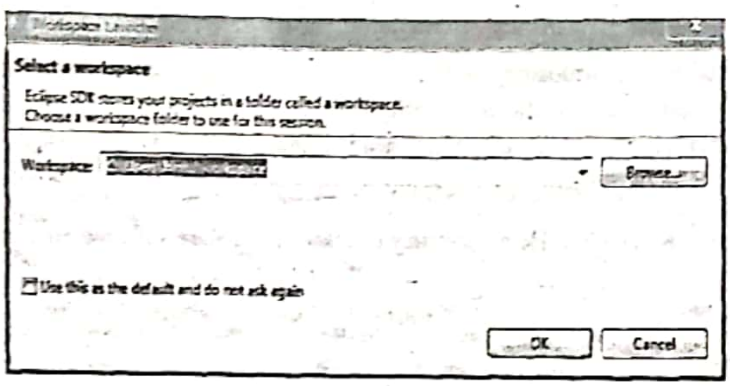


Figure: Workspace Launcher Dialog Box

In the above figure, the location of workspace folder is set in order to store the project files.

4. Now, the home page of an eclipse is displayed,

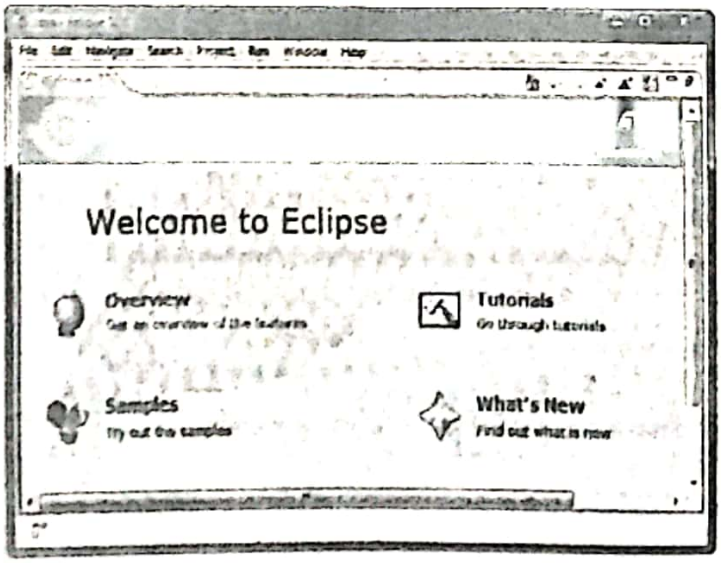


Figure: Eclipse Home Page

5. Click on workbench option. This will show another dialog box with blank windows. They get populated when android applications are developed.

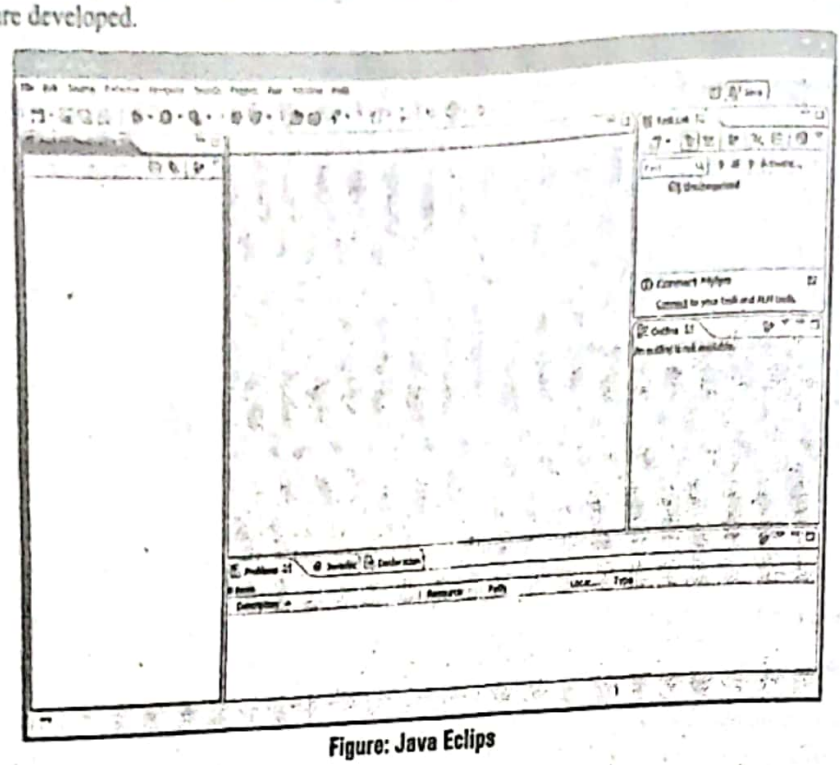


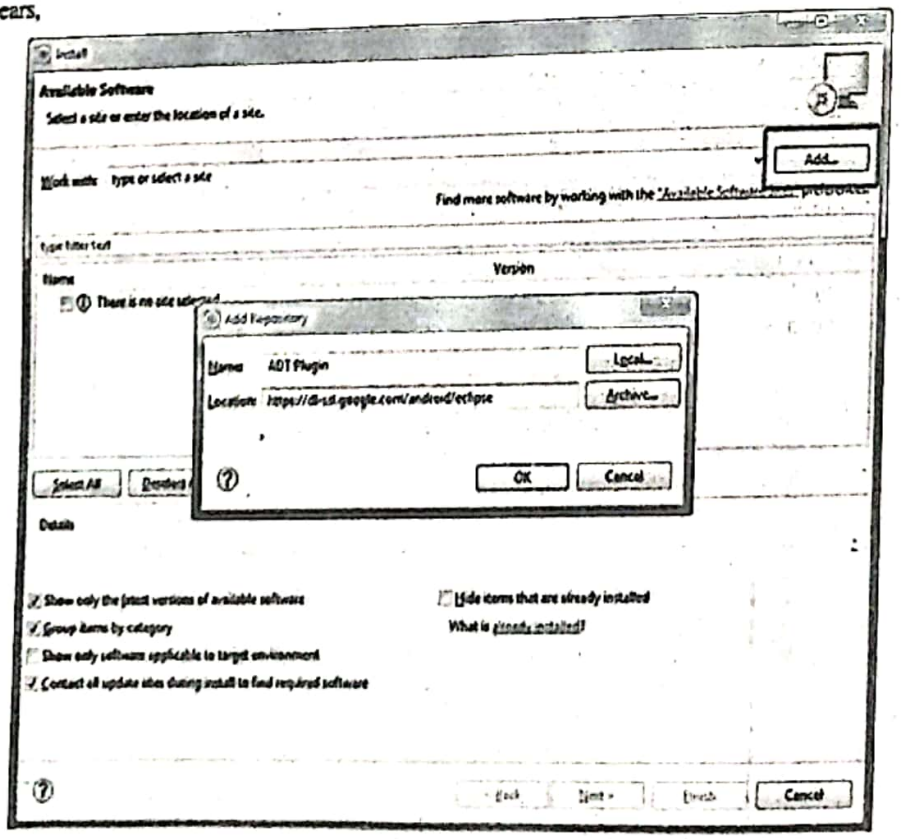
Figure: Java Eclips

Installation of Android Development Tools (ADT)

Eclipse IDE can be installed by downloading the latest eclipse binaries from https://www.eclipse.org/downloads/. Eclipse can be started on windows machine by double clicking on eclipse.exe. Similarly, to start eclipse on Linux execute the command $/usr/local/eclipse/eclipse. When eclipse successfully starts, then it displays the following screen,

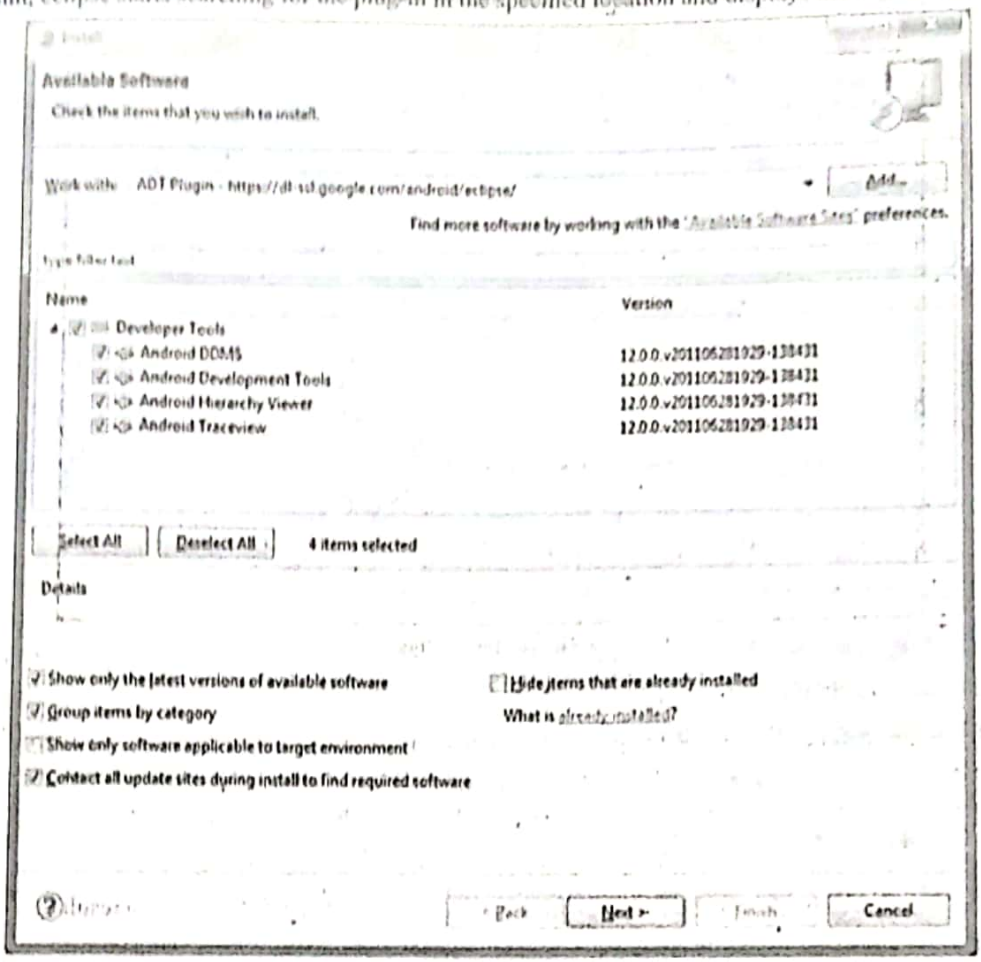
The various steps involved in the installation of ADT plyg-in are as follows,

(i) Initially, launch Eclipse and then select Help > software updates > Install new software. As a result the following dialogue box appears,



(ii) In the displayed dialogue box, type the name as ADT plugin and location as https://dl-ssl.google.com/android/eclipse. Then, click on Ok button.

(iii) As a result, eclipse starts searching for the plug-in in the specified location and displays the list of all available plug-ins.



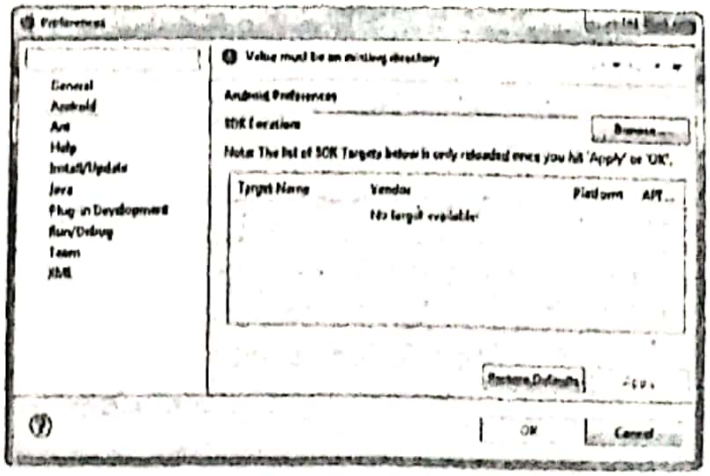
(iv) Click on "SelectAll" button to select the displayed plug-ins and click on "Next>" button and perform the further steps to install android development tools and click on "Finish" button.

Q21. How ADT plug-in is made functional?

Answer :

Making the ADT Plug-in Functional

The ADT plug-in is made functional in the Eclipse IDE by making the plug-in to point the Android SDK. Initially, Eclipse IDE must be launched. Select the window followed by preferences option. A dialog box of preferences is appeared from which Android node must be selected. Now, set the SDK location field in the dialog box to determine the path of Android SDK installation on disk.



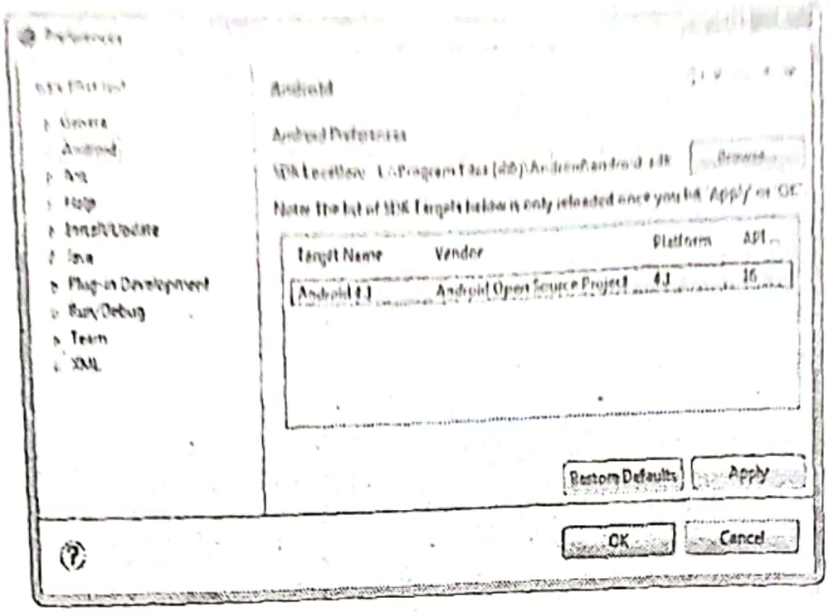


Figure: Preferences Dialog Box

When the path of android SDK is specified the set of SDK targets are displayed. The Android application can be developed and tested against the shown targets.

For this, user must select the Apply button followed by OK button for reloading the SDK targets. Finally, close the preferences dialog box. With this ADT plug-in will be attached to the eclipse.

Q22. Explain the process of creating an android application:

Model Paper-III, Q2(b)

Answer :

Creating an Android Application

The various steps involved in creating an android application are as follows,

1. Open Eclipse and click on File followed by New and Android Project Creator icon.
2. In dialog box shown, select Android Application Project and click on Next.
3. A dialog box displays different fields and the values have to be entered in it is shown below,

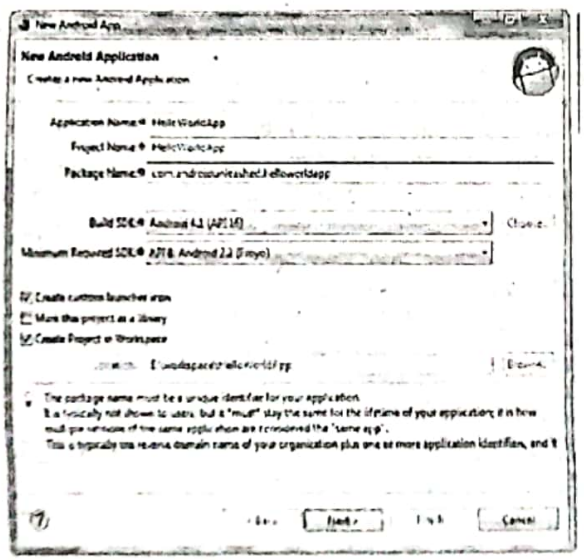
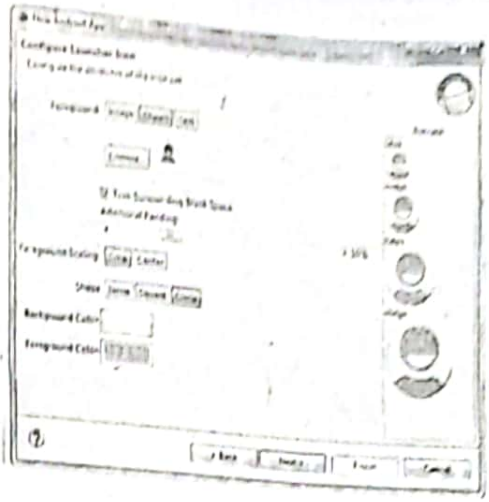


Figure: New Android App

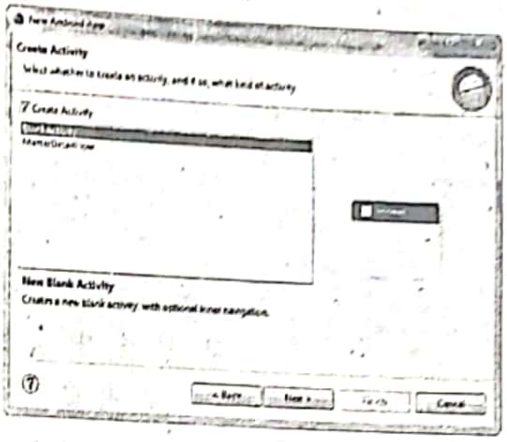
Enter the name of application and project. Select the Android version from Build-SDK drop down list. Select the minimum required SDK as API 8: Android 2.2 (Froyo). Check the create project in windows check box and click on next button.

4. The next dialog box shows configure launcher icon,



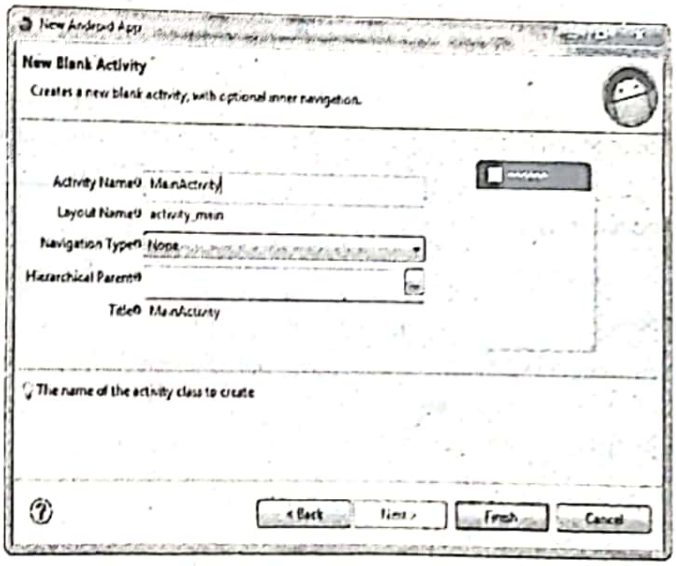
Select one of the Image, Clipart and Text options to define application icon. Select the Clipart or Image through choose button. Now select the foreground scaling, shape Background color and foreground color options. Finally, click on Next button.

5. A new dialog box asks the user about creating an activity.

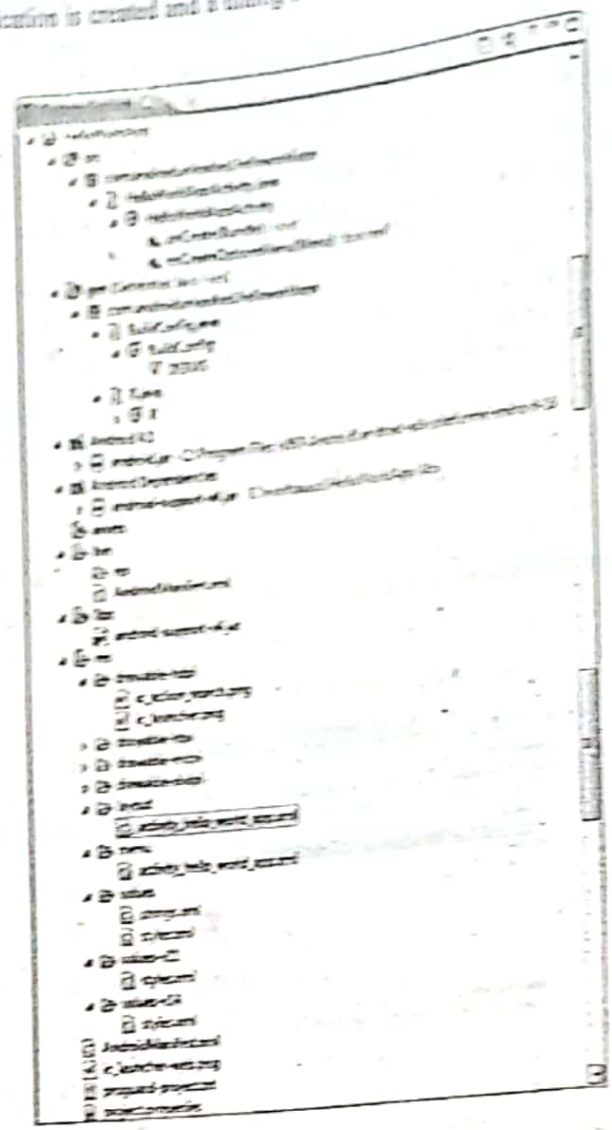


6. Select the type of activity and click on Next.

7. The next dialog box asks for Activity name, layout name, navigation type and title. Set the values for them and click on finish button.



8. Now, with this the Application is created and a dialog box of package Explorer shows the list of files created by ADT plug-in.



There are two important files used in every android application i.e., XML file, Java file.

(i) **XML File**

An XML file present in the folder res>layout>activity-hello-world-app.xml is used for defining the user interface of the application. It includes some controls like textview, button, editText and checkbox shown on the display in an order. The file control defines the way of interaction by a user with an application.

(ii) **Java File**

A Java file present in the src folder contains action code of the controls given in the layout file. Java code is used in handling events occurred through controls in the layout file. The data entered by the user can be fetched and processed in this Java file.

Q23. How to run android application?

Model Paper-I, Q3a

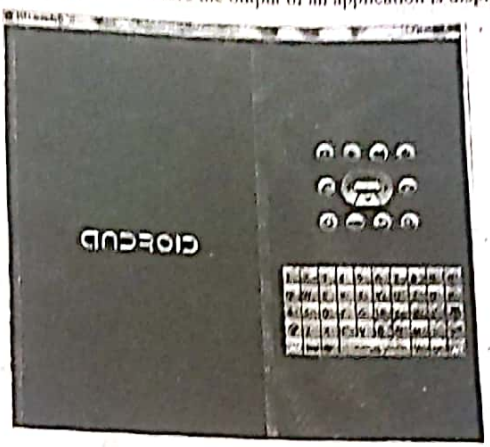
Answer :

Running an Android Application

An Android Application can be run after the configuration file is launched. The following are the different ways for running an android application,

1. Select run icon from Eclipse IDE.
2. Click on Run menu and select Run option.
3. Press Ctrl + F11 keys.

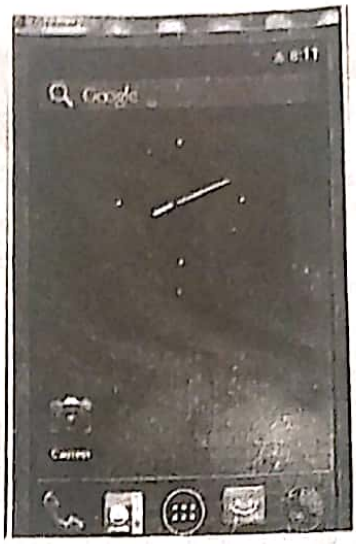
The emulator displays many Booting screens before the output of an application is displayed as shown below,



The first screen has port number of the computer with its emulator 5554 and its AVD name in the title bar of window.
The second screen shows the android logo followed by the default locked home screen.

The home screen is unlocked by dragging the lock towards right side of the screen or by dropping it over the another lock as shown below,

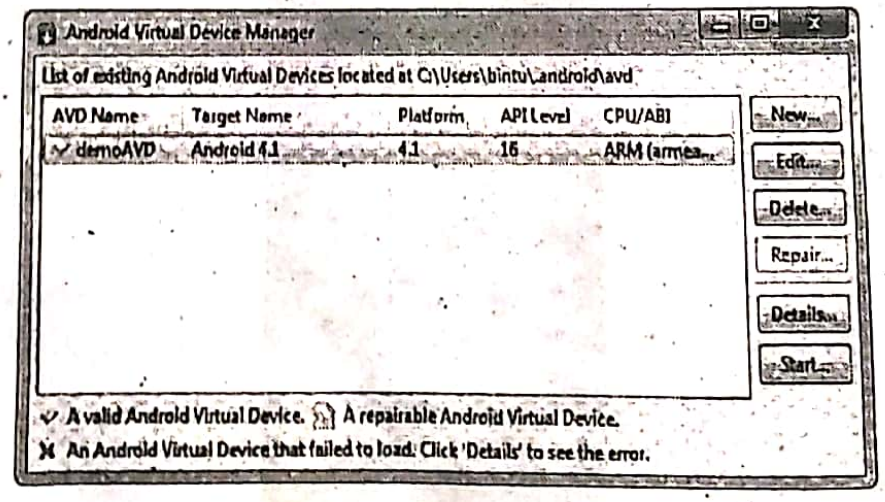
The home screen will be shown when an emulator is unlocked is given below,



The output of an application is displayed below,

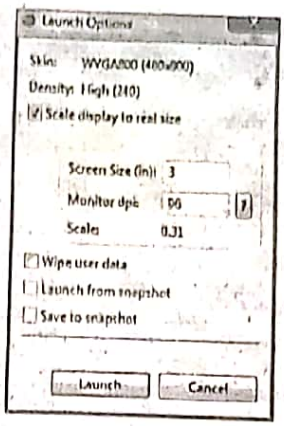
If the screen obtained by an emulator is big then it can be minimized to the size of device by following the below steps,

1. Select the window followed by AVD Manager option.
2. Then, an Android Virtual Device Manager dialog box will be displayed as shown in below program,



3. Choose demo AVD from the dialog box and click on "Start..." button.

4. Select the check box, 'Scale display to real size' and click on launch button to start the emulator as shown below.



Dalvik Virtual Machine (Dalvik VM)

When android applications are run, then the android runtime provides the Dalvik VM environment for deploying and running the android applications. Dan Bornstein has given name to the Dalvik VM which is the Android platform's virtual machine.

It is considered as the specialized virtual machine that is designed mainly for android and is optimized for mobile devices with limited battery, memory and computation capability. When an android application is run and the Android SDK can access all the layout and variable information in the xml files, convert into Java source code and store in R.java file. The R.java class file is compiled into the java byte code files. By using the dx tool it is converted into Dalvik byte code and then saved index format. The Dalvik Executable (.dex) format is optimized for efficient storage and low consumption of memory. The Android Applications cannot be deployed in .dex format. Therefore, the dex code is grouped into an APK file.

Application Package (APK) File

The Java files are converted into DEX code which is grouped with necessary data and resources including the Android Manifest.xml into Application Package. This file has .apk extension. An APK file indicates one application and it is used for distributing an Android Application so that it can be installed on any mobile devices or on emulator. Every APK installed in a device will have a unique IP. It should be signed with certificate the private key of which is held by developer.

Q24. How AVDs are created? Explain.

Answer :

Android Virtual Devices (AVD) **Model Paper-II, Q3(a)**

An Android Virtual Device is used for depicting the configuration of a device. Every android device has distinct configuration. AVDs are created to represent their configuration.

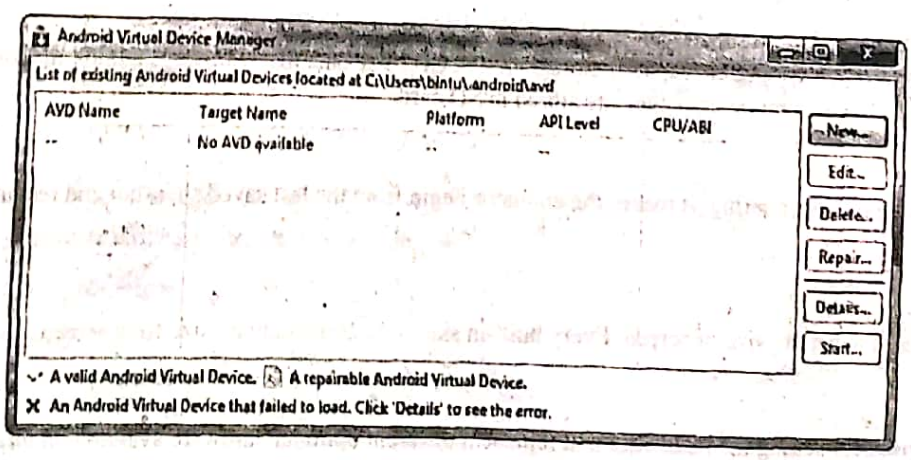
Example

Consider an example of creating an AVD that indicates the configuration of an android device with android version 4.1 of the SDK having 64 MB SD card. AVDs make testing of the android application easier with different configurations.

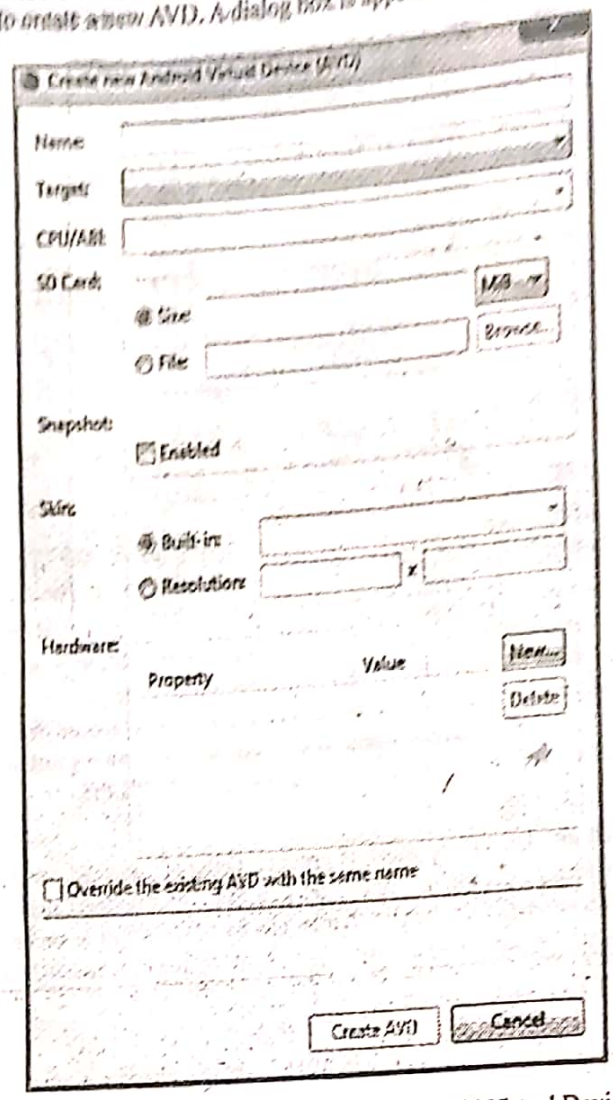
Creation of AVDs in Eclipse

The various steps involved in creating AVDs in Eclipse are as follows,

1. Select window followed by AVD Manager option in Eclipse. A dialog box of Android Virtual Device Manger is appeared as shown below,



2. Click on "New..." button to create a new AVD. A dialog box is appears on screen as shown below,



The following are the different fields available in the 'Create new Android Virtual Device (AVD)" dialog box,

(a) **Name**

This field is used for defining the name of an AVD.

(b) **Target**

This field is used for defining the API level of the target.

(c) **CPU/ABI**

This field is used to define the processor that has to be emulated on device.

(d) **SD Card**

This field is used for extending the capacity storage of a device. The data files with huge audios and videos that cannot be stored in built-in flash memory since they are stored in SD card.

(e) **Snap Short**

This avoids the emulator booting. It makes the emulator begin from the last saved snapshot and resumes to the last saved snapshot.

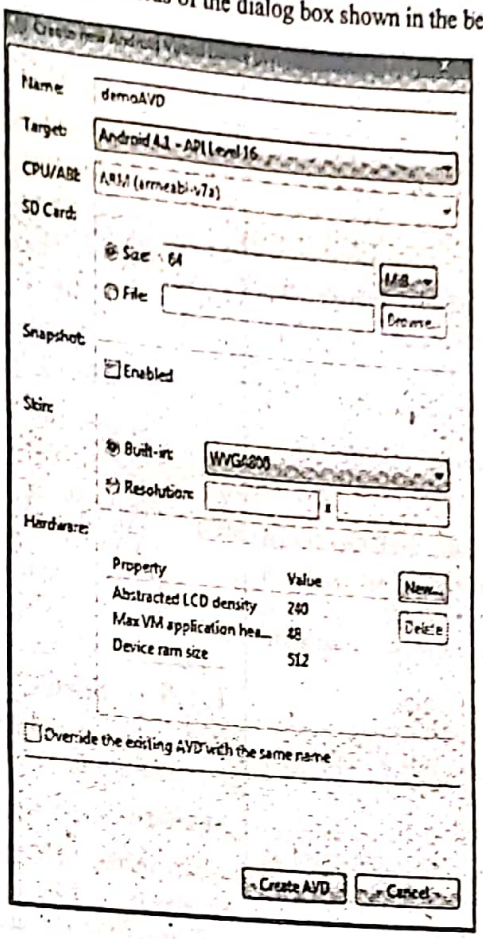
(f) **Skin**

This field is used to set the size of screen. Every built-in skin indicates a defined size for a screen.

(g) **Hardware**

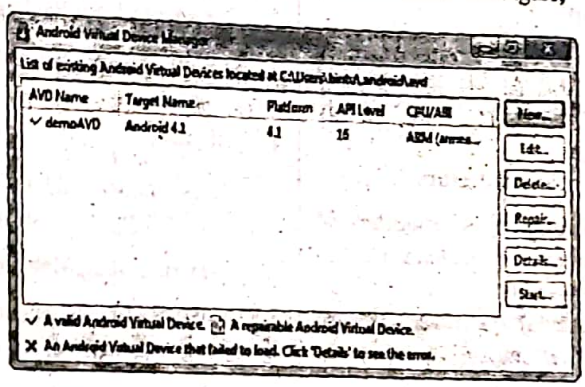
This field is used for setting the properties that represent different optional hardware available in target device.

3. The required values must be entered in the fields of the dialog box shown in the below figure,



Set the Name, Target, SD card and skin options. Select the appropriate properties in hardware section. Click on "Create AVD" button.

4. The dialog box displayed will show the newly created AVD as shown in below figure,



1.1.4 Types of Android Applications, Best Practices in Android Programming

Q25. List and explain various types of android applications.

Answer :

Model Paper-III, Q3(a)

The various types of android applications are as follows,

1. Foreground applications
2. Background applications
3. Intermittent applications
4. Widget applications.

1. **Foreground Applications**

   The applications which will run/work when they are in foreground and suspended when they are not visible are known as foreground applications.

   Example

   Games, Map mashups, etc.

2. **Background Applications**

   The applications which will run in background are called as background applications. They have limited interaction with the user and are not visible.

   Example

   Call screening applications and Auto responders, etc.

3. **Intermittent Applications**

   The applications which will run mostly in the background and have some proper interaction with the user are known as intermittent applications. The set up and working of these applications will be silent. It sends notifications to the users only when required.

   Example

   Media Player

4. **Widget Applications**

   The applications which can only be represented as widgets on the home screen are known as widget applications.

   Example

   Analog clock, Calculator etc.

Q26. What are the best practices in android programming? Explain.

Answer : Model Paper-I, Q3(b)

The best practices for Android design and development are as follows,

1. **Make Use of Recommended Architecture**

   While initiating Android development, if coding standard is not maintained and architecture is not followed, then problems might occur with addition of more features. Some of these problems include the occurrence of bugs, slow improvement in the development of an application. They can be avoided by following coding standard and architecture in the project.

2. **Maintain Code Quality**

   If the quality of the code is maintained good. Then the performance of the application will be good. Quality of every piece of code in application is important in addition to MVP/MVUM/MVC etc.

3. **Make use of Debugging Tools**

   The android debug database is a powerful library for debugging databases and shared preferences in android applications.

4. **Avoid Deep Levels in Layouts**

   The UI becomes slow and layouts become difficult to manage if deep hierarchy of views are used. They can be avoided by using appropriate ViewGroup.

5. **Understand the Context in Android**

   Understanding context in android and usage of it is very much important to avoid memory leaks in android.

6. **Make use of Activity Lifecycle**

   Proper use of activity lifecycle solves many problems in android application development.

7. **Optimize the Build Speed**

   The long build times slow down development process. So, it would be better to optimize the build speed.

8. **Secure Android Application**

   An Android application can be easily decompiled and reverse engineered. It must be secured as much as possible. With this, user trust and device integrity must be preserved.

1.1.5 Android Tools

Q27. Explain different types of tools available in Android.

Answer : Model Paper-II, Q3(b)

Android SDK provides various tools and utilities that are used for creating, testing and debugging the projects. The various android tools are as follows,

1. **Android Emulator**

   The implementation of android Virtual Machine is designed to be run in AVD on development computer.

2. **Android Asset Packaging Tool (AAPT)**

   It is used to develop distributable android package files (.apk).

3. **Logcat**

   It is a utility used for viewing and filtering the output of android logging system.

4. **Dalvik Debug Monitoring Service (DDMS)**

   DDMS is used to monitor as well as control emulators on which applications are debugged.

5. **Android Debug Bridge (ADB)**

   It is a client-server application which provides link to virtual and physical devices. It allows to copy the files, install compiled application packages and even to run shell commands.

6. **Android Virtual Device and SDK Managers**

   It is used to create AVDs and manage SDK packages. AVD hosts emulator that runs a specific build of android allowing to specify the supported SDK version, screen resolution, SD card storage available and hardware capabilities.

Additional Tools

Android SDK provides various additional tools. Some of them are as follows,

1. **SQLite3**

   SQLite3 is a database tool which is used to access SQLite database files created and used by Android.

2. **ProGuard**

   ProGuard is a tool that is used to shrink and obfuscate code by replacing class, variable and method names with unnecessary alternatives.

3. **DX**

   It converts Java ".class" bytecode into Android ".dex" bytecode.

4. **Lint**

   It analyzes the application along with its resources to suggest any improvements and optimizations.

1.2 ANDROID APPLICATION COMPONENTS

1.2.1 Android Manifest File

Q28. Give brief introduction on android manifest file. Write its usage.

Answer :

Manifest file is a configuration file that is created by the ADT. It is a type of XML file that determines the overall structure and application information such as activities, intents and services. In addition, it also contains permissions of the application and metadata.

The components of the application must be declared between Android OS and the application. If declaration is not done in this file then the OS will not consider them.

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

All the components of the application will be enclosed in <application>--</application> tags. The manifest file can contain activity, service, receiver or provider components to specify the components.

The tag <manifest> is the root element containing the below attributes,

- **android:** It recognizes the android namespace that can offer system attributes of an application.
- **package:** The value of it is java package and its name will be unique.
- **versionCode/versionName:** The versionCode determines the current application version and the versionName determines the version number that is to be displayed to users.

Some of the other tags are as follows,

- **<uses-sdk>:** It determines the maximum, minimum and preferred API level that are needed for application operation. The attributes that are used with it are android:minSdkVersion, android:targetsdkversion and android:maxSdkVersion.
- **<application>tag:** This tag is embedded with the <manifest> tag. It acts as the root element of all the application components tags. The attributes such as label and icon are used to depict label and icon resources in android devices.
- **<activity> tag:** This tag is embedded within the <application> tag.
- **<intent-filter :** This tag is used to interact with the application and services other than these default tags these are even commonly used tags such as <service>, <receiver>, <provider> and <user_permission> tags.

1.2.2 Externalizing Resources like Values, Themes, Layouts, Menus etc

Q29. Write short notes on the following,

(i) Creating values

(ii) Creating themes.

Answer :

(i) Creating Values

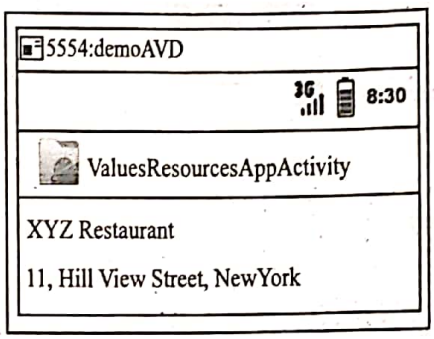
The various types of resources in the values directory are strings, colors, dimensions and string or integer arrays. The values will be stored in XML files within the res/values folder. It is better to maintain separate XML file for every resource type in values directory. The names of resources must be in lowercase letters, numbers, underscore symbols and period. Creation of values resources is shown in the below example,

Example

```
<LinearLayout xmlns : android = "http://schemas.android.com/apk/res/android"
    android : orientation = "vertical"
    android : layout_width = "match_parent"
    android : layout_height = "match_parent">
    <TextView
        android : id = "@+id/name_view"
        android : layout_width = "match_parent"
        android : layout_height = "wrap_content"
        android : text = "@string/name"/>
    <TextView
        android : id = "@+id/address_view"
        android : layout_width = "match_parent"
        android : layout_height = "wrap_content"
    <TextView
        android : id = "@+id/message_view"
        android : layout_width = "match_parent"
        android : layout_height = "wrap_content"
        android : text = "@string/message"/>
```

```
</LinearLayout>
package com.androidunleashed.valuesresourcesapp;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
public class ValuesResources extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savelInstanceState);
        setContentView(R.layout.activity_values_resources_app);
        string Address= "XYZ Restaurant 11, Hill View Street, NewYork"
        TextView addressView = (TextView) findViewById(R.id.address_view);
        addressView.setText(Address);
    }
}
```

Output



(ii) Creating Themes

A complete activity or application can be applied with a style element by using the android: theme attribute. This attribute is added to the android manifest file. It applies the style to all the views or activities within the activity or application.

Example

Activity_main.xml

```
<RelativeLayout xmlns:android="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" >
  <TextView
    android:id="@+id/customText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Welcome to" />
```

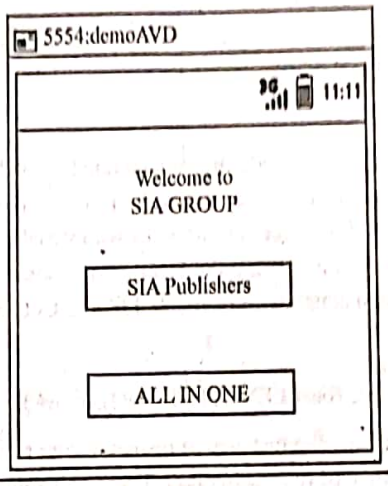
```
    <TextView
        android:id="@+id/bigText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/customText"
        android:text="SIA Group" />
    <Button
        android:id="@+id/simpleTextButton"
        android:layout_below="@id/bigText"
        style="@style/ButtonStyle"
        android:text="SIA Publihsers" />
    <Button
        android:id="@+id/bigTextButton"
        android:layout_below="@id/simpleTextButton"
        style="@style/ButtonStyle"
        android:text="ALL IN ONE" />
</RelativeLayout>
```

MainActivity.java

```
import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
public class MainActivity extends Activity
{
    private Button btn1, btn2;
    private TextView t1, t2;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        t1 = (TextView) findViewById(R.id.customText);
        t2 = (TextView) findViewById(R.id.bigText);
        btn1 = (Button) findViewById(R.id.simpleTextButton);
        btn1.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View arg0)
            {
                // TODO Auto-generated method stub
                t1.setTextAppearance(getApplicationContext(), R.style.SimpleStyle);
            }
        });
```

```
        btn2 = (Button) findViewById(R.id.bigTextButton);
        btn2.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View arg0) {
                // TODO Auto-generated method stub
                t2.setTextAppearance(getApplicationContext(), R.style.SimpleStyle_BigPurple);
            }
        });
    }
}
```

Output



Q30. Discuss in brief about the following,

(i) Creating layouts

(ii) Creating menus.

Answer :

(i) Creating Layouts

Layouts allow the user to decouple the presentation layer from business logic with the help of designing UI layouts in XML instead of developing them in code. They can be used for defining UI for visual components such as Activities, Fragments and widgets. They must be inflated into user interface, after they are defined in XML. This can be done in activity by using setContentView. The fragment views are inflated by using inflate method from inflator object that is passed in fragments onCreateView handler.

User of layouts to create screens in XML is a usually activity in Android. When layout is decoupled from code the optimized layouts can be created for different hardware configurations, like different screen sizes, orientation or presence of keyboards and touchscreens. Every layout definition is stored in a separate file in res/layout folder. The file name will be resource identifier.

```
<?xml version = "1.0" encoding = "utf – 8"?>
<LinearLayout xmlns : android = "http://scheme android.com/apk/res/android"
android : orientation = "vertical"
android : layout_width = "fill_parent"
android : layout_height = "fill_parent">
<TextView
    android : layout_width = "fill_parent"
    android : layout_height = "wrap_content"
    android : text = "@string/hello"
/>
</LinearLayout>
```

(ii) Creating Menus

Menu layouts can be designed in XML by creating menu resources. They can be used to define activity as well as creating menus in the applications and to even provide the same options available while developing menus in code. A menu when defined will be inflated in application through inflate method of MenuInflator Service in onCreateOptionsMenu method. The menus are stored in separate files in res/menu folder. The filename becomes resource identifier.

```
<?xml version = "1.0" encoding = "utf – 8"?>
<menu xmlns : android = "http://scheme android.com/apk/res/android">
</item android : id = "@ + id/menu_refresh"
</item android : title = "@ string/refresh_mi"
</item android : id = "@ + id/menu_settings"
</item android : title = "@ string/setting_mi"/>
</menu>
```

Q31. Discuss how resources are used in code.

Answer :

A class called R is used to access resources in code. It is a generated class depending upon external resources. It is built upon compilation of project. It consists of static subclasses for every resource type for which atleast one resource is created. The subclasses of R does not hide its associated resources as variables with variable names by matching the resource identifiers. The variables hold values in the form of integers indicating the location of resource in resource table rather than an instance of resource. The resource variable is passed when constructor or method like setContentView accepts resource identifier.

```
setContentView(R.layout.main);
Toast.makeText(this, R.string.app_error, Toast.LENGTH_LONG).show( );
```

The helper methods are used for extracting the instance of resource from resource table when it is required. The resource table is represented in application as an instance of resources class. These methods are used to perform lookups on application present resource table. Thus, they cannot be static.

Make use of getResources method on application context for assessing the application's Resource instance,

```
Resource r = getResources( );
```

The resources class contains getters for every available resource types and works by passing in resource ID of which the instance is desired. The below code depicts the usage of helper methods to return selection of resource values,

```
Resource r = getResources( );
CharSequence st = myResources.getText(R.string.stop_message);
Drawable icon =myResources.getDrawable(R.drawable.app_icon);
int OpaqueBlue = myResources.getColor(R.ColorOpque_blue);
float bwidth = myResources.getDimension(R.dimen.standard_boarder);
Animation tranOut;
tranOut = AnimationUtils.loadAnimation.(this, R.anim.spin_shrink_fade);
objectAnimator animator = (ObjectAnimator) AnimatorInflator.loadAnimator(this, R.anim.my_Animator);
string[ ] sA;
sA = = myResources.getIntArray(R.array.s_array);
int[ ] intArray = r.getIntArray(R.array.integer_array);
```

The animated resources are inflated into AnimationResources frame by frame. The value can be returned by using getDrawable and by casting return value shown as follows,

```
AnimationDrawable aAnim;
aAnim = (AnimationDrawable)r.getDrawable (R.drawable.frame_by_frame);
```

1.2.3 Resources for Different Devices and Languages, Runtime Configuration Changes

Q32. How resources are created for different devices and languages? Explain.

Answer : Model Paper-III, Q3(b)

The directory structure can be used to create different resource values for particular languages, locations and hardware configurations. Android selects values from them dynamically at runtime by using the dynamic resource selection mechanism. The alternative resource values can be specified by using parallel directory structure in res folder. They hypen (-) can be used for separating the qualifiers specifying the conditions for which alternatives are provided. The various qualifiers that can be used to customize the resource values are as follows,

1. **Language and Region**

   The language is specified by lower case two letter ISO 639-1 language code, region in lower case and uppercase letters ISO 3166-1 alpha -2 language code.

2. **Available Screen Width**

   The minimum screen width needed for using contained resources specified as w < Dimension value> dp. The largest value will not exceed the present available screen width.

3. **Mobile Country Code and Mobile Network Code (MCC/MNC)**

   MCC and MNC depict the country and network optionally that are associated with SIM that is used in device at present.

   The MCC is specified by MCC and a three digit country code. The MNC can be added by using MNC and two or three digit network code.

4. **Minimum Screen Width**

   It is the minimum screen dimension of device specified as sw<Dimension value>dp. It is mostly used while providing various layouts where value must be less screen width required by layout to render correctly.

5. **Screen Size**

   Screen size might be small, medium, large or extra large. It would be better to use smallest screen size and available screen width and height since every category contains devices with various screen sizes.

6. **Available Screen Height**

   The minimum screen height needed to use the contained resources is specified as <Dimension value> dp. The available screen height changes when device orientation changes to reflect present screen height.

7. **Dock Mode**

   It is a car or desk introduced in API level 8.

8. **Night Mode**

   It is a night or not night mode that is used along with dock mode qualifier. It provides an easy method to alter theme and/or color scheme of application to make it suitable to be used at night is car dock.

9. **Screen Aspect Ratio**

   It specifies long or not long for the resources that are designed for wide screen.

10. **Screen Orientation**

    It is a port, land or square.

11. **Navigation Key Availability**

    It is navexposed or navhidden.

12. **Touchscreen Type**

    It is notouch, stylus or finger that allows to provide layouts or dimensions optimized for style of touchscreen input available on host device.

13. **Keyboard Availability**

    It is a keyexposed, keyshidden or keysoft.

14. **Keyboard Input Type**

    It is a nokeys, qwerty or 12 key.

15. **UI navigation Type**

    It is nonav, dpad, trackball or wheel.

16. **Screen Pixel Density**

    It is pixel density in dots for every inch. It may be specified as ldpi, mdpi, hdpi or xhdpi to specify low, medium, high or extra high pixel density assets respectively.

17. **Platfrom Version**

    It is target API level specified as V < API level> that is used for resources restricted to devices running at specified API level or higher.

Q33. Write about runtime configuration changes.

Answer :

Android can handle runtime changes to languages, location and hardware by terminating and by restarting the active Activity. It enforces resource resolution for activity that is to be re-evaluated and appropriate resource values for new configurations. To make the activity to listen to runtime configuration changes, add android:configurationChanges attribute to manifest node by specifying the configuration changes to be handled. The configuration changes that can be specified are as follows,

1. **mcc and mnc**
   It indicates that SIM is detected and mobile country or network code is changed.
2. **Locale**
   It indicates that user has changed device language settings.
3. **Orientation**
   It indicates that screen is rotated in between portrait and landscape.
4. **Keyboard Hidden**
   It indicates that keyboard, d-pad or other input mechanism is exposed or hidden.
5. **fontScale**
   It indicates the user has changed preferred font size.
6. **uiMode**
   It indicates that global UI mode is changed. It occurs if switched in between car, mode, day or night mode etc.
7. **Keyboard**
   It indicates that keyboard is changed.
8. **Smallest Screen Size**
   It indicates that the physical screen size is changed when device is connected to external output device.
9. **Screen Layout**
   It indicates that screen layout is changed and it occurs if a different screen is activated.
10. **Screen Size**
    It indicates that available screen size is changed.

In some situations, multiple events need to be triggered simultaneously.
Multiple events can be selected to handle by separating values with pipe (|).

```
<activity
  android : name = " .MyActivity"
  android : lable = "@string/appname"
  android : configChanges = "screensize | orientation | keyboardHidden">
  <intent_filter>
      <action android : name "android.intent.action.MAIN"/>
      <category android : name = "android.intent.category.LAUNCHER"/>
  </intent_filter>
</activity>
```

1.3 ANDROID APPLICATION LIFECYCLE – ACTIVITIES, ACTIVITY LIFECYCLE, ACTIVITY STATES, MONITORING STATE CHANGES

Q34. Write in short about the following,

(i) Activity

(ii) Activity Lifecycle.

Answer :

(i) Activity

An activity can be defined as a window that is capable of providing an interface of the application to the user. An application may or may not have an activity but it can contain any number of activities. It is mainly used to provide the user interaction.

Creation of an Activity

An activity can be created by extending the activity base class and by using onCreate( ) method. This is illustrated below,

```
package net.learn2develop.Activity101;
import android.app.Activity;
import android.os.Bundle;
public class ActivityCreation extends Activity
```

```
{
    @Override
    public void onCreate(Bundle Instance)
    {
        super.onCreate(Instance);
        setContentView(R.layout.main);
    }
}
```

Every activity created in the application needs to be entered in AndroidManifest.xml file.

AndroidManifest.xml ,

```
<?xml version = "2.0" encoding = "utf-7"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package = "net.develop.ActivityCreation"
    android:versionCode = "2"
    android:versionName = "2.0">
    <uses-sdk
        android: minSdkVersion = "13"
        android:targetSdkVersion = "20"/>
<application
    android:icon = "@drawable/ic_launcher"
    android:label = "@string/androidapp">
<activity
    android:label = "@string/androidapp"
    android:name= "ActivityCreation">
<intent-filter>
    <action android:name="android.intent.action.MAIN"/>
    <category android:name="android.intent.category.LAUNCHER"/>
    </intent-filter>
    </activity>
    </application>
    </manifest>
```

(ii) Activity Lifecycle

The lifecycle of an activity consists of a sequence of events which it goes through its lifetime. These events are defined by the activity base class. The various events are as follows,

(a) Create

This event deals with the creation of an activity and this can be done by calling the onCreate( ) method.

(b) Start

This event deals with the activity becoming visible to users. It can be done by calling the onStart( ) method.

(c) Resume

This event deals with the activity interacting with its user. It is carried out by calling the onResume( ) method.

(d) Pause

This event deals with stopping an activity for temporary purpose and resuming some other activity for completion. It is carried out by calling the onPause( ) method.

(e) Stop

This event deals with the activity becoming invisible to the user. It is carried out by calling the onStop( ) method.

(f) Destroy

This event deals with the activity being destroyed either manually (or) automatically by the system. It is carried out by calling the onDestroy( ) method.

(g) Restart

This event deals with the activity getting restarted after it was stopped. It is carried out by calling the onRestart( ) method.

The lifecycle of an activity can be pictorially represented as,

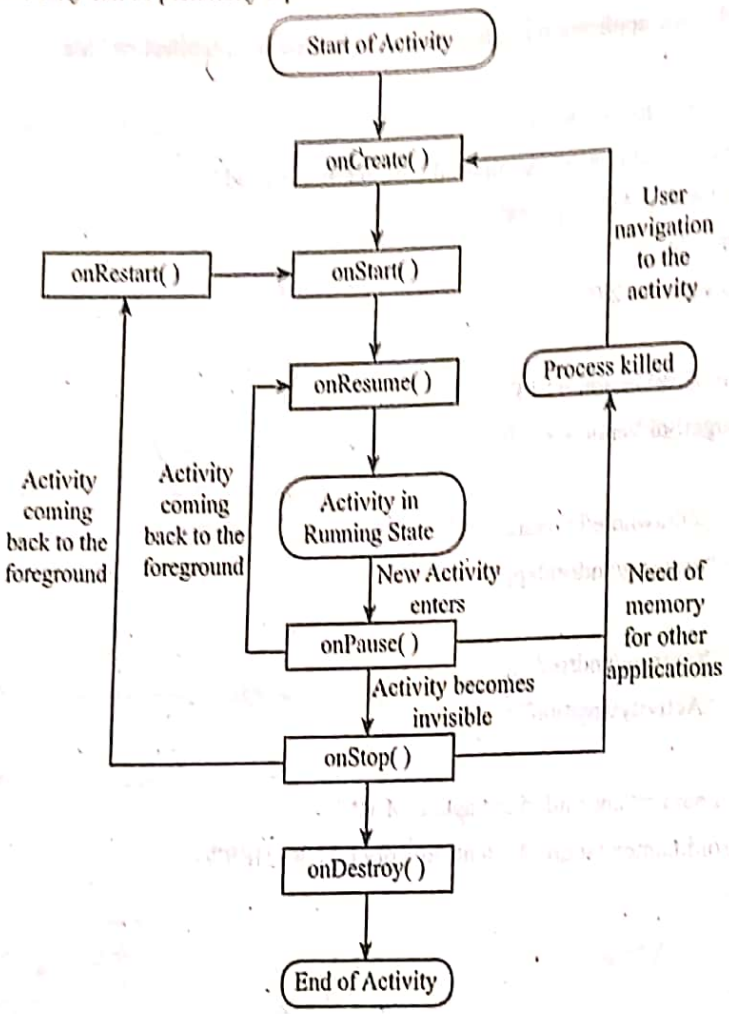


Figure: Activity Lifecycle

The lifecycle of an activity is depicted by the example given below,

1. Create an Android Project named Activity1 using Eclipse.
2. Write the code given below in the Activity1.java file.

```
package net.develop.Activity1;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
public class Activity1 extends Activity
{
    String t = "Activity Lifecycle";
    public void onCreate(Bundle x)
```

```
    {
        super.onCreate(x);
        setContentView(R.layout.main);
        log.d(t, "Executing onCreate( ) event");
    }
    public void onStart( )
    {
        super.onStart( )
        log.d(t, "Executing onStart( ) event");
    }
    public void onRestart( )
    {
        super.onRestart( );
        log.d(t, "Executing onRestart( ) event");
    }
    public void onResume( )
    {
        super.onResume( );
        log.d(t, "Executing onResume( ) event");
    }
    public void onPause( )
    {
        super.onPause( );
        log.d(t, "Executing onPause( ) event");
    }
    public void onStop( )
    {
        super.onStop( );
        log.d(t, "Executing onStop( ) event");
    }
    public void onDestroy( )
    {
        super.onDestroy( );
        log.d(t, "Executing onDestroy( ) event");
    }
}
```

3. Carryout the debugging of the code above by pressing F11 on the Android emulator.
4. Once the activity is loaded, the three events onCreate( ), onStart( ) and onResume( ) will first appear on the Logcat window.
5. Press the back button for the events onPause( ), onStop( ) and onDestroy( ) to appear on the android emulator.
6. When the home button is long pressed, Activities icon gets displayed. Click on it in order to make the events onCreate( ), onStart( ) and onResume( ) to appear.
7. The activity becomes invisible when the phone button on the android emulator is pressed. This leads to the occurrence of onPause( ) and onStop( ) events.
8. In order to make the activity become visible again press the back button in the phone dialer screen. This makes the events onRestart( ), onStart( ) and onResume( ) to appear in the LogCat window.

Q35. List out various states of an activity. Write about monitoring state changes.

Answer :

Activity States

When an activity is created and destroyed, then it move in and out of stack respectively. The transition goes trough four states. They are as follows,

1. **Active**

   If the activity is at top of stack, then it will be visible and focused. Android tries to keep it alive at all the costs by killing activities down the stack as required to assure that it has the required resources. It will be paused when another activity is active.

2. **Paused**

   In certain cases, the activity will be visible but does not have focus. At this point it is paused. It will be reached if a transparent or non-full screen activity is active in front of it. When an activity is paused, it is treated as if it was active. In certain cases, android does not kill a paused activity to recover resources for active activity. It is stopped when it is obscured completely.

3. **Stopped**

   When activity is not visible, then it is stopped. The activity will be in memory and retains its state information. When it is in stopped state, then the data and present UI state must be saved to stop non-critical operations.

4. **Inactive**

   An activity becomes inactive when it is killed or closed. They are removed from activity stack and needs to be restarted before they are displayed or used.

Monitoring State Changes

Activities can react to state changes. This is assured by a series of event handlers which are fired when activity transitions through its full, visible and active lifetimes. The lifetimes of an activity is illustrated in terms of states in the below figure,

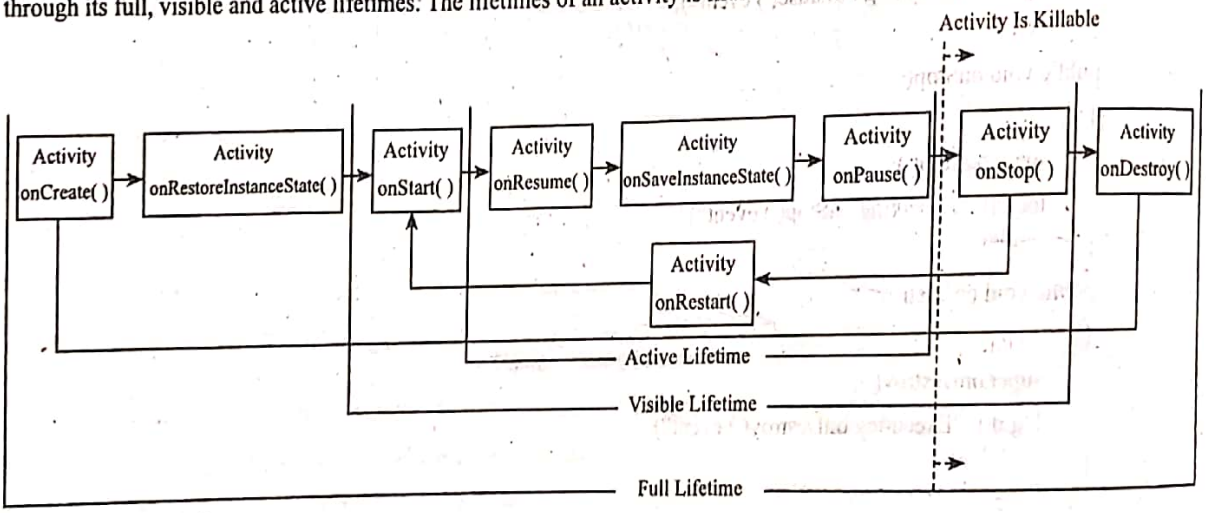


Figure: Lifetimes of an Activity

The below code depicts the stubs for state change method handlers which are available in Activity.

```
package com.paad.Activities;
import android.app.Activity;
import andorid.os.Bundle;
public class StateChange extends Activity
{
```

```
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(saveInstanceState);
}
public void onRestoreInstanceState(Bundle saveInstanceState)
{
    super.onRestoreInstanceState(savedInstanceState);
}
public void onRestart( )
{
    super.onRestart( );
}
public void onStart( )
{
    super.onStart( );
}
public void onResume( )
{
    super.onResume( );
}
public void onSaveInstanceState(Bundle SavedInstanceState)
{
    super.onSaveInstanceState(savedInstanceState);
}
public void onPause( )
{
    super.onPause( );
}
public void onStop( )
{
    super.onStop( );
}
public void onDestory( )
{
    super.onDestroy( );
}
}
```