SUBJECT CODE : 3161612

As per New Syllabus of

GUJARAT TECHNOLOGICAL UNIVERSITY

Semester - VI (IT) (Open Elective - II)

# MOBILE APPLICATION DEVELOPMENT


**Vilas S. Bagad**

M.E. (E&Tc), Microwaves

M.M.S. (Information systems)

Faculty, Institute of Telecommunication Management

Ex-Faculty Sinhgad College of Engineering,

Pune



TECHNICAL PUBLICATIONS

SINCE 1993

An Up-Thrust for Knowledge

# MOBILE APPLICATION DEVELOPMENT

**Subject Code : 3161612**

**Semester - VI (IT) (Open Elective - II)**



Published by :

TECHNICAL PUBLICATIONS
SINCE 1993 An Up-Thrust for Knowledge

Amit Residency, Office No.1, 412, Shaniwar Peth,
Pune - 411030, M.S. INDIA Ph.: +91-020-24495496/97,
Email : sales@technicalpublications.org Website : www.technicalpublications.org

**Printer :**
Yogiraj Printers & Binders
Sr.No. 10/1A,
Ghule Industrial Estate, Nanded Village Road,
Tal. - Haveli, Dist. - Pune - 411041.

ISBN 978-93-90450-08-4

9 789390 450084

Course 18

# PREFACE

The importance of **Mobile Application Development** is well known in computer engineering fields. Overwhelming response to my books on various subjects inspired me to write this book. The book is structured to cover the key aspects of the subject **Mobile Application Development.**

The book uses plain, lucid language to explain fundamentals of this subject. The book provides logical method of explaining various complicated concepts and stepwise methods to explain the important topics. Each chapter is well supported with necessary illustrations, practical examples and solved problems. All chapters in this book are arranged in a proper sequence that permits each topic to build upon earlier studies. All care has been taken to make students comfortable in understanding the basic concepts of this subject.

Representative questions have been added at the end of each section to help the students in picking important points from that section.

The book not only covers the entire scope of the subject but explains the philosophy of the subject. This makes the understanding of this subject more clear and makes it more interesting. The book will be very useful not only to the students but also to the subject teachers. The students have to omit nothing and possibly have to cover nothing more.

I wish to express my profound thanks to all those who helped in making this book a reality. Much needed moral support and encouragement is provided on numerous occasions by my whole family. I wish to thank the **Publisher** and the entire team of **Technical Publications** who have taken immense pain to get this book in time with quality printing.

Any suggestion for the improvement of the book will be acknowledged and well appreciated.

*Author*
*V. S. Bagad*

*Dedicated to God.*

# SYLLABUS

## Mobile Application Development - 3161612

| Credits | Examination Marks | | | | Total Marks |
|---|---|---|---|---|---|
| C | Theory Marks | | Practical Marks | | |
| | ESE(E) | PA(M) | ESE(V) | PA(I) | |
| 3 | 70 | 30 | 30 | 20 | 150 |

1. Overview of Android, Introducing Android, The Android Application Components, The manifest file, Downloading and Installing Android, Exploring the Development Environment, Developing and Executing the first Android Application. **(Chapter - 1)**

2. Using Activities, Fragments and Intents in Android Working with activities, Using Intents, Fragments, Using the Intent Object to Invoke Built –in Application **(Chapter - 2)**

3. Working with the User Interface Using Vies and ViewGroups Working with View Groups, Building data with the AdapterView Class, Designing AutoTextCompleteView, Implementing Screen Orientation, Designing the views programmatically, Handling UI events, Creating Menus **(Chapter - 3)**

4. Storing the Data Persistently, Introducing the Data Storage Options, Using the internal storage, Using the external storage, Using the SQLite Database, Working with content Provider **(Chapter - 4)**

5. Working with Location Services and Maps Working with Google Maps, Working with Geocoding and Reverse Geocoding. **(Chapter - 5)**

6. Working with Graphics and Animation Working with Graphics, Using the Drawable Object, Using the ShapeDrawable object, Hardware Acceleration, Working with Animation **(Chapter - 6)**

7. Audio, Video and Camera Use Media Player, Recording and Playing sound, Creating a sound pool, Using Camera, Recording Video **(Chapter - 7)**

8. Publishing and Distributing Android Application : Signing the Android Application, Versioning the Android Application, Publishing the Android Application **(Chapter - 8)**

# TABLE OF CONTENTS

# 1 Overview of Android

## Syllabus

*Introducing Android, The Android Application Components, The manifest file, Downloading and Installing Android, Exploring the Development Environment, Developing and Executing the first Android Application*

## Contents

## 1.1 Introduction to Android

- Android is an open-source software development platform for creating mobile applications.
- Android is an open source software stack that includes :
  1. Operating system : Linux operating system kernel that provides low level interface with the hardware, memory management and process control.
  2. Middleware : A run time to execute Android applications.
  3. Key mobile applications : Email, SMS, PIM, web browser and etc.
  4. Along with API libraries for writing mobile applications : Including open-source libraries such as SQLite, WebKit and OpenGL ES.
- The components of the underlying OS are written in C or C++, user applications are built for Android in Java. Even the built-in applications are written in Java.
- An important feature of the Android platform is that there's no difference between the built-in applications and applications that is create with the Software Development Kit (SDK).
- Android is only a software. By leveraging its Linux kernel to interface with the hardware, Android runs on many different devices from multiple cell phone manufacturers. Developers write applications in Java.
- The Android software environment and hardware it runs on is shown in Fig. 1.1.1.

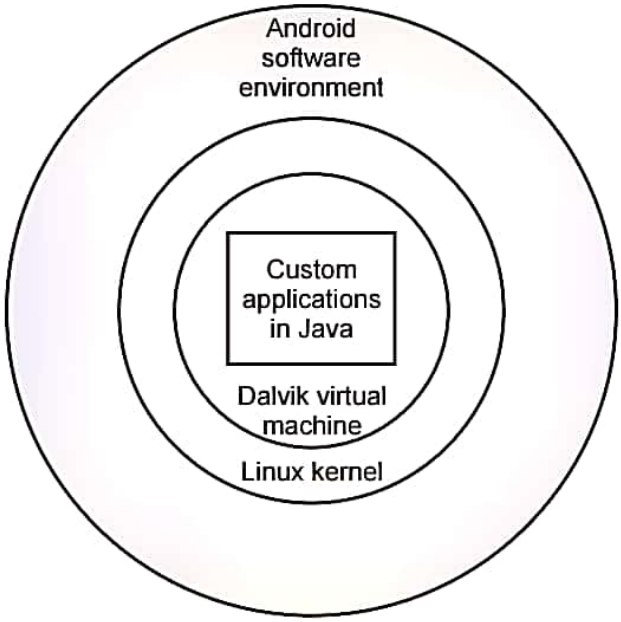



**Fig. 1.1.1 Android OS Design and Features**

## 1.1.1 Advantages and Disadvantages of Android

### Advantages of Android OS

1. **Supports 2D, 3D graphics -** It supports various platforms like 2D and 3D.
2. **Supports multiple languages -** Android supports different languages
3. **Java support -** The Java supporting feature enables developers to enhance more features.
4. **Faster web browser -** Easily loads multimedia so that it makes web browsing faster.
5. **Supports MP4, 3GP, MPEG4, MIDI -** It supports different types of formats. There is no need to convert from one format to another, as it enabled with different formats of audio and video styles.
6. **Video calling -** Faster data connection enables to do video call.
7. **Open source framework -** users can make their own applications and make changes
8. **Uses of tools are very simple**
9. **Social networking integration -** Free to customize the applications and features, using user enabled development.
10. **Better notification system** - It makes users to check important notifications directly from the dashboard.
11. **Low chance of crashing -** The Android OS is very smooth and easy to operate and less chances of crashing down.
12. **Stability** - Stability and security is better than other mobiles OS as it is based on Linux Kernel.

### Disadvantages of Android OS

1. **Slow response** - Compared to 'ios' of Apple, Windows of Microsoft. when we open same app in the ios and Windoes 8. We observe the slow response of the android when we open apps in the different platforms.
2. **Heat -** Compared to other operating systems Android makes use of processes very efficient. This makes processor to get heat. Some hardware companies take care to reduce heat, but it went in vain when we operate it a long time and at low battery.
3. **Advertisement -** When we use an Android app we encounter several adds in between application use, because anyone can make add by inserting some logic in the app program and can interfere in into the phones information.

## 1.2 Android APIs

- Android offers a number of APIs for developing your applications. The following list of core APIs should provide an insight into what's available; all Android devices will offer support for at least these APIs :

1. **android.util** - The core utility package contains low-level classes like specialized containers, string formatters, and XML parsing utilities.
2. **android.os** - The operating system package provides access to basic operating system services like message passing, interprocess communication, clock functions and debugging.
3. **android.graphics** - The graphics API supplies the low-level graphics classes that support canvases, colors and drawing primitives and lets you draw on canvases.
4. **android.text** - The text processing tools for displaying and parsing text.
5. **android.database** - Supplies the low-level classes required for handling cursors when working with databases.
6. **android.content** - The content API is used to manage data access and publishing by providing services for dealing with resources, content providers and packages.
7. **android.view** - Views are the core user interface class. All user interface elements are constructed using a series of Views to provide the user interaction components.
8. **android.widget** - Built on the View package, the widget classes are the "here's one we created earlier" user-interface elements for you to use in your applications. They include lists, buttons and layouts.
9. **com.google.android.maps** - A high-level API that provides access to native map controls that you can use within your application. Includes the MapView control as well as the Overlay and MapController classes used to annotate and control your embedded maps.
10. **android.app** - A high-level package that provides access to the application model. The application package includes the Activity and Service APIs that form the basis for all your Android applications.
11. **android.provider** - To ease developer access to certain standard Content Providers (such as the contacts database), the Provider package offers classes to provide access to standard databases included in all Android distributions.
12. **android.telephony** - The telephony APIs give you the ability to directly interact with the device's phone stack, letting you make, receive and monitor phone calls, phone status and SMS messages.

13. **android.webkit** - The WebKit package features APIs for working with Web-based content, including a WebView control for embedding browsers in your activities and a cookie manager.

## 1.3 Android Architecture

- Following are the different layers in the Android stack :
  1. Linux kernel layer
  2. Native layer
  3. Application framework layer
  4. Applications layer

  Fig. 1.3.1 illustrates android stack.

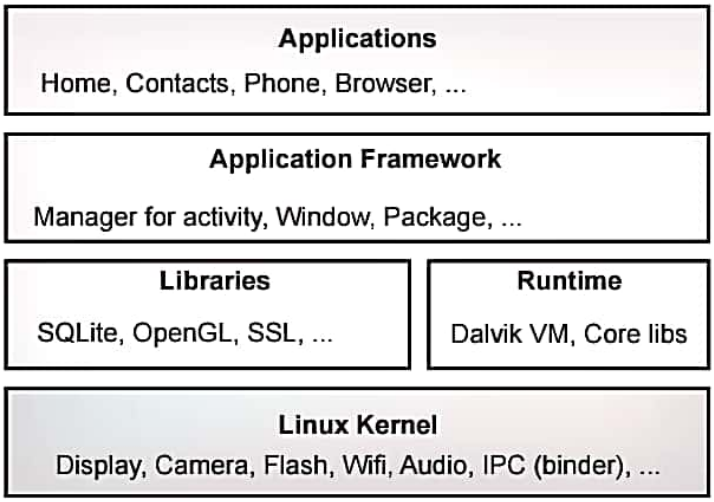



**Fig. 1.3.1 Android stack**

### 1. The Linux kernel layer

- The Linux kernel includes drivers for hardware, networking, file system access and inter-process-communication.
- The Linux kernel is at the bottom of the Android stack. It never really interacts with the users and developers, but is at the heart of the whole system. Its importance stems from the fact that it provides the following functions in the Android system :
  a) Hardware abstraction
  b) Memory management programs
  c) Security settings
  d) Power management software
  e) Other hardware drivers (Drivers are programs that control hardware devices.)

f) Support for shared libraries

g) Network stack

**2. Native code libraries layer**

- The next layer in the Android architecture includes **Android's native libraries**. Libraries carry a set of instructions to guide the device in handling different types of data. For instance, the playback and recording of various audio and video formats is guided by the media framework library.
- The native libraries includes daemons and services (written in C or C++) like browser technology from WebKit, database support via SQLite, advanced graphics support (2D, 3D and animation from scalable games language), audio and video media support from PacketVideo's OpenCORE.

**Android Runtime**

- The Android runtime are written in Java and executing in Dalvik. The core Java packages used for a full-featured Java programming environment and the Dalvik VM, employs services of the Linux-based kernel to provide an environment to host Android applications.
- Dalvik is open-source software. It is the software responsible for running apps on Android devices.

**3. Application framework layer**

- An important block of application framework is application manager. The application managers include windows, contents, activities, telephony, location and notifications.

**4. Application layer**

- The applications are at the topmost layer of the Android stack. An average user of the Android device would mostly interact with this layer (for basic functions, such as making phone calls, accessing the Web browser etc.).
- The layers further down are accessed mostly by developers, programmers and the likes.
- Several standard applications come installed with every device, such as :
  a) SMS client app
  b) Dialer
  c) Web browser
  d) Contact manager

## 1.4 Android Application Framework

- **Application framework** : The application framework provides the classes used to create Android applications. It also provides a generic abstraction for hardware access and manages the user interface and application resources.
- The Android application framework provides everything necessary to implement your average application. The Android application lifecycle involves the following key components :
  1. **Activity Manager** - Controls all aspects of the application lifecycle and activity stack.
  2. **Content Providers** - Allows applications to publish and share data with other applications.
  3. **Resource Manager** - Provides access to non-code embedded resources such as strings, color settings and user interface layouts.
  4. **Notifications Manager** - Allows applications to display alerts and notifications to the user.
  5. **View System** - An extensible set of views used to create application user interfaces.
- The Android application framework includes traditional programming constructs, such as threads and processes and specially designed data structures to encapsulate objects commonly used in mobile applications. Developers can rely on familiar class libraries, such as java.net and java.text. Specialty libraries for tasks such as graphics and database.
- Android applications can interact with the operating system and underlying hardware using a collection of managers. Each manager is responsible for keeping the state of some underlying system service. For example, there is a LocationManager that facilitates interaction with the location-based services available on the handset.
- The ViewManager and WindowManager manage user interface fundamentals. Applications can interact with one another by using or acting as a ContentProvider.
- Built-in applications such as the Contact manager are content providers, allowing third party applications to access contact data and use it in an infinite number of ways. The sky is the limit.

## 1.5 Android Application Components

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

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

### 1.5.1 Manifest File

- Every application must have an AndroidManifest.xml file (with precisely that name) in its root directory. The manifest presents essential information about the application to the Android system, information the system must have before it can run any of the application's code. Among other things, the manifest does the following :
  1. It names the Java package for the application. The package name serves as a unique identifier for the application.
  2. It describes the components of the application - the activities, services, broadcast receivers, and content providers that the application is composed of. It names the classes that implement each of the components and publishes their capabilities (for example, which Intent messages they can handle). These declarations let the Android system know what the components are and under what conditions they can be launched.
  3. It determines which processes will host application components.
  4. It declares which permissions the application must have in order to access protected parts of the API and interact with other applications.
  5. It also declares the permissions that others are required to have in order to interact with the application's components.

6. It lists the Instrumentation classes that provide profiling and other information as the application is running. These declarations are present in the manifest only while the application is being developed and tested; they're removed before the application is published.
7. It declares the minimum level of the Android API that the application requires.
8. It lists the libraries that the application must be linked against.

### 1.5.2 Downloading and Installing Android

- To write Android applications, one must configure programming environment for Java development. The software is available online for download at no cost. Android applications can be developed on Windows, Macintosh, or Linux systems.
- To develop Android applications, you need to have the following software installed on computer :

1. The Java Development Kit (JDK) Version 5 or 6, available for download at http://java.sun.com/javase/downloads/index.jsp.
2. A compatible **Java IDE such as Eclipse** along with its JDT plug-in, available for download at http://www.eclipse.org/downloads/.
3. The **Android SDK**, tools and documentation, available for download at http://developer.android.com/sdk/index.html.
4. The **Android Development Tools** (ADT) plug-in for Eclipse, available for download through the Eclipse software update mechanism. For instructions on how to install this plug-in, see http://developer.android.com/sdk/eclipse-adt.html.

- Although this tool is optional for development, but it is highly recommend to install it.
- The Android SDK comes with five major components : the Android SDK License Agreement, the Android Documentation, Application Framework, Tools and Sample Applications.

## 1.6 Exploring the Development Environment

- Developers have several choices when it comes to integrated development environments (IDEs). Many developers choose the popular and freely available Eclipse IDE to design and develop Android applications.
- Eclipse is the most popular IDE for Android development, and there is an Android plug-in available for facilitating Android development.
- Android applications can be developed on the following operating systems :
- Windows XP (32-bit) or Vista (32-bit or 64-bit)

- Mac OS X 10.5.8 or later (x86 only)
- Linux (tested on Linux Ubuntu 8.04 LTS, Hardy Heron)
- Most developers use the popular Eclipse Integrated Development Environment (IDE) for Android development. The Android development team has integrated the Android development tools directly into the Eclipse IDE. However, developers are not constrained to using Eclipse; they can also use other IDEs.

## 1.7 Android Developing Tools

Various Android development tools are :

### Android SDK

- The Android Software Development Kit (Android SDK) contains the necessary tools to create, compile and package Android applications. Most of these tools are command line based.
- The primary way to develop Android applications is based on the Java programming language.
- Android SDK can be freely downloaded from Android website.

### Android Debug Bridge (ADB)

- The Android SDK contains the Android Debug Bridge (ADB), which is a tool that allows you to connect to a virtual or real Android device, for the purpose of managing the device or debugging your application.

### Android Developer Tools and Android Studio

- Google provides two Integrated Development Environments (IDEs) to develop new applications.
- The Android Developer Tools (ADT) are based on the Eclipse IDE. ADT is a set of components (plug-ins), which extend the Eclipse IDE with Android development capabilities.
- Google also supports an IDE called Android Studio for creating Android applications. This IDE is based on the IntelliJ IDE.
- Both IDEs contain all required functionality to create, compile, debug and deploy Android applications. They also allow the developer to create and start virtual Android devices for testing.

### Dalvik Virtual Machine (DVM)

- Android uses the Dalvik virtual machine with just-in-time compilation to run Dalvik byte code, which is usually translated from Java byte code.

- According to Google's Android documentation, the Dalvik VM is an interpreter-only virtual machine that executes files in the Dalvik Executable (.dex) format, a format that is optimized for efficient storage and memory-mappable execution.
- The virtual machine is register-based and it can run classes compiled by a Java language compiler that have been transformed into its native format.
- Currently Android versions use the Dalvik virtual machine. The latest Android versions introduced a new runtime the Android RunTime.

**Android RunTime (ART)**

- With Android 4.4, Google introduced the Android RunTime (ART) as optional runtime for Android 4.4. It is uses as default runtime for all Android versions after 4.4.
- ART uses Ahead of Time compilation. During the deployment process of an application on an Android device, the application code is translated into machine code. This result in approximately 30 % larger compiles code, but allows faster execution from the beginning of the application.
- This also saves battery life, as the compilation is only done once, during the first start of the application.
- The **dex2oat** tool takes the .dex file created by the Android tool change and compiles that into an Executable and Linkable Format (ELF file). This file contains the dex code, compiled native code and meta-data. Keeping the .dex code allows that existing tools still work.
- The garbage collection in ART has been optimized to reduce times in which the application freezes.

## 1.8 Developing Android Application

- Android applications are primarily written in the Java programming language.
- During development the developer creates the Android specific configuration files and writes the application logic in the Java programming language.
- The ADT or the Android studio tools convert these application files, transparently to the user, into an Android application.
- When developers trigger the deployment in their IDE, the whole Android application is compiled, packaged, deployed and started.

### 1.8.1 Conversion Process from Source Code to Android Application

- The Java source files are converted to Java class files by the Java compiler.
- The Android SDK contains a tool called dx which converts Java class files into a *.dex* (Dalvik Executable) file. All class files of the application are placed in this *.dex* file.
- During this conversion process redundant information in the class files are optimized in the *.dex* file.For example, if the same **String** is found in different class files, the *.dex* file contains only one reference of this **String**.
- These *.dex* files are therefore much smaller in size than the corresponding class files.
- The *.dex* file and the resources of an Android project, e.g., the images and XML files, are packed into an *.apk* (Android package) file. The program *aapt* (Android Asset Packaging Tool) performs this step.
- The resulting *.apk* file contains all necessary data to run the Android application and can be deployed to an Android device via the *adb* tool.

### 1.8.2 Android SDK Features

- Important features of Android SDK are as under.
  1. No licensing, distributions or development fees or release approval processes.
  2. Full multimedia hardware control.
  3. APIs for using sensor hardware including accelerometer and the compass.
  4. APIs for location based services.
  5. Android Inter-Process Communication (IPC).
  6. Shared data storage.
  7. Background applications and processes.
  8. Home screen widgets, live folders.
  9. HTML5 WebKit-based web browser.
  10. GSM, EDGE and 3G networks for telephony and data transfer.
  11. The Android SDK includes development tools which helps compile and debug any app.
  12. Android emulator shows how app will look and behaviour on a real Android device.

## 1.9 Developing Android Application On Eclipse Platform

- **Eclipse** is an Integrated Development Environment, or IDE, which is software that provides all the essential tools needed for editing, running, and debugging your Java programs.
- The Java Development Kit, or JDK, is a set of development tools used in the programming of Java applications.
- The Eclipse IDE requires that a JDK be locally installed. The JDK can be downloaded from the web at http://java.sun.com/javase/downloads/index.jsp.

### Create a new Android Application

1. In Eclipse go to **File->New->Project...**
2. Select an Android Project from the Android Folder and press Next.
3. Fill in the details of your Android application.
   a. **Project Name :** The project name and folder that Eclipse will store the project files.
   b. **Build Target :** The version of the Android SDK that will be used when you build your program. Select a platform that is equal to or lower than the target chosen for the AVD.
   c. **Application Name :** This is the name of the application.
   d. **Package Name :** The namespace that all of the source code will reside under.
   e. **Create Activity :** The name for that class stub that is generated by the plugin.
4. The values that are used in this example are :
   a. **Project Name :** SampleApp
   b. **Build Target :** 2.3.3
   c. **Application Name :** SampleApp
   d. **Package Name :** com.sample.example
   e. **Create Activity :** SampleApp
5. Click on **Finish.**

## 1.10 Short Questions and Answers

**Q.1 Explain Android architecture components.**

**Ans. :** A collection of libraries that help you design robust, testable, and maintainable apps.

- **Room**

Official documentation

Article on how to implement Room Db

Sample implementation

- **Live Data**

Official documentation

Sample implementation

- **ViewModel**

Official documentation

Sample implementation

- **Data Binding**

Official documentation

Sample implementation

- **Lifecycles**

Official documentation

**Q.2 State the tools required for Developing Android Apps.**

**Ans.** Tools for developing Android Apps

1. JDK
2. Eclipse + ADT plugin
3. SDK Tools

**Q.3 What are the main components of Android application ?**

**Ans.**

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

## 1.11 Multiple Choice Questions

**Q.1** *Android is _____ .*

a web server
b web browser
c operating system
d none of these

**Q.2** *What is Android ?*

a Android is a stack of software's for mobility
b Google mobile device name
c Virtual machine
d None of the above

**Q.3** *Who owns Android ?*

a Oracle
b Dalvik
c Open handset alliance
d Google

**Q.4** *Android is specially developed for ____*

a Desktops
b Laptops
c Servere
d Mobile devices

**Q.5** *Android operating system is based on _____.*

a Mac
b Windows
c Linux
d Solaris

**Q.6** *Top most layer of android architecture is _____.*

a applications
b linux kernel
c applications Framework
d system libraries

**Q.7** *What are the layouts available in android ?*

a Linear layout
b Frame layout
c Table layout
d Relative layout
e All of above

**Q.8** *What is an activity in android ?*

a Activity performs the actions on the screen

b Manage the Application content

c Screen UI

d None of the above

**Answer Keys for Multiple Choice Questions**

| | | | | | |
|---|---|---|---|---|---|
| **Q.1** | c | **Q.2** | a | **Q.3** | c |
| **Q.4** | d | **Q.5** | c | **Q.6** | a |
| **Q.7** | e | **Q.8** | a | | |

❑❑❑

# 2 Activities and Fragments

## Syllabus

*Using Activities, Fragments and Intents in Android Working with activities, Using Intents, Fragments, Using the Intent Object to Invoke Built – in Application*

## Contents

## 2.1 Activity Lifecycle

- Application comprise one or more activities. Because of the mobile and UI experience the operation of an activity is governed by a set of states (like a finite state machine) called the lifecycle.
  a) Crashing if the user receives a phone call or switches to another app while using your app.
  b) Consuming valuable system resources when the user is not actively using it.
  c) Losing the user's progress if they leave your app and return to it at a later time.
  d) Crashing or losing the user's progress when the screen rotates between landscape and portrait orientation.
- To navigate transitions between stages of the activity lifecycle, the Activity class provides a core set of six callbacks: onCreate(), onStart(), onResume(), onPause(), onStop(), and onDestroy().
- The system invokes each of these callbacks as an activity enters a new state.

**1. onCreate()**

- This callback must be implemented, which fires when the system creates your activity. Your implementation should initialize the essential components of your activity: For example, your app should create views and bind data to lists here. Most importantly, this is where you must call setContentView() to define the layout for the activity's user interface.
- When onCreate() finishes, the next callback is always onStart().

**2. onStart()**

- As onCreate() exits, the activity enters the Started state, and the activity becomes visible to the user. This callback contains what amounts to the activity's final preparations for coming to the foreground and becoming interactive.

**3. onResume()**

- The system invokes this callback just before the activity starts interacting with the user. At this point, the activity is at the top of the activity stack, and captures all user input. Most of an app's core functionality is implemented in the onResume() method.
- The onPause() callback always follows onResume().

**4. onPause()**

- The system calls onPause() when the activity loses focus and enters a Paused state. This state occurs when, for example, the user taps the Back or Recents button. When the system calls onPause() for your activity, it technically means your activity is still partially visible, but most often is an indication that the user is leaving the activity, and the activity will soon enter the Stopped or Resumed state.
- An activity in the Paused state may continue to update the UI if the user is expecting the UI to update. Examples of such an activity include one showing a navigation map screen or a media player playing. Even if such activities lose focus, the user expects their UI to continue updating.
- You should **not** use onPause() to save application or user data, make network calls, or execute database transactions.
- Once onPause() finishes executing, the next callback is either onStop() or onResume(), depending on what happens after the activity enters the Paused state.

**5. onStop()**

- The system calls onStop() when the activity is no longer visible to the user. This may happen because the activity is being destroyed, a new activity is starting, or an existing activity is entering a Resumed state and is covering the stopped activity. In all of these cases, the stopped activity is no longer visible at all.
- The next callback that the system calls is either onRestart(), if the activity is coming back to interact with the user, or by onDestroy() if this activity is completely terminating.

**6. onDestroy()**

- The system invokes this callback before an activity is destroyed.
- This callback is the final one that the activity receives. onDestroy() is usually implemented to ensure that all of an activity's resources are released when the activity, or the process containing it, is destroyed.

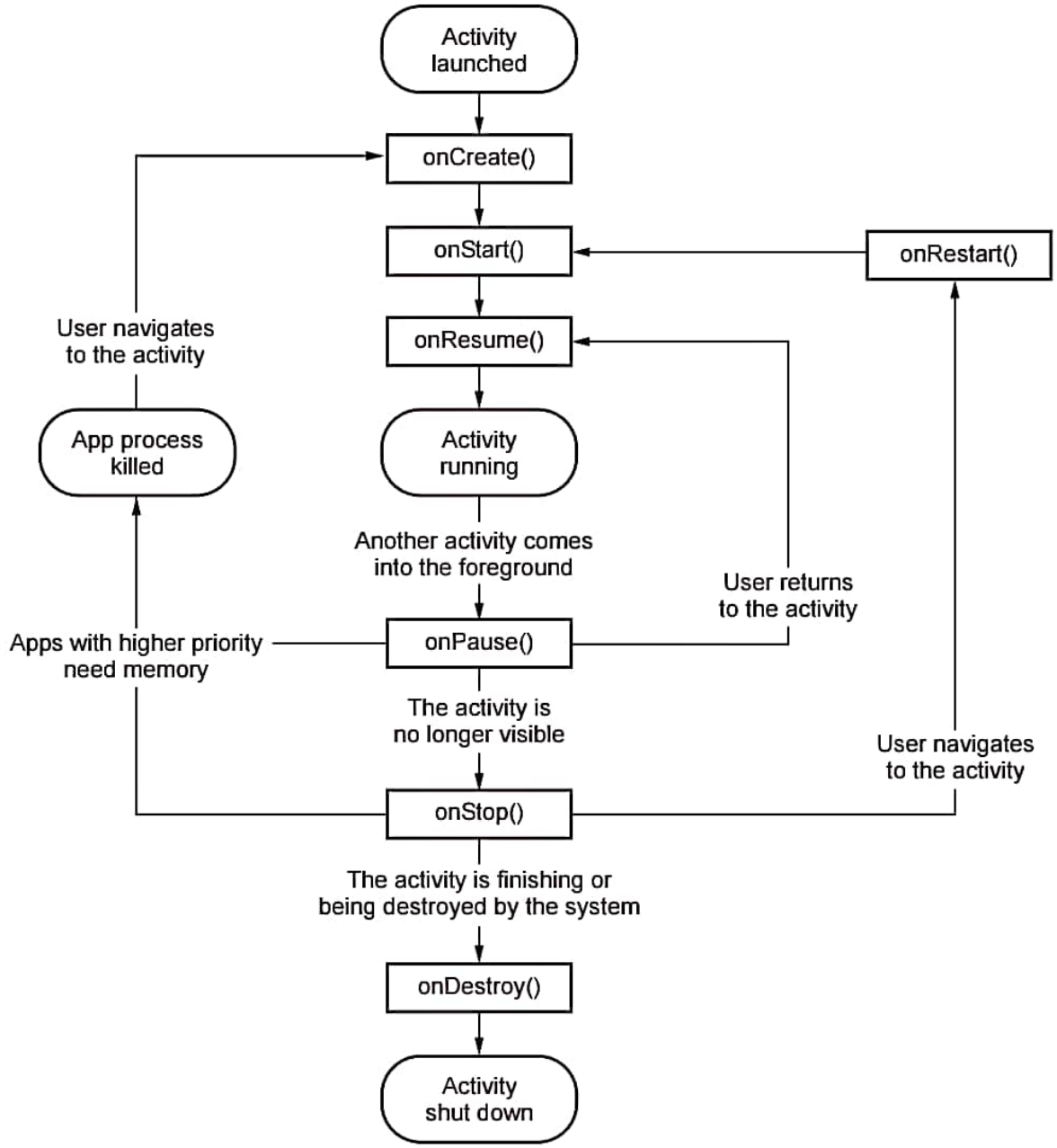



**Fig. 2.1.1**

## 2.2 Fragments in Android

- Fragment class in Android is used to build dynamic User Interfaces. Fragment should be used within the Activity.
- A greatest advantage of fragments is that it simplifies the task of creating UI for multiple screen sizes. A activity can contain any number of fragments.
- A Fragment represents a behavior or a portion of user interface in a FragmentActivity.
- Multiple fragments can be combined in a single activity to build a multi-pane UI and reuse a fragment in multiple activities.

- A fragment can be considered as a modular section of an activity, which has its own lifecycle, receives its own input events, and which you can add or remove while the activity is running (sort of like a "sub activity" that you can reuse in different activities).
- A fragment must always be hosted in an activity and the fragment's lifecycle is directly affected by the host activity's lifecycle. For example, when the activity is paused, so are all fragments in it, and when the activity is destroyed, so are all fragments.
- However, while an activity is running (it is in the resumed lifecycle state), you can manipulate each fragment independently, such as add or remove them. When you perform such a fragment transaction, you can also add it to a back stack that's managed by the activity-each back stack entry in the activity is a record of the fragment transaction that occurred.
- The back stack allows the user to reverse a fragment transaction (navigate backwards), by pressing the Back button.
- When you add a fragment as a part of your activity layout, it lives in a ViewGroup inside the activity's view hierarchy and the fragment defines its own view layout. You can insert a fragment into your activity layout by declaring the fragment in the activity's layout file, as a <fragment> element, or from your application code by adding it to an existing ViewGroup.

### 2.2.1 Importance of Fragments

- There are many use cases for fragments but the most common use cases include :
  1. **Reusing View and Logic Components** - Fragments enable re-use of parts of your screen including views and event logic over and over in different ways across many disparate activities. For example, using the same list across different data sources within an app.
  2. **Tablet Support** - Often within apps, the tablet version of an activity has a substantially different layout from the phone version which is different from the TV version. Fragments enable device-specific activities to reuse shared elements while also having differences.
  3. **Screen Orientation** - Often within apps, the portrait version of an activity has a substantially different layout from the landscape version. Fragments enable both orientations to reuse shared elements while also having differences.

### 2.2.2 Purpose of Fragments

- A fragment is a modular section of an Activity, which has its own lifecycle. It can be added or removed while an Activity is running and can also be reused in different activities.
- **Background :** To create a Fragment you have to subclass the Fragment class. You have to provide a public no-argument constructor, because Android will often re-instantiate a Fragment class when needed (-> state restore).

**Purpose of a Fragment**

- The main purpose of a Fragment is to support a more dynamic UI (tablets, smartphones) and also to make the reuse of UI components a lot easier.
- A Fragment can also exist without its own UI as an invisible worker for the Activity.
- A Fragment is closely tied to the Activity it is in. When the Activity is paused, so are all fragments in it; When the Activity is destroyed, so are all fragments in it.

### 2.2.3 Communication between Fragments and Activity

1. **Fragment.getActivity()** method can return a **FragmentActivity** object. This object is just the Fragment belongs Activity.
2. **FragmentActivity.getSupportFragmentManager()** method will return a **android.support.v4.app.Fragment Manager** instance.
3. **FragmentManager.findFragmentById()** method can retrieve the desired Fragment object with specified id.
4. Call **Fragment.getView().findViewById()** method to get the view controls in that Fragment.
5. Then you can change the view control's property and add event listener.

## 2.3 Life Cycle of Fragment

- Android Fragment is the part of activity, it is also known as sub-activity. There can be more than one fragment in an activity. Fragments represent multiple screen inside one activity.
- Android fragment lifecycle is affected by activity lifecycle because fragments are included in activity.
- Each fragment has its own life cycle methods that is affected by activity life cycle because fragments are embedded in activity.

| No. | Method | Description |
|---|---|---|
| 1. | onAttach(Activity) | It is called only once when it is attached with activity. |
| 2. | onCreate(Bundle) | It is used to initialize the fragment. |
| 3. | onCreateView(LayoutInflater, ViewGroup, Bundle) | Creates and returns view hierarchy. |
| 4. | onActivityCreated (Bundle) | It is invoked after the completion of onCreate() method. |
| 5. | onViewStateRestored (Bundle) | It provides information to the fragment that all the saved state of fragment view hierarchy has been restored. |
| 6. | onStart() | makes the fragment visible. |
| 7. | onResume() | makes the fragment interactive. |
| 8. | onPause() | is called when fragment is no longer interactive. |
| 9. | onStop() | is called when fragment is no longer visible. |
| 10. | onDestroyView() | allows the fragment to clean up resources. |
| 11. | onDestroy() | allows the fragment to do final clean up of fragment state. |
| 12. | onDetach() | It is called immediately prior to the fragment no longer being associated with its activity. |

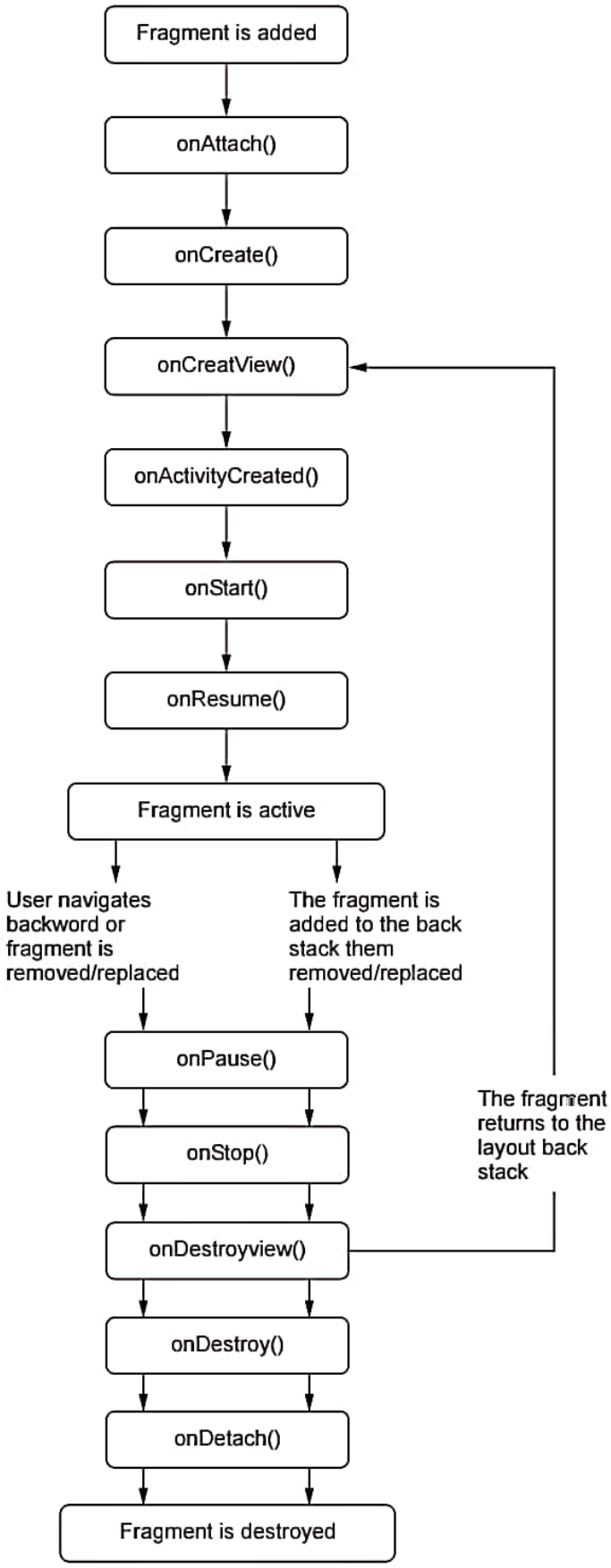



**Fig. 2.3.1 Android fragment life cycle**

## 2.4 Replacing Fragment

- The procedure to replace a fragment is similar to adding one, but requires the **replace()** method instead of **add()**.
- When fragment transactions is performed, such as replace or remove one, it's often appropriate to allow the user to navigate backward and "undo" the change.
- To allow the user to navigate backward through the fragment transactions, you must call **addToBackStack()** before you commit the **FragmentTransaction**.
- When you remove or replace a fragment and add the transaction to the back stack, the fragment that is removed is stopped (not destroyed). If the user navigates back to restore the fragment, it restarts. If you do not add the transaction to the back stack, then the fragment is destroyed when removed or replaced.

Example of replacing one fragment with another :

KOTLINJAVA

```
// Create fragment and give it an argument specifying the article it should show
valnewFragment = ArticleFragment()
Bundle args = Bundle()
args.putInt(ArticleFragment.ARG_POSITION, position)
newFragment.arguments = args

val transaction = supportFragmentManager.beginTransaction().apply {
  // Replace whatever is in the fragment_container view with this fragment,
  // and add the transaction to the back stack so the user can navigate back
  replace(R.id.fragment_container, newFragment)
  addToBackStack(null)
}

// Commit the transaction
transaction.commit();
```

- The **addToBackStack()** method takes an optional string parameter that specifies a unique name for the transaction. The name isn't needed unless you plan to perform advanced fragment operations using the **FragmentManager.BackStackEntry** APIs.

## 2.5 Intent

- Android application components can connect to other Android applications. This connection is based on a task description represented by an Intent object.
- Intents are asynchronous messages which allow application components to request functionality from other Android components.
- Intents are the objects of android. Intent type and intents are mainly useful to perform following things.

- Intents allow you to interact with components from the same applications as well as with components contributed by other applications. For example, an activity can start an external activity for taking a picture.

| Components | Description |
|---|---|
| Starting an activity | By passing an **Internet** object to startActivity() method we can start a new Activity or existing Activity to perform required things : |
| Starting a service | By passing an **Internet** object to startService() method we can start a new Service or send required instructions to an existing Service. |
| Delivering a broadcase | By passing an **Internet** object to sendBroadcast() method we can deliver our message to other app broadcase receivers. |

- Intent is a messaging object which is used to request an action from another app component such as activities, services, broadcast receivers and content providers. Your code can send them to the Android system defining the components you are targeting. For example, via the startActivity() method you can define that the intent should be used to start an activity.
- An intent can contain data via a Bundle. This data can be used by the receiving component.
- In Android the reuse of other application components is a concept known as task. An application can access other Android components to achieve a task. For example, from a component of your application you can trigger another component in the Android system, which manages photos, even if this component is not part of your application. In this component you select a photo and return to your application to use the selected photo.

## 2.6 Intent to Start Another Activity

- An Intent is an object that provides runtime binding between separate components, such as two activities.
- The Intent represents an app's intent to do something. An intent can be used for a wide variety of tasks, one of these task is use of intent to start another activity.
- In MainActivity, add the EXTRA_MESSAGE constant and the sendMessage() code, as shown :

```
const val EXTRA_MESSAGE = "com.example.myfirstapp.MESSAGE"

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
```

```
        }
    /** Called when the user taps the Send button */
    fun sendMessage(view: View) {
        val editText = findViewById<EditText>(R.id.editText)
        val message = editText.text.toString()
        val intent = Intent(this, DisplayMessageActivity::class.java).apply {
            putExtra(EXTRA_MESSAGE, message)
        }
        startActivity(intent)
    }
}
```

Expect Android Studio to encounter Cannot resolve symbol errors again. To clear the errors, press Alt+Enter, or Option+Return on a Mac. You should end up with the following imports:

```
import androidx.appcompat.app.AppCompatActivity
import android.content.Intent
import android.os.Bundle
import android.view.View
import android.widget.EditText
```

Here's what's going on in sendMessage():

- The Intent constructor takes two parameters, a Context and a Class.
- The Context parameter is used first because the Activity class is a subclass of Context.
- The Class parameter of the app component, to which the system delivers the Intent, is, in this case, the activity to start.
- The putExtra() method adds the value of EditText to the intent. An Intent can carry data types as key-value pairs called extras.
- Your key is a public constant EXTRA_MESSAGE because the next activity uses the key to retrieve the text value. It's a good practice to define keys for intent extras with your app's package name as a prefix. This ensures that the keys are unique, in case your app interacts with other apps.
- The startActivity() method starts an instance of the DisplayMessageActivity that's specified by the Intent. Then class can be created.

## 2.7 Implicit and Explicit Intent

- An implicit Intent does not name a specific component. It declares a general action to perform. The action specifies the thing that the app wants to do.
- Background : The system is looking for an appropriate component to start :
- If multiple components are compatible with the action, the system shows a dialog so the user can pick which app to use.

- If there is no appropriate component available on the device that can handle the action, your app will crash immediately !

**Example :** If an app wants to trigger a phone call, it only has to specify the corresponding action (ACTION_DIAL) :

```
Uri number = Uri.parse("tel:4213371337");
Intent callIntent = new Intent(Intent.ACTION_DIAL, number);
if (callIntent.resolveActivity(getPackageManager()) != null) {
        // the system can resolve the Intent
}
```

**Explicit Intent**

- An explicit Intent specifies the component by the fully-qualified class name to start. This is the common case to start a component in the own app.

**Example :**

```
Intent startIntent = new Intent(myContext, AnotherActivity.cla
```

## 2.8 Android Virtual Device (AVD)

- An Android Virtual Device (AVD) is a configuration that defines the characteristics of an Android phone, tablet, Wear OS, Android TV, or Automotive OS device that you want to simulate in the Android Emulator.
- An Android Virtual Device (AVD) is a device configuration that is run with the Android emulator. It works with the emulator to provide a virtual device-specific environment in which to install and run Android apps.
- An AVD contains a hardware profile, system image, storage area, skin, and other properties.
- The avdmanager is a command line tool that allows you to create and manage Android Virtual Devices (AVDs) from the command line. An AVD lets you define the characteristics of an Android handset, Wear OS watch, or Android TV device that you want to simulate in the Android Emulator.
- If you're using Android Studio, then you do not need to use this tool and you can instead create and manage AVDs from the IDE.
- The avdmanager tool is provided in the Android SDK Tools package (25.3.0 and higher) and is located in android_sdk/tools/bin/.

**Syntax**

avdmanager [global options] command [command options]

## 2.9 Mapping Application to Process

- Android applications are primarily written in the Java programming language.
- During development the developer creates the Android specific configuration files and writes the application logic in the Java programming language.
- The ADT or the Android studio tools convert these application files, transparently to the user, into an Android application.
- When developers trigger the deployment in their IDE, the whole Android application is compiled, packaged, deployed and started.

### Conversion Process from Source Code to Android Application

- The Java source files are converted to Java class files by the Java compiler.
- The Android SDK contains a tool called dx which converts Java class files into a *.dex* (Dalvik Executable) file. All class files of the application are placed in this *.dex* file.
- During this conversion process redundant information in the class files are optimized in the *.dex* file.For example, if the same **String** is found in different class files, the *.dex* file contains only one reference of this **String**.
- These *.dex* files are therefore much smaller in size than the corresponding class files.
- The *.dex* file and the resources of an Android project, e.g., the images and XML files, are packed into an *.apk* (Android package) file. The program *aapt* (Android Asset Packaging Tool) performs this step.
- The resulting *.apk* file contains all necessary data to run the Android application and can be deployed to an Android device via the *adb* tool.

### Android SDK Features

- Important features of Android SDK are as under.

1. No licensing, distributions or development fees or release approval processes.
2. Full multimedia hardware control.
3. APIs for using sensor hardware including accelerometer and the compass.
4. APIs for location based services.
5. Android Inter-Process Communication (IPC).
6. Shared data storage.
7. Background applications and processes.
8. Home screen widgets, live folders.
9. HTML5 WebKit-based web browser.

10. GSM, EDGE and 3G networks for telephony and data transfer.
11. The Android SDK includes development tools which helps compile and debug any app.
12. Android emulator shows how app will look and behaviour on a real Android device.

## 2.10 Short Questions and Answers

**Q.1 Define fragments.**

**Ans. :** Fragment is a UI entity attached to Activity. Fragments can be reused by attaching in different activities. Activity can have multiple fragments attached to it. Fragment must be attached to an activity and its lifecycle will depend on its host activity.

**Q.2 What is fragment lifecycle.**

**Ans. :**

- **onAttach()** : The fragment instance is associated with an activity instance.The fragment and the activity is not fully initialized. Typically you get in this method a reference to the activity which uses the fragment for further initialization work.
- **onCreate()** : The system calls this method when creating the fragment. You should initialize essential components of the fragment that you want to retain when the fragment is paused or stopped, then resumed.
- **onCreateView()** : The system calls this callback when it's time for the fragment to draw its user interface for the first time. To draw a UI for your fragment, you must return a View component from this method that is the root of your fragment's layout. You can return null if the fragment does not provide a UI.
- **onActivityCreated()** : The onActivityCreated() is called after the onCreateView() method when the host activity is created. Activity and fragment instance have been created as well as the view hierarchy of the activity. At this point, view can be accessed with the findViewById() method. example. In this method you can instantiate objects which require a Context object.
- **onStart()** : The onStart() method is called once the fragment gets visible.
- **onResume()** : Fragment becomes active.
- **onPause()** : The system calls this method as the first indication that the user is leaving the fragment. This is usually where you should commit any changes that should be persisted beyond the current user session.
- **onStop()** : Fragment going to be stopped by calling onStop()

- **onDestroyView()** : Fragment view will destroy after call this method
- **onDestroy()** :called to do final clean up of the fragment's state but Not guaranteed to be called by the Android platform.

**Q.3 What's the difference between onCreate() and onStart() ?**

**Ans. :** • The onCreate() method is called once during the Activity lifecycle, either when the application starts, or when the Activity has been destroyed and then recreated, for example during a configuration change.

- The onStart() method is called whenever the Activity becomes visible to the user, typically after onCreate() or onRestart().

**Q.4 What is the difference between fragments & activities. Explain the relationship between the two.**

**Ans. :** An Activity is an application component that provides a screen, with which users can interact in order to do something whereas a Fragment represents a behavior or a portion of user interface in an Activity (with its own lifecycle and input events, and which can be added or removed at will).

**Q.5 When should you use a fragment rather than an activity ?**

**Ans. :** • When there are ui components that are going to be used across multiple activities.

- When there are multiple views that can be displayed side by side (viewPager tabs)
- When you have data that needs to be persisted across Activity restarts (such as retained fragments)

**Q.6 What is an intent ?**

**Ans. : Intent**

Intents are messages that can be used to pass information to the various components of android. For instance, launch an activity, open a webview etc.

Two types of intents-

1. **Implicit :** Implicit intent is when you call system default intent like send email, send SMS, dial number.
2. **Explicit :** Explicit intent is when you call an application activity from another activity of the same application.

**Q.7 Difference between Service and Intent Service.**

**Ans. :** • Service is the base class for Android services that can be extended to create any service. A class that directly extends Service runs on the main thread so it will block the UI (if there is one) and should therefore either be used only for short tasks or should make use of other threads for longer tasks.

- IntentService is a subclass of Service that handles asynchronous requests (expressed as "Intents") on demand. Clients send requests through startService(Intent) calls. The service is started as needed, handles each Intent in turn using a worker thread, and stops itself when it runs out of work.

**Q.8 Give Difference between Service, Intent Service, AsyncTask and Threads**

**Ans. :** • **Android service** is a component that is used to perform operations on the background such as playing music. It doesn't has any UI (user interface). The service runs in the background indefinitely even if application is destroyed.

- **AsyncTask** allows you to perform asynchronous work on your user interface. It performs the blocking operations in a worker thread and then publishes the results on the UI thread, without requiring you to handle threads and/or handlers yourself.
- **IntentService** is a base class for Services that handle asynchronous requests (expressed as Intents) on demand. Clients send requests through startService(Intent) calls; the service is started as needed, handles each Intent in turn using a worker thread, and stops itself when it runs out of work.
- A **thread** is a single sequential flow of control within a program. Threads can be thought of as mini-processes running within a main process.

**Q.9 State three common usages of intent and how are they invoked.**

**Ans. :** Android Intents are used to

1. start an activity - startActivity(intent)
2. start a service - startService(intent)
3. deliver a broadcast - sendBroadcast(intent)

**Q.10 What are the four essential states of an Activity ?**

**Ans. :** **States of Activity**

1. Active : if the Activity is active (it can receive user input) and visible
2. Paused : if the Activity is visible but not active
3. Stopped : if the Activity is not visible
4. Destroyed : when the activity process is killed

**Q.11 Differentiate between Broadcast Receivers and Services.**

**Ans. :** A service is used for long running tasks in the background such as playing a music or tracking and updating the user's background location.

- A Broadcast Receiver is a component that once registered within an application executes the onReceive() method when some system event gets triggered. The events the receiver listens to are defined in the AndroidManifest.xml in the intent filters.
- Types of system events that a Broadcast Receiver listens to are : changes in the network, boot completed, battery low, push notifications received etc. We can even send our own custom broadcasts using sendBroadcast (intent).

## 2.11 Multiple Choice Questions

**Q.1** *Which component is not activated by an Intent ?*

a Activity b Services

c ContentProvider d BroadcastReceiver

**Q.2** *When contentProvider would be activated ?*

a Using Intent b Using SQLite

c Using ContentResolver d None of the above

**Q.3** *You can shut down an activity by calling its _____ method.*

a onDestory() b finishActivity()

c finish() d None of the above

**Q.4** *How many ways to start services ?*

a Started b Bound

c a & b d None of the above.

**Answer Keys for Multiple Choice Questions**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Q.1 | c | Q.2 | c | Q.3 | c | Q.4 | c |

**Notes**

# 3 User Interface

## Syllabus

*Working with the User Interface Using Views and ViewGroups Working with View Groups, Building data with the AdapterView Class, Designing AutoTextCompleteView, Implementing Screen Orientation, Designing the views programmatically, Handling UI events, Creating Menus.*

## Contents

## 3.1 Android Graphics Interface

- Android graphical interfaces are usually implemented as XML files (although they could also be dynamically created from Java code).
- An Android UI is conceptually similar to a common HTML page.
- In a manner similar to a web page interaction, when the Android user touches the screen, the controller interprets the input and determines what specific portion of the screen and gestures were involved.
- Based on this information it tells the model about the interaction in such a way that the appropriate "callback listener" or lifecycle state could be called into action.
- Unlike a web application (which refreshes its pages after explicit requests from the user) an asynchronous Android background service could quietly notify the controller about some change of state (such as reaching a given coordinate on a map) and in turn a change of the view's state could be triggered; all of these without user intervention

## 3.2 Layouts in Android

- Layouts are invisible structured containers used for holding other Views and nested layouts.
- 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. Once your layout has created, you can load the layout resource from your application code, in your Activity.onCreate() callback implementation as shown below -

```
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
}
```

### 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.

1. **Linear Layout** - Linear layout is a view group that aligns all children in a single direction, vertically or horizontally.
2. **Relative Layout** - Relative layout is a view group that displays child views in relative positions.

3. **Table Layout** - Table layout is a view that groups views into rows and columns.
4. **Absolute Layout** - Absolute layout enables you to specify the exact location of its children.
5. **Frame Layout** - The frame layout is a placeholder on screen that you can use to display a single view.
6. **List View** - List view is a view group that displays a list of scrollable items.
7. **Grid View** - Grid view is a view Group that displays items in a two-dimensional, scrollable grid.

## 3.3 Android User Interface

- Android is a widely used OS made for smart phones and tablets. It is an open source project led by Google and it is released under Apache License.
- Android has a very large community that extend its features and creates apps that cover almost all aspects.
- All android applications, called apps, are built on Android UI framework. App interface is the first thing a user sees and interacts with. From the user perspective, this framework keeps the overall experience consistent for every app installed in our smartphone or tablets. At the same time, from the developer perspective, this framework provides some basic blocks that can be used to build complex and consistent user interface (API).
- Android UI interface is divided in three different areas :

1. Home screen
2. All apps
3. Recent screen

- The home screen is the "landing" area when we power our phone on. This interface is highly customizable and themed. Using widgets we can create and personalize our "home" screen.
- All apps is the interface where the app installed are displayed, while recent screens are the list of last used apps.
- Since its born, Android has changed a lot in terms of its features and its interfaces. The growth of the smartphone power made possible creating ever more appealing apps.
- At the beginning, apps in Android did not have a consistent interface and well defined rules so every app had a different approach, navigation structure and buttons position. This caused user confusion and it was one of the most important missing features compared to the iOS.

### 3.3.1 Editable Text View

- The EditText editable text field places the cursor in the text field and automatically displays the on-screen keyboard. You will change attributes of the text entry field so that the keyboard suggests spelling corrections while you type, and automatically starts each new sentence with capital letters.

**For example :**

**android:inputType="textCapSentences"** : Sets the keyboard to capital letters at the beginning of sentences.

**android:inputType="textAutoCorrect"** : Sets the keyboard to show automatic spelling corrections as you enter characters.

**android:inputType="textMultiLine"** : Enables the Return key on the keyboard to end lines and create new blank lines without closing the keyboard.

android:inputType="textPassword" : Sets the characters the user enters into dots to conceal the entered password.

### 3.3.2 Radio and Toggle Buttons

- Radio buttons are input controls that are useful for selecting only one option from a set of options. You should use radio buttons if you want the user to see all available options side-by-side.
- A basic Button is often used to perform some sort of action, such as submitting a form or confirming a selection. A basic Button control can contain a text or image label.
- A CheckBox is a button with two states-checked or unchecked. You often use CheckBox controls to turn a feature on or off or to pick multiple items from a list.
- A ToggleButton is similar to a CheckBox, but you use it to visually show the state. The default behavior of a toggle is like that of a power on/off button.
- A RadioButton provides selection of an item. Grouping RadioButton controls together in a container called a RadioGroup enables the developer to enforce that only one RadioButton is selected at a time.

### 3.3.3 Creating Radio Button

- To create each radio button option, create a RadioButton in your layout. However, because radio buttons are mutually exclusive, you must group them together inside a RadioGroup. By grouping them together, the system ensures that only one radio button can be selected at a time.

```
<?xml version="1.0" encoding="utf-8"?>
<RadioGroupxmlns:android="http://schemas.android.
com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:orientation="vertical">
   <RadioButton android:id="@+id/radio_pirates"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/pirates"
      android:onClick="onRadioButtonClicked"/>
   <RadioButtonandroid:id="@+id/radio_ninjas"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/ninjas"
      android:onClick="onRadioButtonClicked"/>
</RadioGroup>
and within the activity:

public void onRadioButtonClicked(View view) {
   // Is the button now checked?
   boolean checked = ((RadioButton) view).isChecked();

   // Check which radio button was clicked
   switch(view.getId()) {
      case R.id.radio_pirates:
         if (checked)
            // Pirates are the best
         break;
      case R.id.radio_ninjas:
         if (checked)


            // Ninjas rule
         break;
   }
}
```

### 3.3.4 Spinners

- Spinners provide a quick way to select one value from a set of options. Create a spinner like any view and specify the android:entries attribute to specify the set of options :

```
<Spinner
   android:id="@+id/mySpinner"
   android:layout_width="wrap_content"
   android:entries="@array/planets_arrays"
   android:prompt="@string/planets_prompt"
   android:layout_height="wrap_content" />
```

and then specify the string array of options in res/values/planets_array.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
      <string-array name="planets_array">
            <item>Mercury</item>
            <item>Venus</item>
            <item>Earth</item>
            <item>Mars</item>
      </string-array>
</resources>
```

## 3.4 Event Handling

- **Input Events** are used to capture the events, such as button clicks, edittext touch, etc. from the View objects that defined in user interface of our application, when the user interacts with it.
- To handle input events in android, the views must have in place an **event listener**. The View class, from which all UI components are derived contains a wide range event listener interfaces and each listener interface contains an abstract declaration for a callback method.
- To respond to an event of a particular type, the view must register an appropriate event listener and implement the corresponding callback method.
- For example, if button is to respond to a click event it must register to **View.onClickListener** event listener and implement the corresponding **onClick()** callback method. In application when a button click event is detected, the Android framework will call the **onClick()** method of that particular view.
- Generally, to handle input events we use **Event Listeners** and **Event Handling** in android applications to listen for user interactions and to extend a View class, in order to build a custom component.

### Android Event Listeners

- In android, **Event Listener** is an interface in the View class that contains a single call-back method. These methods will be called by the Android framework when the View which is registered with listener is triggered by user interaction with the item in UI.

- Following are the call-back methods which included in event listener interface.
- There are many more event listeners available as a part of View class to use it in our android applications.

## 3.5 Android Toolbox of Standard View

- Android supplies a toolbox of standard Views to help you create simple interfaces. By using these controls (and modifying or extending them as necessary), you can simplify your development and provide consistency between applications.
- The following list highlights some of the more familiar toolbox controls :

**1. TextView**

- A standard read only text label. It supports multiline display, string formatting and automatic word wrapping.

**2. EditText**

- An editable text entry box. It accepts multiline entry and word wrapping.

**3. ListView**

- A View Group that creates and manages a group of Views used to display the items in a list. The standard ListView displays the string value of an array of objects using a Text View for each item.

**4. Spinner**

- Composite control that displays a TextView and an associated ListView that lets you select an item from a list to display in the textbox. It's made from a Text View displaying the current selection, combined with a button that displays a selection dialog when pressed.

**5. Button**

- Standard push-button

6. **CheckBox**

- Two-state button represented with a checked or unchecked box

7. **RadioButton**

- Two-state grouped buttons. Presents the user with a number of binary options of which only one can be selected at a time.

## 3.6 Menus in Android

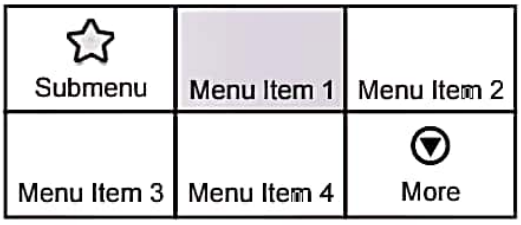
- Menus offer a way to expose application functions without sacrificing valuable screen space. Each activity can specify its own activity menu that's displayed when the device's menu button is pressed.
- Android also supports context menus that can be assigned to any View within an Activity. A View's context menu is triggered when a user holds the middle D-pad

button, depresses the trackball or longpresses the touch screen for around 3 seconds when the View has focus.

- Activity and context menus support submenus, checkboxes, radio buttons, shortcut keys and icons.
- To improve the usability of application menus, Android features a three-stage menu system optimized for small screens :

**1. The Icon Menu**

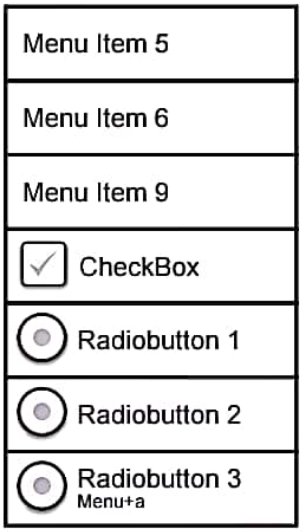
- This compact menu appears along the bottom of the screen when the Menu button is pressed. It displays the icons and text for up to six Menu Items (or submenus).




- This icon menu does not display checkboxes, radio buttons or the shortcut keys for Menu Items, so it's generally good practice not to assign checkboxes or radio buttons to icon menu items, as they will not be available. If more than six Menu Items have been defined, a More item is included that, when selected, displays the expanded menu. Pressing the Back button closes the icon menu.

**2. The Expanded Menu**

- The expanded menu is triggered when a user selects the **More** Menu Item from the icon menu. The expanded menu displays a scrollable list of only the Menu Items that weren't visible in the icon menu. This menu displays full text, shortcut keys and checkboxes/radio buttons as appropriate.



**3. Submenus**

- The traditional "expanding hierarchical tree" can be awkward to navigate using a mouse, so it's no surprise that this metaphor is particularly ill-suited for use on mobile devices.
- The Android alternative is to display each submenu in a floating window. For example, when a user selects a submenu such as the creatively labeled Submenu from menus, its items are displayed in a floating menu Dialog box, as shown in Fig. 3.6.1.


**Fig. 3.6.1**

## 3.7 Android Layout Classes

- Layout Managers (more generally, "layouts") are extensions of the ViewGroup class designed to control the position of child controls on a screen.
- Layouts can be nested, letting you create arbitrarily complex interfaces using a combination of Layout Managers.
- The Android SDK includes some simple layouts to help you construct your UI. It's up to you to select the right combination of layouts to make your interface easy to understand and use.
- The following list includes some of the more versatile layout classes available :

**1. FrameLayout**

- The simplest of the Layout Managers, the Frame Layout simply pins each child view to the top left corner. Adding multiple children stacks each new child on top of the previous, with each new View obscuring the last.

**2. LinearLayout**

- A Linear Layout adds each child View in a straight line, either vertically or horizontally. A vertical layout has one child View per row, while a horizontal layout has a single row of Views.

- The Linear Layout Manager allows you to specify a "weight" for each child View that controls the relative size of each within the available space.

**3. RelativeLayout**

- Using the Relative Layout, you can define the positions of each of the child.
- Views relative to each other and the screen boundaries.

**4. TableLayout**

- The Table Layout lets you lay out Views using a grid of rows and columns. Tables can span multiple rows and columns and columns can be set to shrink or grow.

**5. AbsoluteLayout**

- In an Absolute Layout, each child View's position is defined in absolute coordinates. Using this class, you can guarantee the exact layout of your components, but at a price. Compared to the previous managers, describing a layout in absolute terms means that your layout can't dynamically adjust for different screen resolutions and orientations.

**Working with Layouts**

- Much as web designers use HTML, user interface designers can use XML to define Android application screen elements and layout.
- A layout XML resource is where many different resources come together to form the definition of an Android application screen.
- Layout resource files are included in the /res/layout/ directory and are compiled into the application package at build time. Layout files might include many user interface controls and define the layout for an entire screen or describe custom controls used in other layouts.

## 3.8 Short Questions and Answers

**Q.1 What is view group ? How are they different from views ?**

**Ans. : View :** View objects are the basic building blocks of User Interface (UI) elements in Android. View is a simple rectangle box which responds to the user's actions. Examples are EditText, Button, CheckBox etc. View refers to the android.view. View class, which is the base class of all UI classes.

**ViewGroup :** ViewGroup is the invisible container. It holds View and ViewGroup. For example, LinearLayout is the ViewGroup that contains Button(View), and other Layouts also. ViewGroup is the base class for Layouts.

**Q.2 Difference between RelativeLayout and LinearLayout ?**

**Ans. : Linear Layout** - Arranges elements either vertically or horizontally. i.e. in a row or column.

**Relative Layout** - Arranges elements relative to parent or other elements.

**Q.3 Briefly describe some ways that you can optimize view usage.**

**Ans. :** • **Checking for excessive overdraw :** Install your app on an Android device, and then enable the "Debug GPU Overview" option.

- **Flattening your view hierarchy :** Inspect your view hierarchy using Android Studio's 'Hierarchy Viewer' tool.
- **Measuring how long it takes each view to complete the measure**, layout, and draw phases. You can also use Hierarchy Viewer to identify any parts of the rendering pipeline that you need to optimise.

**Q.4 How to reduce apk size in Android ?**

**Ans. :** • Enable proguard in your project by adding following lines to your release build type.

- Enable shrinkResources .
- Strip down all the unused local resources by adding required resources name in "resConfigs".
- Convert all the images to the webp or vector drawables.

**Q.5 What is an activity in Android ?**

**Ans. :** • An activity is a single, focused thing that the user can do. when ever user click on GUI the next Activity will be start and new GUI set base on coding.

**Q.6 What is a service?**

**Ans. :** • A service is a component in android that's used for performing tasks in the background such as playing Music, location updating etc. Unlike activities, a service does not have a UI. Also, a service can keep running in the background even if the activity is destroyed.

**Q.7 Define and differentiate between the two types of services.**

**Ans. :** Services are largely divided into two categories : **Bound Services** and **Unbound/Started Services**

1. **Bound Services :** An Android component may bind itself to a Service using bindservice(). A bound service would run as long as the other application components are bound to it. As soon as the components call unbindService(), the service destroys itself.
2. **Unbound Services :** A service is started when a component (like activity) calls startService() method and it runs in the background indefinitely even if the original component is destroyed.

## 3.9 Multiple Choice Questions

**Q.1** *The _____ file specifies the layout of your screen.*

a Layout b Manifest

c Strings XML d R

**Q.2** *Which of the following are UI elements that you can use in a window in an Android application ?*

a TextBox b TextView

c EditText d Both B & C

**Q.3** *One of application component, that controls UI and manage user interaction with phone screen is called _____.*

a content providers b activities

c broadcast receivers d services

**Q.4** *A type of service provided by android that helps in creating user interfaces is _____.*

a notifications manager b content providers

c activity manager d view system

**Q.5** *One of application component, that manages database issues is called _____.*

a services b content providers

c broadcast receivers d activities

**Q.6** *One of application component, that manages interaction between operating system and android OS is called _____.*

a broadcast receivers b content providers

c activities d services

**Q.7** *One of application component, that manages application's background services is called _____.*

a activities b broadcast receivers

c services d content providers

**Q.8** *In which directory XML layout files are storeds ?*

a /assets
b /src
c /res/values
d /res/layout

**Q.9** *View Pager is used for _____.*

a swiping activities
b swiping fragments
c paging down list items
d view pager is not supported by android SDK

**Q.10** *What is an Interface in android ?*

a Interface is a class.
b Interface acts as a bridge between class and the outside world.
c Interface is a layout file
d All of the above

**Answer Keys for Multiple Choice Questions**

| | | | | | |
|---|---|---|---|---|---|
| **Q.1** | a | **Q.2** | d | **Q.3** | b |
| **Q.4** | d | **Q.5** | b | **Q.6** | a |
| **Q.7** | c | **Q.8** | d | **Q.9** | b |
| **Q.10** | b | | | | |

❑❑❑

**Notes**

# 4 Storing the Data Persistently

## Syllabus

*Introducing the Data Storage Options, Using the internal storage, using the external storage, using the SQLite Database, Working with content provider.*

## Contents

## 4.1 Storage Data Folder

**Store application-specific data**

- The application data folder is a special hidden folder that your app can use to store application-specific data, such as configuration files.
- The application data folder is automatically created when you attempt to create a file in it. Use this folder to store any files that the user shouldn't directly interact with.
- This folder is only accessible by your application and its contents are hidden from the user and from other Drive apps.
- The application data folder is deleted when a user uninstalls your app from their MyDrive. Users can also delete your app's data folder manually.

### 4.1.1 Application Data Folder Scope

- Before you can access the application data folder, you must request access to the https://www.googleapis.com/auth/drive.appdata scope. For more information about scopes and how to request access to them, refer to Authenticate your users.

### 4.1.2 Creating File in App Data Folder

- To create a file in the application data folder, specify appDataFolder in the parents property of the file and use the files.create method to upload the file to the folder.
- The following example shows how to insert a file into a folder using a client library :

```
File fileMetadata = new File();
fileMetadata.setName("config.json");
fileMetadata.setParents(Collections.singletonList("appDataFolder"));
java.io.FilefilePath = new java.io.File("files/config.json");
FileContent mediaContent = new FileContent("application/json", filePath);
File file = driveService.files().create(fileMetadata, mediaContent)
    .setFields("id")
    .execute();
System.out.println("File ID: " + file.getId());
```

### 4.1.3 Search for Files in App Data Folder

- To search for files in ithe application data folder, set the spaces field to appDataFolder and use the ***files.list*** method.

- The following example shows how to search for files in the application data folder using a client library :

```
FileList files = driveService.files().list()
    .setSpaces("appDataFolder")
    .setFields("nextPageToken, files(id, name)")
    .setPageSize(10)
    .execute();
for (File file : files.getFiles()) {
  System.out.printf("Found file: %s (%s)\n",
      file.getName(), file.getId());
}
```

## 4.2 Using Internal Storage

### Android Read Write Data To Internal File-

- Android data can be saved in internal storage (ROM) , external storage(SD card), shared preferences or SQLite database.
- Android is based on Linux, android file system is Linux based also.
- Android studio provide android device monitor tool for you to monitor and transfer files between android device and your PC.
- All android app internal data files is saved in /data/data/<your app package name> folder like below. For example, my app data internal file is saved in /data/data/com.dev2qa.example folder.
- There are files and cache sub folder under the package name folder.
  1. **Files folder -** android.content.Context's *getFilesDir()* method can return this folder. This folder is used to save general files.
  2. **Cache folder -** android.content.Context's *getCacheDir()* method can return this folder. This folder is used to save cached files.
- When device internal storage space is low, cache files will be removed by android os automatically to make internal storage space bigger. Generally you need to delete the unused cache files in code timely, totally cache file size is better not more than 1 MB.

## 4.3 Using External Storage

### Android External Storage

- Android data can be saved in internal storage ( ROM ) , external storage(SD card), shared preferences or SQLite database.
- Android external storage can be used to write and save data, read configuration files etc.
- External storage such as SD card can also store application data, there's no security enforced upon files you save to the external storage.

- In general there are two types of External Storage :
  1. Primary External Storage: In built shared storage which is "accessible by the user by plugging in a USB cable and mounting it as a drive on a host computer". Example: When we say Nexus 5 32 GB.
  2. Secondary External Storage : Removable storage. Example : SD Card
- All applications can read and write files placed on the external storage and the user can remove them. We need to check if the SD card is available and if we can write to it. Once we've checked that the external storage is available only then we can write to it else the save button would be disabled.
- Firstly, we need to make sure that the application has permission to read and write data to the users SD card, so lets open up the *AndroidManifest.xml* and add the following permissions :

```
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

- Also, external storage may be tied up by the user having mounted it as a USB storage device. So we need to check if the external storage is available and is not read only.

*getExternalStorageState()* is a static method of Environment to determine if external storage is presently available or not. As you can see if the condition is false we've disabled the save button.

1. *Environment.getExternalStorageState() :* returns path to internal SD mount point like "/mnt/sdcard".
2. *getExternalFilesDir() :* It returns the path to files folder inside Android/data/data/application_package/ on the SD card. It is used to store any required files for your app (like images downloaded from web or cache files). Once the app is uninstalled, any data stored in this folder is gone too.

## 4.4 Shared Preferences

- Shared preferences allow you to read and write small amounts of primitive data (as key/value pairs) to a file on the device storage.
- The SharedPreference class provides APIs for getting a handle to a preference file and for reading, writing, and managing this data.
- The shared preferences file itself is managed by the Android framework, and accessible to (shared with) all the components of your app. That data is not, however, shared with or accessible to any other apps.

- The data you save to shared preferences is different from the data in the saved activity state.
- The data in the activity instance state is retained across activity instances in the same user session.
- Shared preferences persist across user sessions, even if your app is killed and restarted or if the device is rebooted.
- Use shared preferences only when you need to save a small amount data as simple key/value pairs. To manage larger amounts of persistent app data use the other methods such as SQL databases.

### 4.4.1 Creating Private and Shared Preferences

- Individual activities can have their own private preferences. These preferences are for the specific Activity only and are not shared with other activities within the application.
- The activity gets only one group of private preferences.
- The following code retrieves the activity's private preferences :

```
import android.content.SharedPreferences;
...
SharedPreferencessettingsActivity = getPreferences(MODE_PRIVATE);
```

- Creating shared preferences is similar. The only two differences are that we must name our preference set and use a different call to get the preference instance :

```
import android.content.SharedPreferences;
...
SharedPreferences settings =
getSharedPreferences("MyCustomSharedPreferences", 0);
```

- You can access shared preferences by name from any activity in the application. There is no limit to the number of different shared preferences you can create. You can have some shared preferences called UserNetworkPreferences and another called AppDisplayPreferences.
- How you organize shared preferences is up to you, the developer.
- However, you want to declare your preference name as a variable (in a base class or header) so that you can reuse the name across multiple activities. For example

```
public static final String PREFERENCE_FILENAME = "AppPrefs";
```

## 4.5 SQLite

- SQLite is a software library that provides a relational database management system.

- The lite in SQLite means light weight in terms of setup, database administration, and required resource.
- A SQLite database is a good storage solution when you have structured data that you need to store persistently and access, search, and change frequently.
- SQLite is self-contained means it requires minimal support from the operating system or external library.
- This makes SQLite usable in any environments especially in embedded devices like iPhones, Android phones, game consoles, handheld media players, etc.
- SQLite is developed using ANSI-C. The source code is available as a big sqlite3.c and its header file sqlite3.h.
- If you want to develop an application that uses SQLite, you just need to drop these files into your project and compile it with your code.
- SQLite databases are very lightweight. Unlike other database systems, there is no configuration, installation required to start working on an SQLite database.
- When you use a SQLite database, all interactions with the database are through an instance of the SQLiteOpenHelper class which executes your requests and manages your database for you.
- The Android SQLite Database requires very little memory (around 250kb), which is available on all android devices.
- Every device has an inbuilt support for SQLite database, which is automatically managed on android right from its creation, execution to querying up process.
- SQLite is an open source database, available on every android database. It supports standard relations database features, like SQL syntax, transactions & SQL statements. SQLite is considerably, the lighter version of SQL database, where most of the SQL commands don't run on SQLite database.
- Once SQLite is in place, it is important to ensure that a feature or command is available in SQLite; only then can it be executed.
- The Basic Advantages of SQLite :
  1. It's a light weight database
  2. Requires very little memory
  3. An Automatically managed database
- The SQLite supports only three Datatypes :
  1. Text(like string) - for storing data type store.
  2. Integer(like int) - for storing integer primary key.
  3. Real(like double) - for storing long values.

### 4.5.1 Creating Database in SQLite

- Unlike other database management systems, there is no CREATE DATABASE command in SQLite. In SQLite, here is how you can create a new database :
- The simplest way to create a new SQLiteDatabase instance for your application is to use the openOrCreateDatabase() method of your application Context, like this :

```
import android.database.sqlite.SQLiteDatabase;
...
SQLiteDatabasemDatabase;
mDatabase = openOrCreateDatabase(
"my_sqlite_database.db",
SQLiteDatabase.CREATE_IF_NECESSARY,
null);
```

## 4.6 Content Provider

- A content provider manages access to a central repository of data. A provider is part of an Android application, which often provides its own UI for working with the data.
- Content providers are primarily intended to be used by other applications, which access the provider using a provider client object.
- Together, providers and provider clients offer a consistent, standard interface to data that also handles inter-process communication and secure data access.
- A content provider presents data to external applications as one or more tables that are similar to the tables found in a relational database.
- A row represents an instance of some type of data the provider collects, and each column in the row represents an individual piece of data collected for an instance.
- Typically you work with content providers in one of two scenarios; you may want to implement code to access an existing content provider in another.
- A content provider coordinates access to the data storage layer in your application for a number of different APIs and components as illustrated in Fig. 4.6.1.
- The content provider and other components include :
    1. Sharing access to your application data with other applications
    2. Sending data to a widget
    3. Returning custom search suggestions for your application through the search framework using SearchRecentSuggestionsProvider.
    4. Synchronizing application data with your server using an implementation of AbstractThreadedSyncAdapter
    5. Loading data in your UI using a CursorLoader

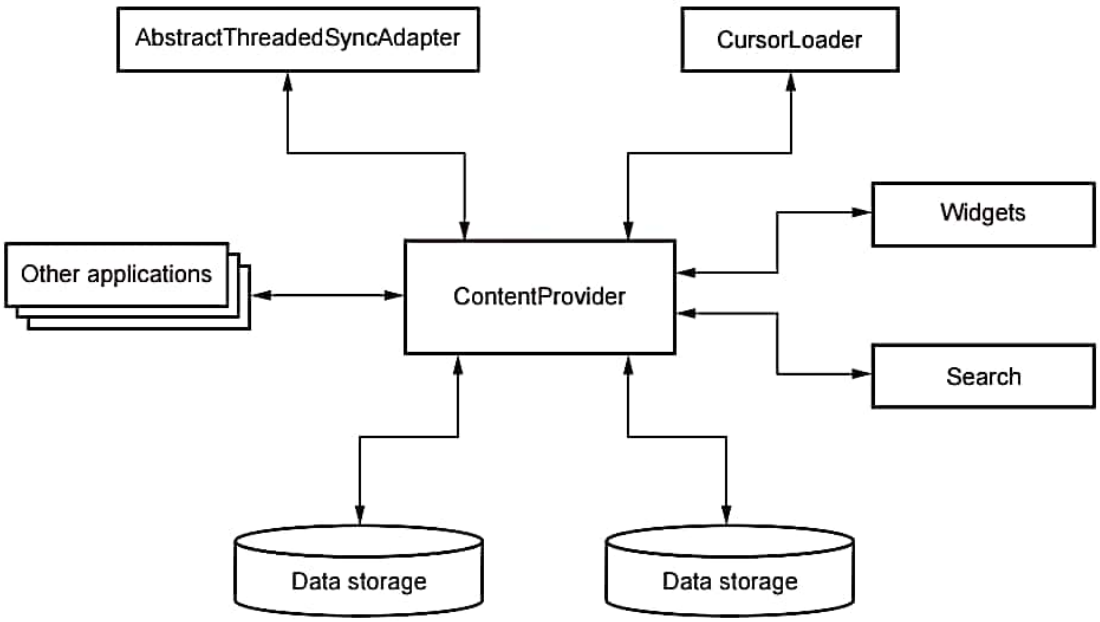



Fig. 4.6.1

## 4.6.1 Content Provider Classes

- A content provider in Android shares data between applications. Each application usually runs in its own process.
- By default, applications can't access the data and files of other applications.
- You can make preferences and files available across application boundaries with the correct permissions and if each application knows the context and path.
- This solution applies only to related applications that already know details about one another.
- In contrast, with a content provider you can publish and expose a particular data type for other applications to query, add, update, and delete, and those applications don't need to have any prior knowledge of paths, resources, or who provides the content.
- The canonical content provider in Android is the contacts list, which provides names, addresses, and phone numbers.
- You can access this data from any application by using the correct URI and a series of methods provided by the Activity and ContentResolver classes to retrieve and store data.

## 4.6.2 Deleting Database Records

- The data modification clauses in SQLite are INSERT, UPDATE, and DELETE statements. It is used for inserting new rows, updating existing values, or deleting rows from the database.

- You can remove records from the database using the remove() method.The remove() method takes few arguments.
- Passing null to the WHERE clause deletes all records within the table. For example, this function call deletes all records within the table called tbl_authors :

```
mDatabase.delete("tbl_authors", null, null);
```

- Most of the time, though,we want to delete individual records by their unique identifiers.
- The following function takes a parameter bookId and deletes the record corresponding to that unique id (primary key) within the table called tbl_books :

```
public void deleteBook(Integer bookId) {
mDatabase.delete("tbl_books", "id=?",
new String[] { bookId.toString() });
}
```

- You need not use the primary key (id) to delete records; the WHERE clause is entirely up to you. For instance, the following function deletes all book records in the table.

```
tbl_books for a given author by the author's unique id:
public void deleteBooksByAuthor(Integer authorID) {
int numBooksDeleted = mDatabase.delete("tbl_books", "authorid=?",
new String[] { authorID.toString() });
}
```

## 4.7 Handling Database in Android

- In the same way that you retrieve data from a provider, you also use the interaction between a provider client and the provider's ContentProvider to modify data.
- You call a method of ContentResolver with arguments that are passed to the corresponding method of ContentProvider.
- The provider and provider client automatically handle security and inter-process communication.

### 4.7.1 Inserting Data

- To insert data into a provider, you call the ContentResolver.insert() method. This method inserts a new row into the provider and returns a content URI for that row.

This snippet shows how to insert a new word into the User Dictionary Provider :

```
// Defines a new Uri object that receives the result of the insertion
lateinit var newUri: Uri
   ...
// Defines an object to contain the new values to insert
val newValues = ContentValues().apply {
    /*
     * Sets the values of each column and inserts the word. The arguments to the
"put"
     * method are "column name" and "value"
     */
    put(UserDictionary.Words.APP_ID, "example.user")
    put(UserDictionary.Words.LOCALE, "en_US")
    put(UserDictionary.Words.WORD, "insert")
    put(UserDictionary.Words.FREQUENCY, "100")

}

newUri = contentResolver.insert(
        UserDictionary.Words.CONTENT_URI,   // the user dictionary content URI
        newValues                          // the values to insert
)
```

- The data for the new row goes into a single ContentValues object, which is similar in form to a one-row cursor. The columns in this object don't need to have the same data type, and if you don't want to specify a value at all, you can set a column to null using *ContentValues.putNull()*.
- The snippet doesn't add the _ID column, because this column is maintained automatically. The provider assigns a unique value of _ID to every row that is added. Providers usually use this value as the table's primary key.
- The content URI returned in newUri identifies the newly-added row, with the following format :

```
content://user_dictionary/words/<id_value>
```

- The <id_value> is the contents of _ID for the new row. Most providers can detect this form of content URI automatically and then perform the requested operation on that particular row.
- To get the value of _ID from the returned Uri, call ContentUris.parseId().

### 4.7.2 Updating Data

- To update a row, you use a ContentValues object with the updated values just as you do with an insertion, and selection criteria just as you do with a query. The client method you use is ContentResolver.update().

- You only need to add values to the ContentValues object for columns you're updating. If you want to clear the contents of a column, set the value to null.
- The following snippet changes all the rows whose locale has the language "en" to a have a locale of null. The return value is the number of rows that were updated :

```
// Defines an object to contain the updated values
val updateValues = ContentValues().apply {
    /*
     * Sets the updated value and updates the selected words.
     */
    putNull(UserDictionary.Words.LOCALE)
}

// Defines selection criteria for the rows you want to update
val selectionClause: String = UserDictionary.Words.LOCALE + "LIKE ?"
val selectionArgs: Array<String> = arrayOf("en_%")

// Defines a variable to contain the number of updated rows
var rowsUpdated: Int = 0

...

rowsUpdated = contentResolver.update(
        UserDictionary.Words.CONTENT_URI,   // the user dictionary content URI
        updateValues,                       // the columns to update
        selectionClause,                    // the column to select on
        selectionArgs                       // the value to compare to
)
```

- You should also sanitize user input when you call ContentResolver.update().

### 4.7.3 Deleting data

- Deleting rows is similar to retrieving row data: you specify selection criteria for the rows you want to delete and the client method returns the number of deleted rows.
- The following snippet deletes rows whose appid matches "user". The method returns the number of deleted rows.

```
// Defines selection criteria for the rows you want to delete
val selectionClause = "${UserDictionary.Words.LOCALE} LIKE ?"
val selectionArgs: Array<String> = arrayOf("user")

// Defines a variable to contain the number of rows deleted
var rowsDeleted: Int = 0
```

```
...

// Deletes the words that match the selection criteria
rowsDeleted = contentResolver.delete(
        UserDictionary.Words.CONTENT_URI,   // the user dictionary content URI
        selectionClause,                    // the column to select on
        selectionArgs                       // the value to compare to
)
```

- You should also sanitize user input when you call ContentResolver.delete().

## 4.8 Short Questions and Answers

**Q.1 How many ways data stored in Android ?**

**Ans. :**

1. Shared Preferences - Store private primitive data in key-value pairs. This sometimes gets limited as it offers only key value pairs. You cannot save your own java types.
2. Internal Storage - Store private data on the device memory
3. External Storage - Store public data on the shared external storage
4. SQLite Databases - Store structured data in a private database. You can define many number of tables and can store data like other RDBMS.

**Q.2 What is SQLite database ?**

**Ans. :** SQLite database : A lightweight SQL database implementation that's available across various platforms (including Android, iPhone, Windows, Linux, and Mac) and fully supported by Android. You can create tables and perform SQL queries against the tables accordingly. You implement an SQLite database in this chapter to handle the persistence of the tasks in the Task Reminder application.

**Q.3 What is SQLite ? How does it differ from client-server database management systems ?**

**Ans. :** SQLite is the open-source relational database of choice for Android applications. The SQLite engine is serverless, transactional, and self-contained. Instead of the typical client-server relationship of most database management systems, the SQLite engine is integrally linked with the application. The library can also be called dynamically, and makes use of simple function calls that reduce latency in database access.

**Q.4 What composes a typical Android application project ?**

**Ans. :** A project under Android development, upon compilation, becomes an .apk file. This apk file format is actually made up of the AndroidManifest.xml file, application code, resource files, and other related files.

**Q.5 What is the importance of settings permissions in app development ?**

**Ans. :** Permissions allow certain restrictions to be imposed primarily to protect data and code. Without these, codes could be compromised, resulting to defects in functionality.

## 4.9 Multiple Choice Questions

**Q.1** *What is a thread in android ?*

a Same as services
b Background activity
c Broadcast Receiver
d Independent dis-patchable unit is called a thread

**Q.2** *Persist data can be stored in Android through* _____.

a shared preferences
b internal/external storage
c SQlite
d network servers.
e all of above

**Q.3** *What is last known location in android ?*

a To find the last location of a phone
b To find known location of a phone
c To find the last known location of a phone.
d None of the above

**Q.4** *How to upgrade SQlite the database from a lower version to higher version in android SQlite ?*

a Using helper Class
b Using cursor
c Using intent
d None of the above

**Q.5** *Shared preferences stores the data in which format?*

a TXT
b XML
c DOC
d None of the above.

**Q.6** *The first step when working with SQLite is to create a class that inherits from a helper class, what is it ?*

a SQLiteOpenHelper class
b SQLiteHelper class
c SQLiteDatabaseHelper class
d SQLiteDatabase class

**Q.7** *Name the method that enables you to obtain the path of the external storage of an Android device.*

a getExternalStorageDirectory()

b getPathExternalStorageDirectory()

c getExternalStorageFile()

d getExternalStoragePath()

**Q.8** *"There are two statements: Statement A : Using Shared Preferences, you can store private primitive data only. Statement B : Using External Storage option, you can store public data on the shared external storage. Which of them are correct ?"*

a Statement A is False, and Statement B is False

b Statement A is True, and Statement B is True

c Statement A is True, and Statement B is False

d Statement A is False, and Statement B is True

**Q.9** *In Android, by default, SQLite database save data in _____ .*

a memory

b external storage

c internal storage

d on the cloud

**Q.10** *There are five different methods to store persistent data. They are: Shared Preference, Internal Storage, External Storage, Network and _____ .*

a core data

b SQLite database

c web service

**Answer Keys for Multiple Choice Questions**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Q.1** | **d** | **Q.2** | **e** | **Q.3** | c | **Q.4** | a | **Q.5** | b |
| **Q.6** | **a** | **Q.7** | **a** | **Q.8** | b | **Q.9** | c | **Q.10** | b |

❑❑❑

# 5 Location Services and Maps

## *Syllabus*

*Working with Location Services and Maps, Working with Google Maps, Working with Geocoding and Reverse Geocoding.*

## *Contents*

# 5.1 Location Services

## 5.1.1 Current Location

- Android location API can be used to track your mobile current location and show in the app.
- To get the current location, create a location client which is LocationClient object, connect it to Location Services using connect() method, and then call its getLastLocation() method.
- This method returns the most recent location in the form of Location object that contains latitude and longitude coordinates and other information as explained above.
- To have location based functionality in your activity, you will have to implement two interfaces -

GooglePlayServicesClient.ConnectionCallbacks

GooglePlayServicesClient.OnConnectionFailedListener

- These interfaces provide following important callback method-

| Sr.No. | Callback Methods & Description |
|---|---|
| 1 | **abstract** void onConnected(Bundle connectionHint)<br>This callback method is called when location service is connected to the location client successfully. You will use connect() method to connect to the location client. |
| 2 | **abstract void onDisconnected()**<br>This callback method is called when the client is disconnected. You will use **disconnect()** method to disconnect from the location client. |
| 3 | **abstract** void onConnectionFailed(ConnectionResult result)<br>This callback method is called when there was an error connecting the client to the service. |

## 5.1.2 Location Listener

- The LocationListener interface, which is part of the Android Locations API is used for receiving notifications from the LocationManager when the location has changed.
- The LocationManager class provides access to the systems location services.
- The LocationListener class needs to implement the following methods.

**1. onLocationChanged(Location location) :** Called when the location has changed.

2. **onProviderDisabled(String provider) :** Called when the provider is disabled by the user.
3. **onProviderEnabled(String provider) :** Called when the provider is enabled by the user.
4. **onStatusChanged(String provider, int status, Bundle extras) :** Called when the provider status changes.

The **android.location** has two means of acquiring location data :

1. **LocationManager.GPS_PROVIDER :** Determines location using satellites. Depending on the conditions, this provider may take a while to return a location fix
2. **LocationManager.NETWORK_PROVIDER :** Determines location based on the availability of nearby cell towers and WiFi access points. This is faster than GPS_PROVIDER.

## 5.2 Working with Google Map

- Google maps are the most popular method of displaying maps.
- Android allows us to integrate Google Maps in application. For this Google provides us a library via Google Play Services for using maps. In order to use the Google Maps API. Register application on the Google Developer Console and enable the API.
- The Google Maps Android API consists of a core set of classes that combine to provide mapping capabilities in Android applications.
- To use the com.google.android.maps package on the Android platform and support all the features related to a MapView, you must use a MapActivity.

### 5.2.1 APIs in Google Maps

- There are several APIs available in Google Maps: SPONSORED SEARCHES interview questions and answers process mapping google map satellite gps location map

**Web APIs**

- Google Maps JavaScript API
- Google Street View Image API
- Google Static Maps API
- Google Maps Embed API

**Web Service APIs**

- Google Maps Directions API
- Google Maps Elevation API
- Google Maps Distance Matrix API
- Google Maps Geocoding API
- Google Maps Geolocation API
- Google Maps Time Zone API
- Google Maps Roads API
- Google Places API Web Service

**Mobile APIs**

- Google Maps Android API
- Google Places API for Android
- Google Maps SDK for iOS
- Google Places API for iOS

## 5.2.2 Steps for Getting the Google Maps API Key

- An API key is needed to access the Google Maps servers.

**Step 1 :** Open Google developer console and signin with your gmail account.

**Step 2 :** Now create new project. You can create new project by clicking on the Create Project button and give name to your project.

**Step 3 :** Now click on APIs and Services and open Dashboard from it.

**Step 4 :** In this open Enable APIS AND SERICES.

**Step 5 :** Now open Google Map Android API.

**Step 6 :** Now enable the Google Maps Android API

**Step 7 :** Now go to Credentials

**Step 8 :** Here click on Create credentials and choose API key

**Step 9 :** Now API your API key will be generated. Copy it and save it somewhere as we will need it when implementing Google Map in our Android project.

## 5.2.3 Elements of Google Map

- Elements of a map are as follows :

**1. GoogleMap -**

- This class is responsible for downloading and displaying map tiles and for displaying and responding to map controls.
- The GoogleMap object is not created directly by the application but is created when MapView or MapFragment instances are created.
- A reference to the GoogleMap object can be obtained within application code via a call to the getMap() method of a MapView, MapFragment or SupportMapFragment instance.

**2. MapView -**

- A subclass of the View class, this class provides the view canvas onto which the map is drawn by the GoogleMap object, allowing a map to be placed in the user interface layout of an activity.

**3. SupportMapFragment -**

- A subclass of the Fragment class, this class allows a map to be placed within a Fragment in an Android layout.

**4. Shapes -**

- The drawing of lines and shapes on a map is achieved through the use of the Polyline, Polygon and Circle classes.

**5. My Location Layer -**

- The My Location Layer displays a button on the map which, when selected by the user, centers the map on the user's current geographical location.
- If the user is stationary, this location is represented on the map by a blue marker.
- If the user is in motion the location is represented by a chevron indicating the user's direction of travel.

**6. Marker -**

- The purpose of the Marker class is to allow locations to be marked on a map. Markers are added to a map by obtaining a reference to the GoogleMap object associated with a map and then making a call to the addMarker() method of that object instance.
- The position of a marker is defined via Longitude and Latitude.
- Markers can be configured in a number of ways, including specifying a title, text and an icon.
- Markers may also be made to be "draggable", allowing the user to move the marker to different positions on a map.

**7. UiSettings -**

- The UiSettings class provides a level of control from within an application of which user interface controls appear on a map.

## 5.2.4 Adding a Map to an Application

- The simplest way to add a map to an application is to specify it in the user interface layout XML file for an activity.
- The following example layout file shows the SupportMapFragment instance added to the activity_map_demo.xml file created by Android Studio :

```
<fragment 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"
  android:id="@+id/map"
  tools:context=".MapDemoActivity"
  android:name="com.google.android.gms.maps.SupportMapFragment"/>
```

## 5.2.5 Types of Google Map

- There are four types of Google maps.
  1. Road map - It is default map type. It displays the default road map view.
  2. Satellite - It display Google Earth satellite images.
  3. Hybrid - It display the mixture of normal and satellite views.
  4. Terrain - It display a physical map based on territory information
- The type of map displayed can be modified dynamically by making a call to the setMapType() method of the corresponding GoogleMap object, passing through one of the following values :
  1. **GoogleMap.MAP_TYPE_NONE -** An empty grid with no mapping tiles displayed.
  2. **GoogleMap.MAP_TYPE_NORMAL -** The standard view consisting of the classic road map.
  3. **GoogleMap.MAP_TYPE_SATELLITE -** Displays the satellite imagery of the map region.
  4. **GoogleMap.MAP_TYPE_HYBRID -** Displays satellite imagery with the road map superimposed.
  5. **GoogleMap.MAP_TYPE_TERRAIN -** Displays topographical information such as contour lines and colors.

### 5.2.6 Displaying the User's Current Location

- The user's current location may be displayed on the map by obtaining a reference to the GoogleMap object associated with the displayed map and calling the setMyLocationEnabled() method of that instance, passing through a value of true.
- When the map is ready to display, the onMapReady() method of the activity is called. This method will also be called when the map is refreshed within the onRequestPermissionsResult() method.

### 5.2.7 User Interface (UI) Controls of Google Maps

- Google maps provides various user-friendly controls to user to interact with map.
- We can add, customize and disable these controls. Some controls are default provided like :
  1. **Zoom -** It display the "+" and "-" button to changing the zoom level of the map. This control appear in bottom right corner of the map.
  2. **Pan -** Pan Control is used for panning the map.
  3. **Map type -** It provides map type options such as Satellite, Road map and Terrain. It appears on the top right corner of the map.
  4. **Street view -** It contains Pegman icon which can be used to get the street view of a particular location, whenever it is dragged.

### 5.2.8 Features of Google Maps

- Following are the features provided by Google Maps.

1. It searches places and route directions.
2. It provides distance information.
3. It helps to find traffic details and navigation.
4. It displays street views
5. It receives verbal instructions.
6. It provides location sharing and location editing.

### 5.2.9 Difference between Google Maps and Google Earth

- Google Maps contains all the information about navigation. It provides lightweight point-to-point navigation with only a small hint of the satellite.
- Google Earth displays the full 3D view of satellite data and only a small subset of location information. Google Erath does not provide point-to-point navigation.

## 5.3 Geocoding

- Geocoding is the process of transforming a description of a location-such as a pair of coordinates, an address, or a name of a place-to a location on the earth's surface.
- Geocoding can best be described as the process of converting a textual based geographical location (such as a street address) into geographical coordinates expressed in terms of longitude and latitude.
- Geocoding can done by entering one location description at a time or by providing many of them at once in a table. The resulting locations are output as geographic features with attributes, which can be used for mapping or spatial analysis.
- Geocoding can be achieved using the Android Geocoder class. An instance of the Geocoder class can, for example, be passed a string representing a location such as a city name, street address or airport code.
- The geocoder will attempt to find a match for the location and return a list of Address objects that potentially match the location string, ranked in order with the closest match at position 0 in the list.
- A variety of information can then be extracted from the Address objects, including the longitude and latitude of the potential matches.

### 5.3.1 Use of Geolocations

- Geolocation can be used to determine time zone and exact positioning coordinates, such as for tracking wildlife or cargo shipments.
- Both mobile and desktop devices can use geolocation. There are apps which use location once, and there are apps which continuously track your location as well.
- Some famous apps using Geolocation are -

1. Uber / Lyft ↔ Cab booking
2. Google Maps (of course) ↔ Map services
3. Swiggy / Zomato ↔ Food delivery
4. Fitbit ↔ Fitness app
5. Instagram / Facebook ↔ For tagging photos

### 5.3.2 Effectiveness of Geocoder

- A system that provides geocoding is called a geocoding service (or just geocoder).
- To assess the effectiveness of a geocoder, you need to use two basic parameters :

1. **The size of the database.** The fuller the base, the more accurate and detailed the response to the request will be. With a large database, the address can be specified up to the house even in a small town.
2. **Geocoding speed.** The speed of a geocoder is determined by the number of requests processed per second. The more requests the geocoder is able to handle, the more users the system can service.

## 5.4 Reverse Geocoding Revgeocode

- **Reverse geocoding** is the process of transforming a (latitude, longitude) coordinate into a (partial) address.
- As reverse geocoding converts geographic latitude longitude coordinates to address so the amount of detail in a reverse geocoded location description may vary, for example, one might contain the full street address of the closest building, while another might contain only a city name and postal code.
- A reverse geocoding Application Programming Interface (API) is a web service. It enables developers to add to their mapping applications reverse geocoding features.
- The end-user will be able to search a latitude and longitude location and obtain the exact street address or largest or nearest city.
- An API for reverse geocoding is important because the latitude and longitude of an area remain the same. This makes it a reliable way to locate streets, landmarks, and other places.

### 5.4.1 Difference between Forward and Reverse Geocoding

- Forward geocoding is the process of looking up a plain-text address or place name (e.g. Girnar Mountain), whereas reverse geocoding is performed by passing latitude and longitude values of a desired location to the API.
- If successful, both types of geocoding return an extensive array of location-related data as well as multiple potential results along with confidence scores.

## 5.5 Short Questions and Answers

**Q.1** *What is alarm manager broadcast receiver service ?*

**Ans. :** An Alarm manager is used to trigger some code at a specific time. It uses the Android SDK's alarm service and runs independently of the application's lifecycle.

To start an Alarm manager you need to first get the instance from the system. Then pass the PendingIntent which would get executed at a future time that you specify.

**Q.2** *Explain types of alarm in android.*

**Ans. :** There are different types of Alarm manager that can be invoked :

1. setInExactAndRepeating - This doesn't trigger the alarm at the exact time.
2. setExact - Setting an alarm manager using this ensures that the OS triggers the alarm at almost the exact time.
3. setExactAndAllowWhileIdle - This method came up with Android M. These types of alarms are allowed to get executed even in low power modes.

**Q.3** *How do you pass the data to sub-activities android ?*

**Ans. :** Using with Bundle, we can pass the data to sub activities.

```
Bundle bun = new Bundle();
bun.putString("EMAIL", "contact@technical.com");
```

**Q.4** *What is sleep mode in android ?*

**Ans. :** Sleep mode means that the CPU will be sleeping and it doesn't accept any commands from an Android device except Radio interface layer and alarm.

**Q.5** *What is google maps ?*

**Ans. :** Google maps is a web-based mapping service that is designed and developed by Google. It contains geographical data and provides routes and information to the client.

**Q.6** *What is google maps API ?*

**Ans. :** • The Google maps API is a robust tool that allows users to create a custom map or searchable map.

- It also helps users to add several other features with the created maps. For exampleCheck-in functions, data synchronization, plan routes, mark or share location, etc.

**Q.7** *What are the properties of info window ?*

**Ans. :** Following are the properties of Info window :

1. **Content -** By using this option we can pass our content in string format.
2. **Position -** By using this option we can choose the position of the Info window.
3. **MaxWidth -** It specifies the maximum width of the info window in pixels

**Q.8** *What is the difference between geocoding and reverse geocoding ?*

**Ans. :** Geocoding starts with an address and gives you back a latitude/longitude point (a geocode). Reverse geocoding starts with a latitude/longitude point, and gives you back an address.

## 5.6 Multiple Choice Questions

**Q.1** *What is a GCM in Android ?*

a Goggle Could Messaging for chrome

b Goggle Count Messaging

c Goggle Message pack

d None of the above

**Q.2** *What is transient data in Android ?*

a Permanent data

b Secure data

c Temporary data

d Logical data

**Q.3** *Which element is used to display Google map in your UI ?*

a View

b Map

c MapView

d None of the above

**Q.4** *For using Google map which permission you will specify in the AndroidManifest.xml ?*

a USEMAP

b USE_GOOGLE_MAP

c INTERNET

d None of the above.

**Q.5** *How will you add marker on map ?*

a Directly use draw method.

b You cannot add marker on the map.

c Implement an Overlay class and override the draw() method.

d None of the above.

**Q.6** *What are the different location provider are available that you can use to obtain location data ?*

a LocationManager.GPS_PROVIDER

b LocationManager.NETWORK_PROVIDER

c LocationManager.PASSIVE_PROVIDER

d All of the above.

**Q.7** *The difference between android API and google API is ____.*

a The google API includes Google Maps and other Google-specific libraries. The Android one only includes core Android libraries

b The google API one only includes core android libraries. The Android includes Google Maps and other Google-specific libraries

c None of the above

d All of the above

**Q.8** *Which of the following applications are processed locally on the phone ?*

a Google earth

b Google maps

c Google voice

d None of the mentioned

**Answer Key for Multiple Choice Questions**

| | | | |
|---|---|---|---|
| **Q.1** | a | **Q.2** | d |
| **Q.3** | c | **Q.4** | c |
| **Q.5** | c | **Q.6** | d |
| **Q.7** | a | **Q.8** | d |

❑❑❑

# 6 Graphics and Animation

## Syllabus

*Working with Graphics, Using the Drawable Object, Using the Shape Drawable object, Hardware Acceleration, Working with Animation.*

## Contents

## 6.1 Working with Graphics

- Graphics are created by the developer and added to the application. They can be created from results from various operations such as querying, identifying, geoprocessing, geocoding or routing.
- Graphics can also be created from external data sources, but if you want to persist the data in a map or scene then you must use features. Graphics can also be created by the developer as a result of a map click or touch.
- The Android SDK provides a huge set of APIs that allows you to create 2-D and 3-D graphics.
- Android support 2D graphics via its own library in packages android.graphics.drawable and android.view.animation. Take note that Android does not support JDK's AWT and Swing.
- The android.graphics.drawable and android.view.animation packages are where you'll find the common classes used for drawing and animating in two-dimensions.

### 6.1.1 Uses of Graphics

- The following are some common uses for graphics :
  1) Display text on top of a map or scene
  2) Highlight a section of the map or scene by overlaying a polygon graphic
  3) Display results from spatial analysis, such as buffer polygons created around features
  4) Display a route between two locations
  5) Display geometry drawn interactively by the app user
  6) Animate data items that change quickly, such as moving objects

## 6.2 Using Drawable Object

- Android provides a set of View widgets that provide general functionality for a wide array of user interfaces.
- You can also extend these widgets to modify the way they look or behave.
- Custom 2D rendering can be done using the various drawing methods contained in the Canvas class or create Drawable objects for things such as textured buttons or frame-by-frame animations.
- A drawable resource is a general concept for a graphic that can be drawn to the screen.

- When you need to display static images in your app, you can use the Drawable class and its subclasses to draw shapes and images.
- Drawables are used to define shapes, colors, borders, gradients, etc. which can then be applied to views within an Activity.
- Every Drawable is stored as individual files in one of the res/drawable folders.
- There are three ways to define and instantiate a drawable :
  1. using an image saved in your project resources;
  2. using an XML file that defines the Drawable properties;
  3. or using the normal class constructors.

## 6.3 Using the Shape Drawable Object

- A drawable object is that draws primitive shapes.
- When you want to dynamically draw some two-dimensional graphics, a shape drawable object will probably suit your needs. With a *Shape Drawable,* you can programmatically draw primitive shapes and style them in any way imaginable.
- A *Shape Drawable* is an extension of Drawable, so you can use one where ever a Drawable is expected - perhaps for the background of a view, set with *setBackgroundDrawable()*. Of course, you can also draw your shape as its own custom View, to be added to your layout however you please. Because the *Shape Drawable* has its own draw() method, you can create a subclass of View that draws the *Shape Drawable* during *theView.onDraw()* method.
- A Shape Drawable takes a Shape object and manages its presence on the screen. If no shape is given, then the *Shape Drawable* will default to a rectangle shape.
- Shape *Drawables* are XML files which allow to define a geometric object with colors, borders and gradients.
- The advantage of using XML shape *Drawables* is that they automatically adjust to the correct size.
- A *Shape Drawable* is an extension of *Drawable,* so you can use one where ever a Drawable is expected - perhaps for the background of a view, set with *setBackgroundDrawable()*.

## 6.4 Hardware Acceleration

- Android 3.0 (API Level 11) introduced a new rendering pipeline to allow applications to benefit from hardware-accelerated 2-D graphics.

- You can use hardware acceleration if you have complex computations like scaling, rotating, translation of images.
- Hardware Acceleration is used to improve the performance and aesthetics of your activities and views.
- All the SDK Views, layouts, and effects support hardware acceleration.
- Android gives you the option to enable or disable hardware acceleration at multiple level
  1) Application level
  2) Activity level
  3) Window level
  4) View level

**1. Application level**

- In your Android manifest file, add the following attribute to the <application> tag to enable hardware acceleration for your entire application:

```
<application android:hardwareAccelerated="true" ...>
```

**2. Activity level**

- If your application does not behave properly with hardware acceleration turned on globally, you can control it for individual activities as well.
- To enable or disable hardware acceleration at the activity level, you can use the android:hardwareAccelerated attribute for the <activity> element.
- The following example enables hardware acceleration for the entire application but disables it for one activity:

```
<application android:hardwareAccelerated="true">
   <activity ... />
   <activity android:hardwareAccelerated="false" />
</application>
```

**3. Window level**

- If you need even more fine-grained control, you can enable hardware acceleration for a given window with the following code:

```
getWindow().setFlags(
   WindowManager.LayoutParams.FLAG_HARDWARE_ACCELERATED,
   WindowManager.LayoutParams.FLAG_HARDWARE_ACCELERATED);
```

**4. View level**

- You can disable hardware acceleration for an individual view at runtime with the following code:

```
myView.setLayerType(View.LAYER_TYPE_SOFTWARE, null);
```

## 6.5 Working with Animation

- Animations can add visual cues that notify users about what's going on in your app.
- They are especially useful when the UI changes state, such as when new content loads or new actions become available.
- Animations also add a polished look to your app, which gives it a higher quality look and feel.
- Android Supports three types of Animation :
  1) **Property animation**
     a) Property animators were introduced in Android 3.0 (API level 11). It is a powerful framework that can be used to animate almost anything.
     b) You can use a property animator to animate almost any property on a target object.
     c) Property animators are extremely useful and are used extensively for animating fragments in android.
  2) **View animations**
     a) Animations that can be applied for views to rotate, move and stretch.
  3) **Frame animations**
     a) Frame-by-frame animations produce a sequence of Drawables, each of which is displayed for a specified duration.

### 6.5.1 Clock for Animated Graphics

- The methods (functions) provided by the android class SystemClock are used to create animated graphics.
- The class consists of core timekeeping facilities. To use the methods. we have to import the class by adding the header statement, **import android.os.SystemClock;**
- There are three clocks we can use to keep time :
  1. **currentTimeMillis()** gives the current time and date expressed in milliseconds since the epoch. This clock can be set by the user or the phone network.
  2. **uptimeMillis()** gives the active time lapse in milliseconds since the system was booted. This clock stops when the process is in a blocked or a sleep state such as waiting for an I/O event or executing Thread.sleep().
  3. **elapsedRealtime()** gives the counts in milliseconds since the system was booted, including the time when the process is blocked or in a sleep state.

### 6.5.2 Controlling Timing Events

- There are several ways to control the timing events in an animation process :
  1. **Thread.sleep**(millis) and **Object.wait**(millis) are standard blocking functions that can be used to generate desired time delays. When these functions are executed, the **uptimeMillis()** clock stops. The thread can be woken up by the function **Thread.interrupt()**.
  2. **SystemClock.sleep**(millis) is a utility function very similar to **Thread.sleep**(millis), except that it ignores **InterruptedException.**
  3. We can use the Handler class to schedule asynchronous callbacks at an absolute or relative time. A handler object uses the **uptimeMillis()** clock to keep time. It requires an event loop to wait for an event to happen.
  4. We can use the AlarmManager class to access the system alarm services such as triggering one-time or recurring events when the thread is in a blocked state.

## 6.6 Alarms in Android

- In Android, alarms allow you to schedule your application to run at some point in the future.
- Alarms can be used for a wide range of applications, from notifying a user of an appointment to something more sophisticated, such as having an application start, checking for software updates, and then shutting down.
- Alarms (based on the AlarmManager class) give you a way to perform time-based operations outside the lifetime of your application. For example, you could use an alarm to initiate a long-running operation, such as starting a service once a day to download a weather forecast.
- An alarm works by registering an Intent with the alarm; at the scheduled time, the alarm broadcasts the Intent.
- Android automatically starts the targeted application, even if the Android handset is asleep.
- Android manages all alarms somewhat as it manages the NotificationManager-via an AlarmManager class.

**Characteristics of alarms :**

- The characteristics of alarms are :
  1. They let you fire Intents at set times and/or intervals.
  2. You can use them in conjunction with broadcast receivers to start services and perform other operations.

3. They operate outside of your application, so you can use them to trigger events or actions even when your app is not running, and even if the device itself is asleep.
4. They help you to minimize your app's resource requirements. You can schedule operations without relying on timers or continuously running background services.

They let you fire Intents at set times and/or intervals.

## 6.7 Download Manager

- The download manager is a system service that handles long-running HTTP downloads.
- Clients may request that a URI be downloaded to a particular destination file.
- The download manager was introduced in Android 2.3 (API level 9).
- One big advantage of Android Download Manager is that it optimizes the handling of long-running downloads in the background.
- The download manager handles HTTP connections, monitors connectivity changes, reboots, and ensures each download completes successfully.
- If you want to download large files/streaming, then you can't use Retrofit or Volley, both recommend using DownloadManager instead, which supports resuming and progress notifications.download manager will conduct the download in the background, taking care of HTTP interactions and retrying downloads after failures or across connectivity changes and system reboots.
- Apps that request downloads through this API should register a broadcast receiver for ACTION_NOTIFICATION_CLICKED to appropriately handle when the user clicks on a running download in a notification or from the downloads UI.
- The application must have the Manifest.permission.INTERNET permission to use this class.

## 6.8 Multiple Choice Questions

**Q.1** *Resource manager is used for ?*

a Controls all aspects of the application lifecycle and activity stack.

b Allows applications to publish and share data with other applications.

c Provides access to non-code embedded resources such as strings, color settings and user interface layouts.

d Allows applications to display alerts and notifications to the user.

**Q.2** *What is a correct statement about an XML layout file?*

a A layout PNG image file

b A file used to draw the content of an Activity

c A file that contains all application permission information

d A file that contains a single activity widget.

**Q.3** *Which is the most popular software used for creating 2D animation for use in web pages ?*

a Corel draw b Flash

c Banner creator d Maya

**Q.4** *What is the purpose of the image switcher ?*

a The image switcher enables images to be displayed with animation.

b The image switcher displayed images without animation.

c An image switcher allows you to add some transitions on the images.

d Option A and C are correct.

**Answer Keys for Multiple Choice Questions**

| | | | | | |
|---|---|---|---|---|---|
| Q.1 | c | Q.2 | b | Q.3 | b |
| Q.4 | d | | | | |

❑❑❑

# 7 Audio, Video and Camera Use

## *Syllabus*

*Media Player, Recording and Playing sound, Creating a sound pool, Using Camera, Recording Video*

## *Contents*

## 7.1 Media Player

- Media player is a part of the Android multimedia framework that plays audio or video from the resource directory and gallery. It also streams music or video from a URL.
- By using media player class audio or video files from application (raw) resources can be accessed.
- Standalone files in file system or from a data stream arriving over a network connection and play audio or video files with the multiple playback options such as play, pause, forward, backward, etc.

### 7.1.1 User Media Player

- The media player class primarily handles the playback of audio and video within android application.
- You can play the media stored in application resources, local files, Content Providers or streamed from a network URL using the media player.
- Media player's management audio and video are handled in the form of state machines. Following are the transitions in the state machine
  1) Initialize the media player with media to play.
  2) Prepare the media player for the playback.
  3) Start the playback.
  4) Pause or stop the playback prior to its completing.
  5) The playback is complete.
- To play a media resource, you need to create an instance of MediaPlayer class, then initialize it with media source and prepare it for playback.

## 7.2 Recording and Playing Sound

- The audio track and audio record classes let you record audio directly from the audio output hardware.
- The audio record class is used to record audio directly from the hardware buffers.
- The audio track class is used to directly play the raw audio into the hardware buffers.

### 7.2.1 Media Formats

- Depending on the codecs used, Android phones can support and play several audio file types.

## 1. Audio Formats

- The Table 7.2.1 describes the media format for audio and codec support built into the Android platform.

| Format Codec | Encoder | Decoder | Details | Supported File Types (s) /Container Formats |
|---|---|---|---|---|
| AAC LC | • | • | Support for mono /stereo/5.0/5.1 content with standard sampling rates from 8 to 48 kHz. | • 3GPP (.3gp)<br>• MPEG-4 (mp4 . m4a)<br>• ADTS raw AAC (.aac dedoce in Andriod 3.1+ encode in Android 4.0 +, ADIF nor supported)<br>• MPEG-TS (.ts, not seekable, Android 3.0+) |
| HE- AACv1 (AAC+) | • (Andriod 4.1 +) | • | | |
| HE- AACv2 (enhanced AAC+) | | • | Support for stereo/5.0/5.1 content with standard sampling rates from 8 to 48 kHz. | |
| AAC ELD (enhanced low delay AAC) | • (Andriod 4.1 +) | • (Andriod 4.1 +) | Support for mono /stereo content with standard sampling rates from 16 to 48 kHz. | |
| AMR-NB | • | • | 4.75 to 12.2 kbps sampled @ 8 kHz. | 3GPP (.3gp) |
| AMB-WB | • | • | 9 rates from 6.60 kbit/s to 23.85 kbit/s kbit @ 16 kHz. | 3GPP (.3gp) |
| FLAC | | • (Android 3.1+) | Mono/Stereo (no multichaneel). Sample rates up to 48 kHz (but up to 44.1 kHz is recommenced on devices with 41.1 kHz output, as the 48 to 44.1 kHz downsampler does not include a low-pass filter). 16-bit recommended ; no dither applied for 24-bit. | FLAC (.flac) only |
| MP3 | | • | Mono /Stereo 8-320 kbps constant (CBR) or variable bit rate (VBR) | MP3 (.mp3) |
| MIDI | | • | MIDI Type 0 and 1. DLS Verserion 1 and 2. XMF and Mobile. XMF Support for ringtone forms RTTTL/RTX, OTA, and iMelody | • Type 0 and 1 (.mid, .xmf, .mxmf)<br>• RTTL/RTX (.rtttl. .rtx)<br>• OTA (.ota)<br>• iMelody (.imy) |

| | | | | |
|---|---|---|---|---|
| Vorbis | | • | | • Ogg (.ogg)<br>• Matroska (.mkv, Android 4.0+) |
| PCM/ WAVE | • (Anddrioid 4.1 +) | • | 8-and 16 bit linear PCM (rates up to limit to hardware). Sampling rates for raw PCM recordings at 8000, 16000 and 44100 Hz. | Wave (.wav) |

**Table 7.2.1**

### 2. Video Formats

- Depending on the codec used to compress the video container format, it may still not play on your Android phone even though it generally is supported.
- To watch your videos still, you can convert the video into another format or choose an Android compatible conversion right away.
- The codecs supported by Android phones are H.263, H.264, MPEG-4, and V8. This leads to the following video container formats to be supported by Android devices :
  - 3GP
  - MKV
  - MP4
  - TS
  - WEBM

## 7.2.2 Playing Audio

- To play local audio in the supported formats, first we should put the local audio file into the res/raw folder. For example put this sample_audio.mp3 into res/raw/sample_audio.mp3.
- Now we can use the MediaPlayer in order to playback any local files :

```
MediaPlayer mediaPlayer = MediaPlayer.create(this, R.raw.sample_audio);
mediaPlayer.start();
```

- On call to start() method, the music will start playing from the beginning. If this method is called again after the pause() method, the music would start playing from where it left off and not from the beginning. To play back audio from a local file path, simply do :

```
Uri myUri = "....." ; // initialize Uri here
MediaPlayer mediaPlayer = new MediaPlayer () ;
mediaPlayer . setAudioStreamType(AudioManager . STREAM_MUSIC);
mediaPlayer . setDataSource(getApplicationContext( ), myUri);
// or just mediaPlayer . setDataSource(mFileName) ;
mediaPlayer . prepare () ; // must call prepare first
mediaPlayer . start (); // then start
```

### 7.2.3 Playing Video

- In Android, VideoView is used to display a video file.
- It can load images from various sources (such as content providers or resources) taking care of computing its measurement from the video so that it can be used for any layout manager, providing display options such as scaling and tinting.

**VideoView code In XML Android**

```
< VideoView
android:id = "@+id/simpleVideoView"
android:layout_width = "fill_parent"
android:layout_height = "fill_parent" / >
```

**MediaController In VideoView**

- MediaController is a class which is used to provide the controls for the video playback. If a video is simply played using the VideoView class then the user will not be given any control over the playback of the video which will run until the end of the video is reached.
- This issue can be addressed by attaching an instance of the MediaController class to the VideoView instance.
- The MediaController will then provide a set of controls allowing the user to manage the playback (such as seeking backwards/forwards and pausing in the video timeline).

## 7.3 Creating a SoundPool

- The SoundPool helps developers to make a collection of samples, to load them into memory, not only from a resource of the application APK, but also from a folder in the file system.
- Creating a SoundPool preloads the audio tracks used by your application and optimizes their require management.

e.g. - All the sound effects required for a particular level in a game

- You can create a *SoundPool* class to manage audio when your application requires low audio latency or will be playing multiple audio streams simultaneously (such as a game with multiple sound effects).
- You can setup *SoundPool* to make a sound in specific stream type. There are several types of stream as follows :

| Stream Type | Description |
|---|---|
| AudioManager.STREAM_ALARM | The audio stream for alarms |
| AudioManager.STREAM DTMF | The audio stream for DTMD Tones |
| AudioManager.STREAM_MUSIC | The audio stream for music playback |
| AudioManager.STREAM_NOTIFICATION | The audio stream for notifications |
| AudioManager.STREAM_RING | The audio stream for the Phone ring |
| AudioManager.STREAM_SYSTEM | The audio stream for system sounds |
| AudioManager.STREAM_VOICE_CALL | The audio stream for phone calls |

- AudioManager allows you to adjust the volume on the different audio streams. For example, you are listening to songs on the device while playing games, you can change the volume of the game without affecting the volume of the song.
- Other unmentioned thing is which audio device will produce a sound.
- Using STREAM_MUSIC the sound will be produced through one audio device (phone speaker, earphone, bluetooth speaker or something else) connected to the phone.
- Using STREAM_RING the sound will be produced through all audio device connected to the phone. This behaviour might be differed for each devices.

## 7.4 Using Camera

- Android provides working with camera in two ways :

### 1. Camera Intent

**a.** The easiest way to take a picture from within your application is by using an Intent and applying the constants from the MediaStore class ACTION_IMAGE_CAPTURE.

e.g.

```
startActivityForResult (
new Intent (MediaStore.ACTION_IMAGE_CAPTURE),
        TAKE_PICTURE);
```

b. This launches a Camera application to take the photo, providing your users with the full suite of camera functionality without you having to rewrite the native Camera application.

c. Once users are satisfied with the image, the result is returned to your application within the Intent received by the onActivityResult handler.

**2. Camera API**

a. To access the camera hardware directly, you need to add the CAMERA permission to your application manifest :

```
<uses-permission android:name=" android.permission.CAMERA"/>
```

b. Use the Camera class to adjust camera settings, specify image preferences, and take pictures.

c. To access the Camera, use the static open method on the Camera class:Camera camera = Camera.open();

d. When you are finished with the Camera, remember to free the camera resources by calling release :

```
camera.release();
```

## 7.5 Recording Video

- Android has two methods for recording video-

**1) Intents to Record Video**

a. The easiest way to record a video from within your application is by using an Intent and applying the constants from the MediaStore class ACTION_VIDEO_CAPTURE.

b. Starting a new Activity with this Intent launches the native video recorder, allowing users to start, stop, review, and retake their video.

**2) Media Recording API -**

a. You can use the MediaRecorder class to record video.

b. To record any media in Android, your application needs the CAMERA and RECORD_AUDIO and/or RECORD_VIDEO permissions as applicable :

```
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission android:name="android.permission.RECORD_VIDEO"/>
<uses-permission android:name="android.permission.CAMERA"/>
```

c. Media Recorder manages recording as a state machine. The order in which you manage and configure the Media Recorder is important.

- The transitions in the state machine are as follows

1) Create a new Media Recorder.

2) Unlock the Camera and assign it to the Media Recorder.

3) Specify the input sources to record from.
4) Select a profile to use for Android 2.2 and above, or define the output format and specify the audio and video encoder, frame rate, and output size.
5) Select an output file.
6) Assign a preview Surface.
7) Prepare the Media Recorder for recording.
8) Record.
9) End the recording.

d. When the recoding is finished free all the resources on your Media Recorder object

```
mediaRecorder.release()
```

## 7.6 Short Questions and Answers

**Q.1 What is SoundPool ?**

**Ans. :** • A SoundPool is a collection of samples that can be loaded into memory from a resource inside the APK or from a file in the file system.

- The SoundPool library uses the MediaPlayer service to decode the audio into a raw 16-bit PCM mono or stereo stream. This allows applications to ship with compressed streams without having to suffer the CPU load and latency of decompressing during playback.

**Q.2 How camera is initialize ?**

**Ans. :** Camera initialization

- The Camera.open method will turn on and initialize the camera. At this point it is ready for you to modify settings, configure the preview surface, and take pictures, as shown in the following sections.

**Q.3 What is Android Media player ?**

**Ans. :** Media player class can be used to control playback of audio/video files and streams in Android devices

**Q.4 What is ANR in Android ?**

**Ans. :** ANR stands for Application Not Responding. It is a notification or pop-up displayed by the Android platform whenever the application is performing too many functions at a time and if it is suddenly not responding for a long time to the user action.

## 7.7 Multiple Choice Questions

**Q.1** *What is the most common file type of media supported for audio playback on the Android ?*

a midi b mp3

c ogg d wav

**Q.2** *Point out the correct statement.*

a Voice Actions for Android allows you to use spoken commands to control your Android phone

b The Android Market is not available on Android phones through the Market application

c Video is accessed with the microphone button inside the Google search box on the Android home screen

d None of the mentioned

**Q.3** *Which class is used for playback of audio and video within Android Application ?*

a MediaPlayer class b AudioPlayer class

c VideoPlayer class d AudioVideoManager class

**Q.4** *Which method is used to get the duration of the length of the media being played ?*

a getDuration() b getCurrentDuration()

c getMediaDuration() d getAudioDuration()

**Q.5** *Which class lets you record audio directly from hardware buffers ?*

a AudioRecord (Correct) b AudioBuffer

c SetAudio d PlayAudio

**Q.6** *Which class is used to play raw audio directly into the hardware buffers ?*

a AudioTrack (Correct) b PlayAudio

c AudioRecord d SetAudio

**Q.7** _____ *preloads the audio tracks into your application?*

a SoundPool b AudioBuffer

c VideoBuffer d SoundBuffer

## Answer Keys for Multiple Choice Questions

| | | | | | |
|---|---|---|---|---|---|
| Q.1 | b | Q.2 | a | Q.3 | a |
| Q.4 | a | Q.5 | a | Q.6 | a |
| Q.7 | a | | | | |

❑❑❑

# 8 Publishing and Distributing Android Application

## Syllabus

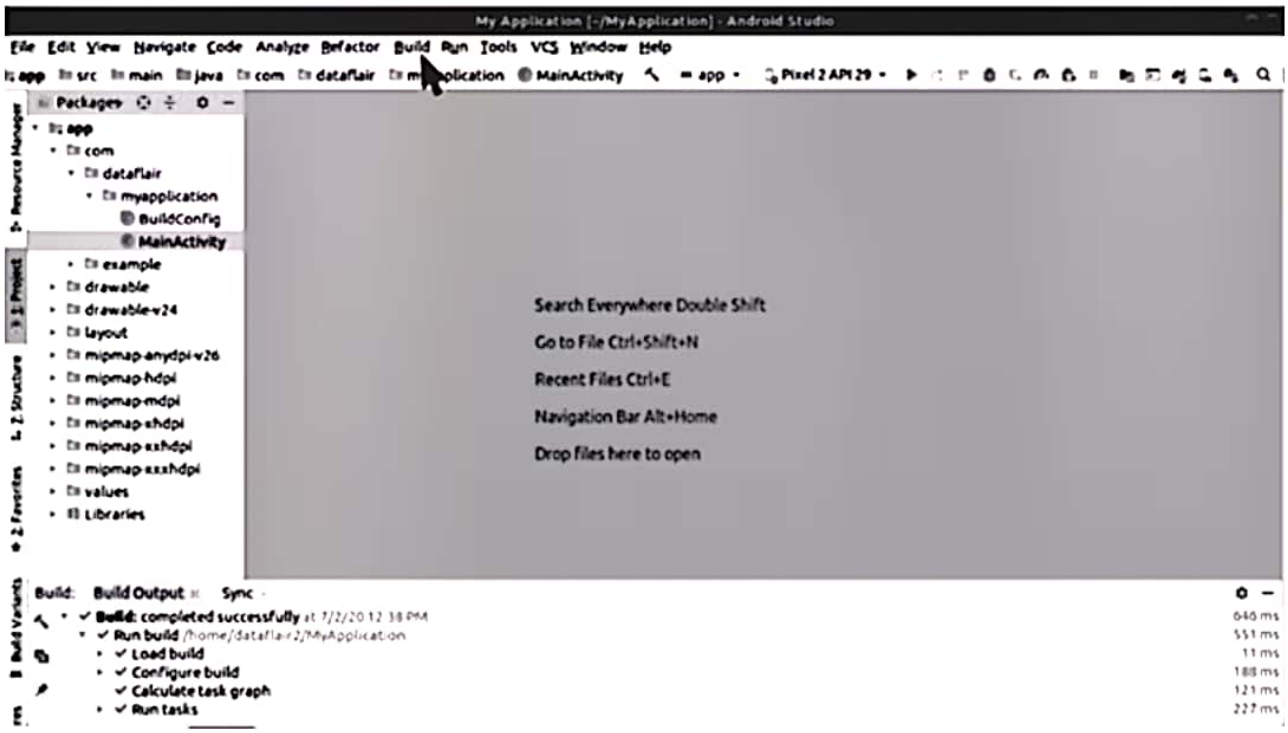
*Signing the Android Application, Versioning the Android Application, Publishing the Android Application.*

## Contents

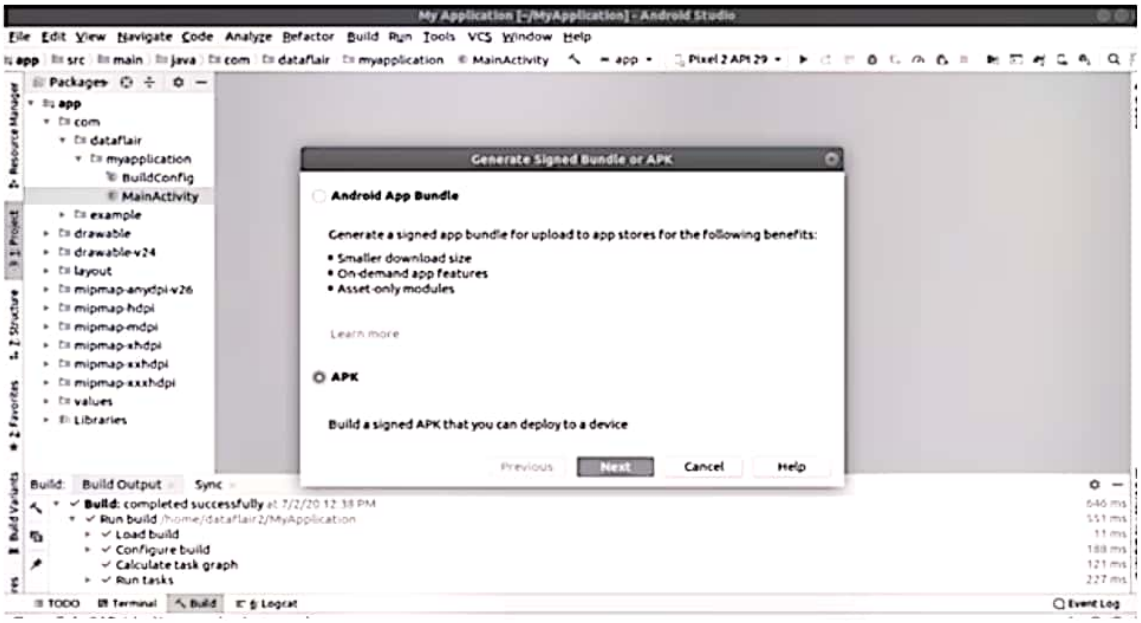
## 8.1 Signing the Android Application

- The Android platform requires every application file-that is, yourappname.apk-to be digitally signed in order to run on a device or emulator; without a signature, an application simply won't run.
- The signing requirement is entirely transparent to most developers until it's time to publish an application for others to use.
- When you're publishing an application for distribution, the application needs to be signed with a nondebug signature. Fortunately, the applications can be self-signed, meaning a certificate authority isn't required. This keeps the complexity and cost down considerably compared to the signing process required for other mobile platforms.
- The Export Android Application wizard simplifies the process of creating and signing a release build of your application package.

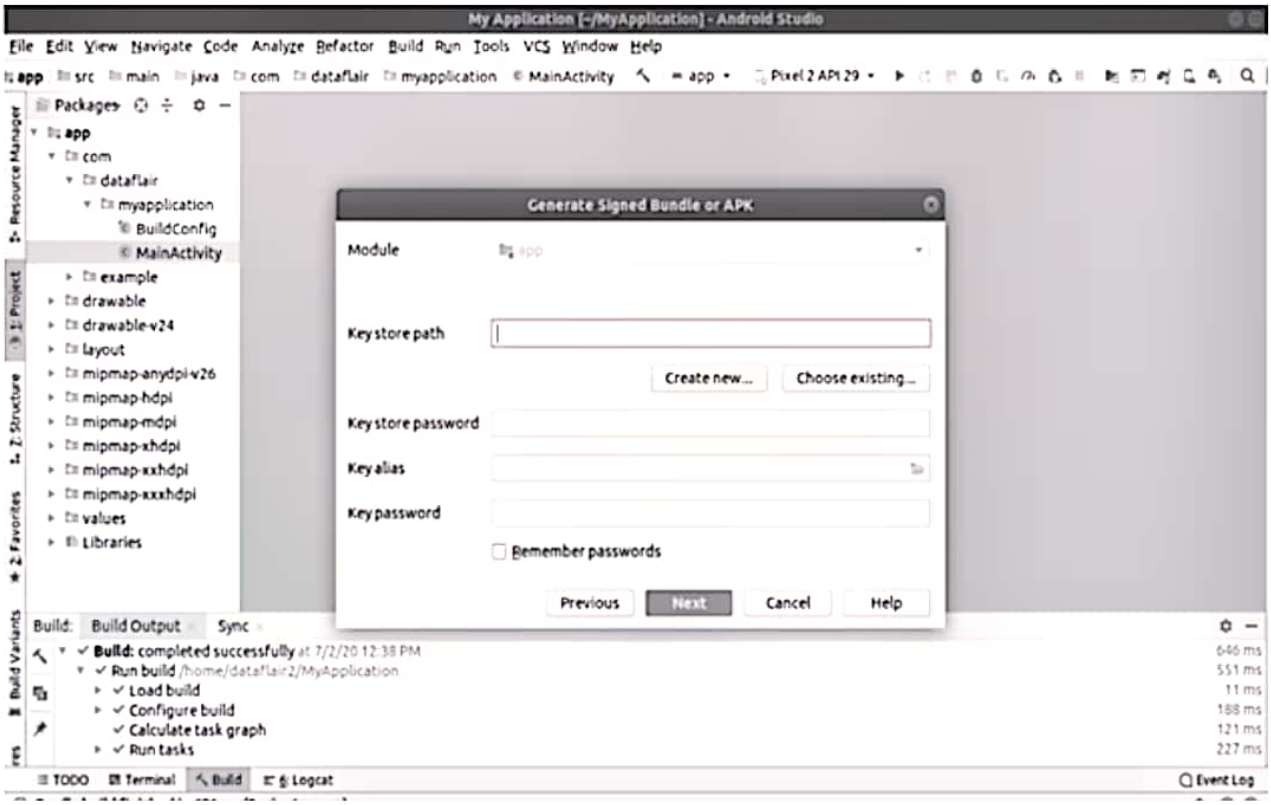
### 1. Open the application in Android Studio



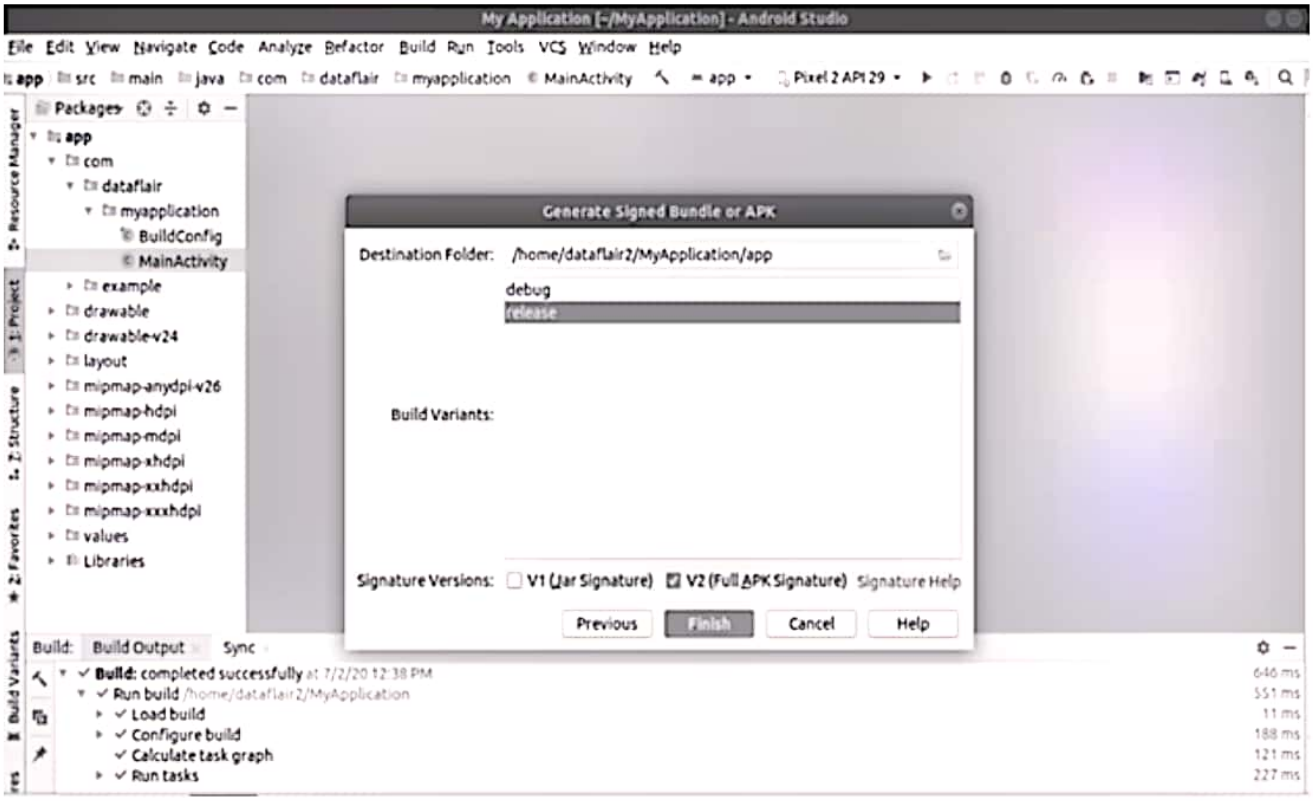
## 2. Generate a signed .apk file from Android Studio.




## 3. Fill the details and press Next.



**4. And after that choose the Variant and version according to choice and requirement. After that, press Finish.**




- Having selected a signing certifi cate, the next step is to select an output destination for your package. The wizard will then compile, sign, and zip-align the package.

## 8.2 Publishing Android App

- Publishing an application involves digitally signing it and uploading it to the appropriate platforms.
- There are various possibilities in which the application can be released, such as on Google Play, Websites or directly to the Users.
- Google Play uses application package names as unique identifiers and will not allow you to upload a duplicate package name.
- To release an app on Google Play, follow these simple steps :
  1. Add some promotional material - Screen shots, videos and interesting feature of app.
  2. Configure options - Information such as language, country, type, category, etc.

- Publish the release version - Once application is ready to released, we can click on the Publish button on the console and in a few minutes, the application would be available throughout the world to download.

### 8.2.1 Process of Publishing an Android Application

- The steps of the publishing process can be summarized as follows.
  1. Select an appropriate AppStore
  2. Read and understand the policies and agreements of the selected AppStore
  3. Quality test
  4. Determine the content rating for the Android application
  5. Determine the country (or the countries) to distribute
  6. Confirm the overall size, platform and the screen compatibility ranges
  7. Decide the revenue model
  8. Decide how to bill or collect the revenue (e.g. In-App or using Google Pay)
  9. Set the price (or prices)
  10. Localization
  11. Prepare promotional graphics, videos and screencasts
  12. Build and upload
  13. Plan for Beta release
  14. Complete AppStore listing
  15. Support users after launch

## 8.3 Distribution of Android App

- Android applications are distributed as Android package files (.APK).
- Google Play is a robust platform that supports us to release, sell and distribute application throughout the world.
- Another way to get great distribution is to partner with device manufacturers and mobile operators, who often select applications to pre-load onto devices prior to purchase. Look for special developer programs that can help you foster partnerships and other distribution relationships with manufacturers, carriers, and the like.
- Various third-party sites also offer distribution channels. These sites have different agreement types and different payment models, so you should research them carefully before using them.

## 8.4 App Characteristics

### 8.4.1 Performance of App

- Performance of app plays important role in getting 5 star rating in all marketplaces like playstore, app store or windows store.
- If application takes more than 10 seconds to load then it will not be used by users.
- If application takes too long time to process data then it will not be used by users.
- If application takes too long time to switch between screens then it will not be used by users.
- The performance of app is important factor which decides the success of app.

**Improving Performance of App**

- Understand your target device. Most developers forget about device, they classify device based on operating system, but forget about configuration of devices. Always classify devices based on specification sheet. Try developing apps based on low specification mobile which will automatically run in all mobiles.
- Understand your tools. Try to understand the tools that are used to develop the mobile app. Better understanding of tools helps to make important architectural design. Example - Understanding about phone gap helps to develop app for multiple platforms.
- Understand core concepts of the language used to build app. Understanding core programming language used to develop app will help to avoid performance issues. Example - A extra string comparison in jscript will surely reduce the performance.
- Understand the library. Try to understand the library used in tools. If you simply call third party methods for simple operation then it will increase battery usage and reduce the performance. Always use standard codes for simple task.

### 8.4.2 Modifiability of App

- Modifiability helps to release multiple version of app more easily.
- Modifiability is achieved by developing as multiple units instead of single unit.
- If any bugs arise after launch then it is easy to modify the unit instead of changing everything in code.
- Modifiability is minimizing the technical risks and cost impact of changes in software.

- In order to achieve modifiability as a system quality, software architects need to envision and incorporate modifiability support in the system's design cycle.
- The architectural design supports the modifiability requirements of a system.
- The modifiability quality of a system can be expressed in terms of cohesion and coupling. Coupling measures the mutual association strength between the system's software components. Where cohesion, on the other hand, is a measure for the number of internal relationships between the responsibilities of a software component.

### 8.4.3 Availability of App

- Availability refers to continuous working of application in both offline and in online mode.
- Today mobile users are travelling across multiple cell sites which frequently disturb the wireless internet connectivity to mobile.
- This interruption should not affect the mobile app. It is possible to achieve high availability by effectively managing offline data. High availability can be provided by giving effective synchronization mechanism.

### 8.4.4 Security of App

- Mobile apps should satisfy stringent requirements for data security and privacy. This is no longer the exclusive domain of the enterprise : Organizations of every size and function are subject to mounting ethical and legal pressure to control and protect the information under their purview.
- Fiduciary responsibility and internal and external policies exist to govern what organizations must do in this regard, from data storage to disaster recovery, encryption to secure updating.
- By definition, internet access and mobile devices carry inherent security risks, including but not limited to the apps that run on them.

## 8.5 Short Questions and Answers

**Q.1 List out at least four versions of Android.**

**Ans. :**

- Android alpha (1.0)
- Android beta (1.1)
- Cupcake (1.5)
- Doughnut (1.6)
- Eclair (2.0 - 2.1)
- Froyo (2.2 - 2.2.3)
- Gingerbread (2.3 - 2.3.7)
- Honeycomb (3.0 - 3.2.6)

- Ice Cream Sandwich (4.0 - 4.0.4)
- Jelly Bean (4.1 - 4.3.1)
- KitKat (4.4 - 4.4 w)

**Q.2 Each application can have zero or more activities. True or False ? Justify in short.**

**Ans. :** True, An Android application can contain zero or more activities. When your application has more than one activity, you may need to navigate from one activity to another.

## 8.6 Multiple Choice Questions

**Q.1** *Android applications must be signed _____ .*

a after they are installed

b before they are installed

c never

d within two weeks of installation

**Q.2** *The emulated device for android _____ .*

a runs the same code base as the actual device, all the way down to the machine layer.

b is more of a simulator, and acts as a virtual machine for the Android device.

c runs the same code base as the actual device, however at a higher level.

d an imaginary machine built on the hopes and dreams of baby elephants.

**Q.3** *The _____ file specifies the layout of your screen.*

a layout file

b manifest file

c strings XML

d R file

**Q.4** *What runs in the background and doesn't have any UI components ?*

a Intents

b Content Providers

c Services

d Applications

**Q.5** *What is AAPT ?*

a Android Asset Processing Tool.

b Android Asset Providing Tool.

c Android Asset Packaging Tool.

d Android Asset Packaging Technique

**Q.6** *What is contained within the manifest xml file ?*

a The source code

b The list of strings used in the app

c The permissions the app requires

d None of the above

**Answer Keys for Multiple Choice Questions**

| | | | | | |
|---|---|---|---|---|---|
| **Q.1** | b | **Q.2** | a | **Q.3** | a |
| **Q.4** | c | **Q.5** | c | **Q.6** | c |

❑❑❑

**Notes**

# SOLVED MODEL QUESTION PAPER

**(As Per 2018 Pattern)**

## Mobile Application Development

**Semester - VI (IT) (Open Elective - II)**

**Time : $2\frac{1}{2}$ Hours]** **[Total Marks : 70**

**Instructions**

1) **Attempt all questions.**

2) **Make suitable assumptions wherever necessary.**

3) **Figures to the right indicate full marks.**

**Q.1** **a)** *Explain android toolbox of standard view.* **[Refer section 3.5]** **[3]**

**b)** *Explain android activity lifecycle.* **[Refer section 2.1]** **[4]**

**c)** *Explain different android APIs.* **[Refer section 1.2]** **[7]**

**Q.2** **a)** *How to create radio button.* **[Refer section 3.3]** **[3]**

**b)** *Explain android graphics interface.* **[Refer section 3.1]** **[4]**

**c)** *Explain android application framework.* **[Refer section 1.4]** **[7]**

**OR**

**c)** *Explain event handling.* **[Refer sections 3.4]** **[7]**

**Q.3** **a)** *List steps for communication between fragments and activity.* **[Refer section 2.2]** **[3]**

**b)** *Explain application specific folder.* **[Refer section 4.1]** **[4]**

**c)** *Explain android application components.* **[Refer section 1.5]** **[7]**

**OR**

**Q.3** **a)** *How android can be used to get current location.* **[Refer section 5.1]** **[3]**

**b)** *Explain SQLite database.* **[Refer section 4.5]** **[4]**

**c)** *What is android layout ? Explain different layout classes and working with layouts.* **[Refer section 3.7]** **[7]**

**Q.4** **a)** *Explain media player in android.* **[Refer section 7.1]** **[3]**

**b)** *Write a note on content provider classes.* **[Refer section 4.6]** **[4]**

**c)** *What are shared preferences in mobile application development ?* **[Refer section 4.4]** **[7]**

**OR**

**Q.4** **a)** *Explain drawable object.* **[Refer section 6.2]** **[3]**

**b)** *Explain working with graphics and uses of graphics.* **[Refer section 6.1]** **[4]**

**c)** *Explain geocoding and use of geolocations.* **[Refer section 5.3]** **[7]**

**Q.5** **a)** *Explain recording video.* **[Refer section 7.5]** **[3]**

**b)** *Explain publishing android app.* **[Refer section 8.2]** **[4]**

**c)** *Explain working with animation.* **[Refer section 6.5]** **[7]**

**OR**

**Q.5** **a)** *Explain signing the android application.* **[Refer section 8.1]** **[3]**

**b)** *Explain recording and playing sound.* **[Refer section 7.2]** **[4]**

**c)** *Write a a short note on :* **[Refer section 8.4]**

*A) Performance of App* *B) Modifiability of App*

*C) Availability of App* *D) Security of App* **[7]**

❑❑❑

***Notes***

***Notes***