A quality product by
Brainheaters™ LLC

Brainheaters Notes

MAD | Computer Semester-7

SERIES 313 - 2018 (A.Y 2021 - 22)

BH.Index

(Learn as per the Priority to prepare smartly)

MODULE-1

Q1. Explain OOPS (P4 - Appeared 1 Time) (3-7M)

ANS: Object-oriented programming (OOPS):

- Object-oriented programming is a method used for designing a program using classes and objects. Object-oriented programming is also called the core of java.
- Object-oriented programming organizes a program around objects and well-defined interfaces. This can also be characterized as data controlling for accessing the code.
- In this type of approach, programmers define the data type of a data structure and the operations that are applied to the data structure.

This implies software development and maintenance by using some of the concepts:

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

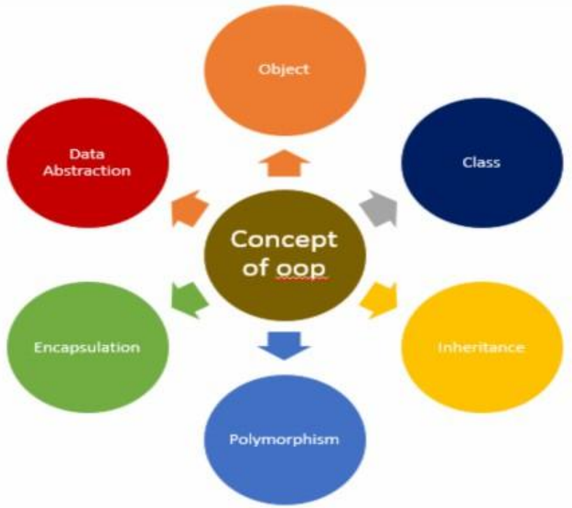


Figure: Concept of OOP

Objects:

- Objects are always called as instances of a class. Objects are created from class in java or any other languages. Objects are those that have state and behavior. Objects are abstract data types (i.e., objects behaviour is defined by a set of values and operations).
- These objects always correspond to things found in the real world, i.e., real entities. So, they are also called the run time entity of the world. These are self–contained which consists of methods and properties which makes data useful.
- Objects can be both physical and logical data. It contains addresses and takes up some space in memory. Some of the examples of objects are a dog, chair, tree etc. When we treat

animals as objects, it has states like colour, name, breed etc., and behaviours such as eating, wagging the tail etc.

Suppose, we have created a class called My book, we specify the class name followed by the object name, and we use the keyword new.

Example 1:

```
Public class Mybook {
int x=10;
Public static void main (String args []) {
Mybook Myobj= new Mybook ();
System.out.println(MyObj.x);
}
}
```

In the above example, a new object is created, and it returns the value of x which may be the number of books.

```
Mybook Myobj= new Mybook ();
```

This is the statement used for creating objects.

```
System.out.println(Myobj.x);
```

This statement is used to return the value of x of an object.

- We can also create multiple objects in the same class and we can create in one class and access it in another class. This method is used for better organization of classes and always remember that name of the java file and the class name remains the same.

Classes:

- Classes are like an object constructor for creating objects. Collection of objects is said to be a class. Classes are said to be logical quantities. Classes don't consume any space in the memory. Class is also called a template of an object. Classes have members

which can be fields, methods and constructors. A class has both static and instance initializers.

A class declaration consists of:

- Modifiers
- Class name
- Keywords
- The class body within { } brackets.

A class keyword is used to create a class. A simplified general form of class definition is given below:

```
class classname {
type instance variable 1;
type instance variable 2;
.
.
.
type instance variable n;
type methodname 1 (parameter list) {
// body od method
}
type methodname 2 (parameter list) {
// body od method
}
type methodname (parameter list) {
// body od method
}
 }
```

- The variables or data defined within a class are called instance variables. Code is always contained in the methods. Therefore, the methods and variables defined within a class are called members

of the class. All the methods have the same form as main () these methods are not specified as static or public.

Abstraction:

- Abstraction is a process which displays only the information needed and hides the unnecessary information. We can say that the main purpose of abstraction is data hiding. Abstraction means selecting data from a large number of data to show the information needed, which helps in reducing programming complexity and efforts.
- There are also abstract class and abstract methods. An abstract class is a type of class that declares one or more abstract methods. An abstract method is a method that has a method definition but not implementation. Once we have modelled our object using data abstraction, the same sets of data can also be used in different applications—abstract classes, generic types of behaviours and object-oriented programming hierarchy.
- Abstract methods are used when two or more subclasses do the same task in different ways and through different implementations. An abstract class can have both the methods, i.e., abstract methods and regular methods.

Abstract class example:

```
//abstract parent class
    Abstract class animal {
     //abstract method
   public abstract void sound ( ) ;
     }
  Public class lion extends animal {
   Public void sound ( ) {
```

```
System.out.println (" roar " );
}
public Static void main ( String args [ ] ) {
 animal obj = new lion ( );
obj. sound ();
}
}
```

Output:

Roar

Inheritance:

- Inheritance is a method in which one object acquires/inherits another object's properties, and inheritance also supports hierarchical classification. The idea behind this is that we can create new classes built on existing classes, i.e., when you inherit from an existing class, we can reuse methods and fields of the parent class. Inheritance represents the parent-child relationship.
- For example, a whale is a part of the classification of marine animals, which is part of class mammal, which is under that class of animal. We use hierarchical classification, i.e., top-down classification.
- If we want to describe a more specific class of animals such as mammals, they would have more specific attributes such as teeth; cold-blooded, warm-blooded, etc. This comes under the subclass of animals where animals come under superclass. The subclass is a class which inherits properties of the superclass. This is also called a derived class.

- A superclass is a base class or parental class from which a subclass inherits properties.We use inheritance mainly for method overriding and R: To inherit a class, we use the extend keyword.

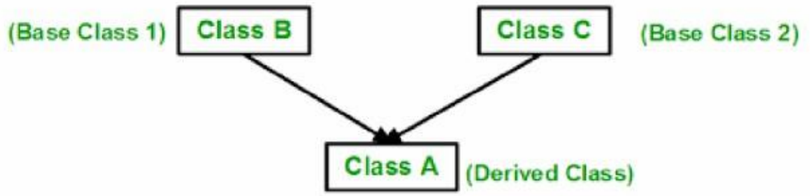


Figure: Inheritance

There are five types of inheritance:

- Single inheritance
- multilevel inheritance
- multiple inheritance
- hybrid inheritance
- hierarchical inheritance

Single level

- In this one class i.e., derived class inherits properties from its parental class. This enables code reusability and also adds new features to the code. Example: class b inherits properties from class a.
- Class A is base or parental class and class b is derived class.

Syntax:

```
Class a {
...
}
Class b extends class a {
```

...
}

Multilevel

- This one class is derived from another class which is also derived from another class i.e., this class has more than one parental class, hence it is called multilevel inheritance.

Syntax:

```
Class a {
....
}
Class b extends class a {
....
}
Class c extends class b {
...
}
```

Hierarchical level

- In this one parental class has two or more derived classes or we can say that two or more child classes has one parental class.

Syntax:

```
Class a {
...
}
Class b extends class a {
..
}
Class c extends class a {
```

```
..
}
```

Hybrid inheritance

- This is the combination of multiple and multilevel inheritance and in java multiple inheritance is not supported as it leads to ambiguity and this type of inheritance can only be achieved through interfaces.
- Consider that class a is the parental or base class of class b and class c and in turn class b and class c are parental or base class of class d. Class b and class c are derived classes from class a and class d is derived class from class b and class c.
- Following program creates a super class called add and a subclass called sub, uses extend keyword to create a subclass add.

Example:

```
// a simple example of inheritance
//create a superclass
Class Add {
int my;
int by;
void setmyby (int xy, int hy) {
my=xy;
by=hy;
}
}
/create a sub class
class b extends add {
int total;
void sum () {
```

```
public Static void main (String args [ ] ) {
b subOb= new b ( );
subOb. Setmyby (10, 12);
subOb. Sum ( ) ;
System.out.println("total =" + subOb. Total);
}
}
```

It gives output as – total = 22

Polymorphism:

- Polymorphism refers to many forms, or it is a process that performs a single action in different ways. It occurs when we have many classes related to each other by inheritance.
- Polymorphism is of two different types, i.e., compile-time polymorphism and runtime polymorphism. One of the examples in Compile time polymorphism is when we overload a static method in java. Run time polymorphism is also called a dynamic method dispatch is a method in which a call to an overridden method is resolved at run time rather than compile time.
- In this method, the overridden method is always called through the reference variable. By using method overloading and method overriding, we can perform polymorphism. Generally, the concept of polymorphism is often expressed as one interface, multiple methods. This reduces complexity by allowing the same interface to be used as a general class of action.

Example:

```
public class Bird {
…
Public void sound ( ) {
System.out.println ( " birds sounds " );
}
}
public class pigeon extends Bird {
…
@override
public void sound ( ) {
System.out.println( " cooing " ) ;
}
}
public class sparrow extends Bird ( ) {
….
@override
Public void sound ( ){
System.out.println( " chip " ) ;
}
}
```

- In the above example, we can see common action sound () but there are different ways to do the same action. This is one of the examples which shows polymorphism.

Encapsulation:

- Encapsulation is one of the concepts in OOPs concepts; it is the process that binds together the data and code into a single unit and keeps both from being safe from outside interference and misuse. In this process, the data is hidden from other classes and

can be accessed only through the current class's methods. Hence, it is also known as data hiding.

- Encapsulation acts as a protective wrapper that prevents the code and data from being accessed by outsiders. These are controlled through a well-defined interface.
- Encapsulation is achieved by declaring the variables as private and providing public setter and getter methods to modify and view the variable values. In encapsulation, the fields of a class are made read-only or write-only. This method also improves the re-usability. Encapsulated code is also easy to test for unit testing.

Example:

```
class animal {
// private field
private int age;
//getter method
Public int getage ( ) {
return age;
}
//setter method
public void setAge ( int age ) {
this. Age = age;
}
}
class Main {
public static void main (String args []);
//create an object of person
Animal a1= new Animal ();
//change age using setter
A1. setAge (12);
```

```
// access age using getter
System.out.println(" animal age is " + a1. getage ( ) );
}
}
```

Output: Animal age is 12

- In this example, we declared a private field called age that cannot be accessed outside of the class. To assess age, we used public methods. These methods are called getter and setter methods. Making age private allows us to restrict unauthorized access from outside the class. Hence this is called data hiding.

Q2. Explain Core java OR Difference between Core java and Advanced java (P4 - Appeared 1 Time) (3-7M)

ANS:

Core Java	Advanced Java
1. To develop general purpose applications.	1. To develop online application and mobile application
2. Without core java no one can develop any advanced java applications.	2. Whereas advanced java only deals with some specialization like database, DOM(web), networking etc.
3. OOP, data types, operators, functions, loops, exception handling, threading etc.	3. Apart from the core java parts it has some specific sections like database connectivity, web

	services, servlets etc.
4. It uses only one tier architecture that is why it is called a 'stand alone' application.	4. It uses two tier architecture i.e. client side architecture and server side or backend architecture.
5. Core java programming covers the swings, socket, awt, thread concept, collection object and classes.	5. Advanced java is used for web based applications and enterprise applications.

MODULE-2

Q1. Explain Software Engineering with SDLC (P4 - Appeared 1 Time) (3-7M)

ANS: Software Development Cycle (SDLC):

- Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality softwares. The SDLC aims to produce high-quality software that meets or exceeds customer expectations, reaches completion within time and cost estimates.
    1. SDLC is the acronym for Software Development Life Cycle.
    2. It is also called the Software Development Process.
    3. SDLC is a framework defining tasks performed at each step in the software development process.
    4. ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.
- SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The following figure is a graphical representation of the various stages of a typical SDLC.

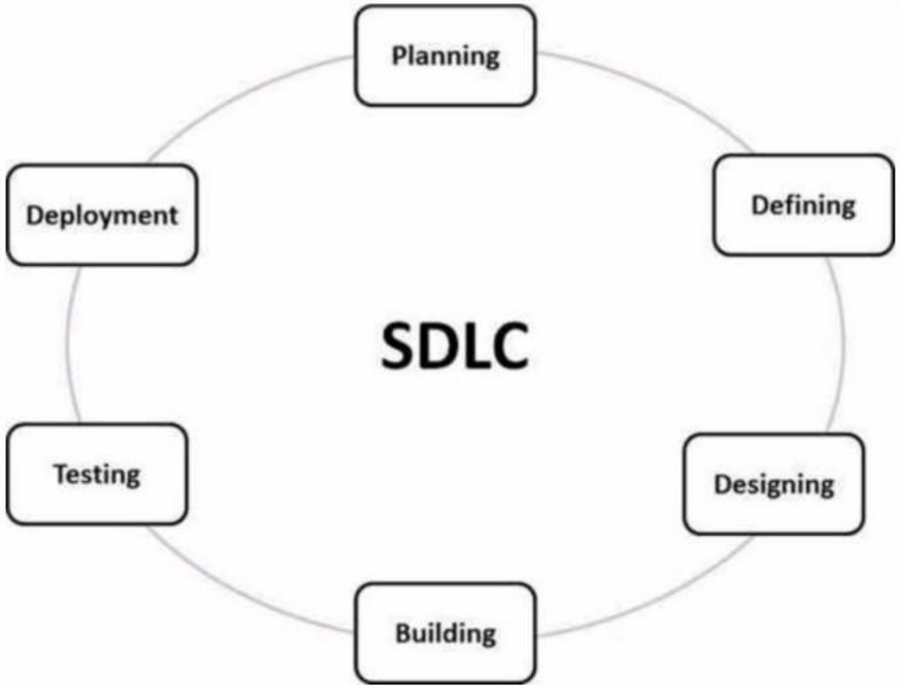


Figure: Stages of SDLC

Stage 1: Planning and Requirement Analysis

- Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.
- Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the

various technical approaches that can be followed to implement the project successfully with minimum risks.

Stage 2: Defining Requirements

- Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an SRS (Software Requirement Specification) document which consists of all the product requirements to be designed and developed during the project life cycle.

Stage 3: Designing the Product Architecture

- SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.
- This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.
- A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

Stage 4: Building or Developing the Product

- In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized

manner, code generation can be accomplished without much hassle.

- Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

Stage 5: Testing the Product

- This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

Stage 6: Deployment in the Market and Maintenance

- Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).
- Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

Q2. Explain DFD (P4 - Appeared 1 Time) (3-7M)

ANS: Data Flow Diagram (DFD):

- DFD is the abbreviation for Data Flow Diagram. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself.
- DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart. Data Flow Diagrams can be represented in several ways.
- The DFD belongs to structured-analysis modeling tools. Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in software-system processes.

Components of DFD

The Data Flow Diagram has 4 components:

- Process:
  Input to output transformation in a system takes place because of process function. The symbols of a process are rectangular with rounded corners, oval, rectangle or a circle. The process is named a short sentence, in one word or a phrase to express its essence
- Data Flow:
  Data flow describes the information transferring between different parts of the systems. The arrow symbol is the symbol of data flow. A relatable name should be given to the flow to determine the information which is being moved. Data flow also represents material along with information that is being moved. Material shifts are modeled in systems that are not merely informative. A given

flow should only transfer a single type of information. The direction of flow is represented by the arrow which can also be bi-directional.

- Warehouse:
  The data is stored in the warehouse for later use. Two horizontal lines represent the symbol of the store. The warehouse is simply not restricted to being a data file rather it can be anything like a folder with documents, an optical disc, a filing cabinet. The data warehouse can be viewed independent of its implementation. When the data flows from the warehouse it is considered as data reading and when data flows to the warehouse it is called data entry or data updation.
- Terminator:
  The Terminator is an external entity that stands outside of the system and communicates with the system. It can be, for example, organizations like banks, groups of people like customers or different departments of the same organization, which is not a part of the model system and is an external entity. Modeled systems also communicate with terminators.

Figure: Basic Structure of DFD

Rules for creating DFD

- The name of the entity should be easy and understandable without any extra assistance(like comments).

- The processes should be numbered or put in an ordered list to be referred to easily.
- The DFD should maintain consistency across all the DFD levels.
- A single DFD can have maximum processes upto 9 and minimum 3 processes.

Levels of DFD

DFD uses hierarchy to maintain transparency thus multi level DFD's can be created. Levels of DFD are as follows:

- 0-level DFD
- 1-level DFD:
- 2-level DFD:

Advantages of DFD

- It helps us to understand the functioning and the limits of a system.
- It is a graphical representation which is very easy to understand as it helps visualize contents.
- Data Flow Diagram represents a detailed and well explained diagram of system components.
- It is used as part of the system documentation file.
- Data Flow Diagrams can be understood by both technical or nontechnical people because they are very easy to understand.

Disadvantages of DFD

- At times DFD can confuse the programmers regarding the system.
- Data Flow Diagrams take a long time to be generated, and many times due to these reasons analysts are denied permission to work on it.

Q3. Explain SQL database (P4 - Appeared 1 Time) (3-7M)

ANS: Structured Query Language (SQL):

- SQL is a language to operate databases; it includes database creation, deletion, fetching rows, modifying rows, etc.
- SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in a relational database.
- SQL is the standard language for Relational Database Systems. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.

Significance of SQL:

- Allows users to access data in the relational database management systems.
- Allows users to describe the data.
- Allows users to define the data in a database and manipulate that data.
- Allows embedding within other languages using SQL modules, libraries & pre-compilers.
- Allows users to create and drop databases and tables.
- Allows users to create views, stored procedure, functions in a database.
- Allows users to set permissions on tables, procedures and views.

SQL Process:

- When you are executing an SQL command for any RDBMS, the system determines the best way to carry out your request and SQL engine figures out how to interpret the task.

- These components are -
    1. Query Dispatcher
    2. Optimization Engines
    3. Classic Query Engine
    4. SQL Query Engine, etc.

Following is a simple diagram showing the SQL Architecture -

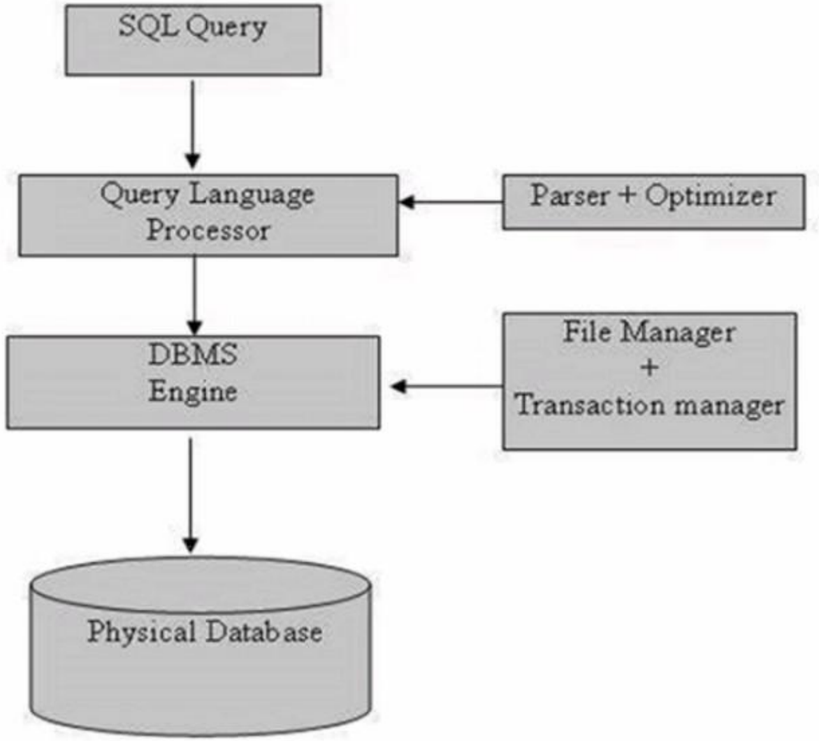


Figure: SQL Architecture

SQL Commands:

DDL- Data Definition Language

No.	Commands	Descriptions
1.	CREATE	Creates a new table, a view of a table, or other object in the database.
2.	ALTER	Modifies an existing database object, such as a table
3.	DROP	Deletes an entire table, a view of a table or other objects in the database.

DML-Data Manipulation Language

No.	Commands	Descriptions
1.	SELECT	Retrieves certain records from one or more tables
2.	INSERT	Creates a record.
3.	UPDATE	Modifies records.
4.	DELETE	Deletes records.

DCL-Data Control Language

No.	Commands	Descriptions
1.	GRANT	Gives a privilege to the user.
2.	REVOKE	Takes back privileges granted from users.

MODULE-3

Q1. Explain Android System with Architecture (P4 - Appeared 1 Time) (3-7M)

ANS: Android System Architecture:

- Android architecture contains different number of components to support any android device needs. Android software contains an open-source Linux Kernel having collection of number of C/C++ libraries which are exposed through an application framework services.
- Among all the components Linux Kernel provides main functionality of operating system functions to smartphones and Dalvik Virtual Machine (DVM) provide platform for running an android application.

The main components of android architecture are following:-

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

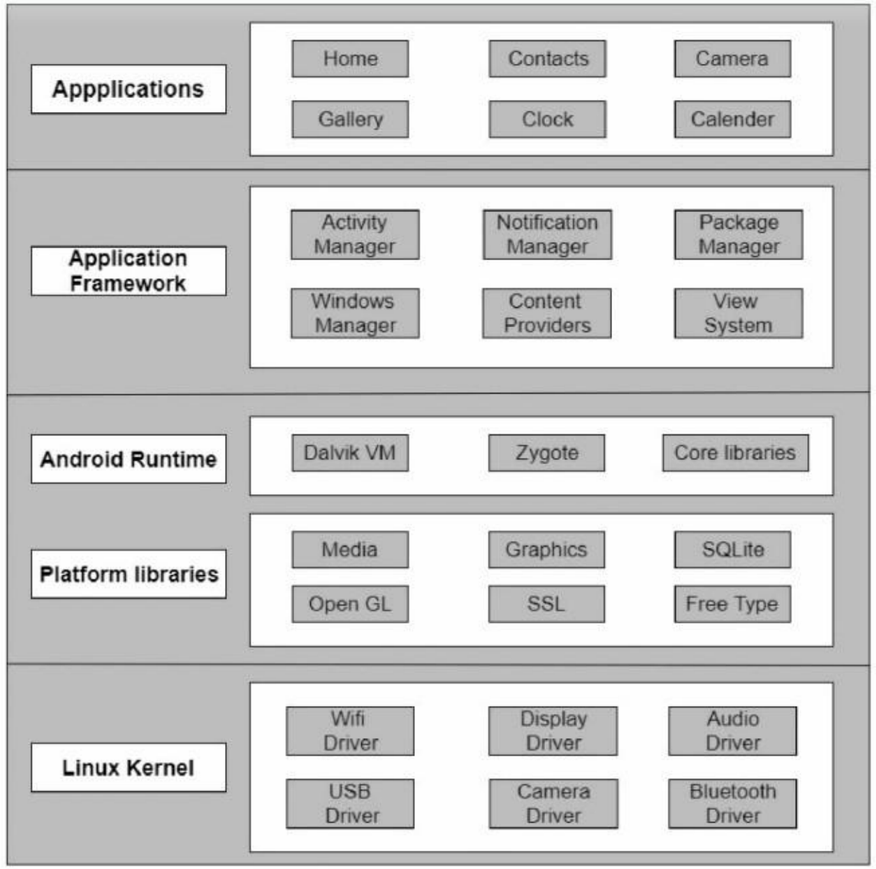


Figure: Pictorial representation of android architecture

Applications –

- Applications is the top layer of android architecture. The pre-installed applications like home, contacts, camera, gallery etc

and third party applications downloaded from the play store like chat applications, games etc. will be installed on this layer only.
- It runs within the Android run time with the help of the classes and services provided by the application framework.

Application framework –

- Application Framework provides several important classes which are used to create an Android application. It provides a generic abstraction for hardware access and also helps in managing the user interface with application resources.
- Generally, it provides the services with the help of which we can create a particular class and make that class helpful for the Applications creation.
- It includes different types of services activity manager, notification manager, view system, package manager etc. which are helpful for the development of our application according to the prerequisite.

Application runtime –

- The Android Runtime environment is one of the most important part of Android. It contains components like core libraries and the Dalvik virtual machine(DVM). Mainly, it provides the base for the application framework and powers our application with the help of the core libraries.
- Like Java Virtual Machine (JVM), Dalvik Virtual Machine (DVM) is a register-based virtual machine and specially designed and optimized for android to ensure that a device can run multiple instances efficiently. It depends on the layer Linux kernel for threading and low-level memory management. The core libraries

enable us to implement android applications using the standard JAVA or Kotlin programming languages.

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

- Media library provides support to play and record audio and video formats.
- Surface manager responsible for managing access to the display subsystem.
- SGL and OpenGL both cross-language, cross-platform application program interface (API) are used for 2D and 3D computer graphics.
- SQLite provides database support and FreeType provides font support.
- Web-Kit This open source web browser engine provides all the functionality to display web content and to simplify page loading.
- SSL (Secure Sockets Layer) is security technology to establish an encrypted link between a web server and a web browser.

Linux Kernel –

- Linux Kernel is the heart of the android architecture. It manages all the available drivers such as display drivers, camera drivers, Bluetooth drivers, audio drivers, memory drivers, etc. which are required during the runtime.
- The Linux Kernel will provide an abstraction layer between the device hardware and the other components of android architecture. It is responsible for management of memory, power, devices etc.

The features of Linux kernel are:

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

Q2. What is Setup Android Environment (P4 - Appeared 1 Time) (3-7M)

ANS: Android Environment Setup:
Install the Java Development Kit

- Download and install the Java Development Kit (JDK). Unity requires the 64-bit version JDK 8 (1.8).

Download the Android SDK

- You can install the Android SDK using command line tools or through Android Studio. Android Studio provides an easy to use GUI based tool, but installs additional software on your computer.
- Using the command line tools is a smaller download and does not install additional software, but it can be more challenging to use.

Install the Android SDK using the command line tools

- Install or unpack the Android SDK. After installing, open the Android SDK Manager and add: at least one Android SDK Platform, the

Platform Tools, the Build Tools, and the USB drivers if you're using Windows.

To install an Android platform SDK and the associated tools:

1. Download the Android Software command line tool.
2. Unzip the tools folder to a location on your hard drive.
3. Open a command-prompt window.
4. Navigate to the bin folder in the location where you unzipped the tools folder: "install folder" > tools > bin
5. Use the sdkmanager command line tool to retrieve the list of packages that you can install. The installable packages include the Platform SDKs, Build Tools, Platform tools, and other tools.
   sdkmanager –list
6. Select a version of the Platform SDK to install. Platform SDKs take the following form in the list: platforms;android-xx. The xx indicates the SDK level. The larger the number, the newer the package. Typically, you can install the latest available version. But, there might be cases in which Google has released a new version of the SDK that causes errors when you build your Unity Project. In that case you must uninstall the SDK and install an earlier version. The general format of the command for package installation is sdk manager <package name>. You can install the corresponding Platform Tools and Build Tools at the same time.
   Example: sdkmanager "platform-tools" "platforms;android–27" "build-tools;27.0.3"
7. If you are running on Windows, install the USB device drivers. sdkmanager "extras;google;usb_driver" This installs the SDK in a directory named "platforms" in the directory in which you unzipped the tools folder. Example: c:\<install folder>\platforms

Install the SDK using Android Studio

1. Install Android studio from the Android developer portal. The Android developer portal provides detailed installation instructions. Note: Android Studio provides some ease of use benefits, but it is not fully tested for compatibility with Unity installs. If you encounter errors, Unity recommends using the command line method.
2. When installing the Android platform SDK and other tools, you can typically install the latest available version. There might be cases in which Google has released a new version of the SDK that causes errors when you build your Unity Project. In that case you must uninstall the SDK and install an earlier version.
3. Install the associated Platform and Build tools at the same time. If you are running on Windows, install the USB device drivers.

Enable USB debugging on your device

- To enable USB debugging, you must enable Developer options on your device. To do this, find the build number in your device's Settings menu. The location of the build number varies between devices. The stock Android setting can be found by navigating to Settings > About phone > Build number.
- For specific information on your device and Android version, refer to your hardware manufacturer. Build number as displayed in Android 5.0 (Lollipop) on a Samsung Galaxy Note 3 Note: On Android versions prior to 4.2 (Jelly Bean), the Developer options aren't hidden. Go to Settings > Developer options, then enable USB debugging.
- After you navigate to the build number using the instructions above, tap on the build number seven times. A pop-up notification saying "You are now X steps away from being a developer" appears, with

"X" being a number that counts down with every additional tap. On the seventh tap, Developer options are unlocked.

- Connect your device to your computer using a USB cable. If you are developing on a Windows computer, you might need to install a device specific USB driver. See the manufacturer web site for your device for additional information.
- The setup process differs for Windows and macOS, and is explained in detail on the Android developer website. For more information on connecting your Android device to the SDK, refer to the Running Your App section of the Android Developer documentation.
- Go to Settings > Developer options, and check the USB debugging checkbox to enable debug mode when the device is connected to a computer via USB. Developer options as displayed in Android 5.0 (Lollipop) - Samsung Galaxy Note 3

Configure the Android SDK path in Unity

- The first time you create a Project for Android (or if Unity later fails to locate the SDK), Unity asks you to locate the folder in which you installed the Android SDK.
- If you installed the SDK using the sdkmanager, you can find the folder in <android tools install location>\platforms\<android sdk folder>.

Example:

c:\<android tools install location>\platforms\android-27

- If you installed the SDK when you installed Android Studio, you can find the location in the Android Studio SDK Manager. To open the SDK Manager from Android Studio, click Tools > Android > SDK Manager or click SDK Manager in the toolbar.

- SDK Manager toolbar button: To change the location of the Android SDK, in the menu bar go to Unity > Preferences > External Tools.

Download and set up the Android NDK

- If you are using the IL2CPP scripting backend for Android, you need the Android Native Development Kit (NDK). It contains the toolchains (such as compiler and linker) needed to build the necessary libraries, and finally produce the output package (APK). If you are not targeting the IL2CPP back end, you can skip this step.
- Download Android NDK version r13b (64-bit) from the NDK Downloads web page. Extract the android-ndk folder to a directory on your computer and note the location.
- The first time you build a Project for Android using IL2CPP, you are asked to locate the folder in which you installed the Android NDK. Select the root folder of your NDK installation.
- To change the location of the Android NDK, in the Unity Editor, navigate to the menu: Unity > Preferences to display the Unity Preferences dialog box. Here, click External Tools.

Q3. Explain Building Blocks of Android Application (P4 - Appeared 1 Time) (3-7M)

ANS: Android Core Building Blocks:

- An android component is simply a piece of code that has a well defined life cycle e.g. Activity, Receiver, Service etc.
- The core building blocks or fundamental components of android are activities, views, intents, services, content providers, fragments and AndroidManifest.xml.

Figure: Android Core Building Blocks

Activity

- An activity is a class that represents a single screen. It is like a Frame in AWT.

View

- A view is the UI element such as button, label, text field etc. Anything that you see is a view.

Intent

- Intent is used to invoke components. It is mainly used to:
    1. Start the service
    2. Launch an activity
    3. Display a web page
    4. Display a list of contacts
    5. Broadcast a message
    6. Dial a phone call etc.

Example, you may write the following code to view the webpage.

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

Service

- Service is a background process that can run for a long time. There are two types of services: local and remote. Local service is accessed from within the application whereas remote service is accessed remotely from other applications running on the same device.

Content Provider

- Content Providers are used to share data between the applications.

Fragment

- Fragments are like parts of activity. An activity can display one or more fragments on the screen at the same time.

AndroidManifest.xml

- It contains information about activities, content providers, permissions etc. It is like the web.xml file in Java EE.

Android Virtual Device (AVD)

- It is used to test the android application without the need for mobile or tablet etc. It can be created in different configurations to emulate different types of real devices.

Q4. Define Activity Lifecycle (P4 - Appeared 1 Time) (3-7M)

ANS: Android:

- Android is an open-source operating system which is based on Linux with a Java programming interface for mobile devices like

Smartphones (Touch Screen Devices who support Android OS).

- It comprises a multiple API to support location-based services such as GPS. It also has extensive support for multimedia hardware control to perform playback or recording using camera and microphone.
- It supports multitasking, we can move from one task window to another and multiple applications can run simultaneously. It will give a chance to reuse the application components and the replacement of native applications.

Android Activity Life Cycle?:

- As a user navigates through the app, Activity instances in your app transition through different stages in their life-cycle. The Activity class provides a number of callbacks that allow the activity to know that a state has changed: that the system is creating, stopping, or resuming an activity, or destroying the process in which the activity resides.

In general, activity lifecycle has seven callback methods:

1. onCreate()
2. onStart()
3. onResume()
4. onPause()
5. onStop()
6. onRestart()
7. onDestroy()

Now let's get into the details of Android Activity Life cycle methods and callbacks. Take a look at the below figure to understand the life cycle.

- Program starts from a main() function in different programming languages. Similarly, android initiates the program within an activity with a call to onCreate() callback method.
- There is a sequence of callback methods that starts up an activity and then tear down in different methods shown in the above

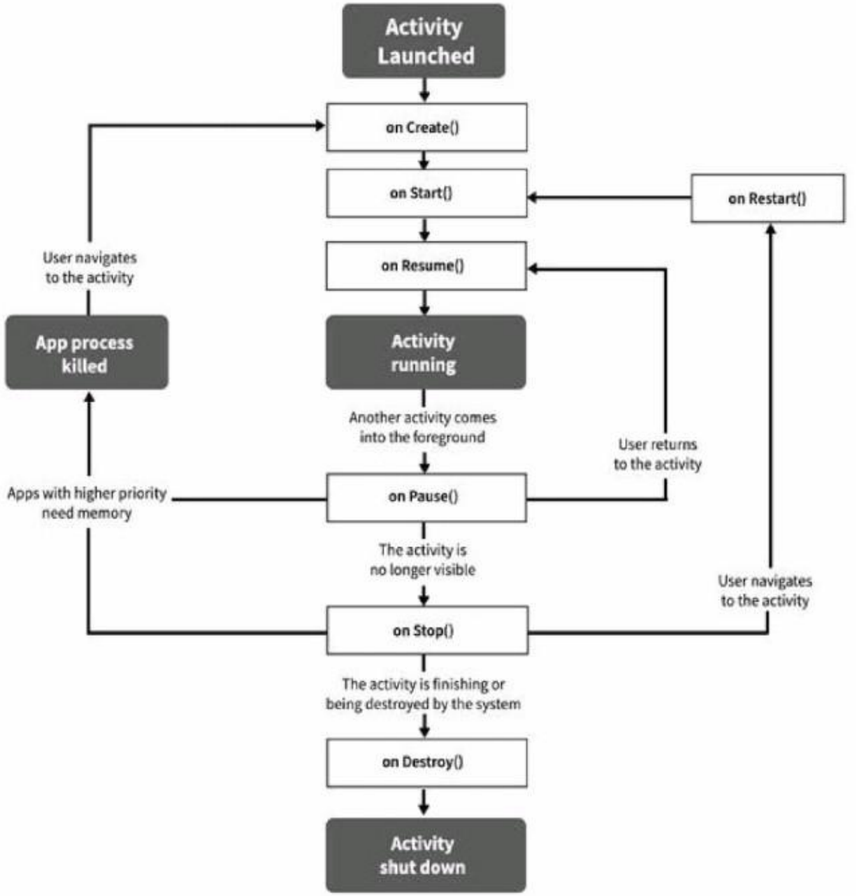


Figure: Activity Lifecycle in Android

Activity lifecycle diagram:

1. onCreate(): In this state, the activity is created.
2. onStart(): This callback method is called when the activity becomes visible to the user.
3. onResume(): The activity is in the foreground and the user can interact with it.
4. onPause(): Activity is partially obscured by another activity. Another activity that's in the foreground is semi-transparent.
5. onStop(): The activity is completely hidden and not visible to the user.
6. onRestart(): From the Stopped state, the activity either comes back to interact with the user or the activity is finished running and goes away. If the activity comes back, the system invokes onRestart()
7. onDestroy(): Activity is destroyed and removed from the memory.

So these are the various methods of the Activity Life Cycle. Now let's see the situations where the life cycle methods and states will occur.

- When you open the app it will go through below states: onCreate() –> onStart() –> onResume()
- When you press the back button and exit the app
  onPaused() — > onStop() –> onDestroy()
- When you press the home button
  onPaused() –> onStop()
- After pressing the home button, again when you open the app from a recent task list
  onRestart() –> onStart() –> onResume()
- After dismissing the dialog or back button from the dialog
  onResume()

- If a phone is ringing and user is using the app
  onPause() –> onResume()
- After the call ends
  onResume()
- When your phone screen is off
  onPaused() –> onStop()
- When your phone screen is turned back on
  onRestart() –> onStart() –> onResume()

Q5. Differentiate Intents Fragments and Fragment Lifecycle OR Explain Android Fragments (P4 - Appeared 1 Time) (3-7M)

ANS: Android Fragments:

- A Fragment is a piece of an activity which enables more modular activity design. It will not be wrong if we say, a fragment is a kind of sub-activity.

Significance of fragments:

- A fragment has its own layout and its own behaviour with its own life cycle callbacks.
- You can add or remove fragments in an activity while the activity is running.
- You can combine multiple fragments in a single activity to build a multi-pane UI.
- A fragment can be used in multiple activities.
- Fragment life cycle is closely related to the life cycle of its host activity which means when the activity is paused, all the fragments available in the activity will also be stopped.

- A fragment can implement a behaviour that has no user interface component.
- Fragments were added to the Android API in Honeycomb version of Android which API version 11.

You create fragments by extending Fragment class and You can insert a fragment into your activity layout by declaring the fragment in the activity's layout file, as a <fragment> element.

- Prior to fragment introduction, we had a limitation because we can show only a single activity on the screen at one given point in time. So we were not able to divide the device screen and control different parts separately.

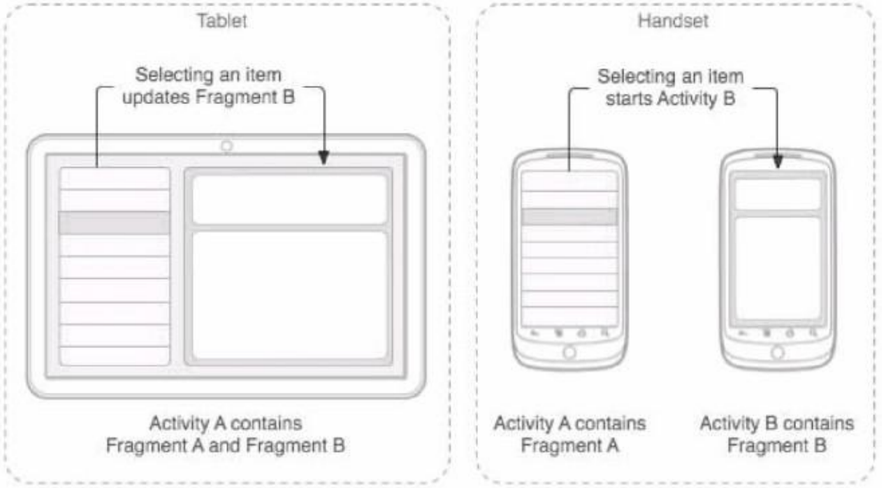


Figure: UI modules defined by fragments

- But with the introduction of fragment we got more flexibility and removed the limitation of having a single activity on the screen at a time. Now we can have a single activity but each activity can

consist of multiple fragments which will have their own layout, events and complete life cycle.

- Following is a typical example of how two UI modules defined by fragments can be combined into one activity for a tablet design, but separated for a handset design.
- The application can embed two fragments in Activity A, when running on a tablet-sized device. However, on a handset-sized screen, there's not enough room for both fragments, so Activity A includes only the fragment for the list of articles, and when the user selects an article, it starts Activity B, which includes the second fragment to read the article.

Fragment Lifecycle:

Android fragments have their own life cycle very similar to an android activity. This section briefs different stages of its life cycle.

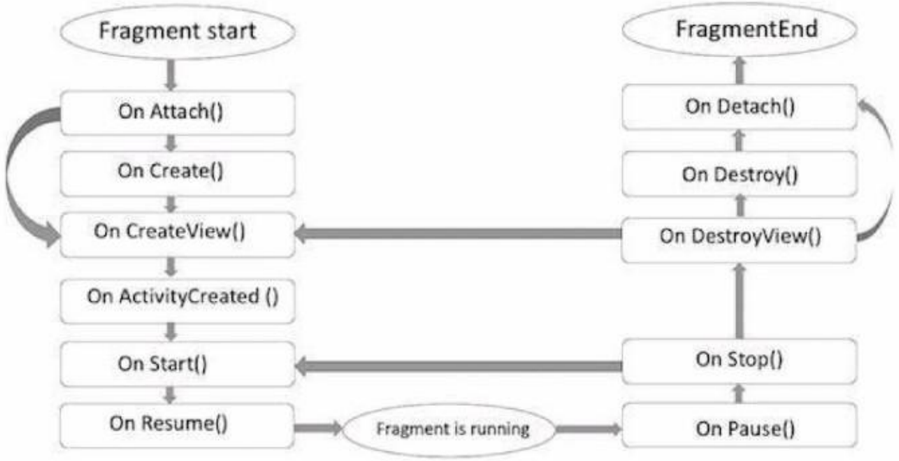


Figure: Fragments Life Cycle

Fragment lifecycle

- 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 instances have been created as well as the view hierarchy of the activity. At this point, the 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(): onDestroy() called to do final clean up of the fragment's state but Not guaranteed to be called by the Android platform.

Types of Fragments

Basically fragments are divided as three stages as shown below.

- Single frame fragments - Single frame fragments are using for hand hold devices like mobiles, here we can show only one fragment as a view.
- List fragments - fragments having special list view is called as list fragment
- Fragments transaction - Using with fragment transaction. we can move one fragment to another fragment.

Usage of Fragments:

This involves a number of simple steps to create Fragments.

- First of all decide how many fragments you want to use in an activity. For example, let's use two fragments to handle landscape and portrait modes of the device.
- Next based on the number of fragments, create classes which will extend the Fragment class. The Fragment class has above-mentioned callback functions. You can override any of the functions based on your requirements.
- Corresponding to each fragment, you will need to create layout files in XML file. These files will have layout for the defined fragments.
- Finally modify activity file to define the actual logic of replacing fragments based on your requirement.

MODULE-4

Q1. Create Custom Layouts (P4 - Appeared 1 Time) (3-7M)

ANS: Steps to Custom Layouts:

1. Click the View tab, then click the Slide Master button in the Presentation Views group. The slide master appears.
2. On the Slide Master tab, in the Edit Master group, choose Insert Layout. A new layout appears in the left pane.
3. Again on the Slide Master tab, in the Master Layout group, click the Insert Placeholder button's down arrow and choose one of the 8 placeholder types.
4. Drag on the slide to size and place the placeholder.
5. Place more placeholders, laying them out as needed.
6. When you're done, click the layout in the left pane, and display the Slide Master tab. In the Edit Master group, click the Rename button. Enter a name and click Rename.
7. The presentation now contains the new layout and you can choose it the same way you'd choose any of the standard layouts for any slide.
8. If you want to use the layout in the future, save the file as a template (.potx, or .potm if it contains macros) or theme (thmx).

Q2. Work with UI Components and Events OR Explain Event Handling in android (P4 - Appeared 1 Time) (3-7M)

ANS: Event Handling:

- Events are a useful way to collect data about a user's interaction with interactive components of Applications. Like button presses or screen touch etc.
- The Android framework maintains an event queue as a first-in, first-out (FIFO) basis. You can capture these events in your program and take appropriate action as per requirements.

There are following three concepts related to Android Event Management -

- Event Listeners - An event listener is an interface in the View class that contains a single callback method. These methods will be called by the Android framework when the View to which the listener has been registered is triggered by user interaction with the item in the UI.
- Event Listeners Registration - Event Registration is the process by which an Event Handler gets registered with an Event Listener so that the handler is called when the Event Listener fires the event.
- Event Handlers - When an event happens and we have registered an event listener for the event, the event listener calls the Event Handlers, which is the method that actually handles the event.

Event Listeners & Event Handlers:

Event Handler	Event Listeners	Description
onClick()	OnClickListener()	This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. Use onClick() event handler to handle such an event.
onLongClick()	OnLongClickListener()	This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. for one or more seconds. Use the onLongClick() event handler to handle such events.
onFocusChange()	OnFocusChangeListener()	This is called when the widget loses its focus ie. user goes away from the view item. Use onFocusChange() event handler to handle such events.
onKey()	OnFocusChangeListener()	This is called when the user is focused on the item and presses or releases a hardware key on the device. Use onKey() event handler to handle such an event.
onTouch()	OnTouchListener()	This is called when the user presses the key, releases the key, or any movement gesture on the screen. Use onTouch() event handler to handle such an event.

onMenuItemClick()	onCreateContextMenuItemListener()	This is called when the context menu is being built(as the result of a sustained "long click)

Event Listeners Registration

- Event Registration is the process by which an Event Handler gets registered with an Event Listener so that the handler is called when the Event Listener fires the event.

Q3. What is Tab Layout (P4 - Appeared 1 Time) (3-7M)

ANS: Tab Layout:

- TabLayout is used to implement horizontal tabs. TabLayout is released by Android after the deprecation of ActionBar.TabListener (API level 21). TabLayout is introduced in the design support library to implement tabs.
- Tabs are created using the newTab() method of the TabLayout class. The title and icon of Tabs are set through setText(int) and setIcon(int) methods of TabListener interface respectively.

Tabs of layout are attached over TabLayout using the addTab(Tab) method.

```
TabLayout tabLayout = (TabLayout)findViewById(R.id.tabLayout);
tabLayout.addTab(tabLayout.newTab().setText("Tab 1"));
tabLayout.addTab(tabLayout.newTab().setText("Tab 2"));
tabLayout.addTab(tabLayout.newTab().setText("Tab 3"));
```

Example of TabLayout using ViewPager:

File: activity.xml

- Create an activity.xml file with TabLayout and ViewPager view components.

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

   <android.support.design.widget.TabLayout
      android:id="@+id/tabLayout"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:background="#1db995">
   </android.support.design.widget.TabLayout>

   <android.support.v4.view.ViewPager
      android:id="@+id/viewPager"
      android:layout_width="355dp"
      android:layout_height="455dp"
      app:layout_constraintTop_toBottomOf="@+id/tabLayout"
      tools:layout_editor_absoluteX="8dp" />

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

File: build.gradle

```
implementation 'com.android.support:design:26.1.0'
```

File: MainActivity.java

- In this file, Implement two additional listeners
    1. addOnPageChangeListener(listener) of ViewPager which makes slides the different fragments of tabs
    2. addOnTabSelectedListener(listener) of TabLayout which select the current tab on tab selection.

```
package tablayout.example.com.tablayout;
import android.support.design.widget.TabLayout;
import android.support.v4.view.ViewPager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {
   TabLayout tabLayout;
   ViewPager viewPager;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      tabLayout=(TabLayout)findViewById(R.id.tabLayout);
      viewPager=(ViewPager)findViewById(R.id.viewPager);
      tabLayout.addTab(tabLayout.newTab().setText("Home"));
      tabLayout.addTab(tabLayout.newTab().setText("Sport"));
      tabLayout.addTab(tabLayout.newTab().setText("Movie"));
      tabLayout.setTabGravity(TabLayout.GRAVITY_FILL);
```

```
        final MyAdapter adapter = new
MyAdapter(this,getSupportFragmentManager(),
tabLayout.getTabCount());
        viewPager.setAdapter(adapter);
        viewPager.addOnPageChangeListener(new
TabLayout.TabLayoutOnPageChangeListener(tabLayout));
         tabLayout.addOnTabSelectedListener(new
TabLayout.OnTabSelectedListener() {
            @Override
            public void onTabSelected(TabLayout.Tab tab) {
               viewPager.setCurrentItem(tab.getPosition());
            }
            @Override
            public void onTabUnselected(TabLayout.Tab tab) {
            }
            @Override
            public void onTabReselected(TabLayout.Tab tab) {
            }
        });
    }
}
```

File: MyAdapter.java

```
package tablayout.example.com.tablayout;
import android.content.Context;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentPagerAdapter;
import android.support.v4.app.FragmentManager;
public class MyAdapter extends FragmentPagerAdapter {
```

```
  private Context myContext;
  int totalTabs;
  public MyAdapter(Context context, FragmentManager fm, int totalTabs) {
    super(fm);
    myContext = context;
    this.totalTabs = totalTabs;
  }
  // this is for fragment tabs
  @Override
  public Fragment getItem(int position) {
    switch (position) {
      case 0:
        HomeFragment homeFragment = new HomeFragment();
        return homeFragment;
      case 1:
        SportFragment sportFragment = new SportFragment();
        return sportFragment;
      case 2:
        MovieFragment movieFragment = new MovieFragment();
        return movieFragment;
      default:
        return null;
    }
  }
// this counts total number of tabs
  @Override
  public int getCount() {
    return totalTabs;
  }
```

```
}
```

Now create different fragment files for all different tabs.

File: HomeFragment.java

```
package tablayout.example.com.tablayout;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
public class HomeFragment extends Fragment {
  public HomeFragment() {
    // Required empty public constructor
  }
  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container,
                Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    return inflater.inflate(R.layout.fragment_home, container, false);
  }
}
```

File: fragment_home.xml

```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
```

```
    tools:context="tablayout.example.com.tablayout.HomeFragment">
    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="@string/home_fragment" />
</FrameLayout>
```

File: SportFragment.java

```
package tablayout.example.com.tablayout;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
public class SportFragment extends Fragment {
    public SportFragment() {
        // Required empty public constructor
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_sport, container, false);
    }
}
```

File: fragment_sport.xml

```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context="tablayout.example.com.tablayout.SportFragment">
   <!-- TODO: Update blank fragment layout -->
   <TextView
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:gravity="center"
      android:text="@string/sport_fragment" />
</FrameLayout>
```

File: MovieFragment.java

```
package tablayout.example.com.tablayout;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class MovieFragment extends Fragment {
   public MovieFragment() {
      // Required empty public constructor
   }
   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup container,
                            Bundle savedInstanceState) {
```

```
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_movie, container, false);
    }
}
```

File: fragment_movie.xml

```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="tablayout.example.com.tablayout.MovieFragment">
    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="@string/movie_fragment" />

</FrameLayout>
```

File: strings.xml

```
<resources>
    <string name="app_name">TabLayout</string>

    <!-- TODO: Remove or change this placeholder text -->
    <string name="home_fragment">Home Fragment</string>
    <string name="sport_fragment">Sport Fragment</string>
    <string name="movie_fragment">Movie Fragment</string>
```

Q4. Differentiate RecyclerView and Card View OR Explain Card View using RecyclerView (P4 - Appeared 1 Time) (3-7M)

ANS: Card View using RecyclerView:
RecyclerView is an extended version of ListView and GridView. It works on the ViewHolder design pattern. With the help of RecyclerView, we can add many extra features to our list of data. Before starting our example on implementation of CardView in RecyclerView. We should know what CardView and RecyclerView mean.

- CardView: CardView is an extended version of Framelayout which can be used to show items inside the card format. With the help of CardView, we can add radius, elevation to our items of RecyclerView. CardView gives a rich look and feel to our list of data.
- RecyclerView: RecyclerView is an extended version of ListView. in RecyclerView we can load a large amount of data and items of RecyclerView can have a custom design. RecyclerView works on a ViewHolder design pattern so we have to create a Data class that holds data for RecyclerView and a ViewHolder class which will set data to each item of RecyclerView.

RecyclerView is divided into 3 sections:

- Card Layout.
- Modal Class.
- ViewHolder class.

Step by Step Implementation:
Step 1: Create a New Project

- To create a new project in Android Studio. Note that select Java as the programming language.

Step 2: Add dependency for creating CardView and RecyclerView

- Navigate to the Gradle Scripts > build.gradle(Module:app) and add dependency in the dependency section.

Step 3: Create RecyclerView Card Layout

- Card Layout: Card Layout is used to display a list of data. It is the design of a single item of our RecyclerView. For creating a Card Layout navigate to the app > res > layout > Right-Click on it > New > Layout Resource File > Give a name to it(here card_layout).

Step 4: Create a Model Class for storing data

- Navigate to the app > java > your apps package name > Right-click on it > New > Java and name the Modal Class(here CourseModel). Model Class will store the data which we will display in our Recycler View. Below is the code for the CourseModel.java file.

Step 5: Create Adapter Class for setting data to items of RecyclerView

- Navigate to the app > java > your apps package name > Right Click on it > New > Java Class and name your Adapter Class(Here CourseAdapter). Adapter Class in RecyclerView will get the data from your Modal Class and set that data to your item of RecyclerView. Below is the code for the CourseAdapter.java file.

Step 6: Now we will move towards creating our RecyclerView

- For creating our RecyclerView. Navigate to the app > res > layout > activity_main.xml and add RecyclerView

Step 7: Now we will initialize our RecyclerView in our MainActivity.java

- Navigate to the app > java > your apps package name > MainActivity.java and initialize your RecyclerView. Below is the code for the MainActivity.java file. Now run the app on Emulator and see the output.

Q5. Explain Android Menus (P4 - Appeared 1 Time) (3-7M)

ANS: Android Menus:
Android provides a standard XML format to define menu items. Instead of building a menu, define a menu and all its items in an XML menu resource and load the menu resource as a Menu object. In android, to define menu, create a new folder menu inside of our project resource directory (res/menu/) and add a new XML file to build the menu with the following elements.

Element	Description
<menu>	It's a root element to define a Menu in an XML file and it will hold one or more elements.
<item>	It is used to create a menu item and it represents a single item on the menu. This element may contain a nested <menu> element in order to create a submenu.
<group>	It's optional and invisible for <item> elements. It is used to categorize the menu items so they share properties such as active state and visibility.

Android Different Types of Menus
In android, there are three fundamental types of Menus available to define a set of options and actions in our android applications.
The following are the commonly used Menus in android applications.

- Options Menu

- Context Menu
- Popup Menu

Android Options Menu

- In android, Options Menu is a primary collection of menu items for an activity and it is useful to implement actions that have a global impact on the app, such as Settings, Search, etc.

Android Context Menu

- In android, Context Menu is a floating menu that appears when the user performs a long click on an element and it is useful to implement actions that affect the selected content or context frame.

Android Popup Menu

- In android, Popup Menu displays a list of items in a vertical list that's anchored to the view that invoked the menu and it's useful for providing an overflow of actions that are related to specific content.

Q1. Differentiate Shared Preferences and Shared Preferences Layout

OR Explain Shared Preferences (P4 - Appeared 1 Time) (3-7M)

ANS: Shared Preferences:

- Shared Preferences is the way in which one can store and retrieve small amounts of primitive data as key/value pairs to a file on the device storage such as String, int, float, Boolean that make up your preferences in an XML file inside the app on the device storage. Shared Preferences class provides APIs for reading, writing, and managing this data.

Following are the methods of Shared Preferences

1. contains(String key): This method is used to check whether the preferences contain a preference.
2. edit(): This method is used to create a new Editor for these preferences, through which you can make modifications to the data in the preferences and atomically commit those changes back to the SharedPreferences object.
3. getAll(): This method is used to retrieve all values from the preferences.
4. getBoolean(String key, boolean defValue): This method is used to retrieve a boolean value from the preferences.
5. getFloat(String key, float defValue): This method is used to retrieve a float value from the preferences.
6. getInt(String key, int defValue): This method is used to retrieve an int value from the preferences.

7. getLong(String key, long defValue): This method is used to retrieve a long value from the preferences.
8. getString(String key, String defValue): This method is used to retrieve a String value from the preferences.
9. getStringSet(String key, Set defValues): This method is used to retrieve a set of String values from the preferences.
10. registerOnSharedPreferencechangeListener(SharedPreferences.OnSharedPreferencechangeListener listener): This method is used to register a callback to be invoked when a change happens to a preference.
11. unregisterOnSharedPreferencechangeListener(SharedPreferences.OnSharedPreferencechangeListener listener): This method is used to unregister a previous callback.

Nested classes of Shared Preferences

1. SharedPreferences.Editor: Interface used to write(edit) data in the SP file. Once editing has been done, one must commit() or apply() the changes made to the file.
2. SharedPreferences.OnSharedPreferenceChangeListener(): Called when a shared preference is changed, added, or removed. This may be called even if a preference is set to its existing value. This callback will be run on your main thread.

Creation of Shared Preferences:

- The first thing we need to do is to create one shared preferences file per app. So name it with the package name of your app- unique and easy to associate with the app.
- When you want to get the values, call the getSharedPreferences() method. Shared Preferences provide modes of storing the data (private mode and public mode). It is for backward compatibility- use only MODE_PRIVATE to be secure.

1. public abstract SharedPreferences getSharedPreferences (String name, int mode)

- This method takes two arguments, the first being the name of the SharedPreference(SP) file and the other is the context mode that we want to store our file in.
    1. MODE_PUBLIC will make the file public which could be accessible by other applications on the device
    2. MODE_PRIVATE keeps the files private and secures the user's data.
    3. MODE_APPEND is used while reading the data from the SP file.

Q2. Explain Android Requesting (P4 - Appeared 1 Time) (3-7M)

ANS: Android Requesting:

- Starting from Android 6.0 (API 23), users are not asked for permissions at the time of installation rather developers need to request the permissions at the run time. Only the permissions that are defined in the manifest file can be requested at run time.

Types of Permissions

1. Install-Time Permissions: If the Android 5.1.1 (API 22) or lower, the permission is requested at the installation time at the Google Play Store. If the user Accepts the permissions, the app is installed. Else the app installation is canceled.
2. Run-Time Permissions: If the Android 6 (API 23) or higher, the permission is requested at the run time during the running of the app. If the user Accepts the permissions, then that feature of the app can be used. Else to use the feature, the app requests

permission again. So, now the permissions are requested at runtime. In this article, we will discuss how to request permissions in an Android Application at run time.

Steps for Requesting permissions at run time

- Step 1: Declare the permission in the Android Manifest file: In Android, permissions are declared in the AndroidManifest.xml file using the uses-permission tag.
  <uses-permission
  android:name="android.permission.PERMISSION_NAME"/>
- Step 2: Modify activity_main.xml file to Add two buttons to request permission on button click: Permission will be checked and requested on button click. Open the activity_main.xml file and add two buttons to it.
- Step 3: Check whether permission is already granted or not. If permission isn't already granted, request the user for the permission: In order to use any service or feature, the permissions are required. Hence we have to ensure that the permissions are given for that. If not, then the permissions are requested. Check for permissions: Beginning with Android 6.0 (API level 23), the user has the right to revoke permissions from any app at any time, even if the app targets a lower API level. So to use the service, the app needs to check for permissions every time.
  Syntax:

```
if(ContextCompat.checkSelfPermission(thisActivity,
Manifest.permission.WRITE_CALENDAR)
  != PackageManager.PERMISSION_GRANTED)
{
  // Permission is not granted
```

}

Request Permissions: When PERMISSION_DENIED is returned from the checkSelfPermission() method in the above syntax, we need to prompt the user for that permission. Android provides several methods that can be used to request permission, such as requestPermissions().

Syntax:

```
ActivityCompat.requestPermissions(MainActivity.this,
                                  permissionArray,
                                  requestCode);
```

```
// Function to check and request permission
public void checkPermission(String permission, int requestCode)
{
    // Checking if permission is not granted
    if (ContextCompat.checkSelfPermission(MainActivity.this,
permission) == PackageManager.PERMISSION_DENIED) {
        ActivityCompat.requestPermissions(MainActivity.this, new
String[] { permission }, requestCode);
    }
    else {
        Toast.makeText(MainActivity.this, "Permission already
granted", Toast.LENGTH_SHORT).show();
    }
}
```

This function will show a Toast message if permission is already granted otherwise prompt the user for permission.

- Step 4: Override onRequestPermissionsResult() method: onRequestPermissionsResult() is called when user grant or decline the permission. RequestCode is one of the parameters of this function which is used to check user action for the corresponding requests. Here a toast message is shown indicating the permission and user action.

Q3. Permission at run time (Android 6.0) (P4 - Appeared 1 Time) (3-7M)

ANS: Permissions at run time:
Run-Time Permissions: If the Android 6 (API 23) or higher, the permission is requested at the run time during the running of the app. If the user Accepts the permissions, then that feature of the app can be used. Else to use the feature, the app requests permission again. So, now the permissions are requested at runtime. In this article, we will discuss how to request permissions in an Android Application at run time.

Steps for Requesting permissions at run time

- Step 1: Declare the permission in the Android Manifest file: In Android, permissions are declared in the AndroidManifest.xml file using the uses-permission tag.
  <uses-permission
  android:name="android.permission.PERMISSION_NAME"/>
- Step 2: Modify activity_main.xml file to Add two buttons to request permission on button click: Permission will be checked and requested on button click. Open the activity_main.xml file and add two buttons to it.

- Step 3: Check whether permission is already granted or not. If permission isn't already granted, request the user for the permission: In order to use any service or feature, the permissions are required. Hence we have to ensure that the permissions are given for that. If not, then the permissions are requested. Check for permissions: Beginning with Android 6.0 (API level 23), the user has the right to revoke permissions from any app at any time, even if the app targets a lower API level. So to use the service, the app needs to check for permissions every time.

  Syntax:

```
if(ContextCompat.checkSelfPermission(thisActivity,
Manifest.permission.WRITE_CALENDAR)
  != PackageManager.PERMISSION_GRANTED)
{
   // Permission is not granted
}
```

  Request Permissions: When PERMISSION_DENIED is returned from the checkSelfPermission() method in the above syntax, we need to prompt the user for that permission. Android provides several methods that can be used to request permission, such as requestPermissions().

  Syntax:

```
ActivityCompat.requestPermissions(MainActivity.this,
                                  permissionArray,
                                  requestCode);
```

```
// Function to check and request permission
public void checkPermission(String permission, int requestCode)
{
    // Checking if permission is not granted
    if (ContextCompat.checkSelfPermission(MainActivity.this,
permission) == PackageManager.PERMISSION_DENIED) {
        ActivityCompat.requestPermissions(MainActivity.this, new
String[] { permission }, requestCode);
    }
    else {
        Toast.makeText(MainActivity.this, "Permission already
granted", Toast.LENGTH_SHORT).show();
    }
}
```

This function will show a Toast message if permission is already granted otherwise prompt the user for permission.

- Step 4: Override onRequestPermissionsResult() method: onRequestPermissionsResult() is called when user grant or decline the permission. RequestCode is one of the parameters of this function which is used to check user action for the corresponding requests. Here a toast message is shown indicating the permission and user action.

Q4. Explain Database in Android (P4 - Appeared 1 Time) (3-7M)

ANS: SQL Database:

- SQLite is a opensource SQL database that stores data to a text file on a device. Android comes in with built in SQLite database implementation.
- SQLite supports all the relational database features. In order to access this database, you don't need to establish any kind of connections for it like JDBC,ODBC e.t.c
- The main package is android.database.sqlite that contains the classes to manage your own databases

Database Creation:

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

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

Database Insertion:

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

Database Fetching:

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

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

Q5. Explain Realm-No SQL Database (P4 - Appeared 1 Time) (3-7M)

ANS: Realm Database:

- Realm is a database that is the perfect alternative to SQLite. It's is a NoSQL database designed for mobile platform.
- Its core consists of a self-contained C++ library. It supports Android and iOS both.

Features:

- As Realm is an object store, its typed language-specific APIs map typed objects directly into the Realm file – therefore classes are used as the schema definition.
- Relationships between objects are allowed via "links". Each "link" creates a "backlink" as an inverse relationship to whichever objects are linking to the current object.

- The query results returned by Realm are thread-local views to the current "database version" (as Realm handles concurrency with MVCC architecture), and these views "automatically update" when a transaction is committed from any thread, as long as Realm is able to update its instance version (which is possible on threads that are able to receive change notifications). When this happens, Realm calls change listeners that are added to its query results (if they've changed).
- Each thread-local view returns proxy objects that only read from/write to the database when an accessor method is called, meaning all database access is lazy-loaded. Writes are allowed only while in a write transaction.
- As each query result and each proxy object is a view to the underlying data, any change made to the database is reflected in all objects that point to the same data. Realm generally calls this behavior "zero-copy architecture" (along with the previously mentioned lazy-loaded data access).

Advantages of Realm

- Simplicity – Unlike SQLite the code is much shorter and concise. Besides, instead of using SQL, you just need to deal with objects and object trees in Realm.
- Speed – Despite having a complex algorithm underneath, Realm performs CRUD operations faster. Obtaining objects is very fast since there is no deserialization
- Live Objects – Realm has a no copy behaviour. All fetches are lazy, and the data is never copied until you perform some operations on them. So till then from a query, all that you get is pointers to data. Hence all objects received from Realm are proxy to the database.

Due to this, zero copy is achieved. Whenever you need to access the data you will always get the latest value.

- Great Documentation – Realm documentation is clear and the community support is great.
- JSON Support Built-in – Realm has JSON support. You can directly set data from JSON without the need to create model classes.
- Security – You can secure the database using encryption.
- Reactive Programming – You can observe changes in the database and update the UI accordingly. Using RxJava along with Realm just makes it a lot more simple and fun.
- Built-in Adapters – Realm has Adapter classes for the Android UI components.

Disadvantages of Realm

- No Auto-increment – You cannot auto-increment values in Realm.
- Restrictions on Model classes – Besides the getter setter methods you cannot override methods like hashcode and equals in the Realm Model classes.
- Threading – Real model classes can't be passed from one thread to another. Hence you'll have to query the class again on the different thread.

MODULE-6

Q1. **Differentiate Web services and Parsing OR Difference between web services and API (P4 - Appeared 1 Time) (3-7M)**

ANS:

No.	Web Services	Web API
1.	Web services are a type of API, which must be accessed through a network connection.	APIs are application interfaces, implying that one application can communicate with another application in a standardized manner.
2.	Web service is used for REST, SOAP and XML-RPC for communication.	API is used for any style of communication.
3.	All Web services are APIs.	APIs are not web services.
4.	It doesn't have lightweight design, needs a SOAP convention to send or receive data over the system.	It has a light-weight architecture furthermore, useful for gadgets which have constrained transmission capacity like smart phones

5.	It provides supports only for the HTTP protocol.	It provides support for the HTTP/s protocol: URL Request/Response Headers, and so on.
6.	It is not open source, however, can be devoured by any customer that comprehends xml.	It is an open source and also ships with .NET framework.
7.	Web service supports only XML.	API supports XML and JSON.
8.	Web Services can be hosted on IIS.	Web API can be hosted only on IIS and self.

Q2. Explain JSON Parsing (P4 - Appeared 1 Time) (3-7M)

ANS: JSON:

- JSON stands for JavaScript Object Notation.It is an independent data exchange format and is the best alternative for XML. This chapter explains how to parse the JSON file and extract necessary information from it.
- Android provides four different classes to manipulate JSON data. These classes are JSONArray,JSONObject,JSONStringer and JSONTokenizer.

JSON Elements:

No.	Component	Description
1.	Array([)	In a JSON file , square bracket ([) represents a JSON array
2.	Objects({)	In a JSON file, curly bracket ({) represents a JSON object
3.	Key	A JSON object contains a key that is just a string. Pairs of key/value make up a JSON object
4.	Value	Each key has a value that could be string , integer or double e.t.c

JSON - Parsing

- For parsing a JSON object, Create an object of class JSONObject and specify a string containing JSON data to it. Its syntax is -
  String in;
  JSONObject reader = new JSONObject(in);
- The last step is to parse the JSON. A JSON file consist of different object with different key/value pair e.t.c. So JSONObject has a separate function for parsing each of the component of JSON file. Its syntax is given below -
  JSONObject sys = reader.getJSONObject("sys");
  country = sys.getString("country");
  JSONObject main = reader.getJSONObject("main");
  temperature = main.getString("temp");

- The method getJSONObject returns the JSON object. The method getString returns the string value of the specified key.

No.	Methods	Description
1.	get(String name)	This method just Returns the value but in the form of Object type
2.	getBoolean(String name)	This method returns the boolean value specified by the key
3.	getDouble(String name)	This method returns the double value specified by the key
4.	getInt(String name)	This method returns the integer value specified by the key
5.	getLong(String name)	This method returns the long value specified by the key
6.	length()	This method returns the number of name/value mappings in this object..
7.	names()	This method returns an array containing the string names in this object.

Q3. Access web data with JSON OR Example of android JSON Parsing (P4 - Appeared 1 Time) (3-7M)

ANS: Code:

Activity_main.xml

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

  <TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:layout_marginLeft="75dp"
    android:layout_marginTop="46dp"
    android:text="TextView" />

</RelativeLayout>
```

MainActivity.java

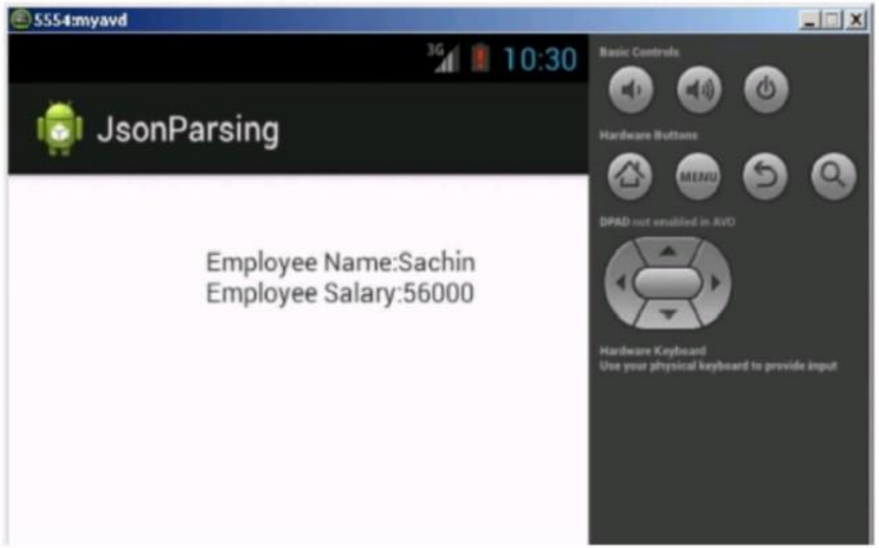
```
package com.javatpoint.json parsing;
import org.json.JSONException;
import org.json.JSONObject;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends Activity {
```

```
public static final String
JSON_STRING="{\"employee\":{\"name\":\"Sachin\",\"salary\":560
00}}";

@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
TextView textView1=(TextView)findViewById(R.id.textView1);
try{
JSONObject emp=(new
JSONObject(JSON_STRING)).getJSONObject("employee");
String empname=emp.getString("name");
int empsalary=emp.getInt("salary");
String str="Employee Name:"+empname+"\n"+"Employee
Salary:"+empsalary;
textView1.setText(str);
}catch (Exception e) {e.printStackTrace();}
}
}
```

Output:

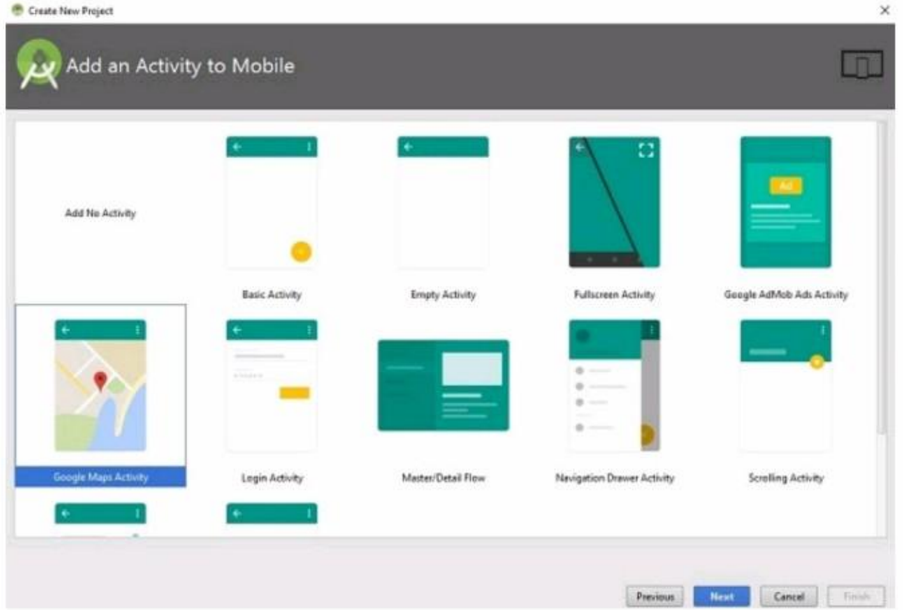

5554:myavd
10:30
JsonParsing
Employee Name:Sachin
Employee Salary:56000
Basic Controls
Hardware Buttons
MENU
DPAD not enabled in AVD
Hardware Keyboard
Use your physical keyboard to provide input

MODULE-7

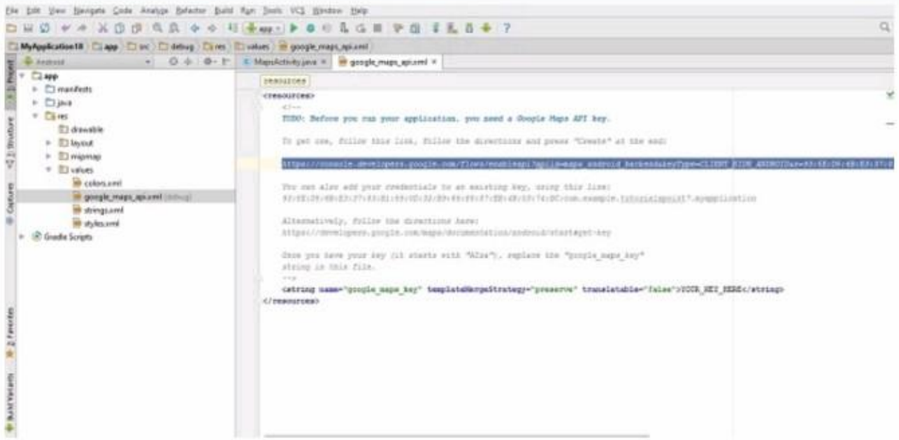
Q1. Creating Google Map (P4 - Appeared 1 Time) (3-7M)

ANS:

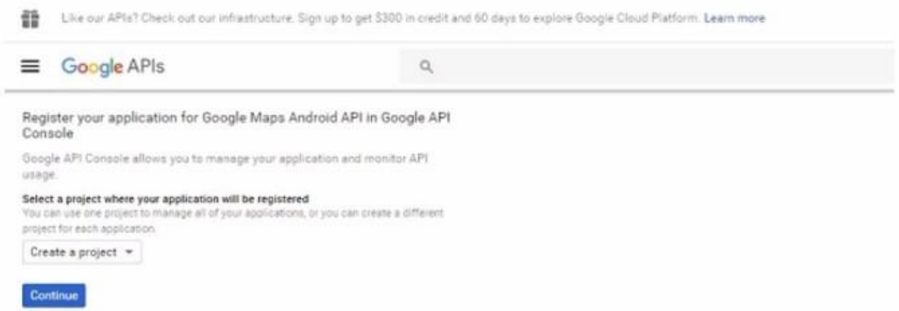
Create a project with google maps activity as shown below -



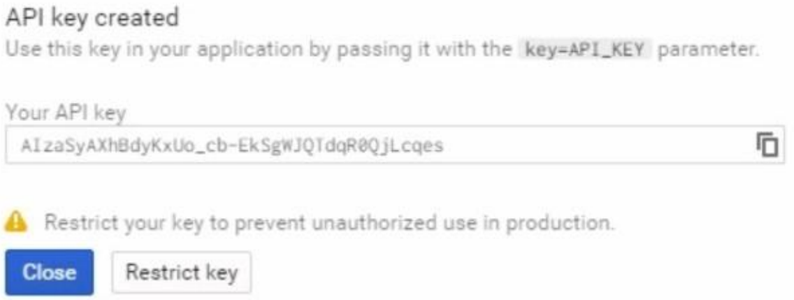
It will open the following screen and copy the console url for API Key as shown below -



Copy this and paste it to your browser. It will give the following screen -



Click on continue and click on Create API Key then it will show the following screen



Here is the content of activity_main.xml.

```
<fragment
xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:map="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:id="@+id/map"

android:name="com.google.android.gms.maps.SupportMapFragm
ent"
  android:layout_width="match_parent"
  android:layout_height="match_parent"

tools:context="com.example.tutorialspoint7.myapplication.MapsActi
vity" />
```

Content of MapActivity.java.

In the below code we have given sample latitude and longitude details

```
import android.support.v4.app.FragmentActivity;
import android.os.Bundle;
```

```
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

public class MapsActivity extends FragmentActivity implements
OnMapReadyCallback {

  private GoogleMap mMap;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_maps);
    // Obtain the SupportMapFragment and get notified when the
map is ready to be used.
    SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
        .findFragmentById(R.id.map);
    mapFragment.getMapAsync(this);
  }

  /**
    * Manipulates the map once available.
    * This callback is triggered when the map is ready to be used.
    * This is where we can add markers or lines, add listeners or
move the camera.
```

```
    * In this case, we just add a marker near Sydney, Australia.
    * If Google Play services is not installed on the device.
    * This method will only be triggered once the user has installed
      Google Play services and returned to the app.
  */

  @Override
  public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
    // Add a marker in Sydney and move the camera
    LatLng TutorialsPoint = new LatLng(21, 57);
    mMap.addMarker(new

MarkerOptions().position(TutorialsPoint).title("Tutorialspoint.com"));

mMap.moveCamera(CameraUpdateFactory.newLatLng(TutorialsPo
int));
  }
}
```

Following is the content of AndroidManifest.xml file.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.tutorialspoint7.myapplication">

  <!--
    The ACCESS_COARSE/FINE_LOCATION permissions are not
required to use
```

```
    Google Maps Android API v2, but you must specify either coarse
or fine
    location permissions for the 'MyLocation' functionality.
  -->

  <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
  <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
  <uses-permission android:name="android.permission.INTERNET" />
  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">

    <!--
      The API key for Google Maps-based APIs is defined as a string
resource.
      (See the file "res/values/google_maps_api.xml").
      Note that the API key is linked to the encryption key used to sign
the APK.
      You need a different API key for each encryption key, including
the release key
      that is used to sign the APK for publishing.
      You can define the keys for the debug and
        release targets in src/debug/ and src/release/.
    -->
```

```
    <meta-data
      android:name="com.google.android.geo.API_KEY"

android:value="AIzaSyAXhBdyKxUo_cb-EkSgWJQTdqR0QjLcqes" />

    <activity
      android:name=".MapsActivity"
      android:label="@string/title_activity_maps">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER"
/>
      </intent-filter>
    </activity>
  </application>

</manifest>
```

Output -

Q2. Differentiate Location service with Location Manager OR Explain android Location based services (P4 - Appeared 1 Time) (3-7M)

ANS: Location Object: The Location object represents a geographic location which can consist of a latitude, longitude, time stamp, and other information such as bearing, altitude and velocity.

No.	Method	Description
1.	float distanceTo(Location dest)	Returns the approximate distance in meters between this location and the given location.
2.	float getAccuracy()	Get the estimated accuracy of this location, in meters.
3.	double getAltitude()	Get the altitude if available, in meters above sea level.
4.	float getBearing()	Get the bearing, in degrees.
5.	double getLatitude(),double getLongitude()	Get the latitude, in degrees. Get the longitude, in degrees.
6.	void setSpeed(float speed)	Set the speed, in meters/second over ground.

Location Quality of service:
The LocationRequest object is used to request a quality of service (QoS) for location updates from the LocationClient.

No.	Methods	Descriptions
1.	setExpirationDuration(long millis)	Set the duration of this request, in milliseconds.

2.	setExpirationTime(long millis)	Set the request expiration time, in millisecond since boot.
3.	setFastestInterval(long millis)	Explicitly set the fastest interval for location updates, in milliseconds.
4.	setInterval(long millis)	Set the desired interval for active location updates, in milliseconds.
5.	setNumUpdates(int num Updates)	Set the number of location updates.
6.	setPriority(int priority)	Set the priority of the request.

Updated Location:

No.	Method	Description
1.	abstract void onLocationChanged(Location location)	This callback method is used for receiving notifications from the LocationClient when the location has changed.

Displaying Location Address:

- Once you have a Location object, Geocoder.getFromLocation() method to get an address for a given latitude and longitude. This method is synchronous, and may take a long time to do its work, so you should call the method from the doInBackground() method of an AsyncTask class.

- The AsyncTask must be subclassed to be used and the subclass will override doInBackground(Params...) method to perform a task in the background and onPostExecute(Result) method is invoked on the UI thread after the background computation finishes and at the time to display the result.
- There is one more important method available in AsyncTask which is execute(Params... params), this method executes the task with the specified parameters.

Q3. Work with 2D Graphics OR Difference between 2D graphics and 3D graphics (P4 - Appeared 1 Time) (3-7M)

ANS:

No.	2D	3D
1.	Biometric-face-recognition-system using 2D model is not faster than 3D	Biometric-face-recognition-system using 3D model is very faster than 2D
2.	2D model databases availability is high	2D model databases availability is less.
3.	Biometric-face-recognition-system using 2D model databases takes into consideration only the two dimensions of the face	Biometric-face-recognition-system using 3D model databases takes into consideration only the three dimensions of the face

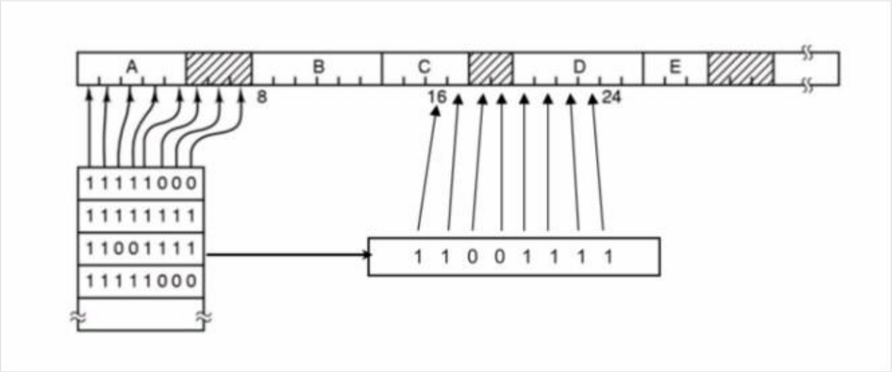
4.	2D represents a face by the intensity variation	3D represents a face by shape variation
5.	In 2D approach, facial features recognized based on measurements such as distance between the eyes, width of the nose	While in 3D approach, recognize the facial features the contours of the nose, chin, eye socket are used.
6.	In 2D face orientation accommodated up to around 15 to 20 degrees	In 3D face orientation up to 90 degrees can be accommodated
7.	Web camera/ digital camera used in 2D face recognition	Stereoscopic/ range camera are used in 3D face recognition.

Q4. Explain Bitmap (P4 - Appeared 1 Time) (3-7M)

ANS: Bitmap:

- A bitmap is a mapping from one system such as integers to bits. It is also known as bitmap index or a bit array.
- The memory is divided into units for bitmap.
- These units may range from a few bytes to several kilobytes.
- Each memory unit is associated with a bit in the bitmap. If the unit is occupied, the bit is 1 and if it is empty, the bit is zero.
- The bitmap provides a relatively easy way to keep track of memory as the size of the bitmap is only dependent on the size of the memory and the size of the units.

An image that clarifies the use of bitmap is as follows -



- The bitmap given in the image writes 1 for the occupied memory unit and 0 for the unoccupied memory unit. The first 5 units are occupied with A and the corresponding entry in the bitmap is 11111.
- The next three units are empty so their entry in the bitmap is 000. After that, 6 units occupy B. So their entry in the bitmap is 11111. This continues and the result obtained in the bitmap for A, B, C, D and E is shown in the image.

Key Features of Bitmap

- The unit size in bitmaps is very important and should be chosen with care.
- If the unit size is smaller, the bitmap will be larger as it will hold the value 0 or 1 for each of the units. Similarly, if the unit size is larger, the bitmap will be smaller.
- The unit size need not be too large, as even with a unit size as small as 3 bytes, only one bit of the bitmap can represent 24 bits.,

Advantage of Bitmap

- The bitmap is quite useful as it provides a way to keep track of the memory using only a little memory for the bitmap table. The size of

the bitmap is purely dependent on the size of the memory as well as the size of the memory unit.

Disadvantage of Bitmap

- A major problem in the bitmap occurs if a 'n' size memory block needs to be occupied by a process. Then a 'n' size vacancy is needed in the bitmap where the values are all zeroes.

Q5. Explain Animation and Frame Animation, Tween Animation and View Animation (P4 - Appeared 1 Time) (3-7M)

ANS: Animation:

An animation resource can define one of two types of animations:

- Property Animation: Creates an animation by modifying an object's property values over a set period of time with an Animator.
- View Animation

There are two types of animations that you can do with the view animation framework:

- Tween animation: Creates an animation by performing a series of transformations on a single image with an Animation
- Frame animation: or creates an animation by showing a sequence of images in order with an AnimationDrawable.

Property animation

- An animation defined in XML that modifies properties of the target object, such as background color or alpha value, over a set amount of time.

file location:

- res/animator/filename.xml- The filename will be used as the resource ID.

compiled resource datatype:

- Resource pointer to a ValueAnimator, ObjectAnimator, or AnimatorSet.

resource reference:

- In Java-based or Kotlin code: R.animator.filename
- In XML: @[package:]animator/filename

Syntax:

```
<set
 android:ordering=["together" | "sequentially"]>

  <objectAnimator
    android:propertyName="string"
    android:duration="int"
    android:valueFrom="float | int | color"
    android:valueTo="float | int | color"
    android:startOffset="int"
    android:repeatCount="int"
    android:repeatMode=["restart" | "reverse"]
    android:valueType=["intType" | "floatType"]/>

  <animator
    android:duration="int"
    android:valueFrom="float | int | color"
    android:valueTo="float | int | color"
    android:startOffset="int"
    android:repeatCount="int"
    android:repeatMode=["restart" | "reverse"]
    android:valueType=["intType" | "floatType"]/>
```

```
    <set>
      ...
    </set>
  </set>
```

- The file must have a single root element: either <set>, <objectAnimator>, or <valueAnimator>. You can group animation elements together inside the <set> element, including other <set> elements.

Elements:

<set>

- A container that holds other animation elements (<objectAnimator>, <valueAnimator>, or other <set> elements). Represents an AnimatorSet.

<objectAnimator>

- Animates a specific property of an object over a specific amount of time. Represents an ObjectAnimator.

Attributes:

android:propertyName

- String. Required. The object's property to animate, referenced by its name. For example you can specify "alpha" or "backgroundColor" for a View object. The objectAnimator element does not expose a target attribute, however, so you cannot set the object to animate in the XML declaration. You have to inflate your animation XML resource by calling loadAnimator() and call setTarget() to set the target object that contains this property.

android:valueTo

- float, int, or color. Required. The value where the animated property ends. Colors are represented as six digit hexadecimal numbers (for example, #333333).

android:valueFrom

- float, int, or color. The value where the animated property starts. If not specified, the animation starts at the value obtained by the property's get method. Colors are represented as six digit hexadecimal numbers (for example, #333333).

android:duration

- int. The time in milliseconds of the animation. 300 milliseconds is the default.

android:startOffset

- int. The amount of milliseconds the animation delays after start() is called.

android:repeatCount

- int. How many times to repeat an animation. Set to "-1" to infinitely repeat or to a positive integer. For example, a value of "1" means that the animation is repeated once after the initial run of the animation, so the animation plays a total of two times. The default value is "0", which means no repetition.

android:repeatMode

- int. How an animation behaves when it reaches the end of the animation. android:repeatCount must be set to a positive integer or "-1" for this attribute to have an effect. Set to "reverse" to have the animation reverse direction with each iteration or "restart" to have the animation loop from the beginning each time.

android:valueTo

- float, int, or color. Required. The value where the animation ends. Colors are represented as six digit hexadecimal numbers (for example, #333333).

android:valueFrom

- float, int, or color. Required. The value where the animation starts. Colors are represented as six digit hexadecimal numbers (for example, #333333).

android:duration

- int. The time in milliseconds of the animation. 300ms is the default.

android:startOffset

- int. The amount of milliseconds the animation delays after start() is called.

android:repeatCount

- int. How many times to repeat an animation. Set to "-1" to infinitely repeat or to a positive integer. For example, a value of "1" means that the animation is repeated once after the initial run of the animation, so the animation plays a total of two times. The default value is "0", which means no repetition.

android:repeatMode

- int. How an animation behaves when it reaches the end of the animation. android:repeatCount must be set to a positive integer or "-1" for this attribute to have an effect. Set to "reverse" to have the animation reverse direction with each iteration or "restart" to have the animation loop from the beginning each time.

android:valueType

- Keyword. Do not specify this attribute if the value is a color. The animation framework automatically handles color values.

Q6. Explain Multimedia in Android OR Explain media player in android (P4 - Appeared 1 Time) (3-7M)

ANS: Media Player:

- Android provides many ways to control playback of audio/video files and streams. One of this way is through a class called MediaPlayer.
- Android is providing MediaPlayer class to access built-in mediaplayer services like playing audio,video e.t.c. In order to use MediaPlayer, we have to call a static Method create() of this class. This method returns an instance of MediaPlayer class. Its syntax is as follows -
  MediaPlayer mediaPlayer = MediaPlayer.create(this, R.raw.song);
- The second parameter is the name of the song that you want to play. You have to make a new folder under your project with name raw and place the music file into it.
- Once you have created the Mediaplayer object you can call some methods to start or stop the music. These methods are listed below.
  mediaPlayer.start();
  mediaPlayer.pause();
- On the 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 is left and not from the beginning.
- In order to start music from the beginning, you have to call reset() method. Its syntax is given below.
  mediaPlayer.reset();

- Apart from the start and pause method, there are other methods provided by this class for better dealing with audio/video files. These methods are listed below -

No.	Methods	Description
1.	isPlaying()	This method just returns true/false indicating the song is playing or not
2.	seekTo(position)	This method takes an integer, and move song to that particular position millisecond
3.	getCurrentPosition()	This method returns the current position of song in milliseconds
4.	getDuration()	This method returns the total time duration of song in milliseconds
5.	reset()	This method resets the media player
6.	release()	This method releases any resource attached with MediaPlayer object
7.	setVolume(float leftVolume, float rightVolume)	This method sets the up down volume for this player
8.	setDataSource(FileDescriptor fd)	This method sets the data source of audio/video file

9.	selectTrack(int index)	This method takes an integer, and select the track from the list on that particular index
10.	getTrackInfo()	This method returns an array of track information

Q7. Explain Broadcast Receiver (P4 - Appeared 1 Time) (3-7M)

ANS: Broadcast Receiver:

- Broadcast Receivers simply respond to broadcast messages from other applications or from the system itself. These messages are sometime called events or intents.
- For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action.

There are following two important steps to make BroadcastReceiver works for the system broadcast intents -

- Creating the Broadcast Receiver.
- Registering Broadcast Receiver

There is one additional steps in case you are going to implement your custom intents then you will have to create and broadcast those intents.
Creating the Broadcast Receiver

- A broadcast receiver is implemented as a subclass of BroadcastReceiver class and overriding the onReceive() method

where each message is received as a Intent object parameter.

```
public class MyReceiver extends BroadcastReceiver {
  @Override
  public void onReceive(Context context, Intent intent) {
    Toast.makeText(context, "Intent Detected.",
Toast.LENGTH_LONG).show();
  }
}
```

Registering Broadcast Receiver

- An application listens for specific broadcast intents by registering a broadcast receiver in AndroidManifest.xml file. Consider we are going to register MyReceiver for system generated event ACTION_BOOT_COMPLETED which is fired by the system once the Android system has completed the boot process.

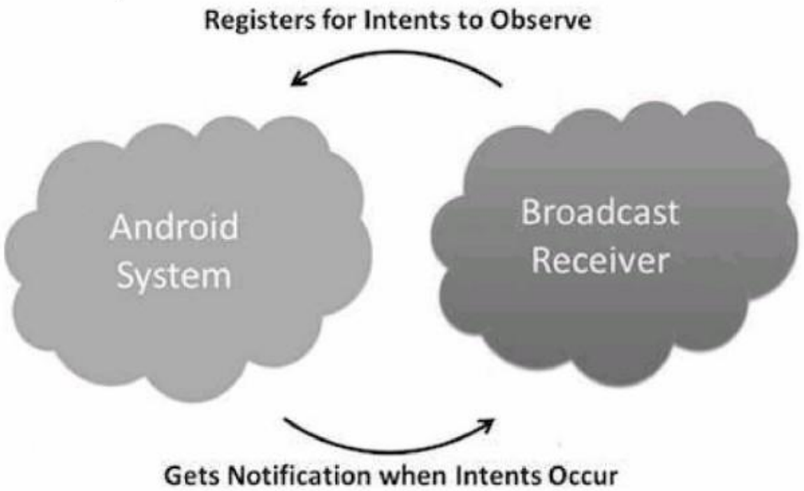


Figure: Broadcast Receivers

Broadcast-Receiver

```
<application
  android:icon="@drawable/ic_launcher"
  android:label="@string/app_name"
  android:theme="@style/AppTheme" >
  <receiver android:name="MyReceiver">

    <intent-filter>
      <action
android:name="android.intent.action.BOOT_COMPLETED">
      </action>
    </intent-filter>

  </receiver>
</application>
```

- Now whenever an Android device gets booted, it will be intercepted by BroadcastReceiver MyReceiver and implemented logic inside onReceive() will be executed.
- There are several system generated events defined as final static fields in the Intent class. The following table lists a few important system events.

No.	Event Constant	Description
1.	android.intent.action.BATTERY_CHANGED	Sticky broadcast containing the charging state, level, and other information about the battery.

2.	android.intent.action.BATTERY_LOW	Indicates low battery condition on the device.
3.	android.intent.action.BATTERY_OKAY	Indicates the battery is now okay after being low.
4.	android.intent.action.BOOT_COMPLETED	This is broadcast once, after the system has finished booting.
5.	android.intent.action.BUG_REPORT	Show activity for reporting a bug.

MODULE-8

Q1. Explain Camera and Taking Picture with Camera OR List out the steps for creating a custom camera interface (P4 - Appeared 1 Time) (3-7M)

ANS: The general steps for creating a custom camera interface for your application are as follows:

- Detect and Access Camera - Create code to check for the existence of cameras and request access.
- Create a Preview Class - Create a camera preview class that extends SurfaceView and implements the SurfaceHolder interface. This class previews the live images from the camera.
- Build a Preview Layout - Once you have the camera preview class, create a view layout that incorporates the preview and the user interface controls you want.
- Setup Listeners for Capture - Connect listeners for your interface controls to start image or video capture in response to user actions, such as pressing a button.
- Capture and Save Files - Setup the code for capturing pictures or videos and saving the output.
- Release the Camera - After using the camera, your application must properly release it for use by other applications.

Camera hardware is a shared resource that must be carefully managed so your application does not collide with other applications that may also want to use it.

Checking camera features

- Once you obtain access to a camera, you can get further information about its capabilities using the Camera.getParameters() method and checking the returned Camera.Parameters object for supported capabilities. When using API Level 9 or higher, use the Camera.getCameraInfo() to determine if a camera is on the front or back of the device, and the orientation of the image.

Capturing pictures

- Once you have built a preview class and a view layout in which to display it, you are ready to start capturing images with your application. In your application code, you must set up listeners for your user interface controls to respond to a user action by taking a picture.
- In order to retrieve a picture, use the Camera.takePicture() method. This method takes three parameters which receive data from the camera. In order to receive data in a JPEG format, you must implement an Camera.PictureCallback interface to receive the image data and write it to a file. The following code shows a basic implementation of the Camera.PictureCallback interface to save an image received from the camera.

```
private PictureCallback mPicture = new PictureCallback() {

    @Override
    public void onPictureTaken(byte[] data, Camera camera) {

        File pictureFile = getOutputMediaFile(MEDIA_TYPE_IMAGE);
        if (pictureFile == null){
```

```
            Log.d(TAG, "Error creating media file, check storage permissions");
            return;
        }

        try {
            FileOutputStream fos = new FileOutputStream(pictureFile);
            fos.write(data);
            fos.close();
        } catch (FileNotFoundException e) {
            Log.d(TAG, "File not found: " + e.getMessage());
        } catch (IOException e) {
            Log.d(TAG, "Error accessing file: " + e.getMessage());
        }
    }
};
```

Trigger capturing an image by calling the Camera.takePicture() method. The following example code shows how to call this method from a button View.OnClickListener.

```
// Add a listener to the Capture button
Button captureButton = (Button) findViewById(R.id.button_capture);
captureButton.setOnClickListener(
    new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // get an image from the camera
            mCamera.takePicture(null, null, picture);
        }
    }
);
```

Capturing videos

- Video capture using the Android framework requires careful management of the Camera object and coordination with the MediaRecorder class. When recording video with Camera, you must manage the Camera.lock() and Camera.unlock() calls to allow MediaRecorder access to the camera hardware, in addition to the Camera.open() and Camera.release() calls.
- Unlike taking pictures with a device camera, capturing video requires a very particular call order.

Order of execution to successfully prepare for and capture video with your application, as detailed below.

1. Open Camera - Use the Camera.open() to get an instance of the camera object.
2. Connect Preview - Prepare a live camera image preview by connecting a SurfaceView to the camera using Camera.setPreviewDisplay().
3. Start Preview - Call Camera.startPreview() to begin displaying the live camera images.
4. Start Recording Video - The following steps must be completed in order to successfully record video:
    1. Unlock the Camera - Unlock the camera for use by MediaRecorder by calling Camera.unlock().
    2. Configure MediaRecorder - Call in the following MediaRecorder methods in this order. For more information, see the MediaRecorder reference documentation.
        1. setCamera() - Set the camera to be used for video capture, use your application's current instance of Camera.

2. setAudioSource() - Set the audio source, use MediaRecorder.AudioSource.CAMCORDER.
3. setVideoSource() - Set the video source, use MediaRecorder.VideoSource.CAMERA.
4. Set the video output format and encoding. For Android 2.2 (API Level 8) and higher, use the MediaRecorder.setProfile method, and get a profile instance using CamcorderProfile.get(). For versions of Android prior to 2.2, you must set the video output format and encoding parameters:
    1. setOutputFormat() - Set the output format, specify the default setting or MediaRecorder.OutputFormat.MPEG_4.
    2. setAudioEncoder() - Set the sound encoding type, specify the default setting or MediaRecorder.AudioEncoder.AMR_NB.
    3. setVideoEncoder() - Set the video encoding type, specify the default setting or MediaRecorder.VideoEncoder.MPEG_4_SP.
5. setOutputFile() - Set the output file, use getOutputMediaFile(MEDIA_TYPE_VIDEO).toString() from the example method in the Saving Media Files section.
6. setPreviewDisplay() - Specify the SurfaceView preview layout element for your application. Use the same object you specified for Connect Preview.

3. Caution: You must call these MediaRecorder configuration methods in this order, otherwise your application will encounter errors and the recording will fail.

4. Prepare MediaRecorder - Prepare the MediaRecorder with provided configuration settings by calling MediaRecorder.prepare().
5. Start MediaRecorder - Start recording video by calling MediaRecorder.start().

5. Stop Recording Video - Call the following methods in order, to successfully complete a video recording:
    1. Stop MediaRecorder - Stop recording video by calling MediaRecorder.stop().
    2. Reset MediaRecorder - Optionally, remove the configuration settings from the recorder by calling MediaRecorder.reset().
    3. Release MediaRecorder - Release the MediaRecorder by calling MediaRecorder.release().
    4. Lock the Camera - Lock the camera so that future MediaRecorder sessions can use it by calling Camera.lock(). Starting with Android 4.0 (API level 14), this call is not required unless the MediaRecorder.prepare() call fails.
6. Stop the Preview - When your activity has finished using the camera, stop the preview using Camera.stopPreview().
7. Release Camera - Release the camera so that other applications can use it by calling Camera.release().

Q2. How to Manage Bluetooth Connection (P4 - Appeared 1 Time) (3-7M)

ANS: Bluetooth Connection:

- The Android platform includes support for the Bluetooth network stack, which allows a device to wirelessly exchange data with other

Bluetooth devices.

- The app framework provides access to the Bluetooth functionality through Bluetooth APIs. These APIs let apps connect to other Bluetooth devices, enabling point-to-point and multipoint wireless features.

Using the Bluetooth APIs, an app can perform the following:

- Scan for other Bluetooth devices.
- Query the local Bluetooth adapter for paired Bluetooth devices.
- Establish RFCOMM channels.
- Connect to other devices through service discovery.
- Transfer data to and from other devices.
- Manage multiple connections.

Key classes and interfaces

BluetoothAdapter

- Represents the local Bluetooth adapter (Bluetooth radio). The BluetoothAdapter is the entry-point for all Bluetooth interaction. Using this, you can discover other Bluetooth devices, query a list of bonded (paired) devices, instantiate a BluetoothDevice using a known MAC address, and create a BluetoothServerSocket to listen for communications from other devices.

BluetoothDevice

- Represents a remote Bluetooth device. Use this to request a connection with a remote device through a BluetoothSocket or query information about the device such as its name, address, class, and bonding state.

BluetoothSocket

- Represents the interface for a Bluetooth socket (similar to a TCP Socket). This is the connection point that allows an app to exchange

data with another Bluetooth device using InputStream and OutputStream.

BluetoothServerSocket

- Represents an open server socket that listens for incoming requests (similar to a TCP ServerSocket). In order to connect two devices, one device must open a server socket with this class. When a remote Bluetooth device makes a connection request to this device, the device accepts the connection and then returns a connected BluetoothSocket.

BluetoothClass

- Describes the general characteristics and capabilities of a Bluetooth device. This is a read-only set of properties that defines the device's classes and services. Although this information provides a useful hint regarding a device's type, the attributes of this class don't necessarily describe all Bluetooth profiles and services that the device supports.

BluetoothProfile

- An interface that represents a Bluetooth profile. A Bluetooth profile is a wireless interface specification for Bluetooth-based communication between devices. An example is the Hands-Free profile. For more discussion of profiles, see Bluetooth profiles.

BluetoothHeadset

- Provides support for Bluetooth headsets to be used with mobile phones. This includes both the Bluetooth Headset profile and the Hands-Free (v1.5) profile.

BluetoothA2dp

- Defines how high-quality audio can be streamed from one device to another over a Bluetooth connection using the Advanced Audio Distribution Profile (A2DP).

BluetoothHealth

- Represents a Health Device Profile proxy that controls the Bluetooth service.

BluetoothHealthCallback

- An abstract class that you use to implement BluetoothHealth callbacks. You must extend this class and implement the callback methods to receive updates about changes in the app's registration state and Bluetooth channel state.

BluetoothHealthAppConfiguration

- Represents an app configuration that the Bluetooth Health third-party app registers to communicate with a remote Bluetooth health device.

BluetoothProfile.ServiceListener

- An interface that notifies BluetoothProfile interprocess communication (IPC) clients when they have been connected to or disconnected from the internal service that runs a particular profile.

Q3. Explain Dalvik Debug Tool (P4 - Appeared 1 Time) (3-7M)

ANS: Dalvik Debug Tool:

- Android ships with a debugging tool called the Dalvik Debug Monitor Server (DDMS), which provides port-forwarding services, screen capture on the device, thread and heap information on the device, logcat, process, and radio state information, incoming call and SMS spoofing, location data spoofing, and more.
- DDMS ships in the tools/ directory of the SDK. Enter this directory from a terminal/console and type ddms (or ./ddms on Mac/Linux) to run it. DDMS will work with both the emulator and a connected

device. If both are connected and running simultaneously, DDMS defaults to the emulator.

Working of DDMS:

- DDMS acts as a middleman to connect the IDE to the applications running on the device. On Android, every application runs in its own process, each of which hosts its own virtual machine (VM). And each process listens for a debugger on a different port.
- When it starts, DDMS connects to adb and starts a device monitoring service between the two, which will notify DDMS when a device is connected or disconnected.
- When a device is connected, a VM monitoring service is created between adb and DDMS, which will notify DDMS when a VM on the device is started or terminated. Once a VM is running, DDMS retrieves the VM's process ID (pid), via adb, and opens a connection to the VM's debugger, through the adb daemon (adbd) on the device. DDMS can now talk to the VM using a custom wire protocol.
- For each VM on the device, DDMS opens a port upon which it will listen for a debugger. For the first VM, DDMS listens for a debugger on port 8600, the next on 8601, and so on.
- When a debugger connects to one of these ports, all traffic is forwarded between the debugger and the associated VM. Debugging can then be processed like any remote debugging session.
- DDMS also opens another local port, the DDMS "base port" (8700, by default), upon which it also listens for a debugger. When a debugger connects to this base port, all traffic is forwarded to the VM currently selected in DDMS, so this is typically where your debugger should connect.

Q4. Explain Emulator Control (P4 - Appeared 1 Time) (3-7M)

ANS: Emulator Control:

- The Android emulator is an Android Virtual Device (AVD), which represents a specific Android device. We can use the Android emulator as a target device to execute and test our Android application on our PC.
- The Android emulator provides almost all the functionality of a real device. We can get incoming phone calls and text messages. It also gives the location of the device and simulates different network speeds. Android emulator simulates rotation and other hardware sensors. It accesses the Google Play store, and much more
- Testing Android applications on emulator are sometimes faster and easier than doing on a real device. For example, we can transfer data faster to the emulator than to a real device connected through USB.
- The Android emulator comes with predefined configurations for several Android phones, Wear OS, tablet, Android TV devices.

Requirement and recommendations
The Android emulator takes additional requirements beyond the basic system requirement for Android Studio. These requirements are given below:

- SDK Tools 26.1.1 or higher
- 64-bit processor
- Windows: CPU with UG (unrestricted guest) support
- HAXM 6.2.1 or later (recommended HAXM 7.2.0 or later)

Install the emulator

- The Android emulator is installed while installing the Android Studio. However some components of emulator may or may not be installed while installing Android Studio. To install the emulator component, select the Android Emulator component in the SDK Tools tab of the SDK Manager.

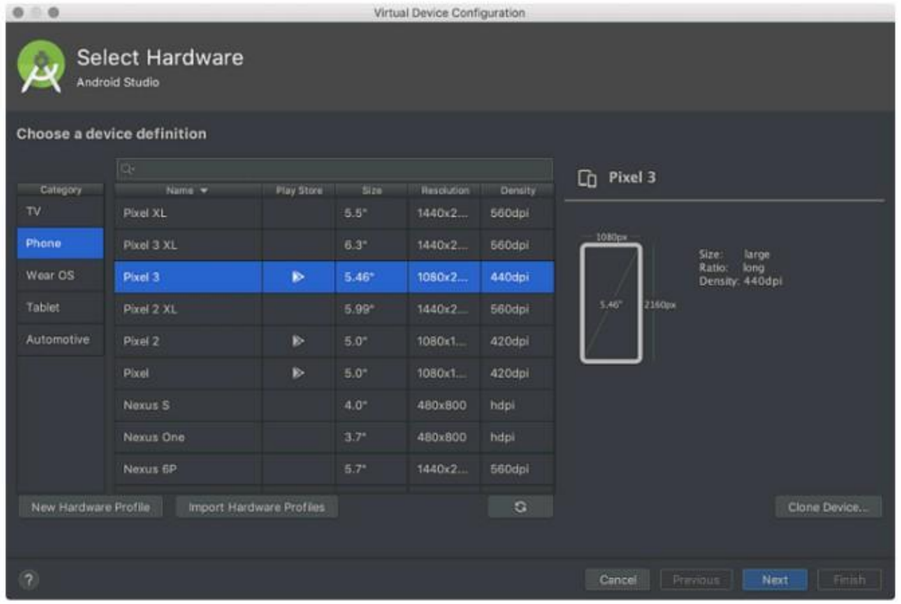
Run an Android app on the Emulator

We can run an Android app form the Android Studio project, or we can run an app which is installed on the Android Emulator as we run any app on a device. To start the Android Emulator and run an application in our project:

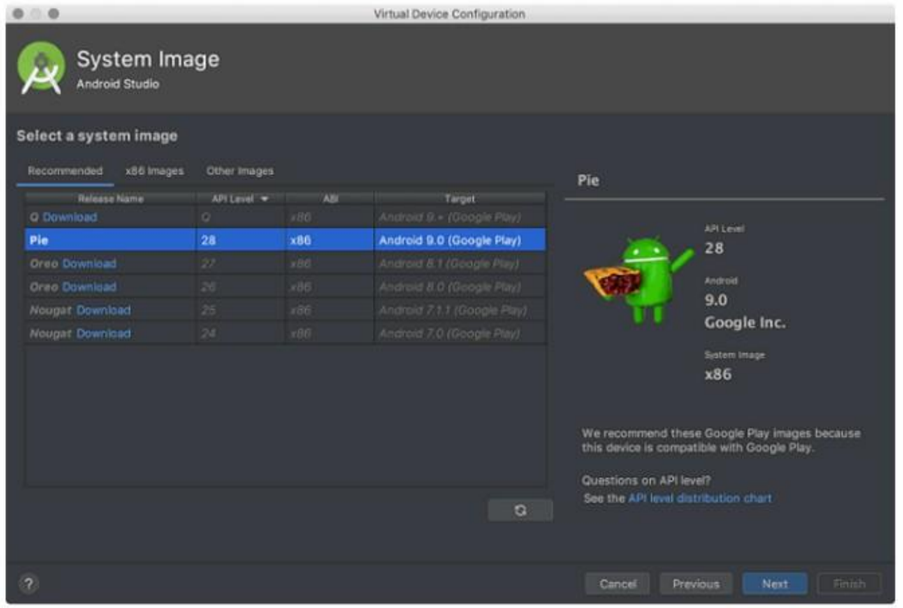
1. In Android Studio, we need to create an Android Virtual Device (AVD) that the emulator can use to install and run your app. To create a new AVD:-

1.1 Open the AVD Manager by clicking Tools > AVD Manager.

1.2 Click on Create Virtual Device, at the bottom of the AVD Manager dialog. Then Select Hardware page appears.

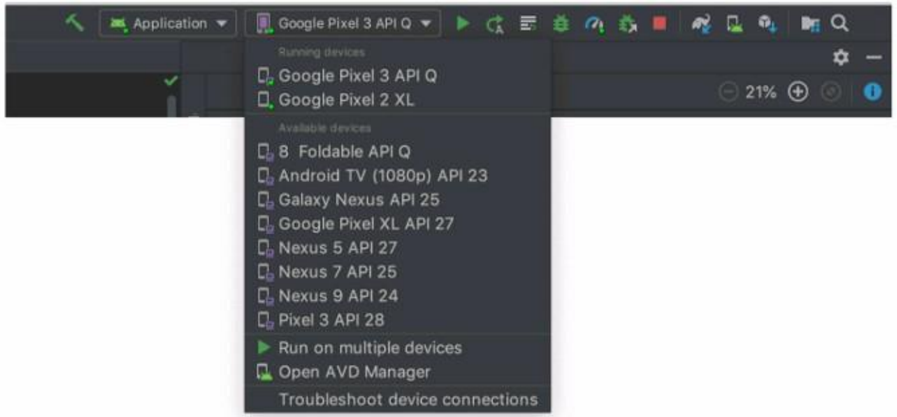


1.3 Select a hardware profile and then click Next. If we don?t see the hardware profile we want, then we can create or import a hardware profile. The System Image page appears.



1.4 Select the system image for the particular API level and click Next. This leads to open a Verify Configuration page.

1.5 Change AVD properties if needed, and then click Finish.
In the toolbar, choose the AVD, which we want to run our app from the target device from the drop-down menu and click Run.



Q5. Explain Work with ADB (P4 - Appeared 1 Time) (3-7M)

ANS: Android Debug Bridge:

- Android Debug Bridge (adb) is a versatile command-line tool that lets you communicate with a device. The adb command facilitates a variety of device actions, such as installing and debugging apps, and it provides access to a Unix shell that you can use to run a variety of commands on a device.

It is a client-server program that includes three components:

- A client, which sends commands. The client runs on your development machine. You can invoke a client from a command-line terminal by issuing an adb command.

- A daemon (adbd), which runs commands on a device. The daemon runs as a background process on each device.
- A server, which manages communication between the client and the daemon. The server runs as a background process on your development machine.

Adb is included in the Android SDK Platform-Tools package.

- To download this package with the SDK Manager, which installs it at android_sdk/platform-tools/. Or if you want the standalone Android SDK Platform-Tools package, you can download it here.

Working of ADB:

- When you start an adb client, the client first checks whether there is an adb server process already running. If there isn't, it starts the server process. When the server starts, it binds to local TCP port 5037 and listens for commands sent from adb clients—all adb clients use port 5037 to communicate with the adb server.
- The server then sets up connections to all running devices. It locates emulators by scanning odd-numbered ports in the range 5555 to 5585, the range used by the first 16 emulators. When the server finds an adb daemon (adbd), it sets up a connection to that port. Note that each emulator uses a pair of sequential ports — an even-numbered port for console connections and an odd-numbered port for adb connections. For example:
  Emulator 1, console: 5554
  Emulator 1, adb: 5555
  Emulator 2, console: 5556
  Emulator 2, adb: 5557
  and so on...

- As shown, the emulator connected to adb on port 5555 is the same as the emulator whose console listens on port 5554.
- Once the server has set up connections to all devices, you can use adb commands to access those devices. Because the server manages connections to devices and handles commands from multiple adb clients, you can control any device from any client (or from a script).

Q6. Execute Application on Real Device (P4 - Appeared 1 Time) (3-7M)

ANS: Run on a real device

Set up your device as follows:

1. Connect your device to your development machine with a USB cable. If you developed on Windows, you might need to install the appropriate USB driver for your device.
2. Perform the following steps to enable USB debugging in the Developer options window:
    1. Open the Settings app.
    2. If your device uses Android v8.0 or higher, select System. Otherwise, proceed to the next step.
    3. Scroll to the bottom and select About phone.
    4. Scroll to the bottom and tap Build number seven times.
    5. Return to the previous screen, scroll to the bottom, and tap Developer options.
    6. In the Developer options window, scroll down to find and enable USB debugging.

Run the app on your device as follows:

1. In Android Studio, select your app from the run/debug configurations drop-down menu in the toolbar.
2. In the toolbar, select the device that you want to run your app on from the target device drop-down menu.

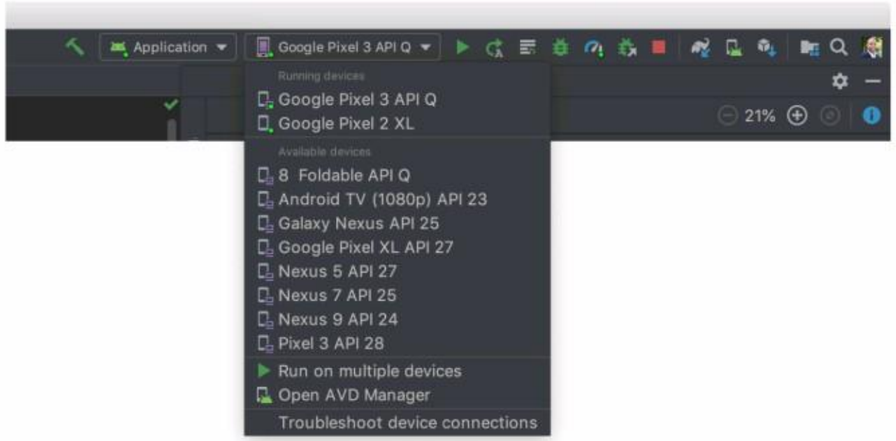


Figure: Target device drop-down menu

3. Click Run
   Android Studio installs your app on your connected device and starts it. You now see "Hello, World!" displayed in the app on your device.

OTHER IMP QUESTIONS:

MODULE-3

Q1. Development with Android its Platforms, Tools, Versions (P4 - Appeared 1 Time) (3-7M)

MODULE-4

Q1. Create Android UI (P4 - Appeared 1 Time) (3-7M)

Q2. Material Design Toolbar (P4 - Appeared 1 Time) (3-7M)

MODULE-7

Q1. Differentiate Geo coding Graphics and Animation (P4 - Appeared 1 Time) (3-7M)

Q2. Firebase with simple CRUD Operation (P4 - Appeared 1 Time) (3-7M)

MODULE-8

Q1. Differentiate Monitor and Manage Wi-Fi (P4 - Appeared 1 Time) (3-7M)

Q2. Differentiate Accelerometer Sensor & Gyroscope. (P4 - Appeared 1 Time) (3-7M)

Q3. What is Connect Real Devices (P4 - Appeared 1 Time) (3-7M)
