



Sub. Code : 3170726

Mobile Application Development

AS PER NEW SYLLABUS - GTU - SEM - VII (CE/CSE) OPEN ELECTIVE - III

- Simplified & Conceptual Approach • Short Questions & Answers
- Multiple Choice Questions with Answers

first edition : aug. 2021

Price : ₹ 160/-
ISBN 978-93-90770-89-2

9 789390 770892



V. S. Bagad

TABLE OF CONTENTS

Chapter - 1 Fundamental	(1 - 1) to (1 - 16)
1.1 Software Development Life Cycle (SDLC)	1 - 2
1.2 Approaches for Developing Application Software.....	1 - 4
1.2.1 Top-Down Approach of Software Development	1 - 4
1.2.2 Bottom-Up Approach of Software Development.....	1 - 5
1.2.3 Modular Programming of Software Development.....	1 - 5
1.3 Waterfall Model of Software Development	1 - 6
1.3.1 Waterfall Model Applications	1 - 8
1.3.2 Advantages of Waterfall Model.....	1 - 8
1.3.3 Disadvantages of Waterfall Model	1 - 8
1.4 Data Flow Diagram	1 - 9
1.4.1 Symbols of Data Flow Diagram	1 - 9
1.4.2 E-R Diagram.....	1 - 10
1.4.3 Symbols used in E-R Diagram.....	1 - 11
1.5 Database Management System (DBMS).....	1 - 12
1.5.1 DDBMS Components	1 - 13
1.5.2 DDBMS Advantages	1 - 14
1.5.3 DDBMS Disadvantages	1 - 14
1.6 Short Questions and Answers.....	1 - 14
1.7 Mutliple Choice Questions with Answers	1 - 15
Chapter - 2 Android OS	(2 - 1) to (2 - 32)
2.1 Introduction to Android.....	2 - 2
2.1.1 Advantages and Disadvantages of Android	2 - 3
2.2 Android APIs	2 - 4
2.3 Android Architecture	2 - 5

2.4 Android Application Framework.....	2 - 7
2.4.1 Development with Android - Platforms	2 - 8
2.4.1.1 Conversion Process from Source Code to Android Application	2 - 8
2.4.1.2 Android SDK Features	2 - 8
2.5 Android Developing Tools.....	2 - 9
2.6 Setup Android Environment	2 - 10
2.7 Building Blocks of Android Application.....	2 - 11
2.7.1 Manifest File	2 - 11
2.7.2 Downloading and Installing Android.....	2 - 12
2.8 Developing Android Application on Eclipse Platform.....	2 - 12
2.9 Activity Lifecycle	2 - 13
2.10 Fragments in Android	2 - 16
2.10.1 Importance of Fragments	2 - 17
2.10.2 Purpose of Fragments	2 - 17
2.10.3 Communication between Fragments and Activity	2 - 18
2.11 Fragment Life Cycle.....	2 - 18
2.12 Replacing Fragment	2 - 21
2.13 Activity Lifecycle	2 - 21
2.14 Intents.....	2 - 24
2.14.1 Intent to Start Another Activity.....	2 - 25
2.14.2 Implicit and Explicit Intent	2 - 26
2.15 Short Questions and Answers.....	2 - 27
2.16 Multiple Choice Questions with Answers.....	2 - 31

**Chapter - 3 Android UI and Component using Fragments
(3 - 1) to (3 - 14)**

3.1 Android Graphics Interface.....	3 - 2
3.2 Create Android UI	3 - 2
3.2.1 Editable Text View	3 - 3
3.2.2 Radio and Toggle Buttons	3 - 3
3.2.3 Creating Radio Button	3 - 4

3.2.4 Spinners	3-5
3.3 Working with Layout.....	3-5
3.4 Work with UI Components and Events.....	3-6
3.5 Material Design Toolbar	3-7
3.6 Android Menus	3-7
3.7 Android Layout Classes	3-9
3.8 Short Questions and Answers.....	3-10
3.9 Multiple Choice Questions with Answers	3-12

Chapter - 4 Database Connectivity (4 - 1) to (4 - 14)

4.1 Storage in Android	4-2
4.1.1 Application Data Folder Scope.	4-2
4.1.2 Creating File in App Data Folder	4-2
4.1.3 Search for Files in App Data Folder	4-2
4.2 Using Internal Storage	4-3
4.3 Working with SD Card / External Storage	4-3
4.4 Shared Preferences Layout.....	4-4
4.4.1 Creating Private and Shared Preferences.	4-5
4.5 Database in Android	4-6
4.5.1 Inserting Data	4-6
4.5.2 Updating Data	4-7
4.5.3 Deleting data.....	4-8
4.6 SQLite.....	4-8
4.6.1 Creating Database in SQLite	4-9
4.7 Content Provider.....	4-10
4.7.1 Content Provider Classes	4-11
4.7.2 Deleting Database Records	4-11
4.8 Short Questions and Answers.....	4-12
4.9 Multiple Choice Questions with Answers	4-13

Chapter - 5 Applicability to Industrial Projects	(5 - 1) to (5 - 12)
5.1 Web Services and Parsing	5 - 2
5.1.1 Types of Web Services	5 - 3
5.1.2 Requirements of Web Services	5 - 3
5.2 JSON Parsing	5 - 4
5.2.1 Characteristics of JSON.....	5 - 5
5.2.2 Applications of JSON.....	5 - 6
5.3 Access Web Data with JSON	5 - 6
5.4 Using AsyncTask and Third Party Library : Retrofit.....	5 - 7
5.4.1 Retrofit.....	5 - 7
5.4.2 Built-in Annotations	5 - 8
5.4.3 Advantages of Android Retrofit.....	5 - 8
5.4.4 Volley	5 - 9
5.5 Short Questions and Answers.....	5 - 9
5.6 Multiple Choice Questions with Answers	5 - 11
Chapter - 6 Advanced Android Development	(6 - 1) to (6 - 32)
6.1 Google Map	6 - 2
6.1.1 APIs in Google Maps	6 - 2
6.1.2 Steps for Getting the Google Maps API Key	6 - 3
6.1.3 Elements of Google Map	6 - 3
6.1.4 Adding a Map to an Application	6 - 4
6.1.5 Types of Google Map	6 - 5
6.1.6 Displaying the User's Current Location	6 - 5
6.1.7 User Interface (UI) Controls of Google Maps	6 - 5
6.1.8 Features of Google Maps	6 - 6
6.1.9 Difference between Google Maps and Google Earth	6 - 6
6.2 Location Service and GPS.....	6 - 6
6.2.1 Current Location	6 - 6
6.2.2 Location Listener.....	6 - 7

6.3 Geocoding.....	6 - 8
6.3.1 Use of Geolocations	6 - 8
6.3.2 Effectiveness of Geocoder	6 - 9
6.4 Reverse Geocoding Revgeocode.....	6 - 9
6.4.1 Difference between Forward and Reverse Geocoding	6 - 9
6.5 Working with 2D Graphics	6 - 9
6.5.1 Uses of Graphics	6 - 10
6.5.2 Using Drawable Object.....	6 - 10
6.5.3 Using the Shape Drawable Object.....	6 - 11
6.5.4 Hardware Acceleration.....	6 - 11
6.6 Working with Animation.....	6 - 12
6.6.1 Clock for Animated Graphics.....	6 - 13
6.6.2 Controlling Timing Events	6 - 13
6.6.3 Bitmap Animation	6 - 14
6.6.4 Frame Animation	6 - 14
6.6.5 Tween Animation in Android.....	6 - 14
6.7 Alarms in Android	6 - 15
6.8 Download Manager	6 - 16
6.9 Multimedia in Android.....	6 - 17
6.9.1 Media Player	6 - 17
6.9.1.1 User Media Player	6 - 17
6.9.2 Play Audio Files	6 - 18
6.9.2.1 Media Formats	6 - 18
6.9.2.2 Playing Audio	6 - 20
6.9.3 Play Video Files	6 - 20
6.9.4 Recording Video	6 - 21
6.9.5 Creating a Sound Pool	6 - 22
6.10 Work in Background Services	6 - 23
6.10.1 Android Background Service Methods	6 - 24
6.11 Notification Services	6 - 24

6.12 Broadcast Receiver	6 - 25
6.13 Introduction to Firebase with Simple CRUD Operation.....	6 - 25
6.13.1 Firebase Realtime Database Feature	6 - 26
6.14 Short Questions and Answers.....	6 - 26
6.15 Multiple Choice Questions with Answers.....	6 - 28

**Chapter - 7 Work with Android System & Development
and Deployment (7 - 1) to (7 - 18)**

7.1 Text to Speech	7 - 2
7.2 Using Camera.....	7 - 2
7.3 Manage Bluetooth Connection.....	7 - 3
7.3.1 Android Permissions.....	7 - 3
7.3.2 Interfaces and Classes	7 - 4
7.4 Monitor and Manage Wi-Fi.....	7 - 5
7.5 Accelerometer Sensor	7 - 6
7.6 Gyroscope	7 - 7
7.7 Dalvik Debug Tool	7 - 8
7.7.1 Logcat	7 - 9
7.7.2 Android Debug Bridge (ADB)	7 - 9
7.8 Signing the Android Application	7 - 10
7.9 Publishing Android App	7 - 13
7.9.1 Process of Publishing an Android Application	7 - 13
7.10 Distribution of Android App.....	7 - 14
7.11 App Characteristics	7 - 14
7.11.1 Performance of App.....	7 - 14
7.11.2 Modifiability of App	7 - 15
7.11.3 Availability of App.....	7 - 15
7.11.4 Security of App	7 - 15
7.12 Short Questions and Answers.....	7 - 16
7.13 Multiple Choice Questions with Answers.....	7 - 16

1

Fundamental

Syllabus

Software Engineering with SDLC, Flow chart, DFD, SQL database.

Contents

- 1.1 Software Development Life Cycle (SDLC)
- 1.2 Approaches for Developing Application Software
- 1.3 Waterfall Model of Software Development
- 1.4 Data Flow Diagram
- 1.5 Database Management System (DBMS)
- 1.6 Short Questions and Answers
- 1.7 Multiple Choice Questions

1.1 Software Development Life Cycle (SDLC)

- SDLC is the acronym of Software Development Life Cycle. It is also called as Software Development Process.
- Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality softwares.
- SDLC is a process that creates a structure of development of software. There are different phases within SDLC, and each phase has its various activities. It makes the development team able to design, create, and deliver a high-quality product.
- The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.
- The Software Development Life Cycle (SDLC) is a structured process that enables the production of high-quality, low-cost software, in the shortest possible production time.
- SDLC is a framework defining tasks performed at each step in the software development process.
- 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.

Different phases of the software development cycle

- | | | |
|--------------------------------------|------------------|-----------------|
| 1. Requirement analysis Phase | 2. Planning | 3. Design Phase |
| 4. Build /Software Development Phase | 5. Testing Phase | |
| 6. Deployment/ Deliver Phase | 7. Maintenance | |

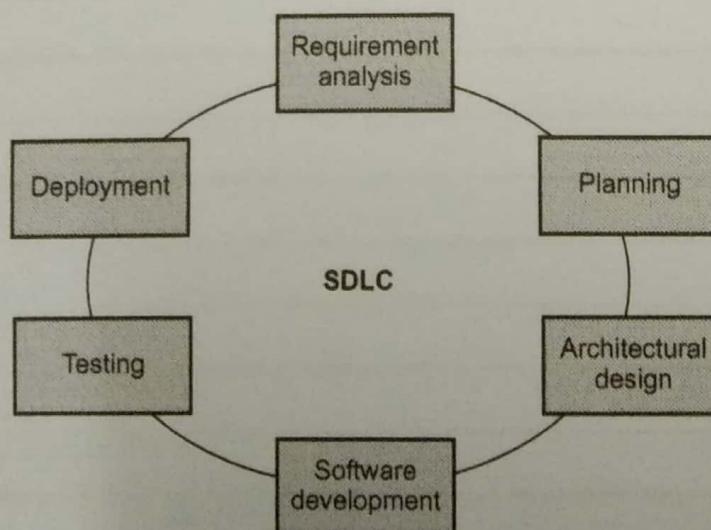


Fig. 1.1.1 SDLC phases

1. Requirement analysis

- During this phase, all the relevant information is collected from the customer to develop a product as per their expectation. Any ambiguities must be resolved in this phase only.
- Business analyst and project manager set up a meeting with the customer to gather all the information like what the customer wants to build, who will be the end-user, what is the purpose of the product. Before building a product a core understanding or knowledge of the product is very important.

2. Planning

- In this stage of the SDLC, the team determines the cost and resources required for implementing the analyzed requirements. It also details the risks involved and provides sub-plans for softening those risks.
- In other words, the team should determine the feasibility of the project and how they can implement the project successfully with the lowest risk in mind.

3. Architectural Design

- After a comprehensive analysis phase, the design phase surfaces. In this third phase, the elements, components, security levels, modules, architecture, interfaces and data of the system are defined and designed to evaluate how the finished system will work and what it will look like.
- The system design is produced in detail to ensure the system will include the necessary features to meet all functional and operational aspects of the project.

4. Software Development

- In this phase, the development team is hard at work writing code and constructing and fine-tuning the technical and physical configurations necessary to build the overall information system.
- Programmers take the design documents and code the modules. They might use various tools and aids such as-
 1. Integrated Development Environments (IDEs) like Eclipse or Android Studio;
 2. Software Development Kits (SDKs);
 3. Application Programming Interfaces (APIs).
 4. Coding guidelines and checklists.

5. Testing

- Testers take over after developers have coded the modules to execute the testing phase. Note that developers typically take care of the unit testing.
- In this phase, the developed software is tested thoroughly and any defects found are assigned to developers to get them fixed.

- Testers address the other types of testing, such as:
 - a) System testing often called Systems Integration Testing (SIT);
 - b) Performance testing;
 - c) Regression testing;
- Testing involves the following tasks:
 - a) Writing test plan and test cases;
 - b) Preparing the "test bed", i.e., test environment with test data;
 - c) Executing test cycles, which could mean test automation scripts;
 - d) Documenting test results, and analyzing them.

6. Deployment

- Software deployment is about deploying the application in the production environment. Once the product is tested, it is deployed in the production environment or first UAT (User Acceptance testing) is done depending on the customer expectation.
- Deployment involves the following:
 - a) Collect the code for deployment in appropriate packages
 - b) Document version/release information, including notes
 - c) Validate the readiness of the target environment
 - d) Configure the various target environments

7. Maintenance

- After the deployment of a product on the production environment, maintenance of the product i.e. if any issue comes up and needs to be fixed or any enhancement is to be done is taken care by the developers.

1.2 Approaches for Developing Application Software

1.2.1 Top-Down Approach of Software Development

- The top-down approach (stepwise design) requires a complete design view of the system before any actual coding begins. It takes the big-picture view of the system and breaks the problem apart into many subsystems that require more design detail.
- The overall perspective of the project is monolithic. There is one design, and all modules are small parts of it. The top-down style of program design is traditionally associated with procedural languages.

1.2.2 Bottom-Up Approach of Software Development

- The **bottom-up** approach, in contrast, emphasizes assembling the big picture by completing many small pieces that work together. These components are then integrated into the larger system. Coding of individual modules, and their testing, commences before the entire design is complete.
- In a bottom-up approach the individual base elements of the system are first specified in detail. These elements are then linked together to form larger subsystems, which then in turn are linked, sometimes in many levels, until a complete top-level system is formed. This strategy often resembles a "seed" model, whereby the beginnings are small but eventually grow in complexity and completeness.
- Proponents of the bottom-up approach argue that this produces reusable code that saves time later in the process. The bottom-up style of program design is traditionally associated with object-oriented languages such as C++ and Java.
- In practice, some blend of top-down and bottom-up is a common compromise. Project managers and architects work on the overall design even as development teams begin building modules and tools. This approach is sometimes called **hybrid design** or **hybrid development**.

1.2.3 Modular Programming of Software Development

- Modular programming is a methodology for software development in which a large program is partitioned into independent smaller units. Each unit contains a set of related functions, types, etc. that can be readily reused in different programming tasks.
- Modular programming is a simple programming technique that can be applied to any programming language. It's as useful in C++ programming as it is in Java programming or programming in C

Advantages of Using Modular Programming

- There are many good reasons for using modular programming :
 1. Code is easier to understand. Instead of pages and pages there are small, discrete blocks of code.
 2. Errors are easier to find. An error will always be localised to a subroutine or function rather than buried somewhere in a mass of code.
 3. Scoping of variables can be controlled more easily.
 4. Duplication of code can be eradicated - rather than retyping the same code again and again, the programmer can create a single procedure.

5. Reuse of code - as well as being reused within a single application, modular programming allows code to be used in multiple applications.
6. Much less code needs to be written.
7. Programs are easier to design - the designer just needs to think about high level functions.
8. Collaborative programming is possible - modular programming enables more than one programmer to work a single application at the same time.
9. Code can be stored across multiple files.

1.3 Waterfall Model of Software Development

- Waterfall model is an example of the sequential model. In this model, the software development activity is divided into different phases, and each phase consists of series of tasks and has different objectives.
- Waterfall model is the pioneer of the SDLC (Software Development Life Cycle) processes. In fact it was the first model which was widely used in the software industry.
- The model is divided into phases, and the output of one phase becomes the input of the next phase. It is mandatory for a phase to be completed before the next phase starts. In short, there is no overlapping in waterfall model.
- In waterfall, the development of one phase starts only when the previous phase is complete. Because of this nature, each step of waterfall model is quite precise well defined. Since the phase's falls from a higher level to a lower level, like a waterfall model, it is named a waterfall model.
- Fig. 1.3.1 shows waterfall model of SDLC.

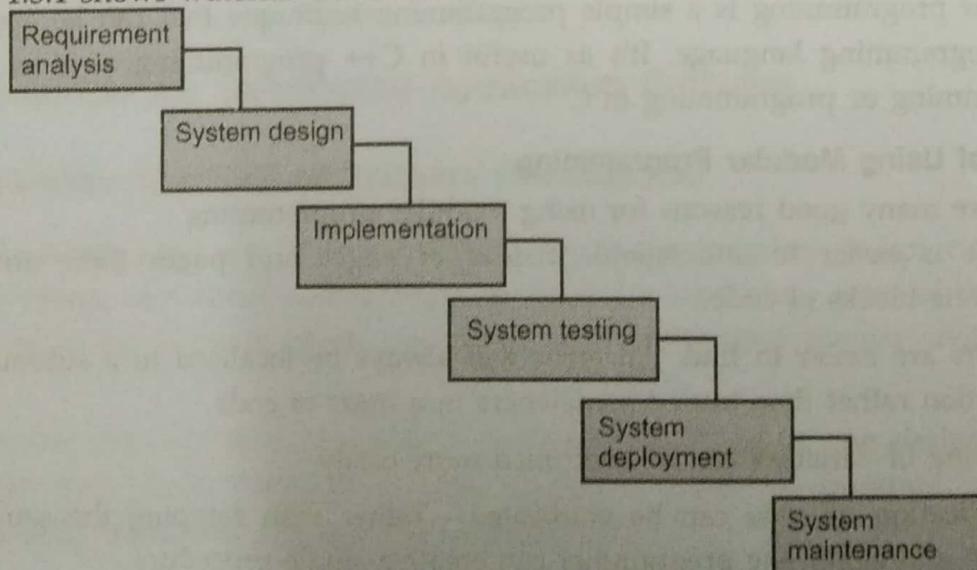


Fig. 1.3.1 Waterfall model

Table 1.3.1 Phases & related activity of waterfall model

Phase	Activity
1. Requirement Analysis :	<ol style="list-style-type: none"> 1. Capture all the requirements. 2. Do brain storming and walk through to understand the requirements. 3. Do the requirements feasibility test to ensure that the requirements are testable or not.
2. System Design	<ol style="list-style-type: none"> 1. As per the requirements, create the design 2. Capture the hardware / software requirements. 3. Document the designs
3. Implementation	<ol style="list-style-type: none"> 1. As per the design create the programmes / code 2. Integrate the codes for the next phase. 3. Unit testing of the code
4. System Testing	<ol style="list-style-type: none"> 1. Integrate the unit tested code and test it to make sure if it works as expected. 2. Perform all the testing activities (Functional and non functional) to make sure that the system meets the requirements. 3. In case of any anomaly, report it. 4. Track your progress on testing through tools like traceability metrics. 5. Report your testing activities.
5. System Deployment	<ol style="list-style-type: none"> 1. Make sure that the environment is up 2. Make sure that there are no defects open. 3. Make sure that the test exit criteria are met. 4. Deploy the application in the respective environment. 5. Perform a sanity check in the environment after the application is deployed to ensure the application does not break.

6. System Maintenance

1. Make sure that the application is up and running in the respective environment.
2. In case user encounters any defect, make sure to note and fix the issues faced.
3. In case any issue is fixed; the updated code is deployed in the environment.
4. The application is always enhanced to incorporate more features, update the environment with the latest features.

Table 1.3.1**1.3.1 Waterfall Model Applications**

- Waterfall model of software development is preferred under the following circumstances:
 1. Requirements are stable and not changed frequently.
 2. Application is small.
 3. There is no requirement that is not understood or not very clear.
 4. The environment is stable.
 5. The tools and technology used is stable and is not dynamic.
 6. Resources are well trained and are available.

1.3.2 Advantages of Waterfall Model

- Advantages of using waterfall model are as follows :

 1. Simple and easy to understand and use.
 2. For smaller projects, waterfall model works well and yield the appropriate results.
 3. Since the phases are rigid and precise, one phase is done one at a time; it is easy to maintain.
 4. The entry and exit criteria are well defined, so it is easy and systematic to proceed with quality.
 5. Results are well documented.

1.3.3 Disadvantages of Waterfall Model

- Disadvantages of using waterfall model :

 1. Cannot adopt the changes in requirements.
 2. It becomes challenging to move back to the phase. For example, if the application has now moved to testing stage and there is a change in requirement; it becomes difficult to go back and change it.

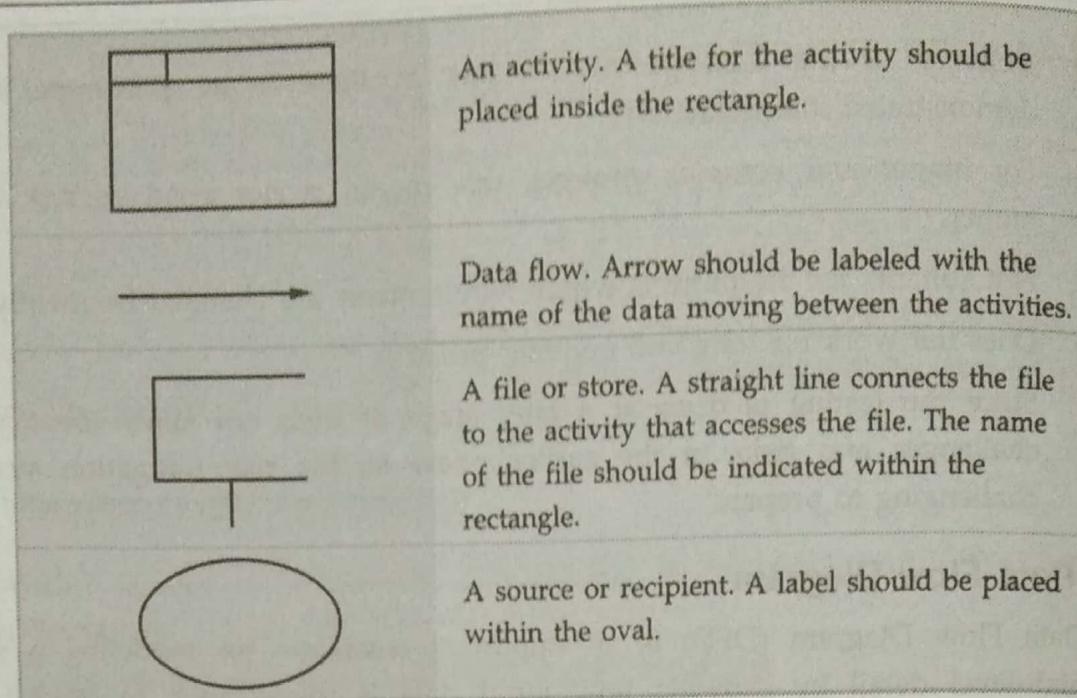
3. Delivery of the final product is late as there is no prototype which is demonstrated intermediately.
4. For bigger and complex projects, this model is not good as risk factor is higher.
5. Not suitable for the projects where requirements are changed frequently.
6. Does not work for long and ongoing projects.
7. Since the testing is done at a later stage, it does not allow identifying the challenges and risks in the earlier phase so the risk mitigation strategy is challenging to prepare.

1.4 Data Flow Diagram

- Data Flow Diagram (DFD) is an important technique for modeling a system's high-level detail by showing how input data is transformed to output results through a sequence of functional transformations.
- DFDs shows relationships among and between the various components in a program or system.
- DFDs consists of four major components : entities, processes, data stores and data flows.
- When to convey information data flows through system and how that data is transformed in the process, DFDs are the method of choice over technical descriptions for three principle reasons.
 1. DFDs are easier to understand by technical and non-technical audiences.
 2. DFDs can provide a high-level system overview, complete with boundaries and connections to other systems.
 3. DFDs can provide a detailed representation of system components.
- A data flow diagram represents the following :
 1. External devices sending and receiving data.
 2. Processes that change that data.
 3. Data flows themselves.
 4. Data storage locations.

1.4.1 Symbols of Data Flow Diagram

- There are only four symbols used to write data flow diagram are as follows :
 1. External Entities -> Rectangular box
 2. Data Flow -> Arrow headed lines
 3. Process -> Bubble (Circle or round corner square)
 4. Data Store -> Narrow opened rectangle



1.4.2 E-R Diagram

- A database is usually modeled as:
 1. a collection of entities,
 2. relationship among entities
- An Entity-Relationship (E-R) model is an abstraction of a relational database design that emphasizes the relationships between pieces of data rather than the mechanics of enforcing those relationships. It primarily consists of an Entity-Relationship diagram and an accompanying Entity-Relationship dictionary.
- An E-R model includes four elements : (i) entities, (ii) attributes, (iii) identifiers, and (iv) relationships.
- Out of the four elements, relationships prove to be the most interesting and the trickiest.
 - (i) **Entity** : An entity is a business object that represents a group or category of data. Example : specific person, company, event, plant.
 - (ii) **Attribute** : An attribute is a sub-group of information within an entity. Attributes define the characteristics of an entity. Example : people have names and addresses.
 - (iii) **Identifier** : An Identifier is a set of one or more attributes that uniquely identify a specific entity.
 - (iv) **Relationships** : Relationships associate entities with one another. Only two entities are associated (binary), but the ER model does allow for more (ternary, quaternary, etc.). However, non-binary relationships cannot survive the schema conversion process, so they rarely used.

Entity Set : An entity set is a set of entities of the same type that share the same properties.

Example : Set of all persons, companies, trees, holidays

1.4.3 Symbols used in E-R Diagram

	entity set		attribute
	weak entity set		multivalued attribute
	relationship set		derived attribute
	identifying relationship set for weak entity set		total participation of entity set in relationship
	primary key		discriminating attribute of weak entity set
	many-to-many relationship		many-to-one relationship
	one-to-one relationship		cardinality limits
	role indicator		ISA (specialization or generalization)
	total generalization		disjoint generalization

Illustrative Example

Example 1.4.1 A publishing company produces scientific books on various subjects. The books are written by authors who specialize in one particular subject. The company employs editors who, not necessarily being specialists in a particular area, each take sole responsibility for editing one or more publications. A publication covers essentially one of the specialist subjects and is normally written by a single author. When writing a particular book, each author works with one editor, but may submit another work for publication to be supervised by other editors. To improve their competitiveness, the company tries to employ a variety of authors, more than one author being a specialist in a particular subject.

Solution :

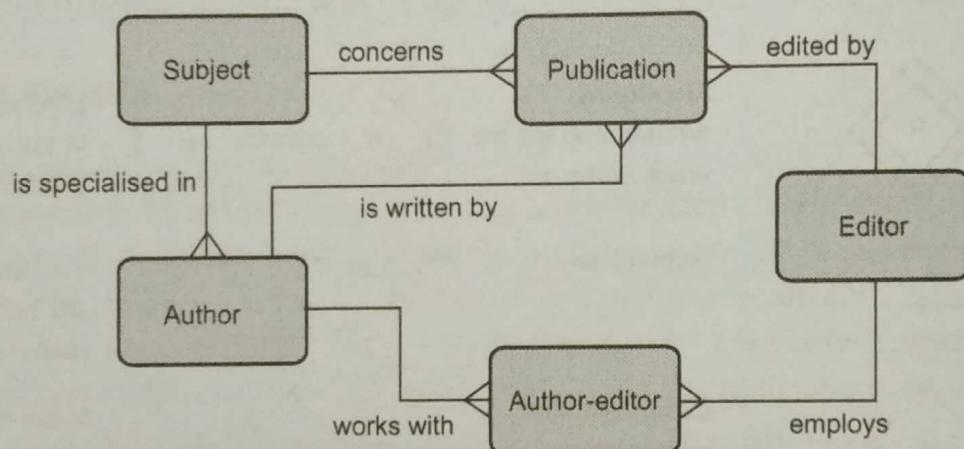


Fig. 1.4.1

Review Questions

1. What is a data flow diagram? Enlist different symbols used in DFDs.
2. Explain E-R diagrams and its components.

1.5 Database Management System (DBMS)

- A distributed database (DDB) is a collection of multiple, logically interrelated databases distributed over a computer network.
- A distributed database management system (D-DBMS) is the software that manages the DDB and provides an access mechanism that makes this distribution transparent to the users.
- A distributed database is a collection of multiple, logically interrelated databases distributed over a computer network.

- Sometimes "distributed database system" is used to refer jointly to the distributed database and the distributed DBMS.

Distributed database system (DDBS) = DDB + D-DBMS

1.5.1 DDBMS Components

- Major components of DDBMS includes the following :-
 1. Computer workstations.
 2. Network hardware and software.
 3. Communications media.
 4. Transaction processor (or, application processor, or transaction manager): Software component found in each computer that requests data.
 5. Data processor or data manager :
- Software component residing on each computer that stores and retrieves data located at the site.
- Data processor may be a centralized DBMS.

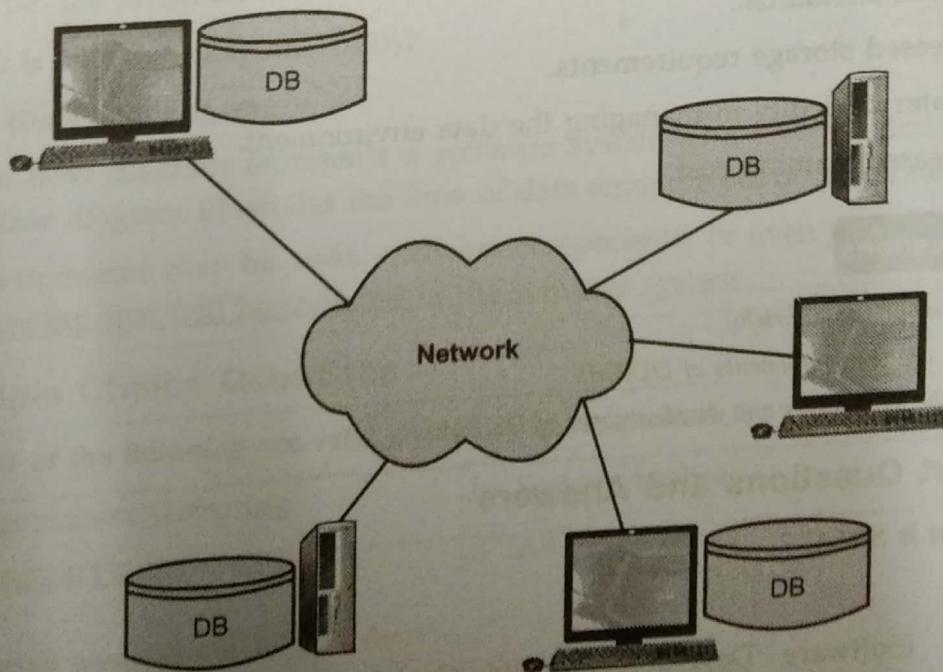


Fig. 1.5.1 DDBMS components

1.5.2 DDBMS Advantages

- Important advantages of DDBMS are :
 1. Data are located near greatest demand site.
 2. Faster data access.
 3. Faster data processing.
 4. Growth facilitation.
 5. Improved communications.
 6. Reduced operating costs.
 7. User-friendly interface.
 8. Less danger of a single-point failure.
 9. Processor independence.

1.5.3 DDBMS Disadvantages

- Disadvantages of DDBMS are listed below -
 1. Complexity of management and control.
 2. Security.
 3. Lack of standards.
 4. Increased storage requirements.
 5. Greater difficulty in managing the data environment.
 6. Increased training cost.

Review Questions

1. Explain DDBMS briefly.
2. What are the components of DDBMS.
3. List the advantages and disadvantages of DDBMS.

1.6 Short Questions and Answers

Q.1 What is SDLC ?

Ans. :

- SDLC (Software Development Life Cycle) is the process of design and development of a product or service to be delivered to the customer that is being followed for the software or systems projects in the Information Technology or Hardware Organizations.

- SDLC framework is useful in delivering the products or customers which ensures high quality and efficient.

Q.2 How many phases are there in the SDLC process ?

Ans. : SDLC process is involved to develop software in a systematic way. So, the SDLC process has a total of 7 phases including -

1. Requirement analysis Phase
2. Planning
3. Design phase
4. Build /Software development phase
5. Testing Phase
6. Deployment / deliver Phase
7. Maintenance

Q.3 What are SDLC models ?

Ans. : SDLC Models

The different SDLC models are Iterative, Waterfall, Agile, V-Shaped and Spiral. Starting from the basic waterfall model the other models have evolved based on the product type, the target audience and the platforms where they have to be executed.

Q.4 What is data flow diagram (DFD)?

Ans. : DFD (Data flow diagram)

- A data flow diagram represents a software system as a labeled, directed graph. Data flow diagram illustrates the flow of data among a set of components.
- The components may be tasks, software components, or even abstractions of the functionality that will be included in the software system.

1.7 Multiple Choice Questions

Q.1 Which of the following are valid step in SDLC framework ?

- | | |
|--|---|
| <input type="checkbox"/> a Requirement Gathering | <input type="checkbox"/> b System Analysis |
| <input type="checkbox"/> c Software Design | <input type="checkbox"/> d All of the above |

Q.2 Waterfall model is not suitable for :

- | | |
|--|---|
| <input type="checkbox"/> a Small projects | <input type="checkbox"/> b Complex projects |
| <input type="checkbox"/> c Accommodating changes | <input type="checkbox"/> d Maintenance Projects |

2

Android OS

Syllabus

Introduction to Android, Android System with Architecture, Android Architecture, Development with Android - Platforms, Tools, Versions, Setup Android Environment, Say Hello to Android Application, Building Blocks of Android Application, Work with Activity, Activity Lifecycle, Intents Fragments, Fragment Lifecycle.

Contents

- 2.1 Introduction to Android
- 2.2 Android APIs
- 2.3 Android Architecture
- 2.4 Android Application Framework
- 2.5 Android Developing Tools
- 2.6 Setup Android Environment
- 2.7 Building Blocks of Android Application
- 2.8 Developing Android Application on Eclipse Platform
- 2.9 Activity Lifecycle
- 2.10 Fragments in Android
- 2.11 Fragment Life Cycle
- 2.12 Replacing Fragment
- 2.13 Activity Lifecycle
- 2.14 Intents
- 2.15 Short Questions and Answers
- 2.16 Multiple Choice Questions

2.1 Introduction to Android

- Android is an open-source software development platform for creating mobile applications.
- Android is an open source software stack that includes :
 1. Operating system : Linux operating system kernel that provides low level interface with the hardware, memory management and process control.
 2. Middleware : A run time to execute Android applications.
 3. Key mobile applications : Email, SMS, PIM, web browser and etc.
 4. Along with API libraries for writing mobile applications : Including open-source libraries such as SQLite, WebKit and OpenGL ES.
- The components of the underlying OS are written in C or C++, user applications are built for Android in Java. Even the built-in applications are written in Java.
- An important feature of the Android platform is that there's no difference between the built-in applications and applications that is create with the Software Development Kit (SDK).
- Android is only a software. By leveraging its Linux kernel to interface with the hardware, Android runs on many different devices from multiple cell phone manufacturers. Developers write applications in Java.
- The Android software environment and hardware it runs on is shown in Fig. 2.1.1.

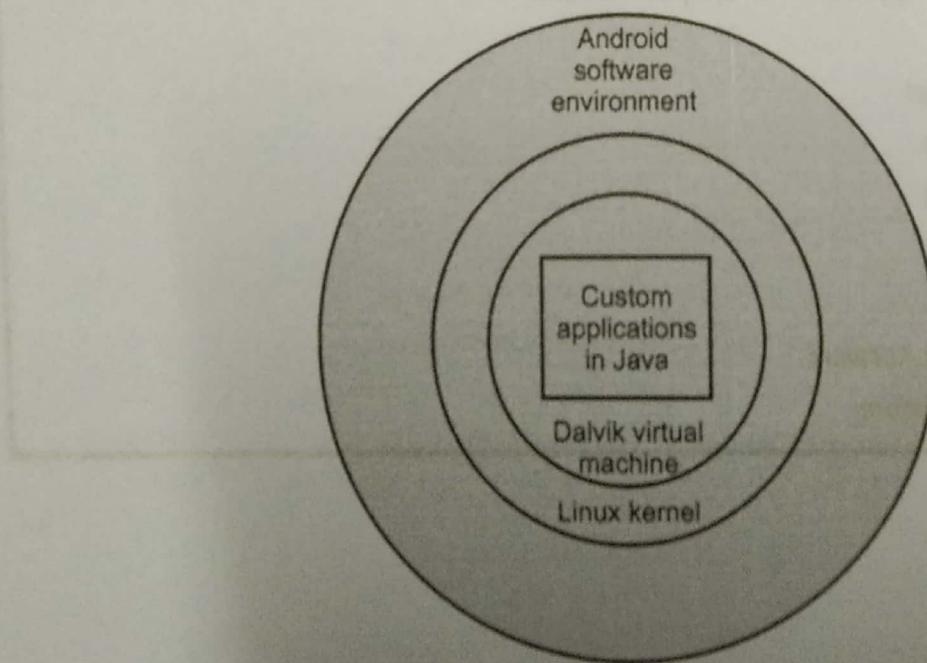


Fig. 2.1.1 Android OS Design and Features

2.1.1 Advantages and Disadvantages of Android

Advantages of Android OS

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

Disadvantages of Android OS

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

2.2 Android APIs

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

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

2.3 Android Architecture

- Following are the different layers in the Android stack :
 - Linux kernel layer
 - Native layer
 - Application framework layer
 - Applications layer

Fig. 2.3.1 illustrates android stack.

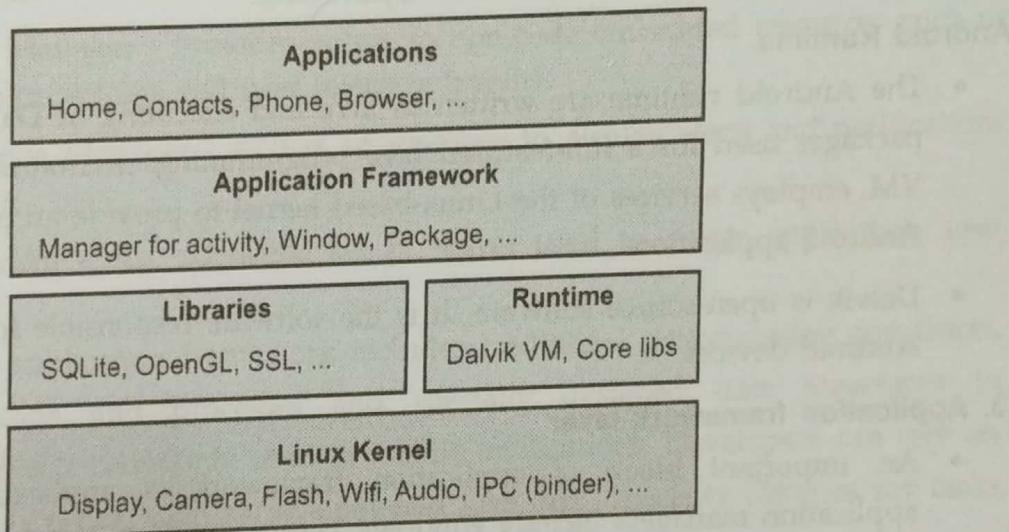


Fig. 2.3.1 Android stack

1. The Linux kernel layer

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

- f) Support for shared libraries
- g) Network stack

2. Native code libraries layer

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

Android Runtime

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

3. Application framework layer

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

4. Application layer

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

2.4 Android Application Framework

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

2.4.1 Development with Android - Platforms

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

2.4.1.1 Conversion Process from Source Code to Android Application

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

2.4.1.2 Android SDK Features

- Important features of Android SDK are as under.
 1. No licensing, distributions or development fees or release approval processes.
 2. Full multimedia hardware control.
 3. APIs for using sensor hardware including accelerometer and the compass.
 4. APIs for location based services.
 5. Android Inter-Process Communication (IPC).
 6. Shared data storage.
 7. Background applications and processes.

8. Home screen widgets, live folders.
9. HTML5 WebKit-based web browser.
10. GSM, EDGE and 3G networks for telephony and data transfer.
11. The Android SDK includes development tools which helps compile and debug any app.
12. Android emulator shows how app will look and behaviour on a real Android device.

2.5 Android Developing Tools

Various Android development tools are :

Android SDK

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

Android Debug Bridge (ADB)

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

Android Developer Tools and Android Studio

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

Dalvik Virtual Machine (DVM)

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

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

Android RunTime (ART)

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

2.6 Setup Android Environment

- Developers have several choices when it comes to integrated development environments (IDEs). Many developers choose the popular and freely available Eclipse IDE to design and develop Android applications.
- Eclipse is the most popular IDE for Android development, and there is an Android plug-in available for facilitating Android development.
- Android applications can be developed on the following operating systems :
- Windows XP (32-bit) or Vista (32-bit or 64-bit)
- Mac OS X 10.5.8 or later (x86 only)
- Linux (tested on Linux Ubuntu 8.04 LTS, Hardy Heron)

- Most developers use the popular Eclipse Integrated Development Environment (IDE) for Android development. The Android development team has integrated the Android development tools directly into the Eclipse IDE. However, developers are not constrained to using Eclipse; they can also use other IDEs.

2.7 Building Blocks of Android Application

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

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

2.7.1 Manifest File

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

4. It declares which permissions the application must have in order to access protected parts of the API and interact with other applications.
5. It also declares the permissions that others are required to have in order to interact with the application's components.
6. It lists the Instrumentation classes that provide profiling and other information as the application is running. These declarations are present in the manifest only while the application is being developed and tested; they're removed before the application is published.
7. It declares the minimum level of the Android API that the application requires.
8. It lists the libraries that the application must be linked against.

2.7.2 Downloading and Installing Android

- To write Android applications, one must configure programming environment for Java development. The software is available online for download at no cost. Android applications can be developed on Windows, Macintosh, or Linux systems.
- To develop Android applications, you need to have the following software installed on computer :
 1. The Java Development Kit (JDK) Version 5 or 6, available for download at <http://java.sun.com/javase/downloads/index.jsp>.
 2. A compatible Java IDE such as Eclipse along with its JDT plug-in, available for download at <http://www.eclipse.org/downloads/>.
 3. The Android SDK, tools and documentation, available for download at <http://developer.android.com/sdk/index.html>.
 4. The Android Development Tools (ADT) plug-in for Eclipse, available for download through the Eclipse software update mechanism. For instructions on how to install this plug-in, see <http://developer.android.com/sdk/eclipse-adt.html>.
- Although this tool is optional for development, but it is highly recommended to install it.
- The Android SDK comes with five major components : the Android SDK License Agreement, the Android Documentation, Application Framework, Tools and Sample Applications.

2.8 Developing Android Application on Eclipse Platform

- Eclipse is an Integrated Development Environment, or IDE, which is software that provides all the essential tools needed for editing, running, and debugging your Java programs.

- The Java Development Kit, or JDK, is a set of development tools used in the programming of Java applications.
- The Eclipse IDE requires that a JDK be locally installed. The JDK can be downloaded from the web at
<http://java.sun.com/javase/downloads/index.jsp>.

Create a new Android Application

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

2.9 Activity Lifecycle

- Applications comprise one or more activities. Because of the mobile and UI experience the operation of an activity is governed by a set of states (like a finite state machine) called the lifecycle.

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

1. **onCreate()**

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

2. **onStart()**

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

3. **onResume()**

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

4. **onPause()**

- The system calls onPause() when the activity loses focus and enters a Paused state. This state occurs when, for example, the user taps the Back or Recents button. When the system calls onPause() for your activity, it technically means

your activity is still partially visible, but most often is an indication that the user is leaving the activity, and the activity will soon enter the Stopped or Resumed state.

- An activity in the Paused state may continue to update the UI if the user is expecting the UI to update. Examples of such an activity include one showing a navigation map screen or a media player playing. Even if such activities lose focus, the user expects their UI to continue updating.

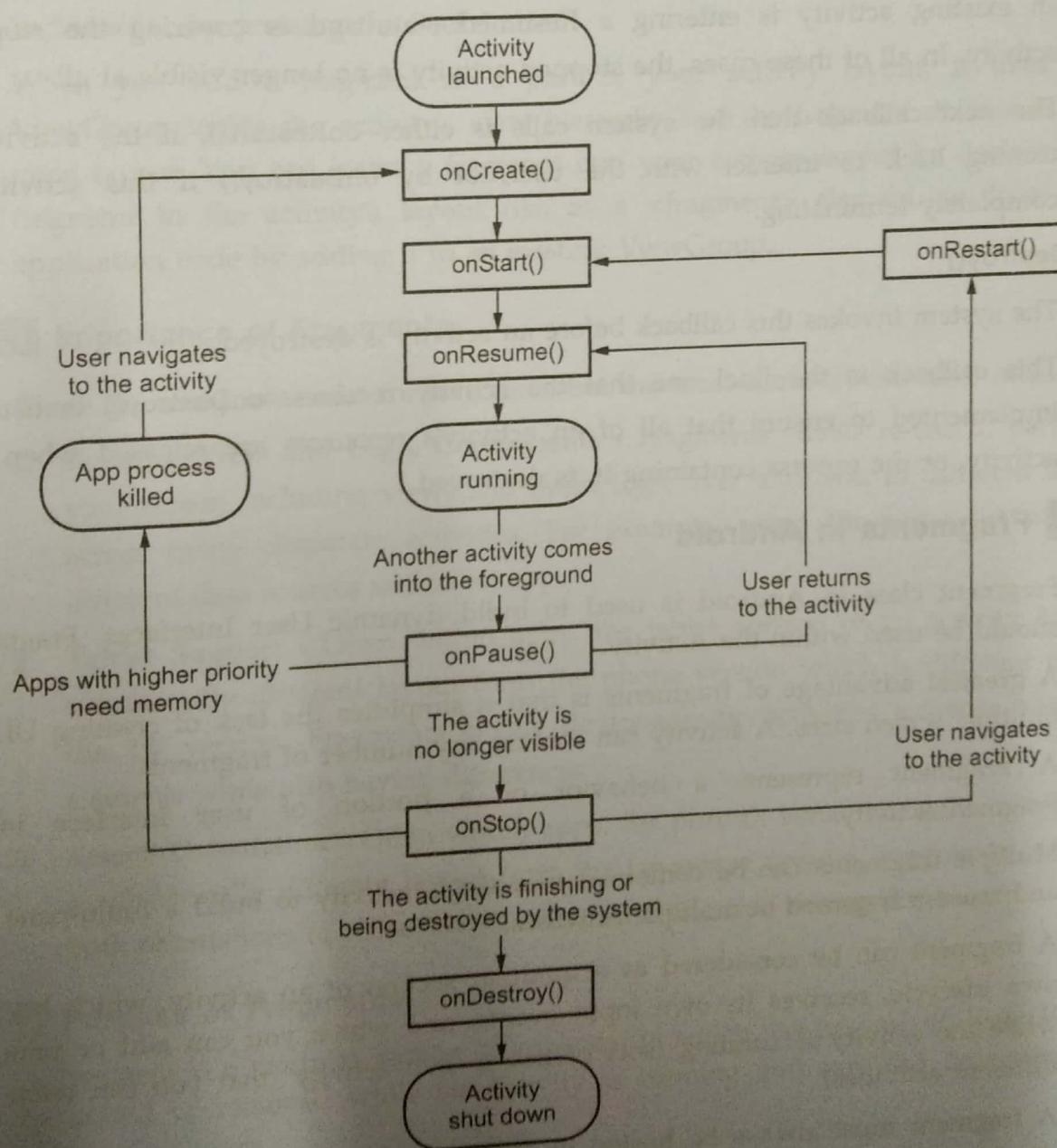


Fig. 2.9.1

- You should **not** use `onPause()` to save application or user data, make network calls, or execute database transactions.
- Once `onPause()` finishes executing, the next callback is either `onStop()` or `onResume()`, depending on what happens after the activity enters the Paused state.

5. `onStop()`

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

6. `onDestroy()`

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

2.10 Fragments in Android

- Fragment class in Android is used to build dynamic User Interfaces. Fragment should be used within the Activity.
- A greatest advantage of fragments is that it simplifies the task of creating UI for multiple screen sizes. A activity can contain any number of fragments.
- A Fragment represents a behavior or a portion of user interface in a FragmentActivity.
- Multiple fragments can be combined in a single activity to build a multi-pane UI and reuse a fragment in multiple activities.
- A fragment can be considered as a modular section of an activity, which has its own lifecycle, receives its own input events, and which you can add or remove while the activity is running (sort of like a "sub activity" that you can reuse in different activities).
- A fragment must always be hosted in an activity and the fragment's lifecycle is directly affected by the host activity's lifecycle. For example, when the activity is

paused, so are all fragments in it, and when the activity is destroyed, so are all fragments.

- However, while an activity is running (it is in the resumed lifecycle state), you can manipulate each fragment independently, such as add or remove them. When you perform such a fragment transaction, you can also add it to a back stack that's managed by the activity-each back stack entry in the activity is a record of the fragment transaction that occurred.
- The back stack allows the user to reverse a fragment transaction (navigate backwards), by pressing the Back button.
- When you add a fragment as a part of your activity layout, it lives in a ViewGroup inside the activity's view hierarchy and the fragment defines its own view layout. You can insert a fragment into your activity layout by declaring the fragment in the activity's layout file, as a <fragment> element, or from your application code by adding it to an existing ViewGroup.

2.10.1 Importance of Fragments

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

2.10.2 Purpose of Fragments

- A fragment is a modular section of an Activity, which has its own lifecycle. It can be added or removed while an Activity is running and can also be reused in different activities.

- **Background :** To create a Fragment you have to subclass the Fragment class. You have to provide a public no-argument constructor, because Android will often re-instantiate a Fragment class when needed (-> state restore).

Purpose of a Fragment

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

2.10.3 Communication between Fragments and Activity

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

2.11 Fragment Life Cycle

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

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

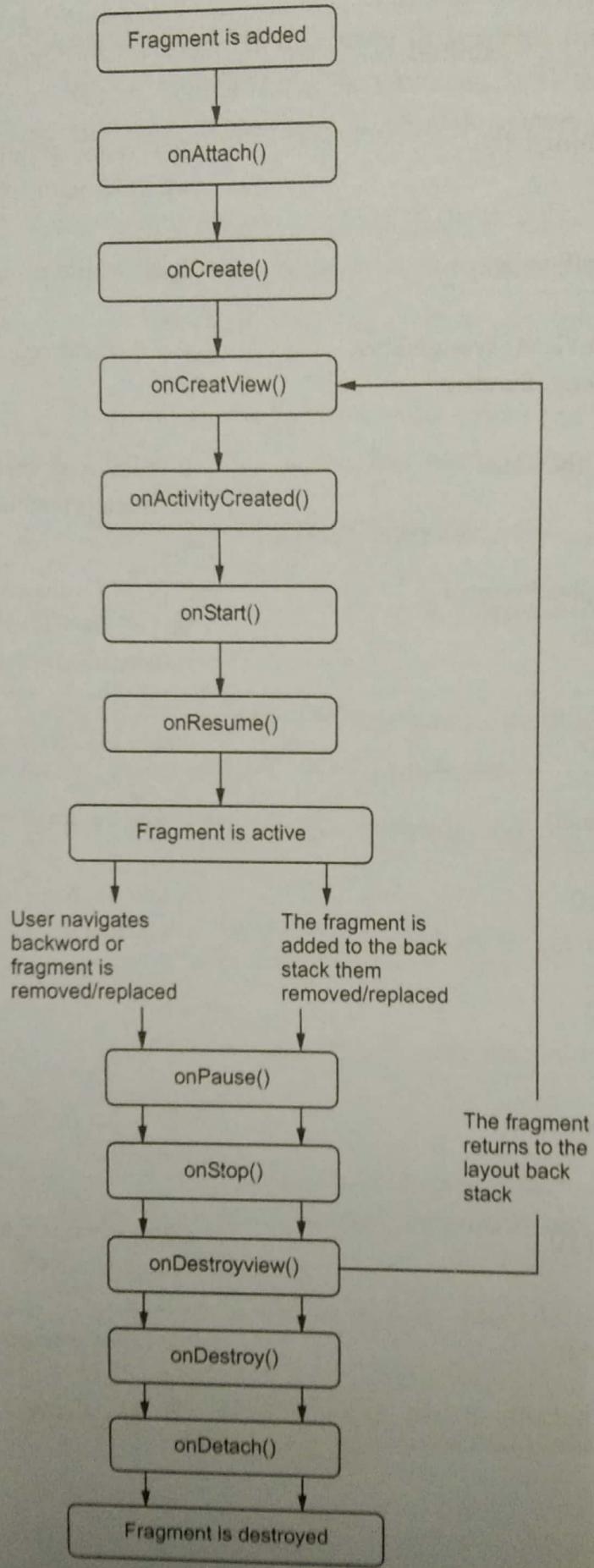


Fig. 2.11.1 Android fragment life cycle

2.12 Replacing Fragment

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

Example of replacing one fragment with another :

KOTLINJAVA

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

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

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

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

2.13 Activity Lifecycle

- Application comprise one or more activities. Because of the mobile and UI experience the operation of an activity is governed by a set of states (like a finite state machine) called the lifecycle.
 - Crashing if the user receives a phone call or switches to another app while using your app.
 - Consuming valuable system resources when the user is not actively using it.

- c) Losing the user's progress if they leave your app and return to it at a later time.
- d) Crashing or losing the user's progress when the screen rotates between landscape and portrait orientation.
- To navigate transitions between stages of the activity lifecycle, the activity class provides a core set of six callbacks : `onCreate()`, `onStart()`, `onResume()`, `onPause()`, `onStop()`, and `onDestroy()`.
- The system invokes each of these callbacks as an activity enters a new state.

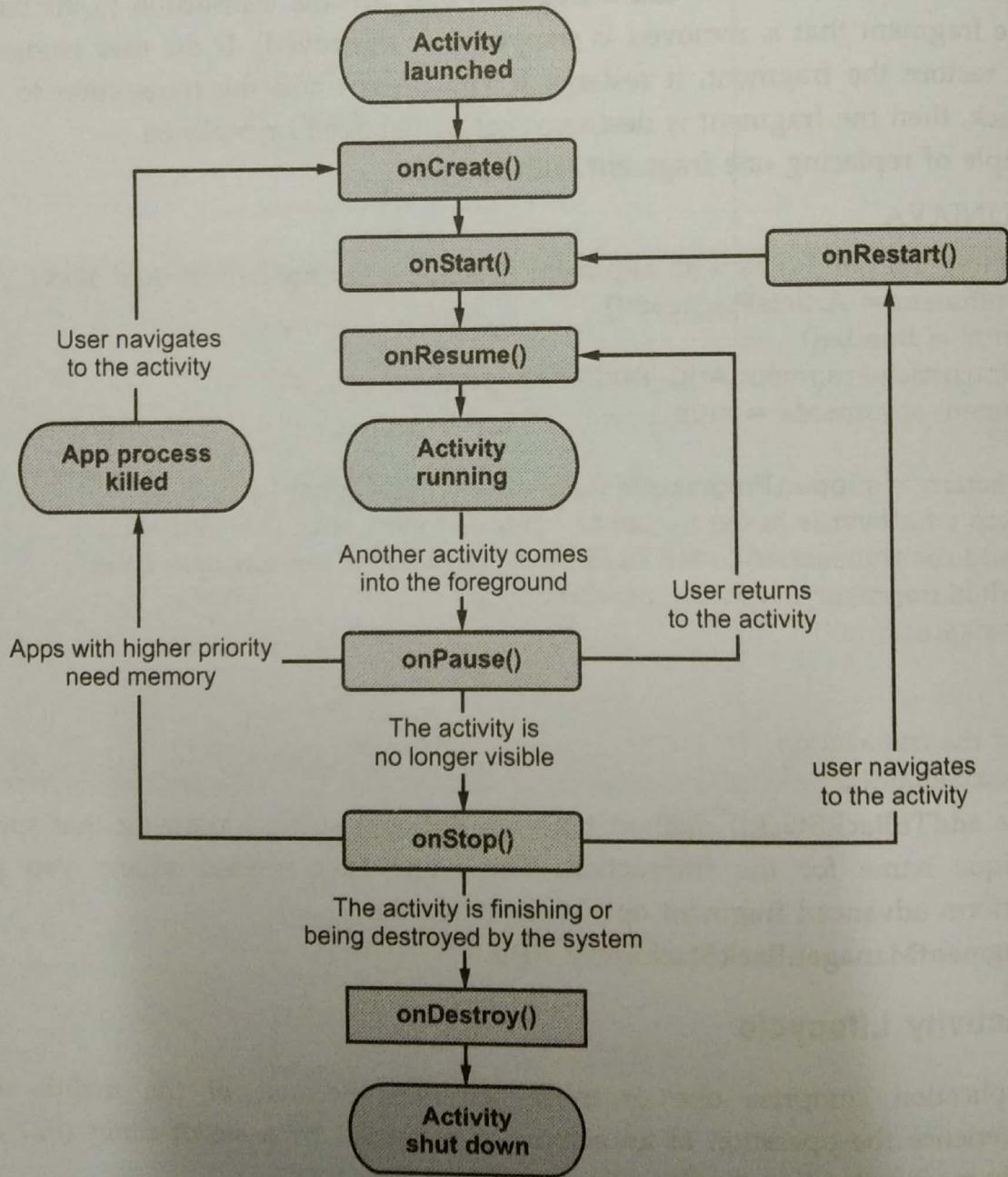


Fig. 2.13.1

1. onCreate()

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

2. onStart()

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

3. onResume()

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

4. onPause()

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

5. onStop()

- The system calls onStop() when the activity is no longer visible to the user. This may happen because the activity is being destroyed, a new activity is starting, or an existing activity is entering a Resumed state and is covering the stopped activity. In all of these cases, the stopped activity is no longer visible at all.

- The next callback that the system calls is either `onRestart()`, if the activity is coming back to interact with the user, or by `onDestroy()` if this activity is completely terminating.

6. `onDestroy()`

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

2.14 | Intents

- Android application components can connect to other Android applications. This connection is based on a task description represented by an Intent object.
- Intents are asynchronous messages which allow application components to request functionality from other Android components.
- Intents are the objects of android. Intent type and intents are mainly useful to perform following things.
- Intents allow you to interact with components from the same applications as well as with components contributed by other applications. For example, an activity can start an external activity for taking a picture.

Components	Description
Starting an activity	By passing an Intent object to <code>startActivity()</code> method we can start a new Activity or existing Activity to perform required things :
Starting a service	By passing an Intent object to <code>startService()</code> method we can start a new Service or send required instructions to an existing Service.
Delivering a broadcast	By passing an Intent object to <code>sendBroadcast()</code> method we can deliver our message to other app broadcast receivers.

- Intent is a messaging object which is used to request an action from another app component such as activities, services, broadcast receivers and content providers. Your code can send them to the Android system defining the components you are targeting. For example, via the `startActivity()` method you can define that the intent should be used to start an activity.
- An intent can contain data via a Bundle. This data can be used by the receiving component.
- In Android the reuse of other application components is a concept known as task. An application can access other Android components to achieve a task. For

example, from a component of your application you can trigger another component in the Android system, which manages photos, even if this component is not part of your application. In this component you select a photo and return to your application to use the selected photo.

2.14.1 Intent to Start another Activity

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

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

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
    /** Called when the user taps the Send button */
    fun sendMessage(view: View) {
        val editText = findViewById<EditText>(R.id.editText)
        val message = editText.text.toString()
        val intent = Intent(this, DisplayMessageActivity::class.java).apply {
            putExtra(EXTRA_MESSAGE, message)
        }
        startActivity(intent)
    }
}
```

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

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

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

- The Intent constructor takes two parameters, a Context and a Class.
- The Context parameter is used first because the Activity class is a subclass of Context.

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

2.14.2 Implicit and Explicit Intent

- An implicit Intent does not name a specific component. It declares a general action to perform. The action specifies the thing that the app wants to do.
- Background : The system is looking for an appropriate component to start :
- If multiple components are compatible with the action, the system shows a dialog so the user can pick which app to use.
- If there is no appropriate component available on the device that can handle the action, your app will crash immediately !

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

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

Explicit Intent

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

Example :

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

2.15 Short Questions and Answers

Q.1 Explain Android architecture components.

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

- Room

Official documentation

Article on how to implement Room Db

Sample implementation

- Live Data

Official documentation

Sample implementation

- ViewModel

Official documentation

Sample implementation

- Data Binding

Official documentation

Sample implementation

- Lifecycles

Official documentation

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

Ans. Tools for developing Android Apps

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

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

Ans.

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

Q.4 Define fragments.

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

Q.5 What is fragment lifecycle.

Ans. : • **onAttach()** : The fragment instance is associated with an activity instance. The fragment and the activity is not fully initialized. Typically you get in this method a reference to the activity which uses the fragment for further initialization work.

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

Q.6 What's the difference between `onCreate()` and `onStart()` ?

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

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

Q.7 What is the difference between fragments & activities. Explain the relationship between the two.

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

Q.8 When should you use a fragment rather than an activity ?

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

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

Q.9 What is an intent ?

Ans. : Intent

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

Two types of intents-

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

Q.10 Difference between Service and Intent Service.

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

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

Q.11 Give Difference between Service, Intent Service, AsyncTask and Threads

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

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

Q.12 State three common usages of intent and how are they invoked.

Ans. : Android Intents are used to

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

Q.13 What are the four essential states of an Activity ?

Ans. : States of Activity

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

Q.14 Differentiate between Broadcast Receivers and Services.

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

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

3

Android UI and Component using Fragments

Syllabus

Create Android UI, Working with Layout, Create Custom Layouts, Work with UI Components and Events, Material Design Toolbar, Tab Layout, Recycler View and Card View, Android Menus.

Contents

- 3.1 Android Graphics Interface
- 3.2 Create Android UI
- 3.3 Working with Layout
- 3.4 Work with UI Components and Events
- 3.5 Material Design Toolbar
- 3.6 Android Menus
- 3.7 Android Layout Classes
- 3.8 Short Questions and Answers
- 3.9 Multiple Choice Questions (MCQs)

3.1 Android Graphics Interface

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

3.2 Create Android UI

- Android is a widely used OS made for smart phones and tablets. It is an open source project led by Google and it is released under Apache License.
- Android has a very large community that extend its features and creates apps that cover almost all aspects.
- All android applications, called apps, are built on Android UI framework. App interface is the first thing a user sees and interacts with. From the user perspective, this framework keeps the overall experience consistent for every app installed in our smartphone or tablets. At the same time, from the developer perspective, this framework provides some basic blocks that can be used to build complex and consistent user interface (API).
- Android UI interface is divided in three different areas :
 1. Home screen
 2. All apps
 3. Recent screen.
- The home screen is the "landing" area when we power our phone on. This interface is highly customizable and themed. Using widgets we can create and personalize our "home" screen.
- All apps is the interface where the app installed are displayed, while recent screens are the list of last used apps.

- Since its born, Android has changed a lot in terms of its features and its interfaces. The growth of the smartphone power made possible creating ever more appealing apps.
- At the beginning, apps in Android did not have a consistent interface and well defined rules so every app had a different approach, navigation structure and buttons position. This caused user confusion and it was one of the most important missing features compared to the iOS.

3.2.1 Editable Text View

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

For example :

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

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

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

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

3.2.2 Radio and Toggle Buttons

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

3.2.3 Creating Radio Button

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

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

and within the activity:

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

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

                // Ninjas rule
                break;
    }
}
```

3.2.4 Spinners

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

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

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

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

3.3 Working with Layout

- Layouts are invisible structured containers used for holding other Views and nested layouts.
- A typical layout defines the visual structure for an android user interface and can be created either at run time using View/ViewGroup objects or you can declare your layout using simple XML file main_layout.xml which is located in the res/layout folder of your project.
- A layout may contain any type of widgets such as buttons, labels, textboxes, and so on. Once your layout has created, you can load the layout resource from your application code, in your Activity.onCreate() callback implementation as shown below -

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

Android Layout types

- There are number of Layouts provided by Android which you will use in almost all the Android applications to provide different view, look and feel.
- 1. **Linear Layout** - Linear layout is a view group that aligns all children in a single direction, vertically or horizontally.
- 2. **Relative Layout** - Relative layout is a view group that displays child views in relative positions.
- 3. **Table Layout** - Table layout is a view that groups views into rows and columns.
- 4. **Absolute Layout** - Absolute layout enables you to specify the exact location of its children.
- 5. **Frame Layout** - The frame layout is a placeholder on screen that you can use to display a single view.
- 6. **List View** - List view is a view group that displays a list of scrollable items.
- 7. **Grid View** - Grid view is a view Group that displays items in a two-dimensional, scrollable grid.

3.4 Work with UI Components and Events

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

Android Event Listeners

- In android, **Event Listener** is an interface in the View class that contains a single call-back method. These methods will be called by the Android framework when

the View which is registered with listener is triggered by user interaction with the item in UI.

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

3.5 Material Design Toolbar

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

1. TextView

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

2. EditText

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

3. ListView

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

4. Spinner

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

5. Button

- Standard push-button

6. CheckBox

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

7. RadioButton

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

3.6 Android Menus

- Menus offer a way to expose application functions without sacrificing valuable screen space. Each activity can specify its own activity menu that's displayed when the device's menu button is pressed.

- Android also supports context menus that can be assigned to any View within an Activity. A View's context menu is triggered when a user holds the middle D-Pad button, depresses the trackball or longpresses the touch screen for around 3 seconds when the View has focus.
- Activity and context menus support submenus, checkboxes, radio buttons, shortcut keys and icons.
- To improve the usability of application menus, Android features a three-stage menu system optimized for small screens :

1. The Icon Menu

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

	submenu	Menu Item 1	Menu Item 2
			More

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

2. The Expanded Menu

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

Menu Item 5
Menu Item 6
Menu Item 9
<input checked="" type="checkbox"/> CheckBox
<input type="radio"/> Radiobutton 1
<input type="radio"/> Radiobutton 2
<input type="radio"/> Radiobutton 3 Menu+a

3. Submenus

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

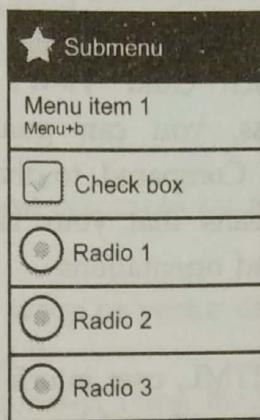


Fig. 3.6.1

3.7 Android Layout Classes

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

1. FrameLayout

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

2. LinearLayout

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

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

3. RelativeLayout

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

4. TableLayout

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

5. AbsoluteLayout

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

Working with Layouts

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

3.8 Short Questions and Answers

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

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

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

Q.2 Difference between RelativeLayout and LinearLayout ?

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

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

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

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

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

Q.4 How to reduce apk size in Android ?

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

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

Q.5 What is an activity in Android ?

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

Q.6 What is a service ?

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

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

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

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

4

Database Connectivity

Syllabus

Storage in Android, Shared Preferences, Shared Preferences Layout, Android Requesting Permission at run time (Android 6.0), Work with SD Card and Files, Database in Android, Realm-No SQL Database.

Contents

- 4.1 Storage in Android
- 4.2 Using Internal Storage
- 4.3 Working with SD Card / External Storage
- 4.4 Shared Preferences Layout
- 4.5 Database in Android
- 4.6 SQLite
- 4.7 Content Provider
- 4.8 Short Questions and Answers
- 4.9 Multiple Choice Questions

4.1 Storage in Android

Store application-specific data

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

4.1.1 Application Data Folder Scope

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

4.1.2 Creating File in App Data Folder

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

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

4.1.3 Search for Files in App Data Folder

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

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

```
FileList files = driveService.files().list()
    .setSpaces("appDataFolder")
    .setFields("nextPageToken, files(id, name)")
    .setPageSize(10)
    .execute();

for (File file : files.getFiles()) {
    System.out.printf("Found file: %s (%s)\n",
        file.getName(), file.getId());
}
```

4.2 Using Internal Storage

Android Read Write Data To Internal File -

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

4.3 Working with SD Card / External Storage

Android External Storage

- Android data can be saved in internal storage (ROM) , external storage (SD card), shared preferences or SQLite database.
- Android external storage can be used to write and save data, read configuration files etc.

- External storage such as SD card can also store application data, there's no security enforced upon files you save to the external storage.
- In general there are two types of External Storage :
 1. Primary External Storage : In built shared storage which is "accessible by the user by plugging in a USB cable and mounting it as a drive on a host computer". Example : When we say Nexus 5 32 GB.
 2. Secondary External Storage : Removable storage. Example : SD Card
- All applications can read and write files placed on the external storage and the user can remove them. We need to check if the SD card is available and if we can write to it. Once we've checked that the external storage is available only then we can write to it else the save button would be disabled.
- Firstly, we need to make sure that the application has permission to read and write data to the users SD card, so lets open up the *AndroidManifest.xml* and add the following permissions :

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

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

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

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

4.4 Shared Preferences Layout

- Shared preferences allow you to read and write small amounts of primitive data (as key/value pairs) to a file on the device storage.
- The SharedPreference class provides APIs for getting a handle to a preference file and for reading, writing, and managing this data.

- The shared preferences file itself is managed by the Android framework, and accessible to (shared with) all the components of your app. That data is not, however, shared with or accessible to any other apps.
- The data you save to shared preferences is different from the data in the saved activity state.
- The data in the activity instance state is retained across activity instances in the same user session.
- Shared preferences persist across user sessions, even if your app is killed and restarted or if the device is rebooted.
- Use shared preferences only when you need to save a small amount data as simple key/value pairs. To manage larger amounts of persistent app data use the other methods such as SQL databases.

4.4.1 Creating Private and Shared Preferences

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

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

- Creating shared preferences is similar. The only two differences are that we must name our preference set and use a different call to get the preference instance :
import android.content.SharedPreferences;
...
SharedPreference settings =

```
getSharedPreferences("MyCustomSharedPreferences", 0);
```

- You can access shared preferences by name from any activity in the application. There is no limit to the number of different shared preferences you can create. You can have some shared preferences called UserNetworkPreferences and another called AppDisplayPreferences.

- How you organize shared preferences is up to you, the developer.

- However, you want to declare your preference name as a variable (in a base class or header) so that you can reuse the name across multiple activities. For example
public static final String PREFERENCE_FILENAME = "AppPrefs";

4.5 Database in Android

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

4.5.1 Inserting Data

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

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

```
// Defines a new Uri object that receives the result of the insertion
lateinit var newUri: Uri

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

}

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

- The data for the new row goes into a single ContentValues object, which is similar in form to a one-row cursor. The columns in this object don't need to have the same data type, and if you don't want to specify a value at all, you can set a column to null using ContentValues.putNull().

- The snippet doesn't add the _ID column, because this column is maintained automatically. The provider assigns a unique value of _ID to every row that is added. Providers usually use this value as the table's primary key.
- The content URI returned in newUri identifies the newly-added row, with the following format :


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

4.5.2 Updating Data

- To update a row, you use a ContentValues object with the updated values just as you do with an insertion, and selection criteria just as you do with a query. The client method you use is ContentResolver.update().
- You only need to add values to the ContentValues object for columns you're updating. If you want to clear the contents of a column, set the value to null.
- The following snippet changes all the rows whose locale has the language "en" to a have a locale of null. The return value is the number of rows that were updated :

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

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

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

...

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

- ```
 selectionArgs // the value to compare to
)
• You should also sanitize user input when you call ContentResolver.update().
```

#### 4.5.3 Deleting data

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

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

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

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

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

#### 4.6 SQLite

- SQLite is a software library that provides a relational database management system.
- The lite in SQLite means light weight in terms of setup, database administration and required resource.
- A SQLite database is a good storage solution when you have structured data that you need to store persistently and access, search, and change frequently.
- SQLite is self-contained means it requires minimal support from the operating system or external library.
- This makes SQLite usable in any environments especially in embedded devices like iPhones, Android phones, game consoles, handheld media players, etc.
- SQLite is developed using ANSI-C. The source code is available as a big sqlite3.c and its header file sqlite3.h.

- If you want to develop an application that uses SQLite, you just need to drop these files into your project and compile it with your code.
- SQLite databases are very lightweight. Unlike other database systems, there is no configuration, installation required to start working on an SQLite database.
- When you use a SQLite database, all interactions with the database are through an instance of the SQLiteOpenHelper class which executes your requests and manages your database for you.
- The Android SQLite Database requires very little memory (around 250 kb), which is available on all android devices.
- Every device has an inbuilt support for SQLite database, which is automatically managed on android right from its creation, execution to querying up process.
- SQLite is an open source database, available on every android database. It supports standard relations database features, like SQL syntax, transactions & SQL statements. SQLite is considerably, the lighter version of SQL database, where most of the SQL commands don't run on SQLite database.
- Once SQLite is in place, it is important to ensure that a feature or command is available in SQLite; only then can it be executed.
- The basic advantages of SQLite :
  1. It's a light weight database.
  2. Requires very little memory.
  3. An automatically managed database.
- The SQLite supports only three Datatypes :
  1. Text(like string) - for storing data type store.
  2. Integer(like int) - for storing integer primary key.
  3. Real(like double) - for storing long values.

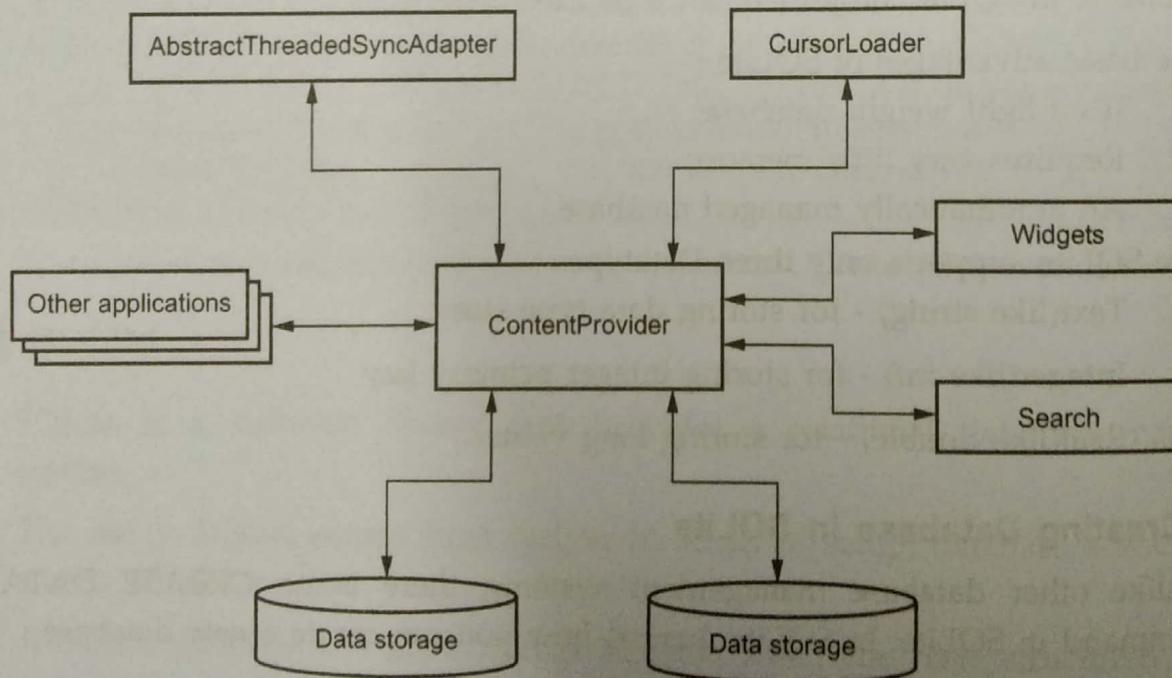
#### 4.6.1 Creating Database in SQLite

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

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

## 4.7 Content Provider

- A content provider manages access to a central repository of data. A provider is part of an Android application, which often provides its own UI for working with the data.
- Content providers are primarily intended to be used by other applications, which access the provider using a provider client object.
- Together, providers and provider clients offer a consistent, standard interface to data that also handles inter-process communication and secure data access.
- A content provider presents data to external applications as one or more tables that are similar to the tables found in a relational database.
- A row represents an instance of some type of data the provider collects, and each column in the row represents an individual piece of data collected for an instance.
- Typically you work with content providers in one of two scenarios; you may want to implement code to access an existing content provider in another.
- A content provider coordinates access to the data storage layer in your application for a number of different APIs and components as illustrated in Fig. 4.7.1.



**Fig. 4.7.1**

- The content provider and other components include :
  1. Sharing access to your application data with other applications
  2. Sending data to a widget
  3. Returning custom search suggestions for your application through the search framework using SearchRecentSuggestionsProvider.

4. Synchronizing application data with your server using an implementation of AbstractThreadedSyncAdapter
5. Loading data in your UI using a CursorLoader

#### 4.7.1 Content Provider Classes

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

#### 4.7.2 Deleting Database Records

- The data modification clauses in SQLite are INSERT, UPDATE, and DELETE statements. It is used for inserting new rows, updating existing values, or deleting rows from the database.
- You can remove records from the database using the remove() method. The remove() method takes few arguments.

• Passing null to the WHERE clause deletes all records within the table. For example, this function call deletes all records within the table called tbl\_authors :

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

- Most of the time, though, we want to delete individual records by their unique identifiers.

• The following function takes a parameter bookId and deletes the record corresponding to that unique id (primary key) within the table called tbl\_books :

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

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

tbl\_books for a given author by the author's unique id:

```
public void deleteBooksByAuthor(Integer authorID) {
 int numBooksDeleted = mDatabase.delete("tbl_books", "authorid=?",
 new String[] { authorID.toString() });
}
```

## 4.8 Short Questions and Answers

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

**Ans. :**

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

### Q.2 What is SQLite database ?

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

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

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

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

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

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

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

**4.9 Multiple Choice Questions****Q.1 What is a thread in android ?**

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

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

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

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

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

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

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

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

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

# 5

## Applicability to Industrial Projects

### Syllabus

*Web services and Parsing, JSON Parsing, Access web data with JSON, Connect to Web Services, Using Async Task & Third Party Library : Retrofit.*

### Contents

- 5.1 Web Services and Parsing
- 5.2 JSON Parsing
- 5.3 Access Web Data with JSON
- 5.4 Using AsyncTask and Third Party Library : Retrofit
- 5.5 Short Questions and Answers
- 5.6 Multiple Choice Questions

## 5.1 Web Services and Parsing

- A web service is a type of software that can help streamline every aspect of a mobile application and integrate it for other web services.
- Web service needed to provide interoperability, means connecting various applications.
- Web services allows different apps to communicate with each other and share the data and services among themselves. For example, an android application can interact with java or .net application using web services.
- Web service is a standardized medium to act as a bridge to communication between the client and server applications on the world wide web.

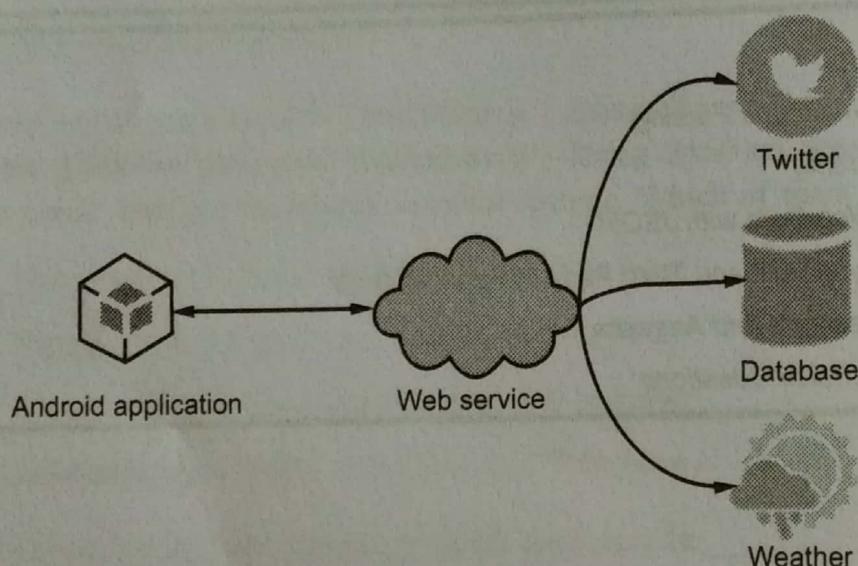


Fig. 5.1.1 Android web services

### Types of web services

- There are 4 types of web services.
  1. Rest
  2. SOAP
  3. UDDI
  4. XML-RPC

### Android web services components

- Android web services architecture has 3 major components.
  1. Publisher
  2. Subscriber
  3. Broker

**Data formats**

- A web service can exchange data in any format, but two formats are the most popular for data exchange between applications :
  1. **XML** - Standard format for data exchange among applications, in order to avoid datatype-mismatch problems.
  2. **JavaScript Object Notation (JSON)** - Text-based open standard for representing data. Uses characters such as brackets ([]), braces ({}), colons (:) and commas (,) to represent data.

**5.1.1 Types of Web Services**

- Popular web services protocols are :
1. **SOAP :**
    - SOAP is known as the Simple Object Access Protocol.
    - SOAP was developed as an intermediate language so that applications built on various programming languages could talk quickly to each other and avoid the extreme development effort.
  2. **WSDL :**
    - WSDL is known as the Web Services Description Language(WSDL).
    - WSDL is an XML-based file which tells the client application what the web service does and gives all the information required to connect to the web service.
  3. **REST :**
    - REST stands for RE presentational state transfer.
    - REST is used to build web services that are lightweight, maintainable and scalable.

**5.1.2 Requirements of Web Services**

- In order to be considered as a "Web Service", the software application must meet the following requirements :
  1. It must use an XML messaging system to communicate and correspond to all commands.
  2. It is not based on any single programming language or operating system.
  3. It is a self-describing and self-contained program that uses XML language.
  4. It must be easily discoverable with a search or find feature.
  5. It must be available over the Internet and can include availability over private or intranet networks as well.

## 5.2 JSON Parsing

- Douglas Crockford specified the JSON format in the early 2000s.
- JSON is very light weight, structured, easy to parse and much human readable.
- JSON stands for JavaScript Object Notation. JSON is a widely used data interchange format.
- JSON is used for serializing and transmitting structured data over a network connection.
- JSON is widely adopted in web services and APIs - enabling web applications to transfer and retrieve data with a common format.
- JSON can be used with many modern programming languages.
- So far the, only choice for open data interchange was the eXtensible Markup Language (XML).
- JSON is best alternative to XML when your android app needs to interchange data with your server.
- JSON is more compatible, lightweight and relatively easy to use compared to other formats for open data sharing.

### Advantages of JSON over XML

- 1) JSON is faster and easier than xml for AJAX applications.
- 2) Unlike XML, it is shorter and quicker to read and write.

### JSON structure

- The structure of a JSON object is as follows :
  - The data are in name/value pairs
  - Data objects are separated by commas
  - Curly braces {} hold objects
  - Square brackets [] hold arrays
- Each data element is enclosed with quotes "" if it is a character or without quotes if it is a numeric value.
- A JSON can be of two structures -
  1. A collection of name value pairs (Object).
  2. A list of values (Array).

### Definition and usage

- Parse the data with JSON.parse() and the data becomes a JavaScript object.
- The string has to be written in JSON format.
- The JSON.parse() method can optionally transform the result with a function.

### JSON object

- A JSON object contains key/value pairs like map.
- The keys are strings and the values are the JSON types.
- Keys and values are separated by comma. The { (curly brace) represents the json object.

### Syntax

```
JSON.parse(string, function)
```

- JSON can actually take the form of any data type that is valid for inclusion inside JSON, not just arrays or objects. So for example, a single string or number would be valid JSON.
- Number, string, boolean, null, object and array are important data types used in JSON.
- Unlike in JavaScript code in which object properties may be unquoted, in JSON only quoted strings may be used as properties.
- JSON provides support for all browsers offered by many languages.

### Serializing and deserializing JSON data :

1. **Serialization** : This is the process of translating data structures or object state into a format that can be stored.
2. **Deserialization** : This is the process where a data structure is extracted from a series of bytes.

### 5.2.1 Characteristics of JSON

- Some important features of JSON are :
  1. **Easy to use** - JSON API offers high-level artificial appearance or effect, which helps you to simplify commonly used use-cases.
  2. **Performance** - JSON is quite fast as it consumes very less memory space, which is especially suitable for large object graphs or systems.
  3. **Free tool** - JSON library is open source and free to use.
  4. **Doesn't require to create mapping** - Jackson API provides default mapping for many objects to be serialized.
  5. **Clean JSON** - Creates clean and compatible JSON result that is easy to read.
  6. **Dependency** - JSON library does not require any other library for processing.

## 5.2.2 Applications of JSON

- Some common applications of JSON are :
  1. JSON useful in transferring data from a server.
  2. Sample JSON file format used in transmit and serialize all types of structured data.
  3. Allows you to perform asynchronous data calls without the need to do a page refresh.
  4. JSON is a tool to transmit data between a server and web applications.
  5. JSON is widely used for JavaScript-based application, which includes browser extension and websites.
  6. Data can be transmitted between the server and web application using JSON.
  7. Web services and Restful APIs use the JSON format to get public data.

## 5.3 Access Web Data with JSON

- There are multiple ways to invoke a JSON web service since there are different ways to supply parameters.
- Parameters in web service invocations are included.
- Invocation is matched to a method, especially in the case of overloaded service methods.

### Passing parameters as part of a URL path

- Specify parameters in name-value pairs after the service URL.
- Parameter names must be formed from method argument names by converting them from camel case to all lower case, dash-separated names.
- Parameters can be passed in any order; there is no need to follow the order of the arguments in the method signatures.
- When a method name is overloaded, the best match is used.
- One can also pass parameters in a URL query.

### Passing parameters as a URL query

- To pass in parameters as request parameters, specify them as-is (camel case) and set them equal to their argument value.
- As with passing parameters as part of a URL path, the parameter order is not important and the best match rule applies for overloaded methods.
- Now you know a few different ways to pass parameters. You can also pass URL parameters in a mixed way.

- Parameter values are sent as strings using the HTTP protocol. Before a matching Java service method is invoked, each parameter value is converted from a string to its target Java type.
- Conversion for common types (e.g., long, string, boolean) happens directly.

## 5.4 Using AsyncTask and Third Party Library : Retrofit

- Android framework provides special instruments to handle data sending using the network. For example, there is AsyncTask. But this can't handle an application lifecycle and this problem can lead to memory leaks. Data may be lost or the application might crash.
- AsyncTask was intended to enable proper and easy use of the UI thread. However, the most common use case was for integrating into UI and that would cause context leaks, missed callbacks or crashes on configuration changes.
- It also has inconsistent behavior on different versions of the platform, swallows exceptions from `doInBackground` and does not provide much utility over using `Executors` directly.
- AsyncTask is designed to be a helper class around Thread and `Handler` and does not constitute a generic threading framework.
- AsyncTasks should ideally be used for short operations (a few seconds at the most.) If you need to keep threads running for long periods of time, it is highly recommended you use the various APIs provided by the `java.util.concurrent` package such as `executor`, `ThreadPoolExecutor` and `FutureTask`.
- An asynchronous task is defined by a computation that runs on a background thread and whose result is published on the UI thread.
- An asynchronous task is defined by 3 generic types, called params, progress and result and 4 steps, called `onPreExecute`, `doInBackground`, `onProgressUpdate` and `onPostExecute`.

### 5.4.1 Retrofit

- Retrofit is a clean, simple and light library for Android.
- Retrofit library is created by square open source, it's a REST client for android and Java. By the use of this library, it is easy to request web services of REST with GET, POST, PUT and much more.
- Retrofit is a REST client for Android, through which you can use its easy interfaces which can work on any android app.
- Retrofit can perform asynchronous and synchronous requests as well.

- It has a feature with that is automatic JSON parsing without any effort.
- This feature alone makes it powerful enough from other networking libraries. So it to make a competitor for others. Although it has more useful features.
- Retrofit used for turns your HTTP API request response into Java interface. Without server communication we cant develop any high level android application. Because when we integrate with server we can access more storage without any traffic.
- Local storage does not give traffic free request response and speed also very slow. Most of developers use volley library because they are dont work with retrofit. Once we work with retrofit REST API, then we cant go other libraries.

#### 5.4.2 Built-in Annotations

- Retrofit supports the following built-in annotations for each of the standard request types :
  1. **GET** : A method that's annotated with @GET is responsible for processing a HTTP GET request, where data is retrieved from a server. This is the annotation we'll be using to retrieve the list of names.
  2. **POST** : A method that's annotated with @POST is responsible for processing a HTTP POST request, where you send data to a server.
  3. **PUT** : This method will process a HTTP PUT request, where we provide some data and ask the server to store it under a specific URL.
  4. **DELETE** : This method will process a HTTP DELETE request, which specifies a resource that should be deleted.
  5. **HEAD** : This method will process a HTTP HEAD request. HEAD is similar to GET, except that a @HEAD method retrieves information without the corresponding response body. By using @HEAD annotations, you can obtain data that's written in a response header, without having to retrieve the rest of that content.

#### 5.4.3 Advantages of Android Retrofit

1. Retrofit android is simple to use. It essentially lets you treat API calls as simple Java method calls, so you only define which URLs to hit and the types of the request/response parameters as Java classes.
2. The entire network call + JSON/XML parsing is completely handled by it (with help from Gson for JSON parsing), along with support for arbitrary formats with pluggable serialization/deserialization.

#### 5.4.4 Volley

- Volley is a networking library it offers great features like synchronous requests, asynchronous requests, prioritization, making multiple requests at the same time, ordered requests and of course caching.
- Android volley has a flexible caching mechanism. When a request is made through volley first the cache is checked for an appropriate response if it is found there than it is returned and parsed else network hit is made.

#### Inbuilt types

- The data returned by an Api plays vital role in implementation of network call and its handling. It is one of the important parameter to decide which option we should use.
- Volley can capture four types of responses automatically through these requests :
  1. StringRequest - This type of request converts the response into a string.
  2. JsonObjectRequest - This type of request and response is automatically converted into a JSONObject.
  3. JSONArrayRequest - This type of request and response is automatically converted into a JSONArray.
  4. ImageRequest - This type of request converts the response into a decoded bitmap.

#### 5.5 Short Questions and Answers

**Q.1 What is a web service ?**

**Ans. : Web service**

- A web service is a way for two machines to communicate with each other over a network.
- A web server running on a computer listens for requests from other computers.
- When a request from another computer is received, over a network, the web service returns the requested resources. This resource could be JSON, XML, an HTML file, images, audio files, etc.

**Q.2 What is an API ?**

**Ans. : An API or Application Programming Interface, is a set of definitions and protocols that allow one application to communicate with another application.**

**Q.3 What is JSON ?****Ans. : JSON**

- JSON [JavaScript Object Notation] is similar to XML in that it also stores and enables you to send data in a standardized format.
- JSON just uses a different, object-based, methodology for systematically storing data.

**Q.4 What is SOAP ?****Ans. :**

- SOAP stands for Simple Object Access Protocol.
- SOAP is known as a transport-independent messaging protocol. SOAP is based on transferring XML data as SOAP messages. Each message has something which is known as an XML document.
- Only the structure of the XML document follows a specific pattern, but not the content. The best part of web services and SOAP is that its all sent via HTTP, which is the standard web protocol.

**Q.5 What is retrofit ?****Ans. : Retrofit is a REST client for Java and Android.**

- Its a simple network library that used for network transactions.
- By using this library we can seamlessly capture JSON response from web service/web API.
- It's easy and fast library to retrieve and upload the data (JSON or any other structured data) via a REST based web service.

**Q.6 What is serializing and deserializing JSON data ?****Ans. : Serializing and deserializing JSON data -**

1. **Serialization** : This is the process of translating data structures or object state into a format that can be stored.
2. **Deserialization** : This is the process where a data structure is extracted from a series of bytes.

**Q.7 What is volley ?****Ans. :**

- Volley is a networking library it offers great features like synchronous requests, asynchronous requests, prioritization, making multiple requests at the same time, ordered requests and of course caching.

# 6

# Advanced Android Development

## Syllabus

*Google Map, Location Service and GPS, Creating Google Map, Work with Location, Location service with Location Manager, Find Current Location, Geo coding Graphics and Animation, Work with 2D Graphics, Bitmap, Animation, Frame Animation, Tween Animation, View Animation, Multimedia in Android, Play Audio Files, Play Video Files Work in Background, Services, Notification Services, Broadcast Receiver Introduction to Firebase with simple CRUD Operation*

## Contents

- 6.1 Google Map
- 6.2 Location Service and GPS
- 6.3 Geocoding
- 6.4 Reverse Geocoding Revgeocode
- 6.5 Working with 2D Graphics
- 6.6 Working with Animation
- 6.7 Alarms in Android
- 6.8 Download Manager
- 6.9 Multimedia in Android
- 6.10 Work in Background Services
- 6.11 Notification Services
- 6.12 Broadcast Receiver
- 6.13 Introduction to Firebase with Simple CRUD Operation
- 6.14 Short Questions and Answers
- 6.15 Multiple Choice Questions with Answers

## 6.1 Google Map

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

### 6.1.1 APIs in Google Maps

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

#### Web APIs

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

#### Web Service APIs

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

#### Mobile APIs

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

### 6.1.2 Steps for Getting the Google Maps API Key

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

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

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

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

**Step 4 :** In this open Enable APIs AND SERVICES.

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

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

**Step 7 :** Now go to Credentials.

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

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

### 6.1.3 Elements of Google Map

- Elements of a map are as follows :

#### 1. GoogleMap -

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

#### 2. MapView -

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

#### 3. SupportMapFragment -

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

**4. Shapes -**

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

**5. My Location Layer -**

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

**6. Marker -**

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

**7. UiSettings -**

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

**6.1.4 Adding a Map to an Application**

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

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:id="@+id/map"
 tools:context=".MapDemoActivity"
 android:name="com.google.android.gms.maps.SupportMapFragment"/>
```

### 6.1.5 Types of Google Map

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

### 6.1.6 Displaying the User's Current Location

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

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

- Google maps provides various user-friendly controls to user to interact with map.

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

### 6.1.8 Features of Google Maps

- Following are the features provided by Google Maps.
  1. It searches places and route directions.
  2. It provides distance information.
  3. It helps to find traffic details and navigation.
  4. It displays street views.
  5. It receives verbal instructions.
  6. It provides location sharing and location editing.

### 6.1.9 Difference between Google Maps and Google Earth

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

## 6.2 Location Service and GPS

### 6.2.1 Current Location

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

- To have location based functionality in your activity, you will have to implement two interfaces -

`GooglePlayServicesClient.ConnectionCallbacks`

`GooglePlayServicesClient.OnConnectionFailedListener`

- These interfaces provide following important callback method -

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

### 6.2.2 Location Listener

- The `LocationListener` interface, which is part of the Android Locations API is used for receiving notifications from the `LocationManager` when the location has changed.
- The `LocationManager` class provides access to the systems location services.
- The `LocationListener` class needs to implement the following methods.
  - `onLocationChanged(Location location)`** : Called when the location has changed.
  - `onProviderDisabled(String provider)`** : Called when the provider is disabled by the user.
  - `onProviderEnabled(String provider)`** : Called when the provider is enabled by the user.
  - `onStatusChanged(String provider, int status, Bundle extras)`** : Called when the provider status changes.

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

- `LocationManager.GPS_PROVIDER`** : Determines location using satellites. Depending on the conditions, this provider may take a while to return a location fix.

2. `LocationManager.NETWORK_PROVIDER` : Determines location based on the availability of nearby cell towers and WiFi access points. This is faster than `GPS_PROVIDER`.

## 6.3 Geocoding

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

### 6.3.1 Use of Geolocations

- Geolocation can be used to determine time zone and exact positioning coordinates, such as for tracking wildlife or cargo shipments.
- Both mobile and desktop devices can use geolocation. There are apps which use location once, and there are apps which continuously track your location as well.
- Some famous apps using Geolocation are -
  1. Uber / Lyft ↔ Cab booking
  2. Google Maps (of course) ↔ Map services
  3. Swiggy / Zomato ↔ Food delivery
  4. Fitbit ↔ Fitness app
  5. Instagram / Facebook ↔ For tagging photos

### 6.3.2 Effectiveness of Geocoder

- A system that provides geocoding is called a geocoding service (or just geocoder).
- To assess the effectiveness of a geocoder, you need to use two basic parameters :
  1. **The size of the database.** The fuller the base, the more accurate and detailed the response to the request will be. With a large database, the address can be specified up to the house even in a small town.
  2. **Geocoding speed.** The speed of a geocoder is determined by the number of requests processed per second. The more requests the geocoder is able to handle, the more users the system can service.

### 6.4 Reverse Geocoding Revgeocode

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

#### 6.4.1 Difference between Forward and Reverse Geocoding

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

### 6.5 Working with 2D Graphics

- Graphics are created by the developer and added to the application. They can be created from results from various operations such as querying, identifying, geoprocessing, geocoding or routing.

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

### 6.5.1 Uses of Graphics

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

### 6.5.2 Using Drawable Object

- Android provides a set of View widgets that provide general functionality for a wide array of user interfaces.
- You can also extend these widgets to modify the way they look or behave.
- Custom 2D rendering can be done using the various drawing methods contained in the Canvas class or create Drawable objects for things such as textured buttons or frame-by-frame animations.
- A drawable resource is a general concept for a graphic that can be drawn to the screen.
- When you need to display static images in your app, you can use the Drawable class and its subclasses to draw shapes and images.
- Drawables are used to define shapes, colors, borders, gradients, etc. which can then be applied to views within an Activity.

- Every Drawable is stored as individual files in one of the res/drawable folders.
- There are three ways to define and instantiate a drawable :
  1. Using an image saved in your project resources;
  2. Using an XML file that defines the Drawable properties;
  3. Or using the normal class constructors.

### 6.5.3 Using the Shape Drawable Object

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

### 6.5.4 Hardware Acceleration

- Android 3.0 (API Level 11) introduced a new rendering pipeline to allow applications to benefit from hardware-accelerated 2-D graphics.
- You can use hardware acceleration if you have complex computations like scaling, rotating, translation of images.
- Hardware Acceleration is used to improve the performance and aesthetics of your activities and views.
- All the SDK Views, layouts, and effects support hardware acceleration.

- Android gives you the option to enable or disable hardware acceleration at multiple level.
  - 1) Application level
  - 2) Activity level
  - 3) Window level
  - 4) View level

### 1. Application level

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

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

### 2. Activity level

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

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

### 3. Window level

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

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

### 4. View level

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

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

## 6.6 Working with Animation

- Animations can add visual cues that notify users about what's going on in your app.
- They are especially useful when the UI changes state, such as when new content loads or new actions become available.

- Animations also add a polished look to your app, which gives it a higher quality look and feel.

- Android supports three types of Animation :

### 1) Property animation

- a) Property animators were introduced in Android 3.0 (API level 11). It is a powerful framework that can be used to animate almost anything.
- b) You can use a property animator to animate almost any property on a target object.
- c) Property animators are extremely useful and are used extensively for animating fragments in android.

### 2) View animations

- a) Animations that can be applied for views to rotate, move and stretch.

### 3) Frame animations

- a) Frame-by-frame animations produce a sequence of Drawables, each of which is displayed for a specified duration.

## 6.6.1 Clock for Animated Graphics

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

## 6.6.2 Controlling Timing Events

- There are several ways to control the timing events in an animation process :
- 1. `Thread.sleep(millis)` and `Object.wait(millis)` are standard blocking functions that can be used to generate desired time delays. When these functions are executed, the `uptimeMillis()` clock stops. The thread can be woken up by the function `Thread.interrupt()`.

2. `SystemClock.sleep(millis)` is a utility function very similar to `Thread.sleep(millis)`, except that it ignores `InterruptedException`.
3. We can use the Handler class to schedule asynchronous callbacks at an absolute or relative time. A handler object uses the `uptimeMillis()` clock to keep time. It requires an event loop to wait for an event to happen.
4. We can use the AlarmManager class to access the system alarm services such as triggering one-time or recurring events when the thread is in a blocked state.

### 6.6.3 Bitmap Animation

- If we want to get the bitmaps in motion, we can easily get them.
- It is possible to animate the bitmaps such as icons, using drawable animation APIs.
- We define them statically using a drawable resource, but it can also be defined at runtime.

### 6.6.4 Frame Animation

- In frame animation, move or modify objects one frame at a time. This frame-by-frame animation is time consuming but is sometimes the only way to create complex animated effects.
- A frame animation is like displaying many pictures in succession to form a moving picture.
- Now a days, Frame animation is rarely used.

### 6.6.5 Tween Animation in Android

- Create starting frames and ending frames and let flash figure out where everything goes in the in-between frames, which is why it's called tweening.
- Tweening is much more fun and easier than frame-by-frame animation. If you can create the animation you want by tweening, it's definitely the way to go.
- Flash offers two types of tweening : motion tweening and shape tweening.
- Tween Animation is basically used to make a GUI of the Activity or Application more interactive and happening.
- It is basically an animation that translates, rotates, and scales any type of view in Android.

- Tween animation takes some parameters such as values, times, duration, size, etc and performs the required animation on that object. It applies to all types of objects.

### Tween Animation Types

- There are four main types of tween animation :
  1. Alpha : Gradient transparency animation
  2. Scale : Gradient size scaling animation
  3. Translate : Screen transition position moving animation
  4. Rotate : Screen transition rotation animation

### Steps for usage of Tween Animation

1. Defining an animation animated object.
  2. Set some properties of the animation.
  3. Start animation with the startAnimation() method.
- In order to perform this animation, we will call static function loadAnimation() of AnimationUtils. The syntax is :
 

```
1. Animation myanimation =
 AnimationUtils.loadAnimation(getApplicationContext(),
 R.anim.this_animation);
```
  - To apply animation to an object, we call startAnimation() method of the object, as follows :
 

```
1. TextView text = (TextView) findViewById(R.id.tv1);
2. text.startAnimation(animation);
```
  - This Animation class has the following functions that are very useful :

Method	Description
start( )	It starts the animation.
setDuration(long duration)	It sets the time duration of the animation.
getDuration( )	It gets the method to be set by the above.
end( )	It ends the animation that was started.
cancel( )	It cancels the working of animation.

### 6.7 Alarms in Android

- In Android, alarms allow you to schedule your application to run at some point in the future.

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

#### Characteristics of alarms :

- The characteristics of alarms are :
  1. They let you fire Intents at set times and/or intervals.
  2. You can use them in conjunction with broadcast receivers to start services and perform other operations.
  3. They operate outside of your application, so you can use them to trigger events or actions even when your app is not running, and even if the device itself is asleep.
  4. They help you to minimize your app's resource requirements. You can schedule operations without relying on timers or continuously running background services.

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

#### 6.8 Download Manager

- The download manager is a system service that handles long-running HTTP downloads.
- Clients may request that a URI be downloaded to a particular destination file.
- The download manager was introduced in Android 2.3 (API level 9).
- One big advantage of Android Download Manager is that it optimizes the handling of long-running downloads in the background.
- The download manager handles HTTP connections, monitors connectivity changes, and ensures each download completes successfully.

- If you want to download large files/streaming, then you can't use Retrofit or Volley, both recommend using DownloadManager instead, which supports resuming and progress notifications. download manager will conduct the download in the background, taking care of HTTP interactions and retrying downloads after failures or across connectivity changes and system reboots.
- Apps that request downloads through this API should register a broadcast receiver for ACTION\_NOTIFICATION\_CLICKED to appropriately handle when the user clicks on a running download in a notification or from the downloads UI.
- The application must have the Manifest.permission.INTERNET permission to use this class.

## 6.9 Multimedia in Android

- The Android platform provides comprehensive multimedia functionality.
- Multimedia generally encompasses all the fun, non-text content : pictures, sound, video.
- One of the most important components of the media framework is the MediaPlayer class.

### 6.9.1 Media Player

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

#### 6.9.1.1 User Media Player

- The media player class primarily handles the playback of audio and video within android application.
- You can play the media stored in application resources, local files, Content Providers or streamed from a network URL using the media player.
- Media player's management audio and video are handled in the form of state machines. Following are the transitions in the state machine.
  - 1) Initialize the media player with media to play.
  - 2) Prepare the media player for the playback.

- 3) Start the playback.
- 4) Pause or stop the playback prior to its completing.
- 5) The playback is complete.
- To play a media resource, you need to create an instance of `MediaPlayer` class, then initialize it with media source and prepare it for playback.

## 6.9.2 Play Audio Files

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

### 6.9.2.1 Media Formats

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

#### 1. Audio Formats

- The Table 6.9.1 describes the media format for audio and codec support built into the Android platform.

Format Codec	Encoder	Decoder	Details	Supported File Types (s) /Container Formats
AAC LC	•	•	Support for mono /stereo/5.0/5.1 content with standard sampling rates from 8 to 48 kHz.	• 3GPP (.3gp) • MPEG-4 (mp4 m4a)
HE-AACv1 (AAC+)	• (Andriod 4.1 +)	•		• ADTS raw AAC (.aac dedoce in Andriod 3.1+ encode in Andriod 4.0 +, ADIF not supported)
HE-AACv2 (enhanced AAC+)		•	Support for stereo/5.0/5.1 content with standard sampling rates from 8 to 48 kHz.	
AAC ELD (enhanced low delay AAC)	• (Andriod 4.1 +)	• (Andriod 4.1 +)	Support for mono /stereo content with standard sampling rates from 16 to 48 kHz.	• MPEG-TS (.ts, not seekable, Andriod 3.0+)
AMR-NB	•	•	4.75 to 12.2 kbps sampled @ 8 kHz.	3GPP (.3gp)

AMB-WB		9 rates from 6.60 kbit/s to 23.85 kbit/s kbit @ 16 kHz.	3GPP (.3gp)
FLAC	(Android 3.1+)	Mono/Stereo (no multichaneel). Sample rates up to 48 kHz (but up to 44.1 kHz is recommenced on devices with 41.1 kHz output, as the 48 to 44.1 kHz downampler does not include a low-pass filter). 16-bit recommended ; no dither applied for 24-bit.	FLAC (.flac) only
MP3		Mono /Stereo 8-320 kbps constant (CBR) or variable bit rate (VBR)	MP3 (.mp3)
MIDI		MIDI Type 0 and 1. DLS Verserion 1 and 2. XMF and Mobile. XMF Support for ringtone forms RTTTL/RTX, OTA, and iMelody	<ul style="list-style-type: none"> <li>• Type 0 and 1 (.mid, .xmf, .mxmf)</li> <li>• RTTTL/RTX (.rtttl, .rtx)</li> <li>• OTA (.ota)</li> <li>• iMelody (.imy)</li> </ul>
Vorbis			<ul style="list-style-type: none"> <li>• Ogg (.ogg)</li> <li>• Matroska (.mkv, Android 4.0+)</li> </ul>
PCM/ WAVE	(Anddrioid 4.1 +)	8-and 16 bit linear PCM (rates up to limit to hardware). Sampling rates for raw PCM recordings at 8000, 16000 and 44100 Hz.	Wave (.wav)

Table 6.9.1

## 2. Video Formats

- Depending on the codec used to compress the video container format, it may still not play on your Android phone even though it generally is supported.

- To watch your videos still, you can convert the video into another format or choose an Android compatible conversion right away.
- The codecs supported by Android phones are H.263, H.264, MPEG-4, and V8. This leads to the following video container formats to be supported by Android devices :
  - 3GP
  - MKV
  - MP4
  - TS
  - WEBM

### 6.9.2.2 Playing Audio

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

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

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

### 6.9.3 Play Video Files

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

**VideoView code In XML Android**

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

**MediaController In VideoView**

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

**6.9.4 Recording Video**

- Android has two methods for recording video -

**1) Intents to Record Video**

- a. The easiest way to record a video from within your application is by using an Intent and applying the constants from the MediaStore class ACTION\_VIDEO\_CAPTURE.
- b. Starting a new Activity with this Intent launches the native video recorder, allowing users to start, stop, review, and retake their video.

**2) Media Recording API -**

- a. You can use the MediaRecorder class to record video.
- b. To record any media in Android, your application needs the CAMERA and RECORD\_AUDIO and/or RECORD\_VIDEO permissions as applicable :

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

- c. Media Recorder manages recording as a state machine. The order in which you manage and configure the Media Recorder is important.
- The transitions in the state machine are as follows
  - 1) Create a new Media Recorder.
  - 2) Unlock the Camera and assign it to the Media Recorder.
  - 3) Specify the input sources to record from.

- 4) Select a profile to use for Android 2.2 and above, or define the output format and specify the audio and video encoder, frame rate, and output size.
  - 5) Select an output file.
  - 6) Assign a preview Surface.
  - 7) Prepare the Media Recorder for recording.
  - 8) Record.
  - 9) End the recording.
- d. When the recording is finished free all the resources on your Media Recorder object
- ```
mediaRecorder.release()
```

6.9.5 Creating a Sound Pool

- The SoundPool helps developers to make a collection of samples, to load them into memory, not only from a resource of the application APK, but also from a folder in the file system.
- Creating a SoundPool preloads the audio tracks used by your application and optimizes their require management.
e.g. - All the sound effects required for a particular level in a game.
- You can create a *SoundPool* class to manage audio when your application requires low audio latency or will be playing multiple audio streams simultaneously (such as a game with multiple sound effects).
- You can setup *SoundPool* to make a sound in specific stream type. There are several types of stream as follows :

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

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

6.10 Work in Background Services

- Android background service is an android component that runs in the background. There is no GUI for users to interact with the android background service object directly, it is usually started in android activity and runs in the same thread of the activity.
- Background/Data notifications are "silent" meaning they do not display any message or play a sound when received by your app.
- They are designed to keep your app's data "up-to-date" by providing a way to "wake up" the app to refresh the data in the background.
- Data notifications are handled within the NotificationExtenderService - This can be set up to receive data notifications when your app is not running, or to override how notifications are shown in the notification shade.

Background/Data Notifications with the REST API

- OneSignal will automatically send the notification as a data notification under the following notification parameter conditions :
 - content_available : true
 - alert is omitted
 - badge is omitted
 - sound is omitted
 - category is omitted
 - thread_id is omitted
 - subtitle is omitted
 - contents is omitted
 - headings is omitted

6.10.1 Android Background Service Methods

- If you want to customize the android background service component, you need to override the below methods.
 1. **onCreate** : When the service object is created, it will run this method. This method runs only once.
 2. **onStartCommand** : This method is invoked every time when the background service is started. You can execute tasks or start child threads in this method. Because the android background service is executed in the main activity thread, so if you do not use child threads and if the task is very time-consuming, the android activity main thread will be blocked by the service object.
 3. **onDestroy** : This method is invoked when the background service is destroyed. You can release related resources in it such as close database connection, write data to file, etc.
 4. **stopSelf** : If you want to stop and exit the running service in your source code, you can call this method.

6.11 Notification Services

- A Service is not a separate process. The Service object itself does not imply it is running in its own process; unless otherwise specified, it runs in the same process as the application it is part of.
- A Service is not a thread. It is not a means itself to do work off of the main thread (to avoid Application Not Responding errors).
- To create a status bar notification, we'll need to use two classes : **Notification** and **NotificationManager**.

1. Notification

- Notification defines the properties of the status bar notification like the icon to display, the text to display when the notification first appears on the status bar and the time to display.

2. NotificationManager

- NotificationManager is an android system service that executes and manages all notifications. Hence we cannot create an instance of the NotificationManager but we can retrieve a reference to it by calling the **getSystemService()** method.

6.12 Broadcast Receiver

- Android broadcast receiver is an android component that is used to broadcast the messages to the system or other applications. Examples are -
 1. Screen has turned off.
 2. Battery is getting low.
 3. A picture was captured,...etc.
- All registered broadcast receiver for a particular event are notified by the Android runtime once the registered event fires.
- BroadCastReceivers register to receive events in which they are interested. When events occurs they are presented as intents, those intents are then broadcast to the system.
- Android routes the intents to BroadCastReceivers that have registered to receive them. Broadcast receivers receive the intent via a call to `onReceive()`. Typical use case :
 1. Register BroadCastReceivers
 2. BroadCast An Intent
 3. Android Delivers Intent to Registered Recipients by calling their `onReceive()` method
- BroadCastReceivers Can Register in two Ways -
 - 1) Statically, In `AndroidManifest.xml`
 - 2) Dynamically, By calling a `registerReceiver()` method

6.13 Introduction to Firebase with Simple CRUD Operation

- Firebase is online cloud based database its stored data in JSON format. CRUD (Create, Read, Update and Delete) operations are performed using Firebase realtime database.
- We can use firebase realtime database in multiple platforms like Android, iOS, Windows and web applications. Dont need to write server side programming language for get the data from server side like php node. We can directly retrieve the data from client side applications.
- Firebase realtime database is a schemaless database in which the data is stored in JSON format. Basically the entire database is a big JSON tree with multiple nodes. So when you plan your database, you need to prepare the JSON structure in way that the data is accessible in easier way by avoiding nesting of child nodes.

6.13.1 Firebase Realtime Database Feature

- Important features of firebase realtime database are -
 1. **Realtime** - It means the data is synchronized with all connected devices within milliseconds of changes being made to the data.
 2. **Offline** - The Firebase Realtime Database persists data on your device. The data is always available, even app offline. The data is automatically synced once the app comes back online.
 3. **Security** - You can define your security rules which controls who has what access to your firebase database. You can also integrate firebase authentication to control access to your data.

6.14 Short Questions and Answers

Q.1 What is alarm manager broadcast receiver service ?

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

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

Q.2 Explain types of alarm in android.

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

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

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

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

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

Q.4 What is sleep mode in android ?

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

Q.5 What is google maps ?

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

Q.6 What is google maps API ?

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

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

Q.7 What are the properties of info window ?

Ans. : Following are the properties of Info window :

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

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

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

Q.9 What is SoundPool ?

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

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

Q.10 How camera is initialize ?

Ans. : Camera initialization

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

Q.11 What is Android Media player ?

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

Q.12 What is ANR in Android ?

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

7

Work with Android System & Development and Deployment

Syllabus

Work with android system Development and Deployment

Text to Speech, Camera, Taking Picture with Camera, Manage Bluetooth Connection, Monitor and Manage Wi-Fi, Accelerometer Sensor & Gyroscope.

Dalvik Debug Tool, Logcat, Emulator Control, Device Control, Work with ADB, Connect Real Devices, Execute Application on Real Device, Publish your Application

Contents

- 7.1 Text to Speech
- 7.2 Using Camera
- 7.3 Manage Bluetooth Connection
- 7.4 Monitor and Manage Wi-Fi
- 7.5 Accelerometer Sensor
- 7.6 Gyroscope
- 7.7 Dalvik Debug Tool
- 7.8 Signing the Android Application
- 7.9 Publishing Android App
- 7.10 Distribution of Android App
- 7.11 App Characteristics
- 7.12 Short Questions and Answers
- 7.13 Multiple Choice Questions with Answers

7.1 Text to Speech

- Android Text To Speech is also referred to as Android TTS. TextToSpeech as the name suggests is used to synthesize speech from the text string.
- Converting text into speech is a feature included in Android from API 21 which allows you to transfer a text to speech. It supports many different languages.
- To convert Text to Speech in Android, `speak()` method of `android.speech.tts.TextToSpeech` class is used.

7.2 Using Camera

- Android provides working with camera in two ways :

1. Camera Intent

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

e.g.

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

- b. This launches a camera application to take the photo, providing your users with the full suite of camera functionality without you having to rewrite the native camera application.
- c. Once users are satisfied with the image, the result is returned to your application within the Intent received by the `onActivityResult` handler.

2. Camera API

- a. To access the camera hardware directly, you need to add the CAMERA permission to your application manifest :
`<uses-permission android:name=" android.permission.CAMERA"/>`
- b. Use the Camera class to adjust camera settings, specify image preferences, and take pictures.
- c. To access the Camera, use the static open method on the Camera class:
`Camera camera = Camera.open();`
- d. When you are finished with the Camera, remember to free the camera resources by calling release :
`camera.release();`

3 Manage Bluetooth Connection

- Bluetooth is a wireless technology standard used by two (or more) devices for communication over short distances. The big advantages of Bluetooth are that it is wireless, inexpensive and automatic.
- Bluetooth is a way to exchange data with other devices wirelessly. The Android platform includes support for the Bluetooth network stack, which allows a device to wirelessly exchange data with other Bluetooth devices.
- Bluetooth uses a technique called spread-spectrum frequency hopping, which allows connection between several devices simultaneously, without them interfering with one another.
- Class 1 Bluetooth supports a maximum transfer speed of 1 Megabit Per Second (Mbps), while class 2 Bluetooth, the common standard on handheld devices, can manage up to 3 Mbps, with a maximum communication distance of up to 10 meters.
- Android provides Bluetooth API to perform several tasks such as :
 1. Scan for other Bluetooth devices
 2. Query the local Bluetooth adapter for paired Bluetooth devices
 3. Connect to other devices through service discovery
 4. Transfer data to and from other devices
 5. Manage multiple connections

7.3.1 Android Permissions

- There is need to declare these permissions in the "AndroidManifest.xml":

```
<manifest ... >
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    ...
</manifest>
```

android.permission.BLUETOOTH : For performing Bluetooth communication, such as requesting a connection, accepting a connection, and transferring data.

android.permission.BLUETOOTH_ADMIN : For discovering local Bluetooth devices.

android.permission.ACCESS_COARSE_LOCATION or

android.permission.ACCESS_FINE_LOCATION: For gathering information about the location of the user. This information may come from the user's own device, or Bluetooth beacons in use at locations such as shops.

7.3.2 Interfaces and Classes

- All of the Bluetooth APIs are available in the `android.bluetooth` package.
- The following are the classes and interfaces you need in order to create Bluetooth connections :

1. 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.

2. 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.

3. 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`.

4. 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`.

5. 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.

6. 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.

7. 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.

8. 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).

9. BluetoothHealth

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

10. 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.

11. BluetoothHealthAppConfiguration

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

12. 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.

7.4 Monitor and Manage Wi-Fi

- The Wi-Fi scanning capabilities provided by the *WifiManager* API can be used to get a list of Wi-Fi access points that are visible from the device.
- The `android.net.wifi.WifiManager` class can be used to manage the wifi connectivity. It can be used to add network, disable network, scan for access points, disconnect etc.

Wi-Fi scanning process

- There are three steps to the scanning process :
 1. Register a broadcast listener for `SCAN_RESULTS_AVAILABLE_ACTION`
 2. Request a scan using `WifiManager.startScan()`
 3. Get scan results using `WifiManager.getScanResults()`

Classes for Wi-fi

- To check the network connection, you typically use the following classes :
 1. `ConnectivityManager` : Answers queries about the state of network connectivity. It also notifies applications when network connectivity changes.

2. **NetworkInfo** : Describes the status of a network interface of a given type (currently either Mobile or Wi-Fi).

7.5 Accelerometer Sensor

- A software sensor combines low-level, raw data from multiple hardware sensors to generate new data that is not only easy to use, but also more accurate.
- Accelerometer measures both the static acceleration and dynamic acceleration.
 1. **Static acceleration** - Gravity field of earth
 2. **Dynamic acceleration** - Caused by the movement of the accelerometer
- The Android platform supports three broad categories of sensors :
 1. **Motion sensors** - These sensors measure acceleration forces and rotational forces along three axes. This category includes accelerometers, gravity sensors, gyroscopes, and rotational vector sensors.
 2. **Environmental sensors** - These sensors measure various environmental parameters, such as ambient air temperature and pressure, illumination, and humidity. This category includes barometers, photometers, and thermometers.
 3. **Position sensors** - These sensors measure the physical position of a device. This category includes orientation sensors and magnetometers.

Using accelerometer in Android devices

- The following code shows you how to get an instance of the default acceleration sensor :

```
private SensorManager mSensorManager;
private Sensor mSensor;
...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
```

Applications of accelerometer -

- List of field applications that use accelerometer :
 - a) Engineering - Car, machines, buildings
 - b) Biology - Animal dynamic body acceleration
 - c) Industry - Turbines, pumps, fans, compressors
 - d) Building and structures - Moving loads on bridges, earthquakes and aftershocks
 - e) Medical application - pedometer, CRP D-Padz
 - f) Transport - Intelligent compaction rollers, trains to detect tilts

- g) Volcanology - Motion of magma in active volcanoes
- h) Consumer electronics - Display screens, hard disk

7.6 Gyroscope

- The gyroscope measures the rate of rotation in rad/s around a device's x, y, and z axis. The sensor's coordinate system is the same as the one used for the acceleration sensor.
- A conventional gyroscope measures orientation, in contrast to MEMS (Micro Electro-Mechanical System) types, which measure angular rate, and are therefore called rate-gyros.
- The gyroscope allows you to determine the angular velocity of an Android device at any given instant. In simpler terms, it tells you how fast the device is rotating around its X, Y, and Z axes.
- A Gyroscope is used to maintain a reference direction or provide stability in navigation, stabilizers, etc.
- Gyroscope or a Gyro sensor in smartphone used to sense angular rotational velocity and acceleration.
- Gyroscopes use the same coordinate system as accelerometers, whereas it applies on all axis that the rotation is positive if the device is rotated counter-clockwise.
- Mobile games able to play using motion sense in mobile phones, tablets, etc, is due to a Gyroscope Sense.
- Most gyroscopes on Android devices are vibrational and measure the rotation of a device with a pair of vibrating arms that take advantage of what is known as the Coriolis effect, which is caused by the Earth's rotation.
- The combination of accelerometer and gyroscope creates a powerful motion sensor. In a single Android motion sensor device there are 3 axis-perpendicular mounted accelerometers and 3 rotational-axis mounted gyroscopes.
- This motion sensor is applicable in health management apps such as movement tracking. Movement such as walking, running, and jumping can easily be recognized by the sensor based on the value of the acceleration and the angular velocity of the device.
- The following code shows you how to get an instance of the default gyroscope :

```
private SensorManager mSensorManager;  
private Sensor mSensor;  
...  
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);  
mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
```

Applications of Gyroscope in Smartphone

1. Motion-sensing GUI
2. Answer phone/open website
3. Image stabilization
4. GPS-inertial navigation
5. Motion-sensing control game

7.7 Dalvik Debug Tool

- The Dalvik Debug Monitor Service (DDMS) is a debugging tool used in the Android platform. The Dalvik Debug Monitor Service is downloaded as part of the Android SDK.
- It uses Android Debug Bridge to connect to a device and help you debug applications running inside a Dalvik virtual machine.
- Some of the services provided by the DDMS are port forwarding, on-device screen capture, on-device thread and heap monitoring, and radio state information.
- In order to trigger the DDMS tool in the machine users need to install three main components on the machine :
 - a) Java SDK
 - b) Android SDK
 - c) Device Specific Drivers
- The main services provided by Dalvik Debug Monitor Server are :
 1. App memory usage statistics (total heap and object allocation statistics)
 2. App thread statistics
 3. Device screen capture
 4. Device file explorer
 5. Incoming call and SMS spoofing
 6. Location data spoofing
 7. Logcat

Dalvik Debug Monitor Service

- When you start the emulator, the Dalvik Debug Monitor Service displays information about the applications invoked and shows a details log file of log messages. You can select an application to view the threads executing and memory allocated for that application.

7.7.1 Logcat

- The Android logging system provides a mechanism for collecting and viewing system debug output.
- Logcat is a command line tool which dumps a log of system messages. It contains stack traces when the device throws an error. This is the type of log which is most usually asked for an issue. It can be collected either by using adb or some app of your choice. A lot of ROMs nowadays provide options to collect it in Developer Settings as well.
- Logs from various applications and portions of the system are collected in a series of circular buffers, which then can be viewed and filtered by the logcat command. You can use logcat from an ADB shell to view the log messages.

Syntax

```
[adb] logcat [<option>] ... [<filter-spec>] ...
```

7.7.2 Android Debug Bridge (ADB)

- Android Debug Bridge (ADB) is a command-line tool that allows you to communicate with a device.
- ADB is used to bridge communication between an emulator instance (Android device) and the background running daemon process (server).
- ADB helps perform different actions like installing or debugging a device and run various commands on a device by providing access to a Unix shell.

Accessing the emulator with ADB

- ADB offers many of the features of DDMS, including the means to :
 - a) View log output
 - b) Access the file system on the emulator or device
 - c) Access SQLite databases on the emulator or device
 - d) Issue shell commands on the emulator or device

Connect Real Devices

- Running Android App for testing in real device is faster than running in AVD.
- Also this can be one of the ways you can use to install your new Android App apk file in your real device.

Execute Application on Real Device

1. Enable Debugging over USB

- Most Android devices can only install and run apps downloaded from Google Play, by default. You will need to enable USB Debugging on your device in order to install your app during development.
- To enable USB debugging on your device, you will first need to enable the "Developer options" menu by going to Settings → About phone → Software information and then tapping the Build number row at the bottom seven times. You can then go back to Settings → Developer options to enable "USB debugging".

2. Plug in your device via USB

- Let's now set up an Android device to run our React Native projects. Go ahead and plug in your device via USB to your development machine.
- Now check that your device is properly connecting to ADB, the Android Debug Bridge, by running adb devices.

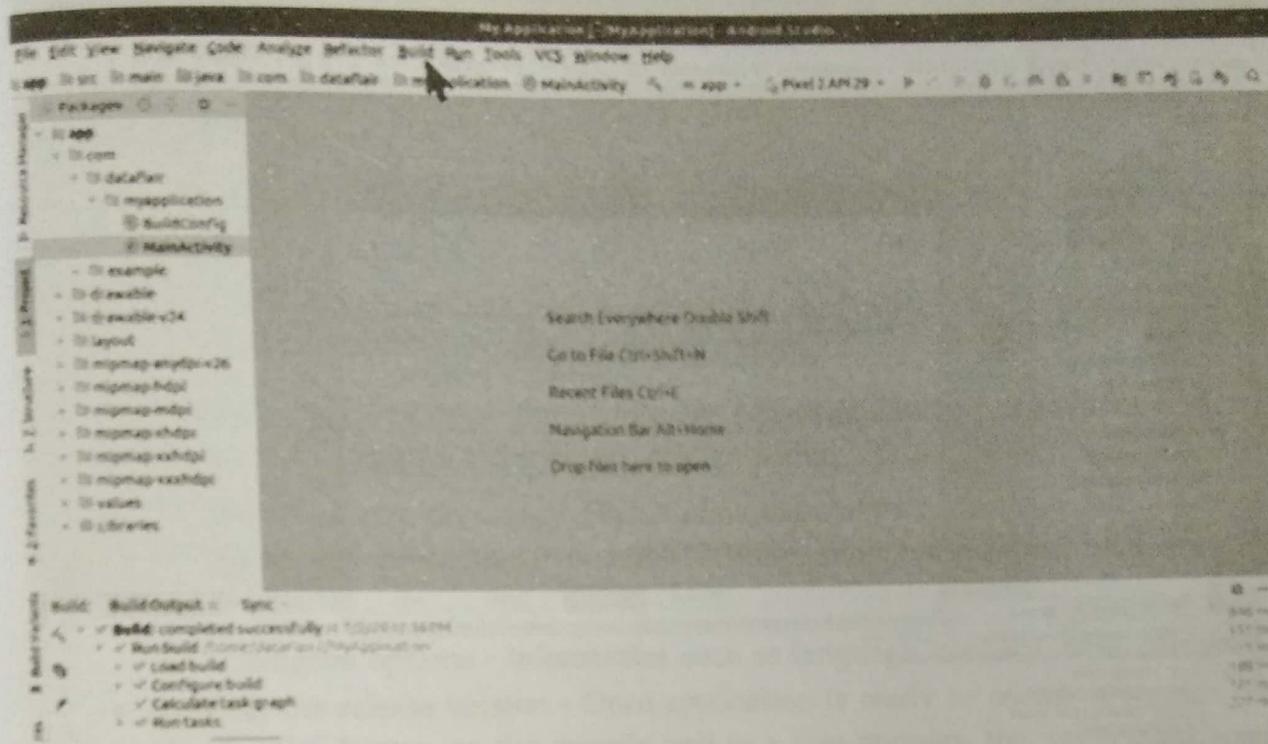
3. Run App

- Type the command prompt to install and launch your app on the device.

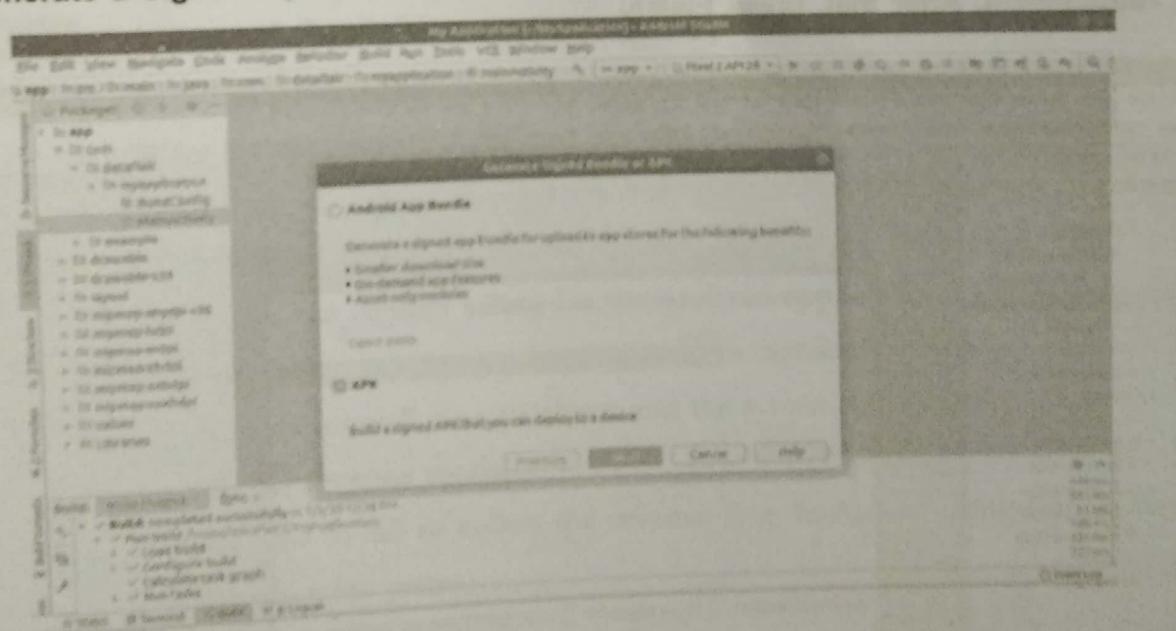
7.8 Signing the Android Application

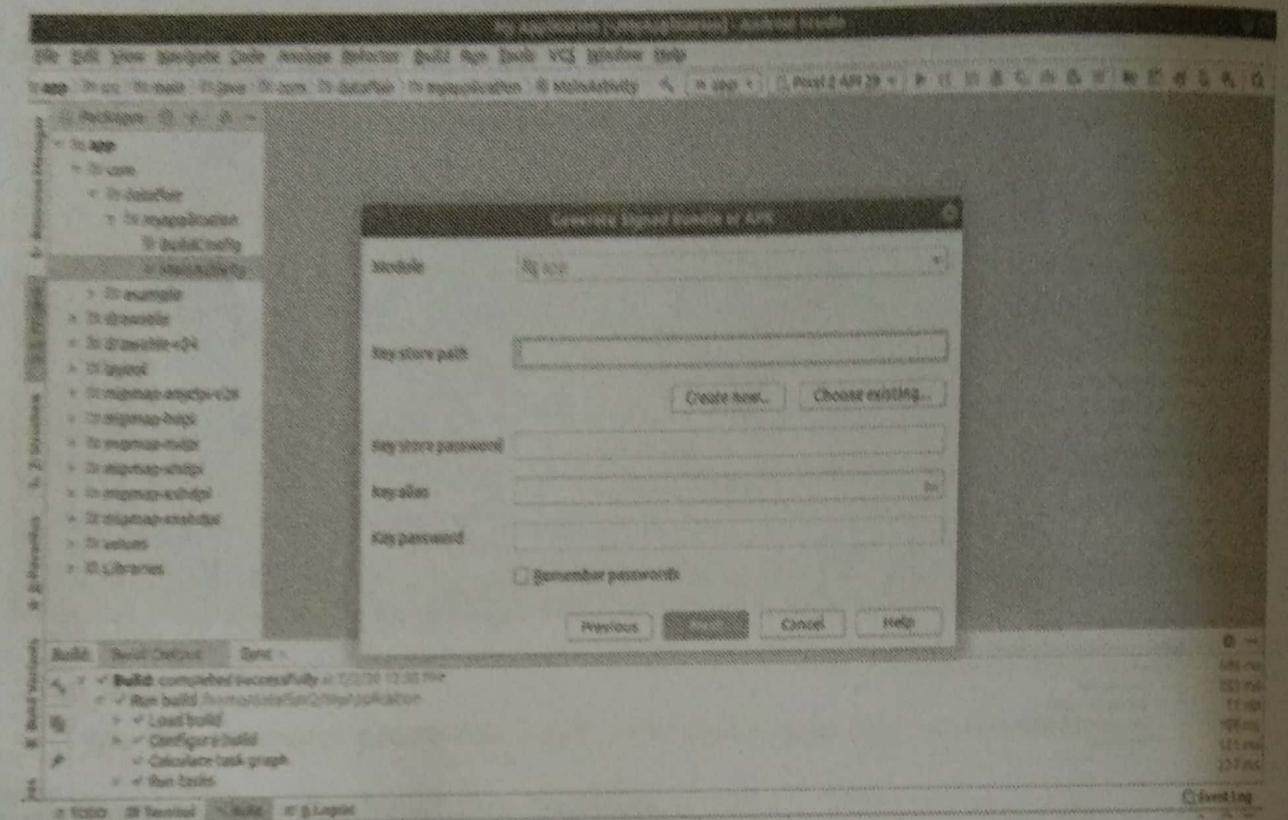
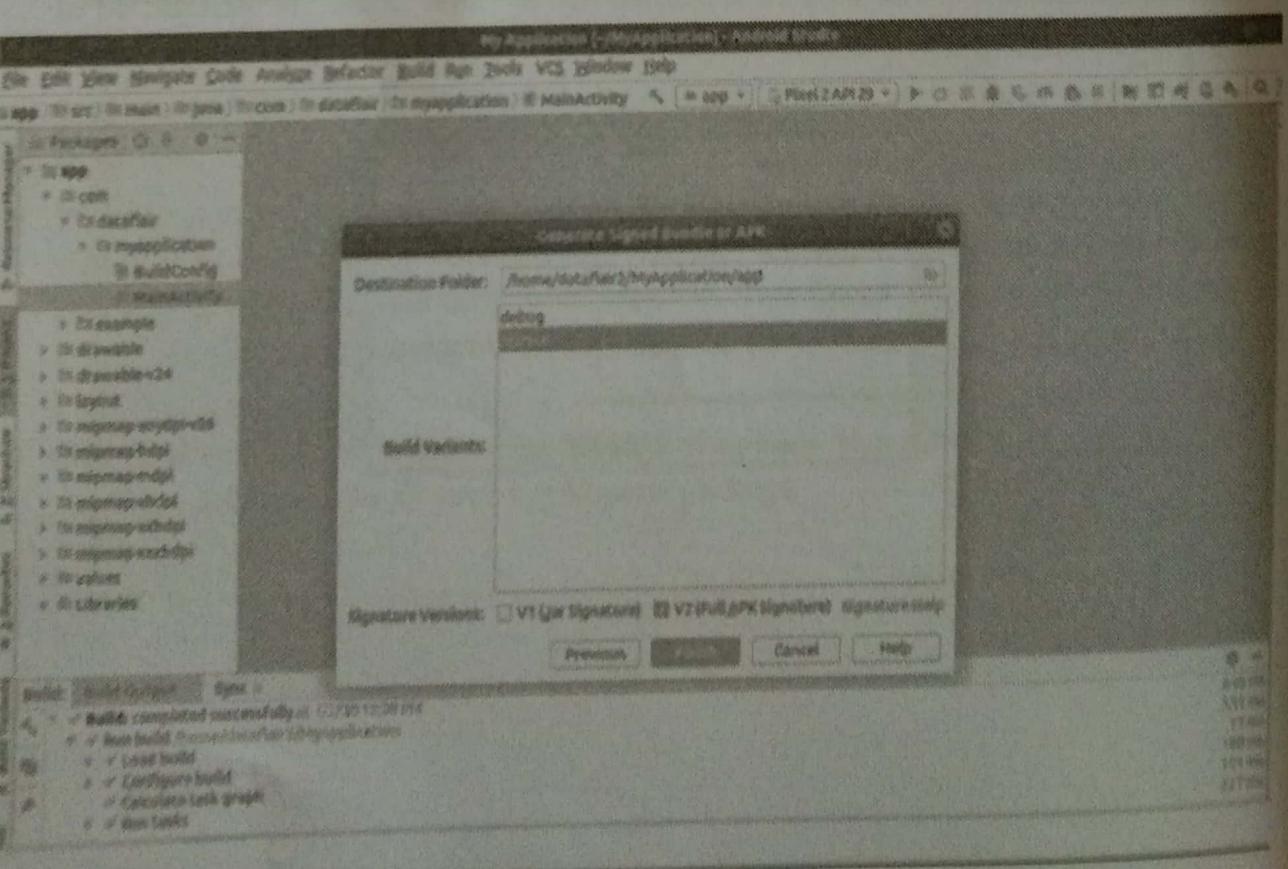
- The Android platform requires every application file - that is, yourappname.apk - to be digitally signed in order to run on a device or emulator; without a signature, an application simply won't run.
- The signing requirement is entirely transparent to most developers until it's time to publish an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

1. Open the application in Android Studio



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



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

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

7.9 Publishing Android App

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

7.9.1 Process of Publishing an Android Application

- The steps of the publishing process can be summarized as follows.
 - Select an appropriate AppStore
 - Read and understand the policies and agreements of the selected AppStore
 - Quality test
 - Determine the content rating for the Android application
 - Determine the country (or the countries) to distribute
 - Confirm the overall size, platform and the screen compatibility ranges
 - Decide the revenue model
 - Decide how to bill or collect the revenue (e.g. In-App or using Google Pay)
 - Set the price (or prices)
 - Localization
 - Prepare promotional graphics, videos and screencasts
 - Build and upload
 - Plan for Beta release

14. Complete AppStore listing
15. Support users after launch

7.10 Distribution of Android App

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

7.11 App Characteristics

7.11.1 Performance of App

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

Improving Performance of App

- Understand your target device. Most developers forget about device, they classify device based on operating system, but forget about configuration of devices. Always classify devices based on specification sheet. Try developing apps based on low specification mobile which will automatically run in all mobiles.
- Understand your tools. Try to understand the tools that are used to develop the mobile app. Better understanding of tools helps to make important architectural design. Example - Understanding about phone gap helps to develop app for multiple platforms.

- Understand core concepts of the language used to build app. Understanding core programming language used to develop app will help to avoid performance issues. Example - A extra string comparison in jscript will surely reduce the performance.
- Understand the library. Try to understand the library used in tools. If you simply call third party methods for simple operation then it will increase battery usage and reduce the performance. Always use standard codes for simple task.

7.11.2 Modifiability of App

- Modifiability helps to release multiple version of app more easily.
- Modifiability is achieved by developing as multiple units instead of single unit.
- If any bugs arise after launch then it is easy to modify the unit instead of changing everything in code.
- Modifiability is minimizing the technical risks and cost impact of changes in software.
- In order to achieve modifiability as a system quality, software architects need to envision and incorporate modifiability support in the system's design cycle.
- The architectural design supports the modifiability requirements of a system.
- The modifiability quality of a system can be expressed in terms of cohesion and coupling. Coupling measures the mutual association strength between the system's software components. Where cohesion, on the other hand, is a measure for the number of internal relationships between the responsibilities of a software component.

7.11.3 Availability of App

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

7.11.4 Security of App

- Mobile apps should satisfy stringent requirements for data security and privacy. This is no longer the exclusive domain of the enterprise : Organizations of every

size and function are subject to mounting ethical and legal pressure to control and protect the information under their purview.

- Fiduciary responsibility and internal and external policies exist to govern what organizations must do in this regard, from data storage to disaster recovery, encryption to secure updating.
- By definition, internet access and mobile devices carry inherent security risks, including but not limited to the apps that run on them.

7.12 Short Questions and Answers

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

Ans. :

- Android alpha (1.0)
- Cupcake (1.5)
- Eclair (2.0 - 2.1)
- Gingerbread (2.3 - 2.3.7)
- Ice Cream Sandwich (4.0 - 4.0.4)
- KitKat (4.4 - 4.4 w)
- Android beta (1.1)
- Doughnut (1.6)
- Froyo (2.2 - 2.2.3)
- Honeycomb (3.0 - 3.2.6)
- Jelly Bean (4.1 - 4.3.1)

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

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

Q.3 What is DDMS?

Ans. : DDMS -

- DDMS stands for Dalvik Debug Monitor Server. It is a debugging tool from the Android software development kit (SDK).
- DDMS able to monitor operations in the emulator as well as real devices.
- DDMS reports the details of each processing thread and time spent whether in the app or Android OS.

7.13 Multiple Choice Questions with Answers

Q.1 Android applications must be signed ____ .

- a after they are installed
- c never

- b before they are installed
- d within two weeks of installation