⊛ Mobile Application Development ⊛

⊛ chapters:

(1) Basic knowledge of OOPs concepts & core Java.
(2) Fundamental
(3) Android OS
(4) Android UI and Component using Fragments
(5) Database Connectivity
(6) Applicability to Industrial Projects
(7) Advanced Android Development
(8) Work with Android System & Development & Deployment.

⊛ Android OS ⊛

Q.① Introduction to Android:

- Android is an operating system & programming platform developed by Google.
- It supports devices like mobile phones & other devices, such as tablets, watch, TV, & auto.
- Android includes a software Development kit (SDK) that helps you write code & assemble software modules to create apps for Android users.
- Android also provides a marketplace (playstore) to distribute apps.

- It conquered around 75% of the global market share by the end of 2020.

- The company named Open Handset Alliance developed Android for the first time that is based on the modified version of the Linux kernel and other open-source software.

- Google sponsered the project at initial stages and in the year 2005, it acquired the whole company.

- In September 2008, the first Android-powered device launch in the market.

Q.② Android System with Architecture:

- Android Architecture contains different number of components to support any android device needs.

- Android software contains an open-source Linux kernel having collection of number of c/c++ libraries which are exposed through an application Framework services.

- Among all the components Linux kernel provides main Functionality of Operating System Functions to Smartphones and

Pages No.
Date

Dalvik Virtual Machine (DVM) provide platform for running an android application.

- The main components of android architecture are following:

1. System Apps
2. Java API Framework
3. Native c/c++ Libraries.
4. Android Runtime
5. Hardware Abstraction Layer (HAL)
6. Linux kernel.

(*) Diagram:

(*) Android Architecture (*)

S.A - Android comes with a set of core applications such as Contacts, Calender, Maps, and a browser.

J.A.F - you use these APIs to control what your app looks like and how it behaves.

N.C.L - You can access OpenGL ES through the native c/c++ libraries to add support for drawing and manipulating 2D and 3D graphics in your app.

A.R - The Android runtime comes with a set of core libraries that implement most of the Java programming language. Each Android app runs in its own process.

H.A.L - HAL provides an interface that expose device hardware capabilities to the higher-level Java API Framework.

L.K - Underneath everything else lies the Linux kernel. Android relies on the kernel for drivers, and also core services such as security and memory mangement.

Page No.
Date

[1] System Apps:

- System apps are pre-installed apps in the system partition with your ROM.
- System app is simply an app placed under '/system/app' folder on an Android device.
- '/system/app' is a read-only folder.
- Android device users do not have access to this partition. Hence, users cannot directly install or uninstall apps to/from it.
- App such as camera, settings, messages, Google play store, etc.

[2] Java API Framework:

- The entire feature-set of the Android OS is available using API's written in the Java language.
- These APIs form the building blocks you need to create Android apps by simplifying the reuse of core, modular system components and services.
- A rich and extensible view system you can use to build an app's UI, including lists, grids, text boxes, buttons, and even an embeddable web browser.

- A Resource Manager, providing access to non-code resources such as localized strings, graphics and layout files.

- A Notification Manager that enables all apps to display custom alerts in the status bar.

- An Activity Manager that manages the lifecycle of apps & provides a common navigation back stack.

- Content Providers that enable apps to access data from other apps, such as the contacts app, or to share their own data.

---

[3] Native c/c++ libraries:

- The Native Development kit (NDK) is a set of tools allows you to use C and C++ code with Android.

- It provides platform libraries you can use to manage activities and access physical device components, such as sensors and touch input.

- The NDK may not be appropiate for most novice Android programmers who need to use only Java code &

Page No.
Date

Framework APIs to develop thier apps.

- However, the NDK can be useful for cases in which you need to do

  - Squeeze extra performance out of device when run games physics simulations.

  - Reuse your own or other developer's C or C++ libraries.

- using Android studio 2.2 and higher, you can use the NDK to compile C & C++ code into a native library & package it into your APK using Gradle.

[4] Android Runtime (ART):

- Android Runtime (ART) is the managed runtime used by applications & some system devices.

- ART and its predecessor Dalvik were originally created specifically for the Android project.

- Google is claiming up to 200% performance improvements overall for ART.

- ART improves garbage collection.

[5] Hardware Abstraction Layer (HAL):

- HAL is responsible for accessing the driver execution hardware.
- A HAL defines a standard interface for hardware vendors to implement, which enables Android to be agnostic about lower-level driver implementations.
- It allows you to implement functionality without affecting or modifying the higher level system.

---

[6] Linux kernel:

- A kernel is the core part of any operating system.
- Android OS is built on top of the Linux kernel, with some architectural changes made by google.
- Most importantly, Linux is a portable platform that can be compiled easily on different Hardware.
- The kernel act as an abstraction layer between the software and Hardware present on the device.

* * *

Page No.
Date

Q.③ Android PlatForms:

- The Android platform is a platform for mobile devices that uses a modified Linux kernel.
- The Android platform was introduced by the Open Handset Alliance in November of 2007.
- Most applications that run on the Android platform are written in Java programming language.
- To create an application for the platform, a developer requires the Android SDK which includes tools and APIs.
- To shorten development time, Android developers typically integrate the SDK.

Q.④ Building Blocks: → in detail ppt

- The building blocks are components that you use as an developer to build Android apps.
- Main Building blocks of Android are:

- Activities
- Intents
- Services
- Content providers
- Broadcast Receivers
- Application Context.

Page No.
Date

Q.5 Activities :

- An application typically has multiple activities, and user flips back and forth among them.

- As such, activities are the most visible part of your application.

- Just like a website consists of multiple pages, so does an Android application consist of multiple activities.

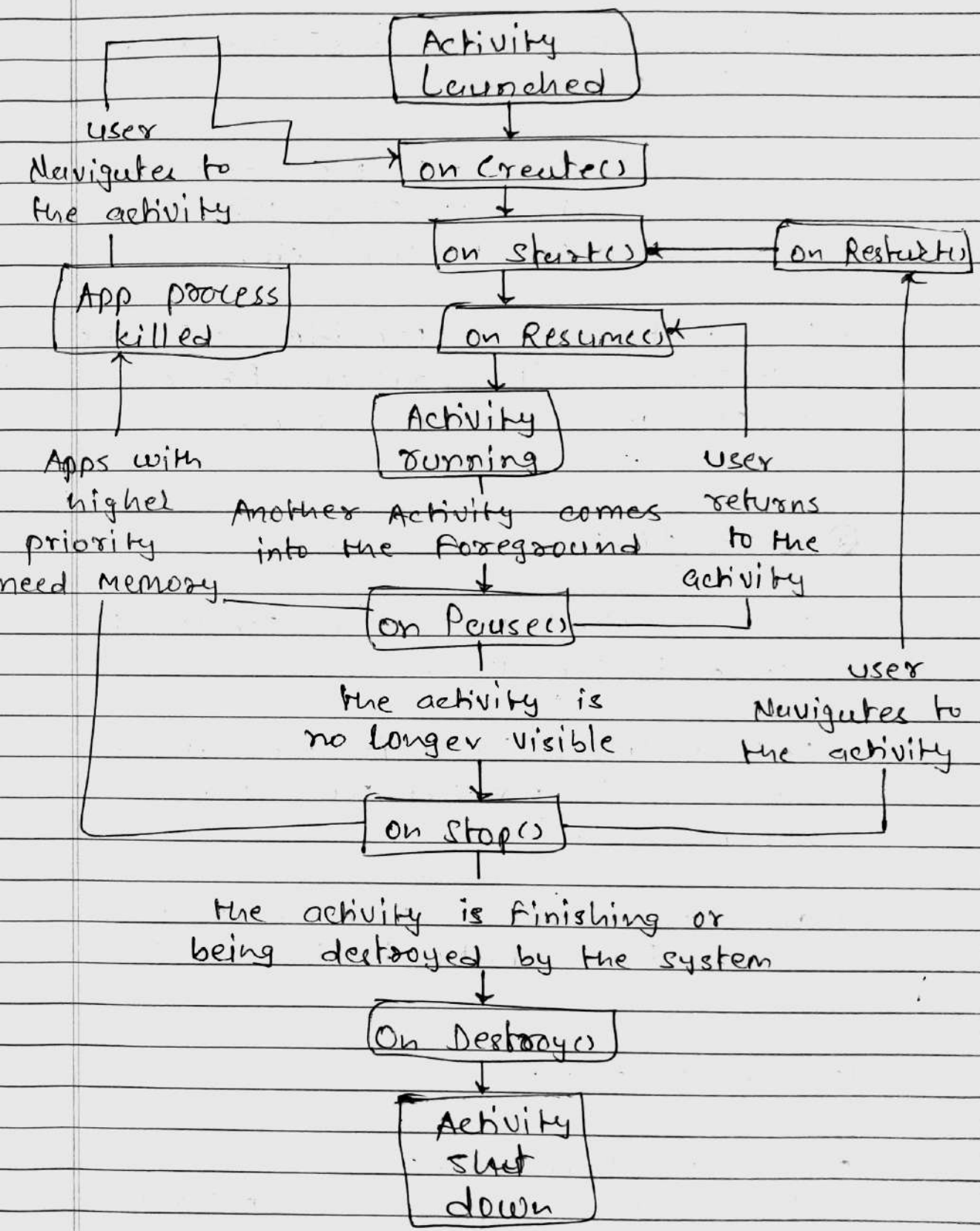
- As a website has a "home page", an Android app has a "main" activity, usually the one that is shown first when you launch the application.

- And just like a website has to provide some sort of navigation among various pages, an Android app should do the same.

- We can jump from an activity of one application to another activity in a completely seperate application.

- For ex, if you are in your contacts apps & you choose to text a friend you'd be launching the activity to compose a text message in the messaging application.

⊛ Activity Life cycle:



⊛ Activity Life Cycle ⊛

① onCreate():

- it is called when the activity is first created.
- this is where all the static work is done like creating views, binding data to lists, etc.
- this method also provides a Bundle containing its previous frozen state, if there was one.

② onStart():

- It is invoked when the activity is visible to the user.
- It is followed by onResume() if the activity is invoked from the background.
- It is also invoked after onCreate() when the activity is first started.

③ onRestart():

- It is invoked after the activity has been stopped and prior to its starting stage.

- Thus, is always followed by onStart() when any activity is ~~set~~ revived from background to on-screen.

---

⑥ onResume():

- It is invoked when the activity starts interacting with the user.
- At this point, the activity is at the top of the activity stack, with a user interacting with it.
- Always followed by onPause() when the activity goes into the background or is closed by the user.

---

⑦ onPause():

- It is invoked when an activity is going into the background but has not yet been killed.
- It is a counterpart to onResume().
- When an activity is launched in front of another activity, this callback will be invoked on the top activity.
- The activity, under the active activity will not be created until the active activity's onPause()

returns, so it is recommended that heavy processing should not be done in this part.

⑧ onStop():

- It is invoked when the activity is not visible to the user.
- It is followed by onRestart() when the activity is revoked from the background, followed by onDestroy().

⑧ onDestroy():

- onDestroy() is called before the activity is destroyed.
- The activity is finishing due to the user completely dismissing the activity or due to finish() being called on the activity.
- The system is temporarily destroying the activity due to a configuration change eg. device rotation or multi-window mode

Page No.
Date

Q.(6) Intents :

- An Intents is to perform an action on the screen.
- It is mostly used to start activity, send broadcast receiver, start services and send message between two activities.
- It is a messaging object which tells what kind of action to be performed.
- the intent's most significant use is the launching of the activity.

(*) Body of Intent :

- action : the general action to be performed such as ACTION_VIEW, ACTION_EDIT etc
- data : the data to operate on, such as a person record in the contacts databases, expressed as a Uri.

(*) Basically two intents are there in android :

- Implicit Intents
- Explicit Intents

---

[1] Implicit Intents :

- Implicit Intent doesn't specify the component.

- In such cases, intent provides information on available components provided by the system that is to be invoked.

- Implicit intents are used without a class name, where Android will help determine an appropriate Activity to handle the intent.

- For example, you may write the following code to view the page.

```
Intent intent = new Intent (Intent.ACTION_VIEW)
intent.setData (Uri.parse ("https://www.own.com"))
startActivity (intent);
```

(2) Explicit Intent:

- Explicit Intent specifies the Component.

- In such a case, intent provides the external class to be invoked.

- Explicit Intent are used with class name, where Android navigate via routes.

```
Intent i = new Intent (getApplicationContext(),
                       ActivityTwo.class);
startActivity(i);
```

Page No.
Date

Q. (7) Fragments:

- A Fragment represents a reusable portion of app's UI.

- A Fragment defines and manages its own layout, has its own lifecycle, and can handle its own input events.

- Fragments cannot live on thier own they must be hosted by an activity or another Fragment.

- Android Fragment is the part of activity, it is also known as sub-activity.

- there can be more than one Fragment in an activity.

- Using with Fragment transaction. we can move one Fragment to another Fragment.

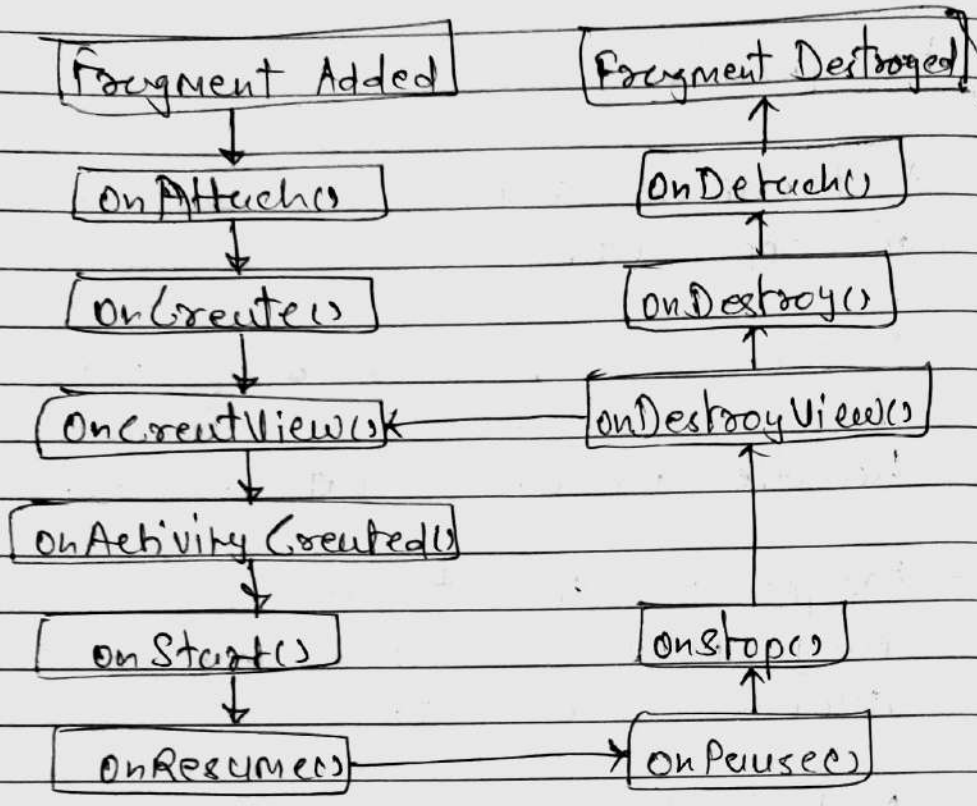
(*) Fragment Life cycle:

(*) onAttach():

- the Fragment instance is associated with an activity instances.

- 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.



① 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.

② 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.

④ 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.

⑤ onStart():

- the onStart() method called once the fragment gets visible.

⑥ onResume():

- Fragment becomes active.

⑦ 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.

④ onStop():

- Fragment going to be stopped by calling onStop().

⑤ onDestroyViews():

- ~~as~~ Fragment view will destroy after call this method.

⑥ onDestroy():

- onDestroy() called to do final clean up of the fragment's state but not guaranteed to be called by the Android platform.

* * *

Page No.
Date

⊛ Chapter: (4)

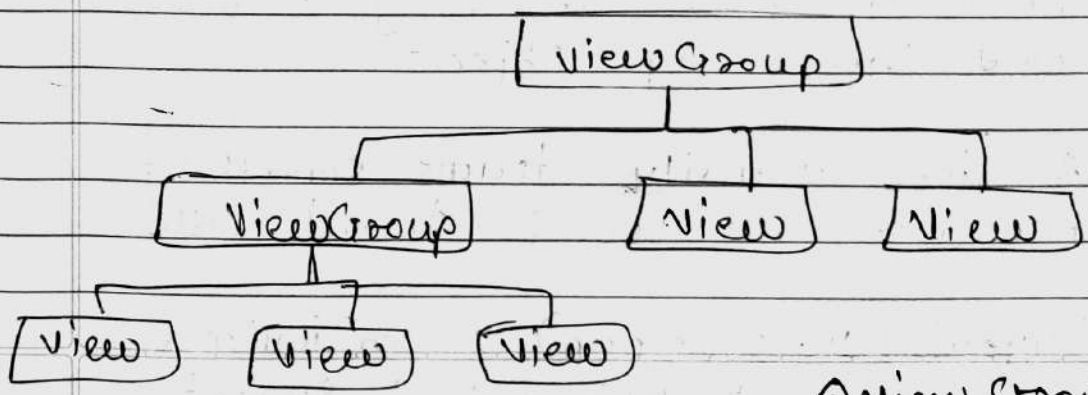
⊛ Android UI and Component, UI Fragments ⊛

Q.① What is Android UI?

- Android provides a variety of pre-built components.
- Layout objects and UI controls that allow to build the graphical user interface for you app.
- Android also provides other UI modules for special interfaces such as dialogs, notifications, and menus.

Q.② What is View Group?

- A ViewGroup is a special view that can contain other views (called children.)
- The view group in the base class for layouts and views containers.
- It's child can be Views or ViewGroup.



⊛ View Group ⊛

⑧ Views:

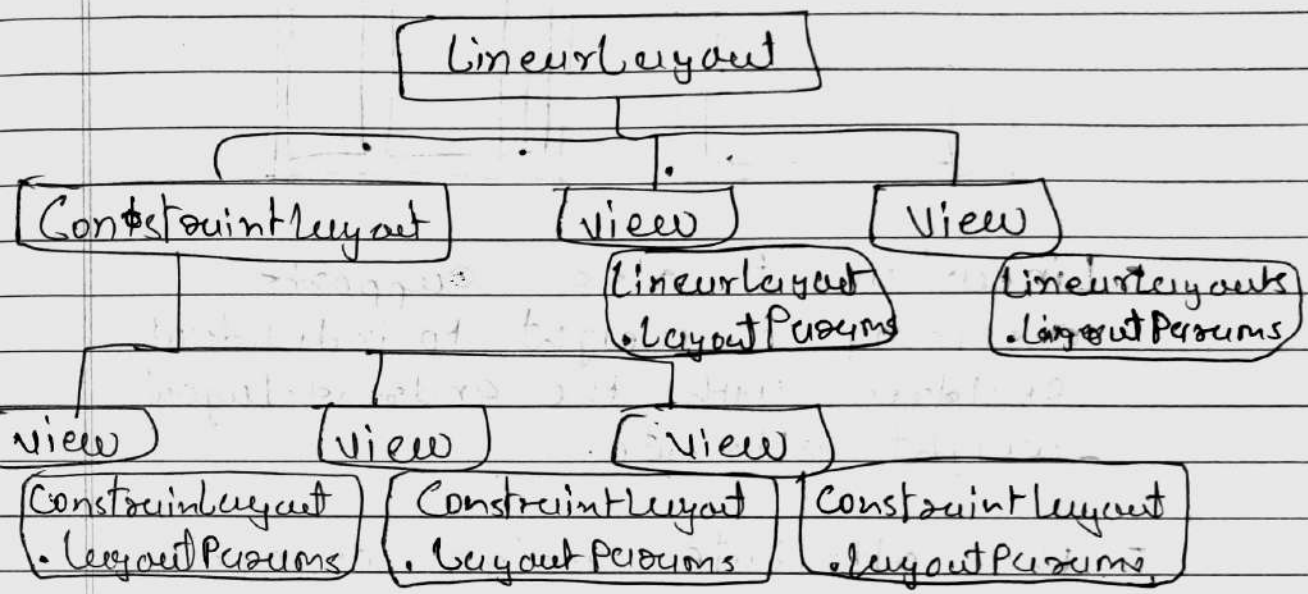
- View class represents the basic building block for user interface components.
- A View occupies a rectangular area on the screen & is responsible for drawing & event handling.
- All of the views in a window are arranged in a single tree.
- View is the base class for widgets, which are used to create interactive UI components (buttons, text fields, etc.)

Q.③ Layouts:

- A layout defines the structure for a user interface in your activity.
- All elements in the layout are built using a hierarchy of View and ViewGroup objects.
- A View usually draws something the user can see & interact with.
- Layout (ViewGroup) is an invisible container that defines that layout

structure for View and other layout (View Group) objects.

- It provide a different layout structure such as LinearLayout or ConstraintLayout.



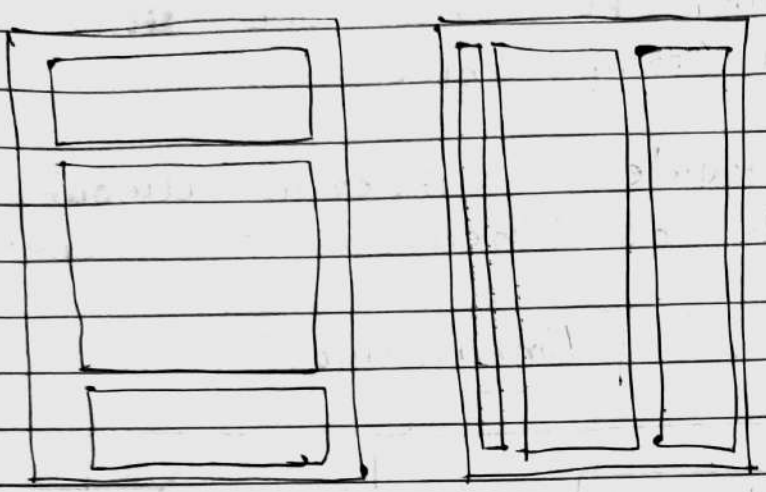
① Layouts ②

[1] Linear Layout:

- LinearLayout is a ViewGroup that aligns all children in a single direction, vertically or horizontally.
- You can specify the layout direction with the android:orientation attribute.
- LinearLayout can be created in two direction:
  - ↳ Horizontal & Vertical.

Vertical Linear Layout

Horizontal Linear Layout.



- LinearLayout also supports assigning a weight to individual children with the android:layout_weight attribute.

- This attribute assigns an important value to a view in terms of how much space it should occupy on the screen.

- A larger weight value allows it to expand to fill any remaining space in the parent view.

[2] Relative Layout:

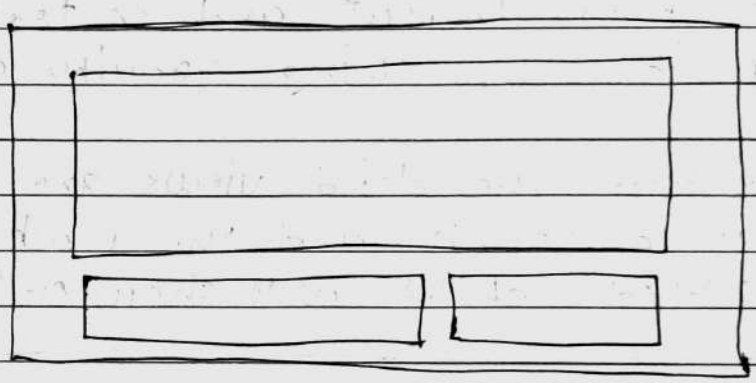
- Relative Layout is a View Group that displays child views in relative positions.

- The position of each views can be specified as relative to sibling elements or in positions relative to the

parent RelativeLayout area.

- As it allows us to position the component anywhere, it is considered as most flexible layout.

- Relative layout is the most used layout after the Linear Layout in Android.



- Relative layout is the most used layout after the Linear Layout in Android.

- In Relative layout, you can use "above, below, left and right" to arrange the component's position in relation to other component.

- In this view group child views can be layered on top of each other.

[3] Frame layout :

- Frame layout is designed to block out an area on the screen to display a single item.

- It is used to specify the position of Views.

- It contains Views on the top of each other to display only single View inside the FrameLayout.

- You can add multiple Children to a FrameLayout and control thier position by using gravity attribute.

- In this, the child views are added in a stack and the most recently added child will show on the top.

Q. (4) Text Views:

- Text view is a UI Component that displays the text to the user on thier Display screen.

Table of Attributes → Darshan PPT.

---

(4) Attributes:

- android:id - This is the ID which uniquely identifies the control.

- android:fontfamily - Font family for the text

- android:text - Text to display in View.

- android:gravity - Specifies how to align the text when the text is smaller than the view.

... etc.

Q. ⓚ EditText :

- A EditText is an overlay over TextView that configures itself to be editable.

- It is the predefined subclass of TextView that includes rich editing capabilities.

ⓚ Attributes :

- android:autoText - textview has a textual input method & automatically corrects some common spelling errors.

- android:drawableBottom : This is the drawable below the text.

- android:editable - If set, specifies that this TextView has an input method.

- android:background : This is a drawable to use as the background.

Q.⑥ Button:

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

- To specify an action when the button is pressed, set a click listener on the button object in the corresponding activity code.

- Every button is ~~stylish~~ styled using the system's default button background it can customize.

⊛ Attributes:

- android:text - this is used to display text on button.

- android:background - this is a drawable to use as the background.

- android:id - this specifies supplies an identifier name for this view.

- android:onClick - this is the name of the method in this view's context to invoke when the view is clicked.

Page No.
Date

20. (7) CardView :

- A CardView is child of FrameLayout with a rounded-corner background along with a specific elevation.

- The main usage of cardview is that helps to give a rich feel and look to the UI design.

- These cards have a default elevation above thier containing view group, so the system draws shadows below them.

- Information inside cards that have a consistent look ~~goes~~ across the platform.

- It can be used for creating items in ListView or inside Recycler View.

21. (8) (*) Attributes :

- Card-View : Card Background Color -
  ↳ can set background color.

- Card-View : card Corner Radius -
  ↳ To set Radius or card corner

- Card-View : cardElevation -
  ↳ To set elevation to the card.

- card-View : card Use Compat Padding - radius &
  ↳ set compatible padding to card view based on elevation.

Q.⑧ ListView:

- ListView is a view which groups several items & display them in vertical scrollable list.

- The list items are button automatically inserted to the list using an Adapter that pulls content from a source such as an array or database.

- An Adapter actually acts as a bridge between UI components and the data source that fill data into UI component.

② Attributes:

- android: divider - This is drawable or color to draw between list items.

- android: dividerHeight - This specifies height of the divider. This could be in px, dp, sp, in, or mm.

- android: entries - Specifies the reference to an array resource that will populate the ListView.

Q.⑨ Adapter:

- An Adapter object acts as a bridge between an AdapterView and the underlying data for

- In Android development, any time we want to show a vertical list of scrollable items we will use a ListView which has data populated using an Adapter.

- The simplest adapter to use is called an ArrayAdapter because the adapter converts an ArrayList of objects into view items loaded into the ListView container.

- The ArrayAdapter fits between an ArrayList (data source) and the ListView (visual representation).

- When your ListView is connected to an adapter, the adapter will instantiate views until the ListView has been fully populated with enough items to fill the full height of the screen. At that point, no additional new items are created in memory.

```
ArrayAdapter<String> itemsAdapter = new
    ArrayAdapter<String>(this, android.R.layout.
        simple_list_item_1, items);
```

10. Recycler View:

- Recycler View is a View Group added as successor of the GridView & ListView.

- Recycler View is mostly used to design the user interface with the fine-grain control

that View.

over the lists & grids of android application.

- To implement a basic RecyclerView three sub-parts are needed.

(1) The Card Layout:

- it is an XML layout which will be treated as an item for the list created by the RecyclerView.

(2) The ViewHolder:

- it is a java class that stores the references to the card layout views that modified during the execution of list.

(3) The Data class:

- it is a Custome java class (Getter-setter) that acts as a structure for holding the information for every item of the RecyclerView.

- The adapter is main part for RecyclerView for displaying the lists.

- onCreateViewHolder:

↳ which deals with the inflation of the card layout as an item.

- onBindViewHolder:
  ↳ which deals with the setting of different data and methods related to clicks on particular items.

Q.(11) Material Design Toolbar:

- The toolbar gave us so much space for customization & creation for ActionBar.
- MaterialToolbar is a Toolbar that implements certain Material features, such as elevation overlays for Dark Themes and centered title.
- Toolbars appear a step above the sheet of Material affected by their actions.
- Unlike Actionbar, its position is not hardcoded it can be place anywhere according to the needed just like any other View in android.
- Use as an ActionBar:
- In an app, the toolbar can be used as an ActionBar in order to provide more customization & a better appearance. All the features of ActionBar such as Menu inflation, ActionBarDrawerToggle, etc. are also supported in Toolbar.

Q.(12) Tab Layout:

- Tab layout is used to implement horizontal tabs.
- population of the tabs to display is done through Tab layout instances.
- Create tabs via new tab() from there we can change the tab's label or icon via TabLayout.Tab.setText(int).
- A TabLayout can be setup with ViewPager
  - ↳ Dynamically create TabItems based on the number of pages, their titles, etc.
  - ↳ Synchronize the selected tab and tab indicator position with page swipes.
- There are two types of tabs:
  - Fixed tabs.
  - Scrollable tabs.
  - ↳ Fixed tabs display all tabs on one screen, with each tab of a fixed width.
  - ↳ Scrollable tabs are displayed without fixed widths. They are scrollable, such that some tabs will remain off-screen until scrolled.

Page No.
Date

2.(13) Menus:

- To provide a familiar and consistent user experience menus are the best.
- It define in seperate XML file and use that file in our application based on our requirements.
- We can use menu APIs to represent user actions and other options in our android application activities.
- The Menus in android applications are:

(1) Options menu
(2) Context menu
(3) Popup menu

[1] Options Menu:

- The options menu is the primary collection of menu items for an activity.
- Options Menu Generally placed on action bar.
- We can declare items for the options menu from either Activity or a Fragment.
- Re-order the menu items with a android:orderInCategory attribute from XML file menu item.

- To Specify the options menu for an Activity, override onCreateOptionsMenu().

- In this method, you can inflate your menu resource (defined in XML) into the menu.

- you can also add menu items using add() retrive items with findItem().

- When the user clicks a menu item from the options menu onOptionsItem Selected() method is used to get callback.

[2] Contextual Menu:

- you can provide a context menu for any view. but they are most often used for items in a RecyclerView, ListView, GridView items.

- there are two ways to provide Contextual ~~Menus~~: actions.

(1) In a floating context menu appears as a floating list of menu items when the user performs a long-click on a view that declares support for a context menu.

(2) In the contextual action mode is a system implementation of ActionMode

that displays a contextual action bar at the top of the screen with action items ex. select all items, Delete Items, etc.

- It can be associate with view using registerForContextMenu() method and pass it the view.

- Menu item inflated to content menu using onCreateContextMenu() method in your Activity or Fragment.

- Menu items click event is handled by onContextItemSelected() in Activity or Fragment.

(3] Popup Menu:

- A Popup Menu displays a Menu in popup window anchored to a view.

- The popup will be shown below the anchored view if there is room (space) otherwise above the view.

- If any keyboard is visible the popup will not overlap until the view is touched.

- Touching outside the popup window will dismiss it.

* —— * —— *

Chapter: ②: Fundamentals

Q.① What is Software Engineering and Explain SDLC.

- Software Engineering is the process of designing, developing, testing and maintaining software.

- It is a systematic and disciplined approach to software development that aims to create high-quality, reliable and maintainable software.

- Software engineering includes a variety of techniques, tools and methodologies, including requirements analysis, design, testing and maintenance.

② key principles of Software Engineering:

- Modularity
- Abstraction
- Encapsulation
- Reusability
- Maintainence
- Testing
- Design
- Agile Methodologies
- Continuous Integration & Development.

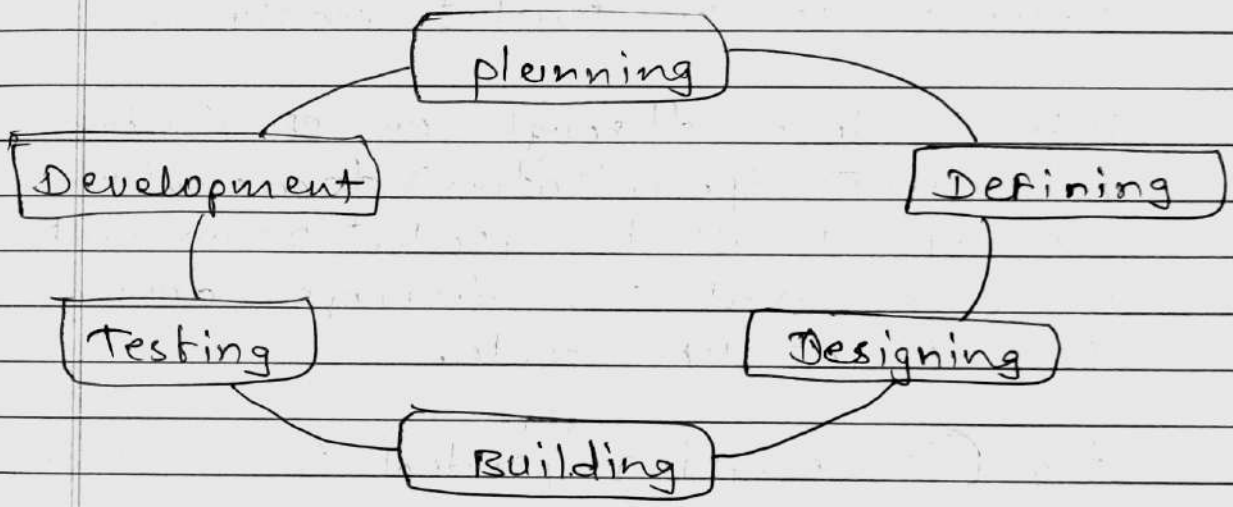
③ Software Development Life cycle ④

- Software Development life cycle is a process of used by software industry to

design, develop and test high qualities.

- The SDLC aims to produce a high-quality software that meets a customer expectation, reaches completion within times and cost estimates.

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



Step:① : Planning & Requirement Analysis.

- Requirement analysis is the most important and fundamental stage in SDLC.

- It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry.

- This information is then used to plan the basic project approach and to conduct product feasibility in the economical operational & technical areas.

Stage: ②: Defining Requirements.

- once the requirement analysis is done the next step is to clearly define the document the product requirements and get them approved from the customer or the market analysts.

- This is done through an SRS (software Requirement Specification) document which consists of all the product requirements to be designed and developed during the project life cycle.

Step: ③: Designing the product Architecture.

- SRS is the Reference for product architectures to come out with the best architecture for the product to be developed.

- Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

Page No.
Date

Stage: (4): Building or Developing the product.

- In this stage of SDLC the actual development starts and the product is built.

- The programming code is generated as per DDS during this stage.

- If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Stage: (5): Testing the Product.

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

Stage: (6): Deployment in the market & once. Maintain

- Once the product is tested & ready to be deployed it is released formally in the appropriate market.

- Sometimes product deveto deployment happens in stages as per the business strategy of that organization.

- the product may first be released in a limited segment & tested in the real business environment (UAT- user acceptance testing).

(*) Following are the most important & popular SDLC models followed in the industry -

- waterfall model
- iterative model
- spiral model
- V-model
- Big Bang Model.

Q.(2) Flow Charts:

- Flow chart is a graphical Representation of an algorithm. programmers often use it as a program-planning to solve a problem.

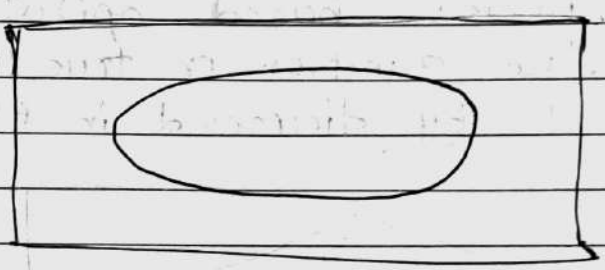
- It makes use of symbols which are connected among them to indicate the flow of information & processing.

- The process of drawing a flowchart for an algorithm is known as "Flowcharting".

(*) Basic Symbols used in Flowchart Designs:

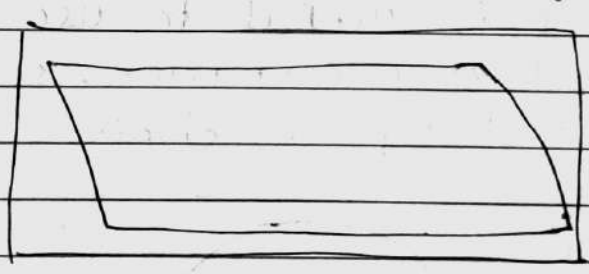
[1] Terminal:

- The oval symbol indicates start, stop and ~~Halt~~ Halt in a program's logic flow. A pause/halt is generally used in program login under some error conditions.
- Terminal is the first and last symbols in the flowcharts



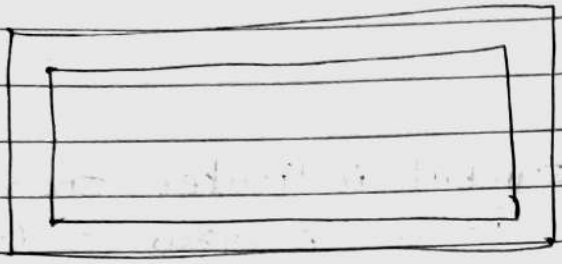
[2] Input/output:

- A parallelogram denotes any functions of input/output type. program instructions that take input from input devices and display output on output devices are indicated with parallelogram in a flowchart.



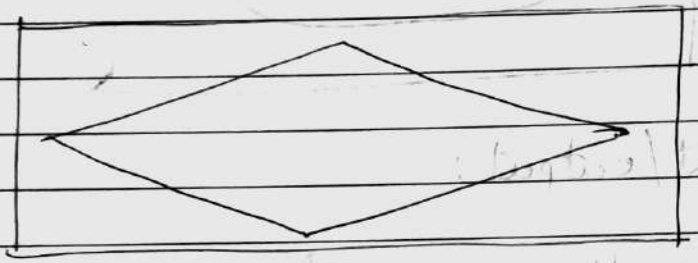
[3] Processing:

- A box represents arithmetic instructions. All arithmetic processes such as adding, subtracting, multiplication & division are indicated by action or ~~press~~ process symbol.



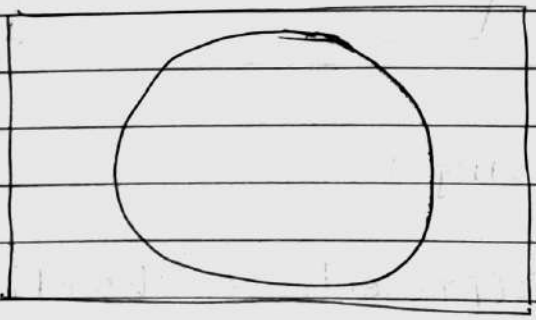
[4] Decision:

- Diamond symbol represents a decision point. Decision based operations such as yes/no question or true/False are indicated by diamond in flowchart.



[5] Connectors:

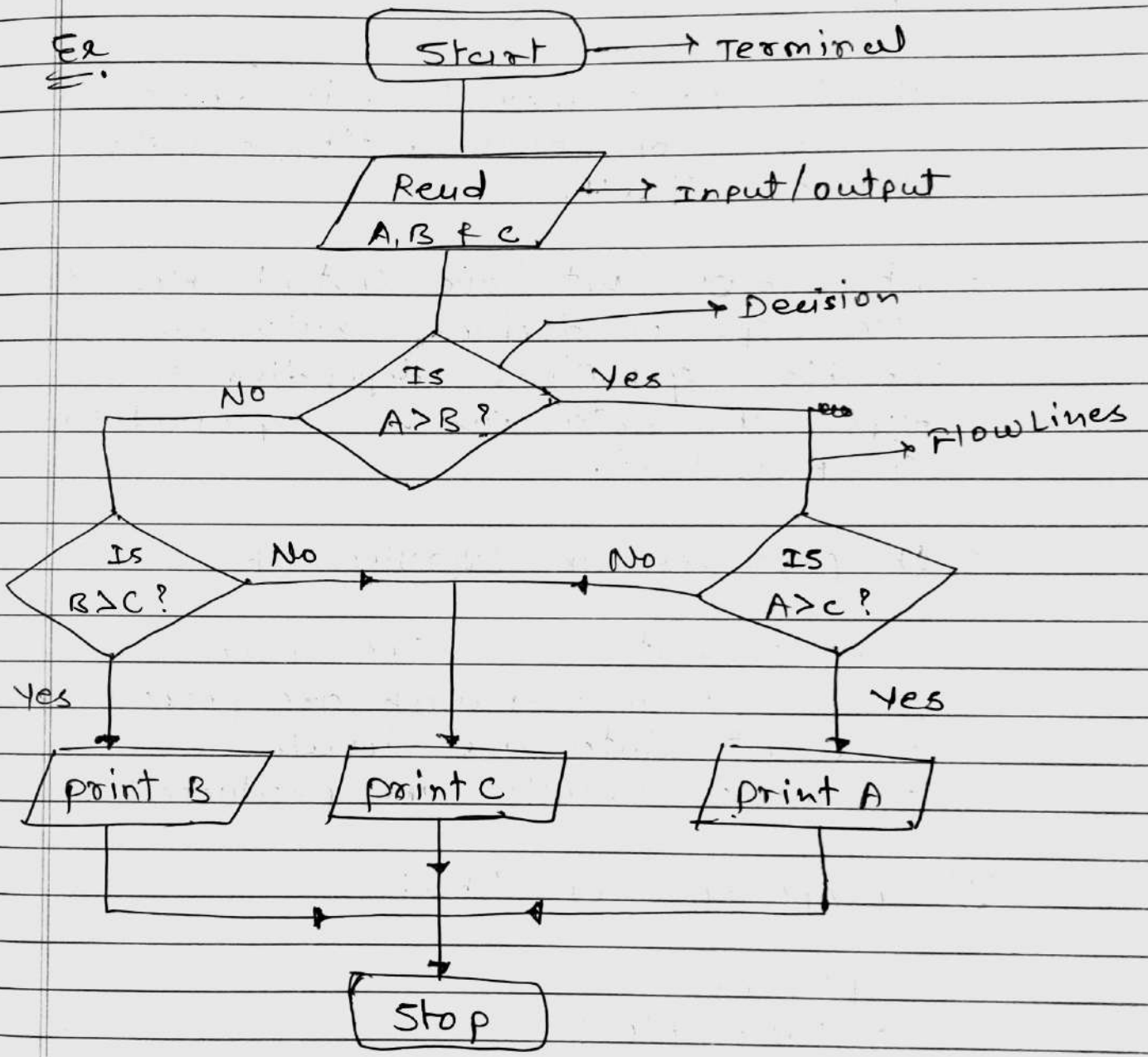
- Whenever flowchart becomes complex or it spreads over more than one page, it is useful to use connectors to avoid any confusions. It is represented by a circle.



[6] Flow Lines:

- Flowlines indicate the exact sequence in which are executed. Arrows represent the direction of flow of control & relation -ship among different symbols of flowchart.

Ex:



Ⓐ Flowchart of Reading A, B & C Ⓐ.

Q.(3) Data Flow Diagrams:

- DFD is the abbreviation for Data Flow Diagram. The flow of data of a system or a process is represented by DFD.

- It also gives insights into the inputs & outputs of each entity and the process itself.

- DFD does not have control flow & no loops or decisions rules are present.
- Specific operations depending on the type of data can be explained by a flowchart.

- It provides an overview of

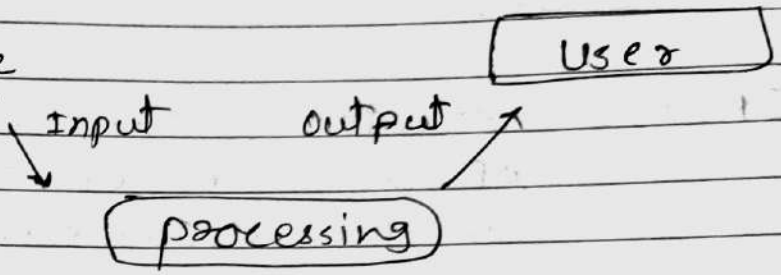
  - what data is system processes.
  - what transformation are performed.
  - what data are stored.
  - what results are produced, etc.

(4) Components of DFD:

- process
- Data Flow
- Warehouse
- Terminator

Page No.
Date

Data Warehouse



⊛ Basic structure of DFD ⊛

Q.(4) SQL Database:

- SQL (Structured Query Language) is a programming language which is used to manage data stored in relational databases like mySQL, MS Access, SQL server, etc.

⊛ SQL commands:

- SQL Commands can be classified into the following groups based on their nature:

  - Data Definition Language (DDL)
  - Data Manipulation Language (DML)
  - Data Control Language (DCL).

(1) Data Definition Language:

- A Data Definition Language (DDL) is a Computer Language which is used to create and modify the structure of databases objects which includes tables, views, schemas, and indexes, etc.

Page No.
Date

① Commands:

CREATE - Creates a new table, a view of a table, or other object in the database.

ALTER - Modifies an existing database object, such as a table.

DROP - Deletes an entire table, a view of a table or other objects in the database.

TRUNCATE - Truncates the entire table in a go.

[2] Data Manipulation Language:

- A Data Manipulation Language (DML) is a computer programming language which is used for adding, deleting and modifying data in a database.

② Commands:

SELECT - Retrives certain records from one or more tables.

INSERT - Creates a Record.

UPDATE - Modifies records

DELETE - Deletes records.

[3] Data control language (DCL):

- Data control language (DCL) is a computer programming language which is used to control access to data stored in a database.

* commands:

GRANT - Gives a privileges to user

REVOKE - takes back privileges granted from user.

* SQL Applications:

- Execute different databases queries against a database.
- Define the data in a databases and manipulate the data.
- Create data in a relational database management system.
- Access data from the relational dbms.
- Create and drop databases and tables.
- Create and ~~manipulate~~ maintain database users.
- Create view, stored procedure, functions in a databases.
- set permissions on tables, procedures and views.

⊛ Chapter: ⑤: Database connectivity ⊛

Q. ① Storage in android and its types.

- Storage on Android where you keep data like photo & music.
- Android provides four types of storage systems:

(1) Internal storage.
(2) External storage (SD card)
(3) Shared preferences
(4) Database

---

[1] Internal storage :-

- 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 internal data files saved in /data/data/<your app package name> like below.

Ex. My app data internal file is saved in data/data/com.dev2qu.example Folder.

- There are files and cache sub folder under the package name folder.

(1) Files Folder :-

- android.content.Context's getFilesDir() method can return this folder.

- This folder is used to save general files.

(2) cache Folder :-

- android.content.Context's getCacheDir() method can return this Folder.

- This folder is used to save cached files.

- When device internal storage space is low, cache files will be removed by android OS automatically to make internal storage space bigger.

- Generally you need to delete the unused cache files in code timely, totally cache files size is better not more than 1 MB.

(2) External storage :-

- Android data can be saved. external storage can be used to write & 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".

Ex: when we say Nexus 5 32 GB.

(2) Secondary External storage :-

- Removable store.

Ex: SD card.

- All Applications can read & write files placed on the External storage and the user can remove them.

- 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.

(1) Environment.getExternalStorageState():

- returns the path to internal SD mount point like "/mnt/sdcard".

(2) getExternalFilesDir():

- it returns the path to files inside Android/data/data/application_package/ on the SD card.

(3) Shared Preference Layout :-

- Shared preferences allow you to read & write small amounts of primitive data (as key/value points) to a file on the device storage.

- The SharedPreference class provides APIs for getting a handle to a preference file and for reading, writing & 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 persists 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 sqlite Databases.

⊗ Creating Shared preferences :-

- in order to use shared preferences, you have to call a method getSharedPreferences().

Ex. SharedPreferences sharedpreferences = getSharedPreferences("settings", Context.MODE_PRIVATE);

- The string settings is the name of the preference file you wish to access if it does not exist, it will be created.

- if you want to use common preference

file and don't want/wish to specify a file name, we can use default s.preferences.

Eg. SharedPreferences sharedpreferences = PreferenceManager.getDefautSharedPreferences (getApplicationContext());

- using this way will default the preference file to be stored as
  - /data/data/com.package.name/shared-Prefs/com.package.name-preferences.xml
- we can write the data inside shared preferences file using SharedPreferences.Editor.
- once editing has been done, one must commit() or apply() the changes made to the file.

---

(4) Databases :-

- Databases are collections of data that are organized and saved for future use.
- Using a Database Management System, we can store any type of data in our databases.
- All we have to do is establish the

database and use one query to perform all of the operations, such as insertion, deletion and searching.

- The query will be passed to the database, which will return the desired output.

- In Android, an SQLite Database is an example of database.

Q.(2) SQLite Database & ~~establishing connection with sqlite database in Android~~.

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

- The ~~lit~~ lite in SQLite means light weight in terms of setup, database administration & required resource.

- A SQLite database is good storage solution when you have structed 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 phones, Android phones, game consoles, etc.

- SQLite is developed using ANSI-C. The source code is available as a big sqlite3.0 and its header file sqlite3.h.

- SQLite databases are very light weight. Unlike other databases systems, there is no configuration, installation required to start working on SQLite database.

- The Android SQLite database requires very little memory (around 250 kb), which ~~merged or android right~~ is available on all around devices.

- 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.

④ the basic advantages of SQLite :

(1) its a light weight database.
(2) Requires very little memory.
(3) An automatically managed database.

⑤ The SQLite supports only three Data types :-

(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.

Q.③ 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.

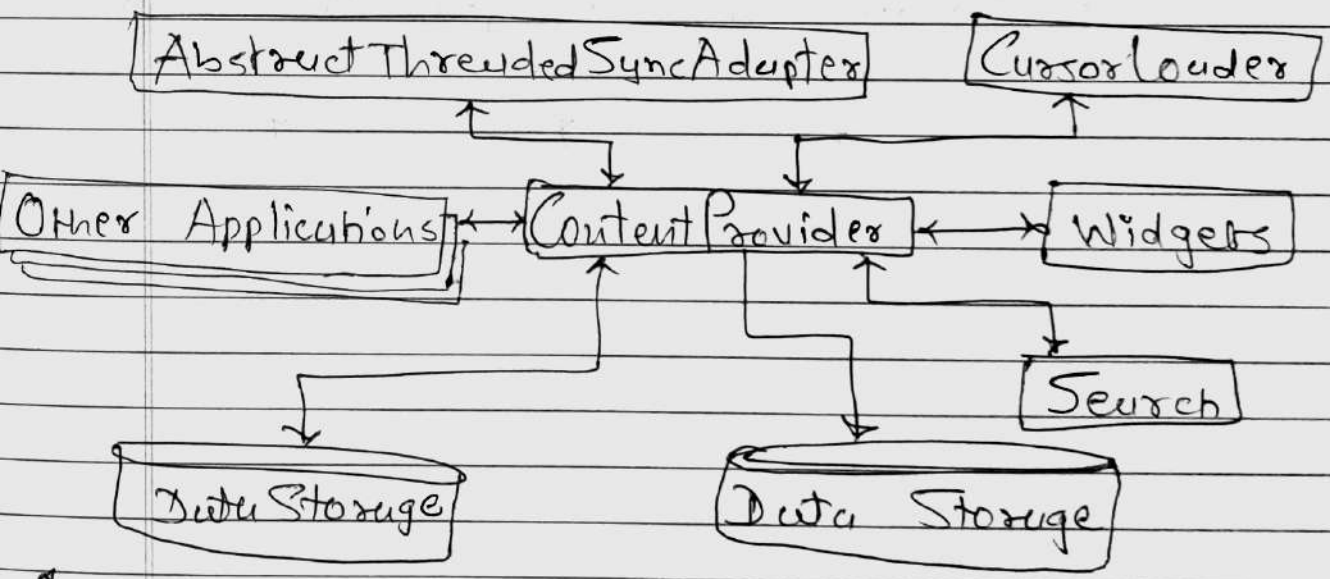
- Together, providers and providers clients offer a consistent, standard interface to

data that also handles inter-process communication and secure data access.

- A content provider presents data to external applications as one or more table that are similar to the tables found in a relational database.

- A row represents an instance of some type of data the provider collects, & 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

(v) The content provider and other components include:

(1) Sharing access to your application data with other application.

(2) Sending data to a widget.

(3) Returning custome search suggestions for your Application through the ~~sewet~~ search framework using SearchRecentSuggestionsProvider.

(4) Synchronizing application data and with your server using an implementation of AbstractThreadedSyncAdapter.

(5) Loading data in your UI using a CursorLoader.

⊛ Chapter: ⑥: Applicability to Industrial Projects ⊛

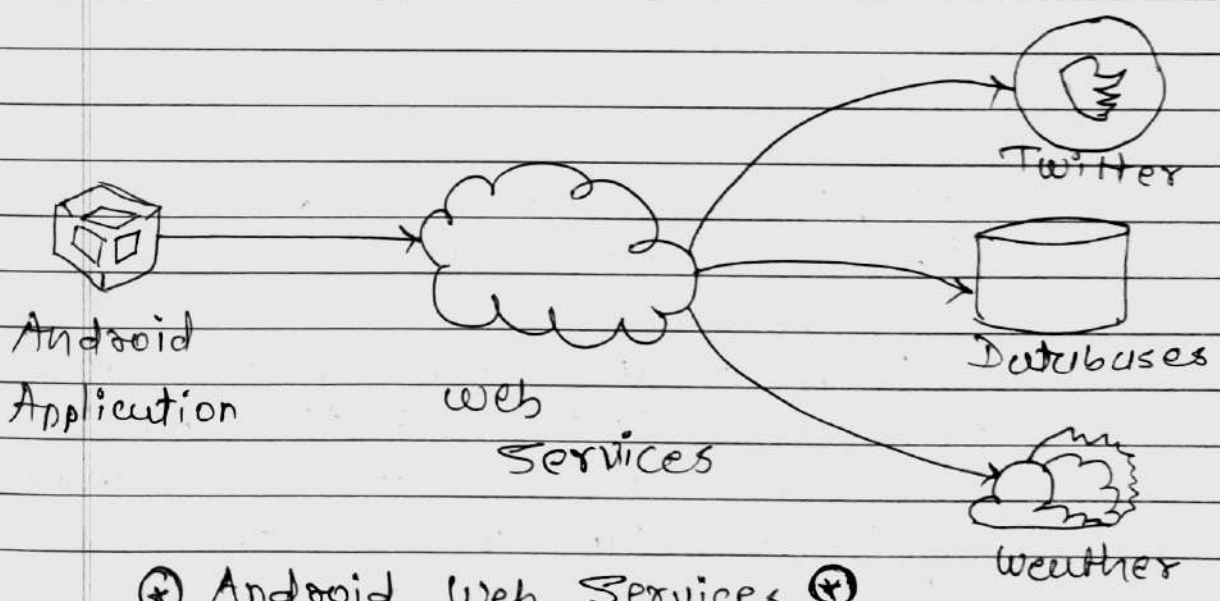
Q.① Web service, its types, Requirements & its application.

- A web service is a type of software that can help streamline every aspect of a mobile application & integrate it for other web services.

- Web services needed to provide interoperability, means connecting various applications.

- Web services allows different apps to communicate with each other & share the data and service among themselves.

Ex. an Android application can interact with java or .net application using web services.



⊛ Android Web Services ⊛

⊛ Types of web Services :-

(1) Rest
(2) SOAP
(3) UDDI
(4) XML-RPC (5) WSDL

Popular types of web Services

[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 language could talk quickly to each other and avoid the extreme development effort.

[2] WSDL :-

- WSDL is known as the Web Service Description language (WSDL).

- WSDL is an XML-based file which tells the client Application what the web service does and gives all information required to connect to the web service.

[3] REST :-

- REST stands for RE presentational state transfer.

- REST is used to build web service that are light weight, maintainable and scalable.

⊛ Android web service 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
  (2) JSON (JavaScript Object Notation)

⊛ Requirements of Web Services :-

- In order d 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 OS.

3. it is a self-declaring & 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 availibility over private or internet networks as well.

Q.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 service 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.

⊛ Advantage 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 ~~in~~ name+ 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 numberic value.

- A JSON can be of two structures:

1. A collection of name value pairs (object).
2. A list of values (Array).

⊛ JSON, Definition & 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 key are strings and the values are the JSON types.

- keys and values are separated by commu. the { (curly braces) 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.

ex. 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 offers by many languages.

(*) Serializing & Deserializing in JSON :-

(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.

(3) Characteristics of JSON :-

(1) Easy to use
(2) Performance
(3) Free tool
(4) Doesn't require to create mapping
(5) Clean JSON
(6) Dependency

(4) Applications of JSON :-

- JSON useful in transfering data from a server.

- Sample JSON file format used in transmit and serialize all types of structured data.

- Allows you to perform asynchronous data calls without the need to do a page refresh.

- JSON is a tool to transmit data between a server and web applications.

- JSON is widely used for Javascript-based application, which includes browser extension and websites.

- Data can be transmitted between the server and web application using JSON.

- Web services and Restful APIs use the JSON format to get public data.

Q. (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 use of overloaded service methods.

(a) Passing parameters as part of URL Path:-

- Specify parameters is name-value pairs after the service URL.

- parameter names must be formed ~~form~~ 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 thier 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 String 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 (eg. Long, String) happens directly.

Q.(4) What is AsyncTask? Explain with Example.

- AsyncTask was intended to enable proper and easy use of the UI thread, However, the most common use was for integrating into UI and that would cause context leaks, missed callbacks or crashes on configuration changes.

- 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. 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.

  - params: Parameters sent to the task upon execution

  - Progress: Progress units shared during the background computation

  - Result: Result obtained from the background computation.

Ex - The following codesnippet must be present in the MainActivity class in order to launch an AsyncTask.

```
MyTask myTask = new MyTask();
myTask.execute();
```

- execute Method is used to start the background thread.

Q.⑤ Third Party Library : 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.~~

- Retrofit used for turns your HTTP API request response into Java interface. Without server communication we cant develop any high level android application.

(2) Retrofit supports below built-in annotations:

(1) GET :-

- A method that's annotated with @GET is responsible for processing a HTTP GET request, where data is retrieved from a server.

(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 :-

- it is similar to GET, except that a @HEAD method retrieves information without the corresponding response body.

Page No.
Date

⊛ Advantages of Retrofit :-

- it is simple to use.

- it essentially lets you treat API calls are as simple as Java method calls, so you only define which URLs to hit and the types of the request/response parameters as Java classes.

Q.5 ~~What is parsing? How we perform parsing using JSON in an Android Application.~~

Q.6 Steps of AsyncTask or methods of AsyncTask.

- there are several ways to do background processing in Android. Two of those ways are:

  - you can do background processing directly, using the `AsyncTask` class.

  - you can do background processing indirectly, using the `Loader` framework and then the `AsyncTaskLoader` class.

- in most situations the Loader framework is better choice, but it's important to know how AsyncTask works.

- Ans starts from here]

- When AsyncTask is executed, it goes through four steps :

(1) onPreExecute() :-

- onPreExecute() is invoked on the UI thread before the task is executed.

- This step is normally used to set up the task, for instance by showing a process bar in the UI.

(2) doInBackground(Params...) :-

- doInBackground (Params...) is invoked on the Background thread immediately after onPreExecute() finishes.

- This step performs a background computation, returns a result, and passes the result to onPostExecute().

- The doInBackground() method can also call publishProgress(Progress...) to publish one or more units of progress.

(3) onProgressUpdate (Progress...) :-

- onProgressUpdate (Progress...) runs on the UI thread after publishProgress (Progress...)

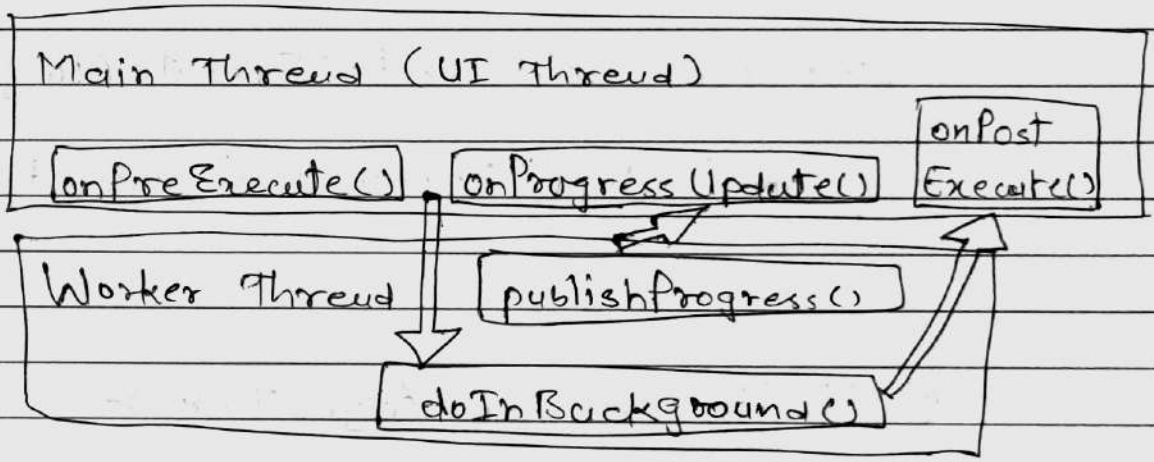
is invoked.

- Use onProgressUpdate() to report any form of progress to the UI thread while the Background computation is executing.

(4) onPostExecute(Result) :-

- onPostExecute(Result) runs on the UI thread after the background computation has finished.

- The result of the background computation is passed to this method as a parameter.



Ex.

```
private class DownloadFileTask extends
        AsyncTask<URL, Integer, Long>{

    protected Long doInBackground(URL...urls){

        int count = urls.length;
        long totalSize = 0;
```

```
for (int i=0; i< count; i++){
    totalSize += Downloader.downloadFile
                    (urls[i]);
    publishProgress((int)((i)/(float) count)*100));
    if (isCancelled()) break;
}
    return totalSize;
}

protected void onProgressUpdate(Integer... progress
    setProgressPercent(progress[0]);
}

protected void onPostExecute(Long result){
    showDialog("Downloaded" + result + " bytes");
}
}
```

⊛ Limitations of Async Task :-

- changes to device configuration cause problems.
- Old AsyncTask objects stay around, and your app may run out of memory or crash.

⊛ When to use AsyncTask :-

- short or for interruptible tasks.

- Tasks that don't need to report back to UI or user.

- Low-priority tasks that can be left unfinished.

Q.⑦ AsyncTask Loader :-

- AsyncTask Loader is the loader equivalent of AsyncTask.

- AsyncTask Loader provides a method, loadInBackground(), that runs on a separate thread.

- the results of loadInBackground() are automatically delivered to the UI thread, by way of the onLoadFinished() LoaderManager callback.

- To define subclass of AsyncTaskLoader :

public static class StringListLoader extends AsyncTaskLoader < List < String >> { }.

Q.⑧ What is parsing ? Discuss how we can perform parsing using JSON in Android Application.

- JSON stands for JavaScript Object Notation.

- JSON is used for data interchange from the server.

- Hence knowing the syntax and its usability is important. JSON is the best alternative for XML and its more readable by humans.

- JSON is language independent. Because of language in-dependency we can program JSON in any language.

- A JSON response from the server consists of many fields. An example of JSON response is given below

```
{ "title" : "JSONParser",
  "array" : [
      { "company" : "Google"
      },
      { "company" : "Facebook"
      },
      { "company" : "Microsoft"
      } ],
  "nested" : {
  "flag" : true,
  "random_number" : 7
  } }
```

- A JSON Data Consits of 4 components:

(1) Array: A JSON Array is enclosed in square brackets [].

(2) object : Data enclosed in curly brackets {}. is a single JSONObject.

(3) keys : Every JSONObject has a key String that's contains certain value.

(4) Value : Every key has a single value that can be of any type : String, double, integer, boolen, etc.

Page No.
Date

⊛ Chapter: ⑦ : Advanced Android Development ⊛

9.① Google Map & APIs in Google Maps.

- 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.
- The Google Maps Android API consist 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.

⊛ APIs in Google Maps :-

⊛ Web APIs :

- Google Maps JavaScript API
- Google Street View Image API

⊛ Web Service APIs :-

- Google Maps Directions API
- Google Maps Elevation API

Page No.
Date

⊛ Mobile APIs :-

- Google Maps Android API
- Google Places API for Android

⊛ Steps for Getting the Google Maps API Key :

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

(1) - open Google developer console and sign in with your gmail account.

(2) - Now create new project. You can Create new project by clicking on the Create Project button and give name to your project.

(3) - Now click on APIs and services and open Dashboard from it.

(4) - In this open Enable APIs AND SERICES.

(5) - Now open Google Map Android API.

(6) - Now enable the Google Maps Android API.

(7) - Now go to Credentials.

(8) - Here click on create credentials and choose API key.

(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 your Android project.

⊛ Elements of Google Map :-

(1) GoogleMap
(2) MapView
(3) SupportMapFragment
(4) Shapes
(5) MyLocation layer
(6) Marker
(7) Ui settings

} Detailed in Technical book pg.no 6-3/4.

⊛ 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 SupportMapFragement 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.SupportMapFragement" />

⊛ 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 views.

(2) Satellite :-

- it displays 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.

⊛ UI - controls of Google Maps :

(1) Zoom
(2) Pan
(3) Map type
(4) Street view

Page No.
Date

⑥ Features of Google Maps :-

- it searches places and route directions.
- it provides distance information.
- it helps to find traffic details and navigation.
- it displays street views.
- it receives voice instructions.
- it provides location sharing and location editing.

---

6. ⑦ 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 expired above.

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

- GooglePlayServicesClient.ConnectionCallbacks
- GooglePlayServicesClient.OnConnectionFailed Listener

Q.⑤ Location Listener :-

- The LocationListener interface, which is part of the Android Locations API is used for receiving notifications from the LocationManager when the location has changed.

- The LocationManager class provides access to the systems location services.

- The LocationListener class needs to implement the following methods.

(1) onLocationChanged(Location location):

- called when the location has changed.

(2) onProviderDisabled(String provider):

- called when the provider is disabled by the ~~provid~~ user.

(3) onProviderEnabled(String provider):

- called when the provider is enabled by the user.

(4) onStatusChanged (String provider, int status, Bundle extras):

- Called when the provider status changes.

- The location android.location has two means of aquiring location data:

(1) LocationManager.GPS_PROVIDER
(2) LocationManager.NETWORK_PROVIDER

Q.(4) Geocoding and Reverse Geocoding with Ex.

(*) 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 discribed as the process of converting a texual based geographical location (such as a street address) into geographical coordinates expressed in terms of longitude & lubitude.

- Geocoding can be done 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 or in the list.

- A variety of information can then be extracted from the Address objects, including the Logitude & Lutitude of the potential matches.

Example: "1600 Amphitheater Parkway, Mountain View, CA".

- can be converted into lutitude & logitude

(-77.03655, 38.89768).

(4) Use of Geolocations :-

- Geolocation can be used to determine time zone and exact positioning coordinates, such as for tracking wildlife or cargo shipments.

- 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

⑧ Reverse Geocoding :-

- 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 closet 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 thier mapping applications reverse geocoding features.

- The end-user will be able to search a latitude and logitude location & obtain the exact street address or largest or nearest city.

Ex. "(-77.0365, 38.897 68)"
- can be converted into "1600 Amphitheater Parkway, Mountain View, CA".

④ Forward and Reverse Geocoding :-

- Forward geocoding is the process of looking up a plain-text address or place name (eg Grimar Mountain), whereas reverse geocoding is performed by passing latitude and longitude values of desired location to the API.

Q.⑤ Explain Graphics & its use.

- Graphics are created by the developer and added to the application.

- They can be created using 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~~.

- Android supports 2D Graphics via its own library in packages android.graphics.drawable and android.view.animation.

⑤ Use of Graphics :-

- The following are some common uses of Graphics:

(1) Display text on top of a map or
(2) Highlighting a section of the map
(3) Display a route between two locations
(4) Animate data items that change quickly, such as moving objects.

8. (6) Explain Drawable Object.

- 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.

- Drawable are used to define shapes, colors, borders, gradients, etc. which can be then be applied to views within an Activity.

- Every Drawable is stored as individual files in one of the 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) Using the normal class constructors.

Q.⑦ Explain 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 for needs.

= with a Shape Drawable, you can programmatically draw primitive shapes and style them in any way imaginable.

- Shape Drawable has its own draw() method, you can create a subclass of View that draws the shape Drawable during the View.onDraw() method.

- A shape Drawable takes a Shape object and ~~has~~ manages its presense on the screen. if no shape is given, then the Shape Drawable will default to a rectangle shape.

- A shape Drawables are XML files which allow to define a geometric object with colors, borders & gradients.

- The advantage of using XML Shape Drawable is that they automatically adjust correct size.

- A Shape Drawable is an extension of Drawable.

Q.6 Hardware Acceleration :-

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

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

↑ same

(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);
```

Q. (9) Animation in Android and its types.

- Animation is the process of adding a motion effect to any view, image, or text.

- With the help of an animation, you can add motion or can change the shape of a specific view.

- Animation in Android is generally used to give your UI a rich look & feel.

- There are three types of animations:

  (1) Property Animation
  (2) View Animation
  (3) Drawable Animation

[1] Property Animation :-

- property Animation is one of the robust frameworks which allows animating almost everything.

- This is one of the powerful and flexible animations which was introduced in Android 3.0.

- Property Animation can be used to add any animation in the checkBox, Radio Buttons, and widgets other than views.

[2] View Animation :-

- View Animation can be used to add animation to a specific view to perform tweened animation on Views.

- Tweened Animation calculates animation information such as size, rotation, start point, and end point.

- These animations are slower and less flexible.

- The example of view animation can be used if we want to expand a specific layout in that place we use View Animation.

- The example of View Animation can be seen in Expandable RecyclerView.

[5] Drawable Animation :-

- Drawable Animation is used if you want to animate one image over another.

- The simple way to understand is it to animate drawable is to load the series of drawable one after another to create an animation.

- A simple example of drawable animation can be seen in apps splash screen or apps logo animation.

Page No.
Date

⊛ Methods of Animation :-

(1) startAnimation() :-

- This method will start the animation.

(2) clearAnimation() :-

- This method will clear the animation running on a specific view.

Q.(10) Explain 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.

Q.(11) 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.

Q. (11) Tween Animation :-

- 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 frames-by-frame animation. If you can create the animation you want by tweening, its definitely the way to go.

- Flash offers two types of tweening:

  (1) motion tweening
  (2) shape tweening

- Tween Animation is basically used to make a GUI of the Activity or Application more interactive & 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 :-

(1) Alpha : Gradient Transperency 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 :

Page No.
Date

```
Animation myAnimation =
    AnimationUtils.loadAnimation(getApplicationContext(),
    R.anim.this_animation);
```

- to apply animation to object, we call startAnimation() method of the object, as follows:

```
TextView text = (textView) findViewById
                    (R.id.tv1);
text.startAnimation (animation);
```

⑧ Methods/Functions of Animation class :-

(1) start() - it starts the animation.
(2) setDuration(long duration) - it set the time duration of the animation.
(3) getDuration() - it gets the method to be set by the above.
(4) end() - it ends the animation that was started.
(5) cancel() - it cancels the working of animation.

---

⑨ Alarms in Android : Tech. Book P.g.no - 6-16.

---

10. ⑫ Download Manager :-

- The download Manager is a system service that handles long-running HTTP downloads.

- clients may request that a URI file be downloaded to a particular Destination.

- One big Advantage of Android Download Manager is that it optimizes the handling of long-running downloads in the background.

Q. (13) Media player in Android.

OR

Audio & Video files in Android.

OR

How to play Audio & Video files in Media player.

- One of the most important components of the media framework is the MediaPlayer class.

- Media player is a part of the Android multimedia framework that plays Audio 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.

- The Media player class primarily handles the playback of audio & video within android application.

- To play a media resource, you need to create an instance of MediaPlayer class, then initialize it with media resource and prepare it for playback.

⊛ 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.

⊛ Playing Audio in Media Player :-

- To play local audio in supported formats, first we should put the local audio

File into res/raw folder.

- For example, put this sample-audio.mp3 into res/raw/sample-audio.mp3.

- Now, we can use the Media player 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 beggining. if this method is called again after the pause() method, the music would start playing from where it left off and not from the beggining.

- 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(fileName);
mediaPlayer.prepare(); // must call prepare first
mediaPlayer.start(); // then start
```

⊛ 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) ~~but~~ 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 timing.

⊛ 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.

⑧ Recording Video :-

- Android has two methods for recording video:

(1) Intents to Record Video
(2) Media Recording API

[1] Intents to Record Video :-

- 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.

- 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 :-

- you can use a MediaRecorder class to record video.

- 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" />
↑ same .RECORD_VIDEO" />
↑ same .CAMERA" />

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

Q.(14) 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 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.

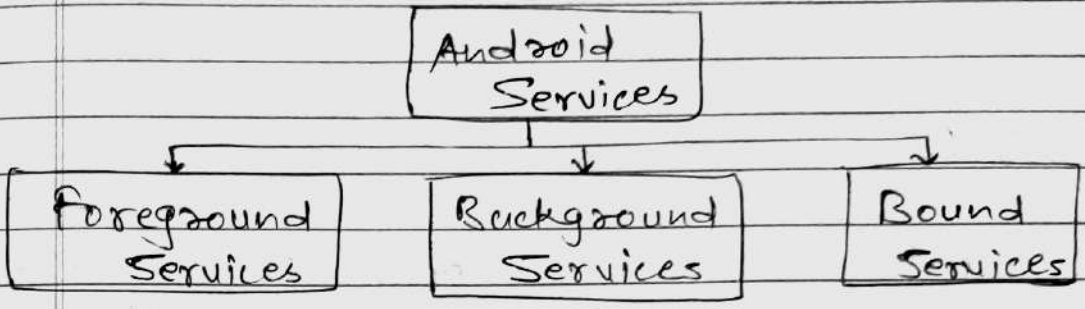
Q.(15) Services and its types and life cycle of Service.

- Service in Android are a special component that facilitates an application to run in the background in order to perform long-running operation tasks.

- The prime aim of a service is to ensure that the application remains active in the background so that the user can operate multiple applications at the same time.

- A service can run continuously in the background even if the application is closed or the user switches to another application.

⊛ Types of Services :-



[1] Foreground Services :-

- Services that notify the user about its ongoing operations are termed as Foreground Services.

- Users can interact with the service by the notification provided about the ongoing task.

- Such as downloading a file, the user can keep track of the progress in downloading and can also pause and resume the process.

[2] Background Services :-

- Background services do not require any user intervention.

- These services do not notify the user about ongoing background tasks and users also cannot access them.

- The process like schedule syncing of data or storing of data fall under this services.

[3] Bound Services :-

- This type of android service allows the components of the application like activity to bound themeselves with it.

- Bound services perform thier task as long as any application component is bound to it.

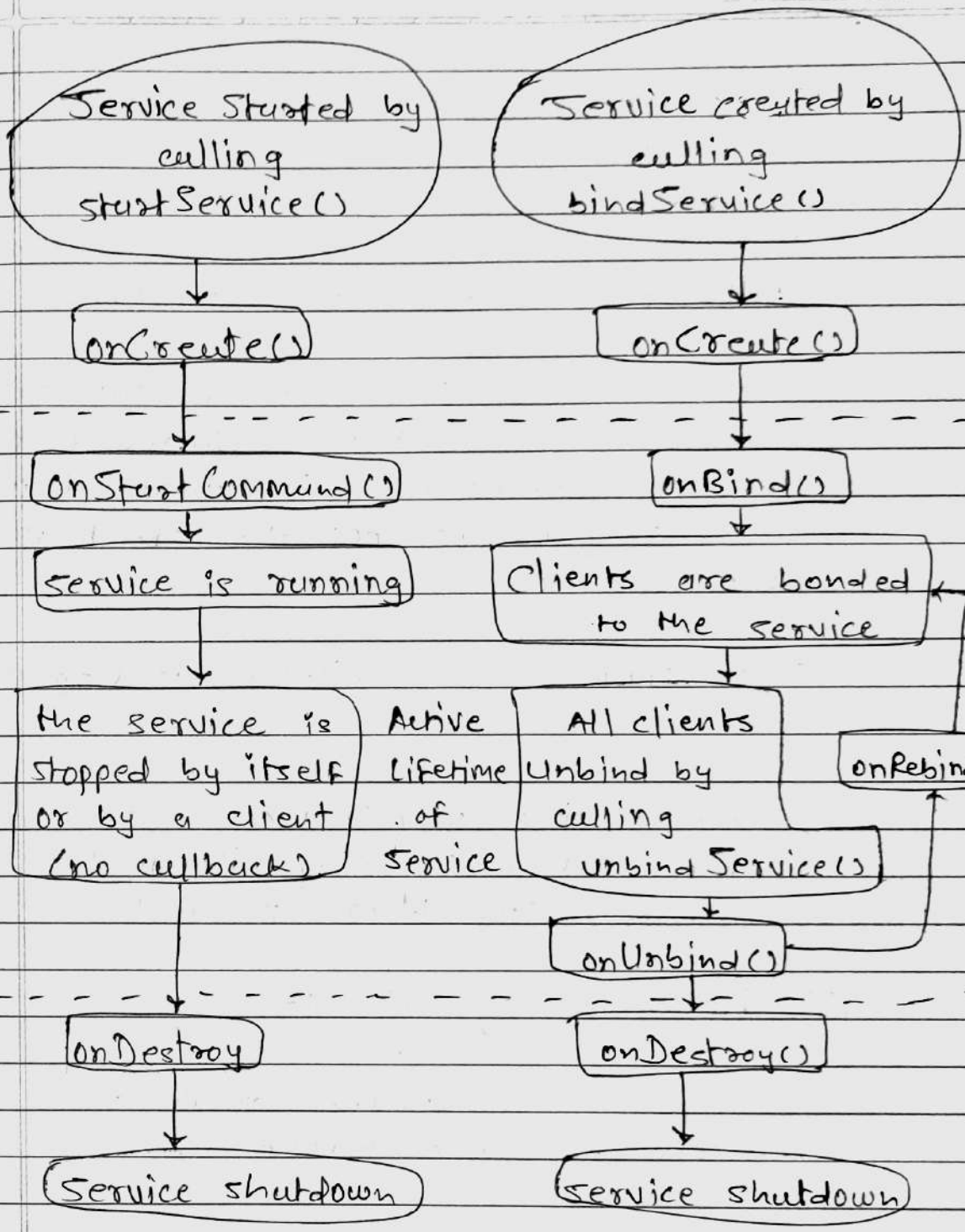
- More than one component is allowed to bind themeselves with a service at a time.

- In order to bind an application component with a service bindService() method is used.

⊛ The Life Cycle of Android Services :-

- In android, services have 2 possible paths to complete its life cycle namely Started and Bounded.

(1) Started Service (unbounded service)
(2) Bounded Service



Unbounded Service Life cycle

Bounded Service Life cycle

⊛ Lifecycle of Service ⊛

___ , ___ , ___ .

[1] Started Service (unbounded Service):-

- By following this path, a service will initiate when an application component calls the startService() method.

- Once initiated, the service can run continuously in the background even if the component is destroyed which was responsible for the start of the service.

- Two option are available to stop the execution of service:

  - By calling stopService() method.
  - The service can stop itself by using stopSelf() method.

[2] Bounded Service:-

- it can be treated as a server in a client-server interface.

- By following this path, android application components can send requests to the service and can fetch results.

- A service is termed as bounded when an application component binds itself with a service by calling bindService() method.

- To stop the execution of this service, all the components must unbind themselves from the service by using unbindService() method.

Ex. - Best Example of Android Service is :

- Playing music in the background

Q. (16) Notification Services :-

- Notification is a kind of message, alert, or status of an application that is visible or available in the Android's UI elements.

- This application could be running in the background but not in use by the user.

- The purpose of a notification is to notify the user about a process that was initiated in the application either by the user of the system.

(*) Types of Notifications :-

(1) Status Bar Notification
(2) Notification drawer Notification
(3) Heads-Up Notification
(4) Lock-Screen Notification

⊛ Notification & Notification Manager :-

- To create a status bar notification, we will need to use two classes:

  (1) Notification
  (2) Notification Manager

⊛ Notification :-

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

⊛ NotificationManager :-

- NotificationManager is an android System service that executes and manages all notifications.

- Hence ~~the~~ we cannot create an instance of the NotificationManager but we can retrieve a reference to it by calling the getSystemService() method.

- Notification Manager manages the notification lifecycle, ensuring a consistent and efficient user experience across different devices & Android versions.

Q.(17) How to set Notification in Android App? Explain it with example.

- To create a notification in Android, you can use the NotificationManager class.

- Here are some steps to create Notification in Android:

(1) Specify a content intent defined with a PendingIntent object.

(2) Pass it to setContent()

(3) Create an explicit intent for an Android in your app.

(4) Set the intent that fires when the users taps the notification.

Eg. NotificationCompat.Builder builder
= new NotificationCompat.Builder (this, CHANNEL_ID)
• setSmallIcon (R.drawable.notification_icon)
• setContentTitle (textTitle)
• setContentText (textContent)
• setPriority (NotificationCompat.PRIORITY_DEFAULT);

Q.(18) Broadcast Receiver :-

- Android broadcast receiver is an android component that is used to broadcast the

Page No.
Date

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 ~~times~~. fires.

- BroadcastReceivers register to receive events in which they are interested. When event occurs they are presented ~~an~~ as intents, those intents are then broadcast to the system.

- Android routes the intents to BroadcastReceivers that have registered to receive them. Broadcast receivers 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.

Q.(19) Write the steps for CRUD operation to use firebase database in android application.

Step:(1) set up firebase in your Android project.

(1) Create firebase project
(2)
(2) Add firebase to your app:

→ Tools → Firebase
- Connect your app with Firebase

(3) Add Firebase SDK:

→ by adding necessary dependencies in your app-level "build.gradle" file.

Step:(2) Initialize Firebase in your Android App.

→ Add Config. file provided by firebase.
→ Initialize firebase in onCreate() method in your mainActivity.

FirebaseApp.initializeApp(this);

Step:3 Perform CRUD operations

FirebaseDatabase database
= FirebaseDatabase.getInstance();

DatabaseReference myRef
= database.getReference("your reference");

// To add data

myRef.child("unique_key").setValue(your data);

Step:4 Security Rules:-

- Configure security rules in firebase.
- console to control access to your db.

Step:5 Testing:-

→ Test CRUD and handle error or exceptions approximately.

* —— * —— *

Page No.
Date

Chapter 8: Work with Android System and Development & Deployment

Q.1 Text to Speech in android.

- 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.

Q.2 Cameras in Android.

- Android provides working with cameras in two ways:

  (1) Camera Intent
  (2) Camera API

Camera Intent :-

- the easiest way to take a picture from within your application is by using an Intent and applying the constants

From the media store class
ACTION_IMAGE_CAPTURE.

Ex. startActivityForResult(
new Intent(MediaStore.ACTION_IMAGE_
CAPTURE), TAKE_PICTURE);

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

Q.(3)

(4) Camera API :-

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

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

- use the Camera class to adjust camera settings, specify image preferences, and take pictures.

- To access the camera, use the static open method on the ~~camera~~

Camera class : Camera camera
= Camera.open();

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

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

Q.(B) 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 service 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:

(1) Java SDK
(2) Android SDK
(3) Device Specific Drivers

- The main services provided by DDMS are:

(1) App memory usage statistics
(2) App Thread statistics
(3) Device screen capture
(4) Device file explore
(5) Incoming call and SMS spoofing
(6) Location data spoofing
(7) Logcat

Q.④ 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 and the background running daemon process.

- 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).

⑤ Accessing the emulator with ADB:-

- ADB offers many of the features of DDMS, including the means to:

(1) View log output
(2) Access the file system on the emulator or device
(3) Access SQLite databases on the emulator or device.
(4) Issues shell commands on the emulator or device.

Q.⑤ Publishing Android app :-

- publishing an application involves digitally singing it and uploading it to the appropiate platforms.

- There are various possibilities in which the application can be released, such as on Google play, Websites or directly to the users.

- Google Play uses application package names as unique identifiers and will not allow you to upload a duplicate package name.

- To release an app on Google Play, Follow these simple steps:

(1) Add some promotional material
- Screen shots, videos & interesting Feature of app.

(2) Configure options - information such as

language, country, type, category, etc.

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

ⓐ process/steps to/of Publishing an Android Application ⓐ

- The steps of the publishing process can be summerized as follows:

(1) select an appropiate Appstore.

(2) Read and Understand the policies and agreements of the selected Appstore

(3) Quality Test

(4) Determine the content rating for the Android application.

(5) Determine the country to distribute

(6) Confirm the overall size, platform & the screen compatibility ranges.

(7) Decide the revenue model.

revenue.

(8) Decide how to bill or collect the (e.g. In-App or using Google play).

(9) Set the price

(10) Localization

(11) Prepare promotional graphics, videos and screencasts.

(12) Build and upload

(13) Plan for Beta release

(14) Complete Appstore listing

(15) Support users after launch.

(*) Characteristics of App :-

(1) Performance of App
(2) Mobility of App (release multiple Versions of App)
(3) Availibility of App
(4) Security of App

Q.(6) Sensors and its types :-

- most Android-powered devices have built-in sensors that measures motion, orientation, & various environmental conditions.

- Android sensors can be used to monitor the three dimensional device movement or change in the environment of the device such as light, proximity, rotation, movements, etc.

Types of Sensors :-

(1) Motion Sensors :-
(2) Position Sensors :-
(3) Environmental 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] Position Sensors :-

- These sensors measure the physical position of a device.

- This category includes orientation sensors and magnetometers.

Page No.
Date

[5] Environmental Sensors :-

- These sensors measure various environmental parameters, such as ambient air temperature & pressure, illumination, and humidity.

- This category includes barometers, photometers and thermometers.

Page No.
Date

(x) Other Questions (*)

Q.① Difference Between JVM and DVM.

Ans.	(2) DVM	(1) JVM
(1)	It is Register based which is designed to run on low memory.	It is Stack based.
(2)	DVM uses its own byte code and runs the ".Dex" file. From Android 2.2 SDK Dalvik has got a Just in Time compiler.	JVM uses Java byte code and runs ".class" file having JIT. (Just In Time).
(3)	DVM has been designed so that a device can run multiple instances of the VM efficiently. Applications are given their own instance.	A single instance of JVM is shared with multiple applications.
(4)	DVM Supports the Android OS only.	JVM Supports multiple OS.
(5)	For DVM very Few Re-tools are available.	For JVM many Re-tools are available.
(6)	There is a constant pool for every application.	It has a constant pool for every class.
(7)	Here the executable is a APK	Here the executable is JAR

Q.(2) Components / Building Blocks of Android.

- An Android Component is a simply a piece of code that has a well defined life cycle. eg. Activity, service, etc.
- The core Building blocks or fundamental components of android are activities, views, intents, services, content, providers, Fragements and AndroidManifest.xml.

(1) Activity :-

- An Activity is a class that represent a single screen.
- it is like a Fraime.

(2) View :-

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

(3) Intent :-

- intent is a used to invoke components.
- it is mainly used to:
  - start the service.
  - Launch an activity
  - Display a web page.

- Following is used to Display a webpage.

```
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.Parse("http://www.youtube.com"));
startActivity(intent);
```

(4) Service :-

- service is a background process that can run for a long time.

- There are two types of services - local & remote.

  - local service is accessed from within the application.

  - remote service is accessed remotely from other applications running on the same device.

(5) Content Providers :-

- Content providers are used to share data between the application.

(6) Fragement :-

- Fragements are like parts of Activity.

- An Activity can display one or more fragements on the screen at the same time.

(7) Android Manifest.xml :-

- it contains informations about activities, content providers, permissions, etc.

- it is like the web.xml file in Java EE.

---

(3) How to perfor CRUD operation in Firebase database write steps.

(1) Create a Firebase project.

- if you don't already have one, you can create a free Firebase at the Firebase console.

(2) Add Firebase to your Android project.

- In Android Studio, follow the steps in the

⊛ Firebase CRUD operation :-

(1) Create a new Firebase project and add your Android app to it.

(2) In the Firebase console, enable the Realtime Database.

(3) Add the Firebase SDK to your Android project.

(4) Create a new Java class to represent your data model.

(5) Create a Firebase Database reference in your MainActivity.java file.

(6) Write code to add, read, update and delete data in the Firebase Database.

(7) Test your app to make sure that CRUD operations are working expected.

: Code :

(1) // Create a new Firebase Database reference.

```
FirebaseDatabase database
    = FirebaseDatabase.getInstance();
```

(2) // Create a new data model object.

MyData myData = new MyData("Rohan", 21);

(3) // Add the data model object to the database. (Create)

database.getReference("myData").setValue(myData);

(4) // Get a ref reference to the data model object.

MyData myData = database.getReference("myData").getValue(MyData.class);

(5) // print the data model object's properties. (Read)

System.out.println(myData.name);
System.out.println(myData.age);

(6) // update the data model object's properties. (Update)

myData.name = "Own";
myData.age = 20;

(7) // Save the updated data model object to the database.

database.getReference("myData").setValue(myData);

(8) // Delete the data model object from the database. (Delete)

```
database.getReference("myData").removeValue();
```

(9) write a code to display Toast message on click Button.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/btnShowToast"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Show Toast"
        android:layout_centerInParent="true" />

</RelativeLayout>
```

MainActivity.java

```
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    @Override

protected void onCreate(Bundle saveInstanceState){

    super.onCreate(SaveInstanceState);
    setContentView(R.layout.activity_main);

 // Find the button by its ID

Button btnShowToast = findViewById(R.id.BtnShowToast);

// set a click listener for the Button

btnShowToast.setOnClickListener(new
                View.OnClickListener() {

    @Override
    public void onClick(View view) {

    // Display a Toast message
Toast.makeText(MainActivity.this, "Button clicked!"
    , Toast.LENGTH_SHORT).show(); }});}}
```

⑧ Write code to insert contact details (cID, cName, cPhoneNumber) in sqlite Database.

→ activity_main.xml

<?xml version="1.0" encoding="UTF-8"?>
<LinearLayout

android:layout_width="match_parent"
" " _height=" " "
" :orientation=" vertical"
tools:context=".MainActivity">

<EditText

android:id="@+id/cId"
" :layout_width="match_parent"
" :layout_height="wrap_content"
" :layout_margin="10dp"
" :hint="customer id" />

<EditText

" " /cName
" "
" "
" "
" :hint="cName"
" :inputtype="text" />

< EditText

" " " /ePhone"
" " "
" " "
" " "
" : hint = " Phone Number "
" : inputType = " Number" />

< Button

android : id = " @ + id / save"
android: layout_width = " match_parent"
android: layout_height = " wrap_content"
android: layout_margin = " 10 dp"
android: text = " save"
android: textAllCaps = " false" />

</ linearLayout>

→ MainActivity.java

import...

import androidx.appCompat.app.AppCompatActivity;

publi class MainActivity extends AppCompatActh {

private EditText cId, eName, cPhone;
private Button save;
private DBhandler dbHandler;

```
@override

protected void onCreate(Bundle saveInstanceState) {
super.onCreate(saveInstanceState);
setContentView(R.layout.activity_main);

cId = findViewById(R.id.cId);
cName = findViewById(R.id.cName);
cPhone = " (R.id.cPhone);
save = " (R.id.save);

dbHandler = new DBHandler(MainActivity.this);

save.setOnClickListener(new View.OnClickL..() {

  @override

public void onClick(View view) {

String cId = eId.getText().toString();
"  cName = cName." . ";
"  cPhone = cPhone." . ";

if(cId.isEmpty() && cName.isEmpty() &&
    cPhone.isEmpty()) {

  Toast.makeText(MainActivity.this,
  "Please enter all Data",
  Toast.LENGTH_SHORT).show();
  return;
}
```

dbHandler.addNewContact(cId, cName, cPhone);

Toast.makeText(MainActivity.this, "Contacts has been added", Toast.LENGTH_SHORT).show();

cId.setText("");
cName.setText("");
cPhone.setText("");

}});}}

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import a.d.s.SQLiteOpenHelper;

→ DBHandler.java

public class DBHandler extends SQLiteOpenHelper {

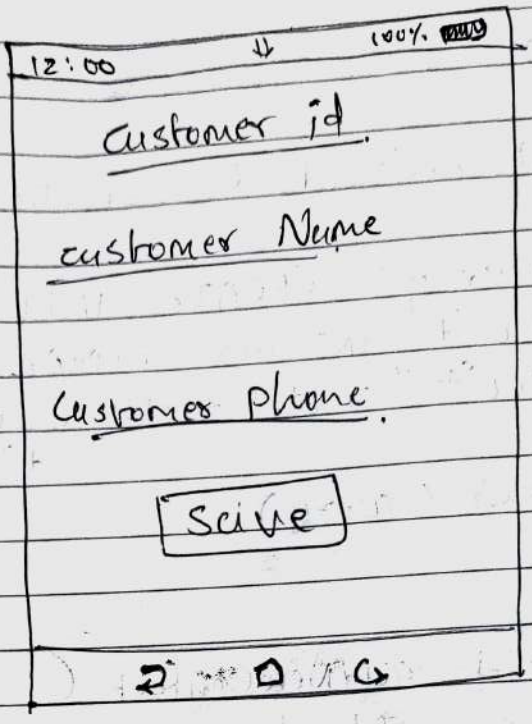
private static final String DB_NAME = "contact.db");
int DB_VERSION = 1;
String TABLE_NAME = "mycontact";
CID_COL = "cid";
CNAME_COL = "cname";
CPHONE_COL = "cphone");

public DBHandler(Context context) {

super(context, DB_NAME, null, DB_VERSION);

}

```
@Override

public void onCreate (SQLiteDatabase db) {

  String query = " CREATE TABLE " + TABLE_NAME
+ " ( " + ID_COL + " INTEGER PRIMARY KEY AUI, "
+ " NAME_COL + " Text, " + PHONE_COL + "Number"
                                    + ");";
  db.execSQL (query);
}

public void addNewContact (
      String cId, String cName, S. cPh.N.) {

SQLiteDatabase db =
   this.getWritableDatabase();

ContentValues values = new ContentValues();
~~Not~~ values.put (ID_COL, cId);
   values.put (NAME_COL, cName);
   values.put (PHONE_COL, cPhone);

db.insert (TABLE_NAME, null, values);
db.close ();
}
  @ override

public void onUpgrade (SQLiteDatabase db,
   int oldVersion, int newVersion) {

   db.execSQL (" DROPE TABLE IF EXISTS "
         + TABLE_NAME );
   onCreate (db); } }
```

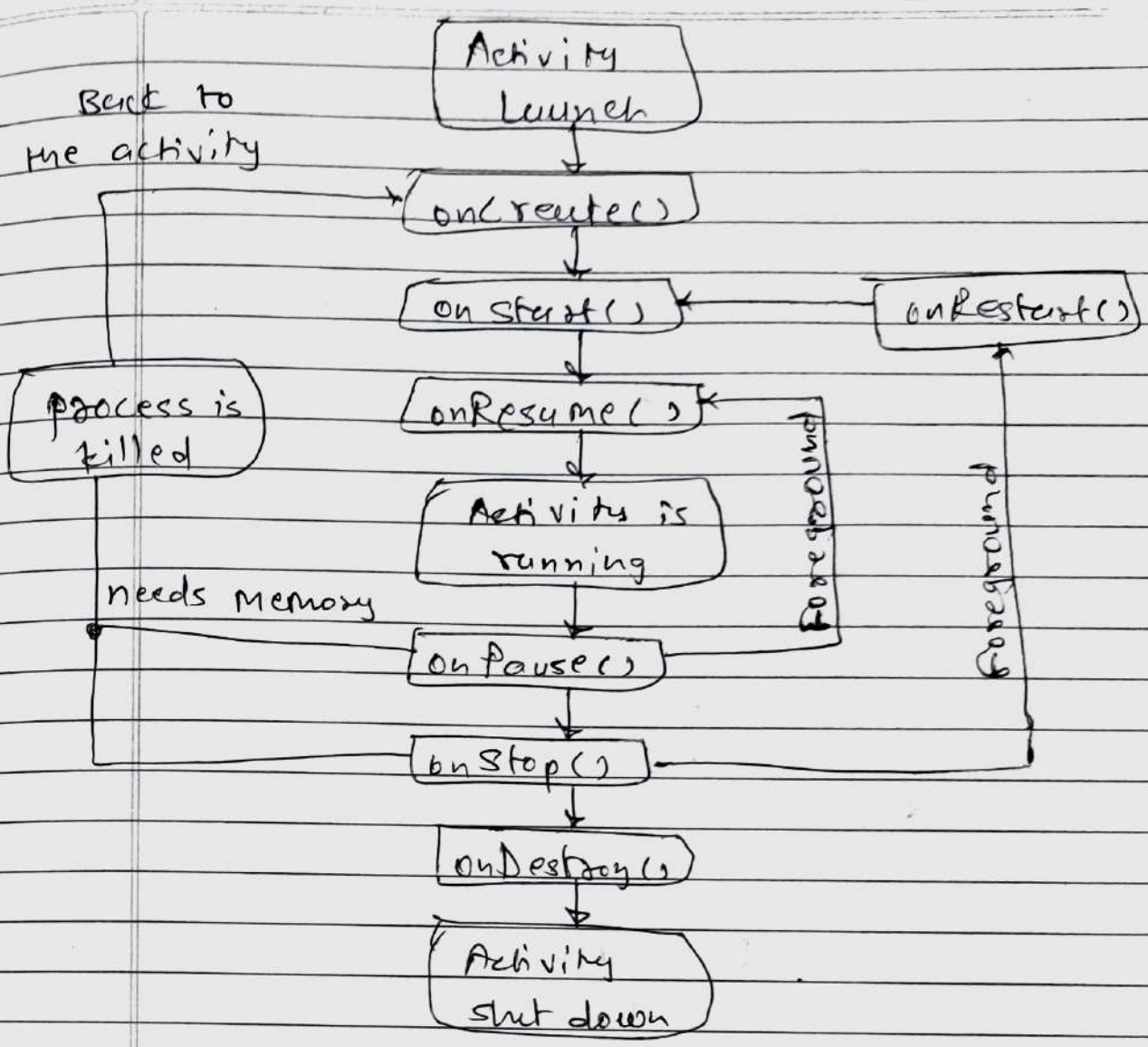


Database: Contacts

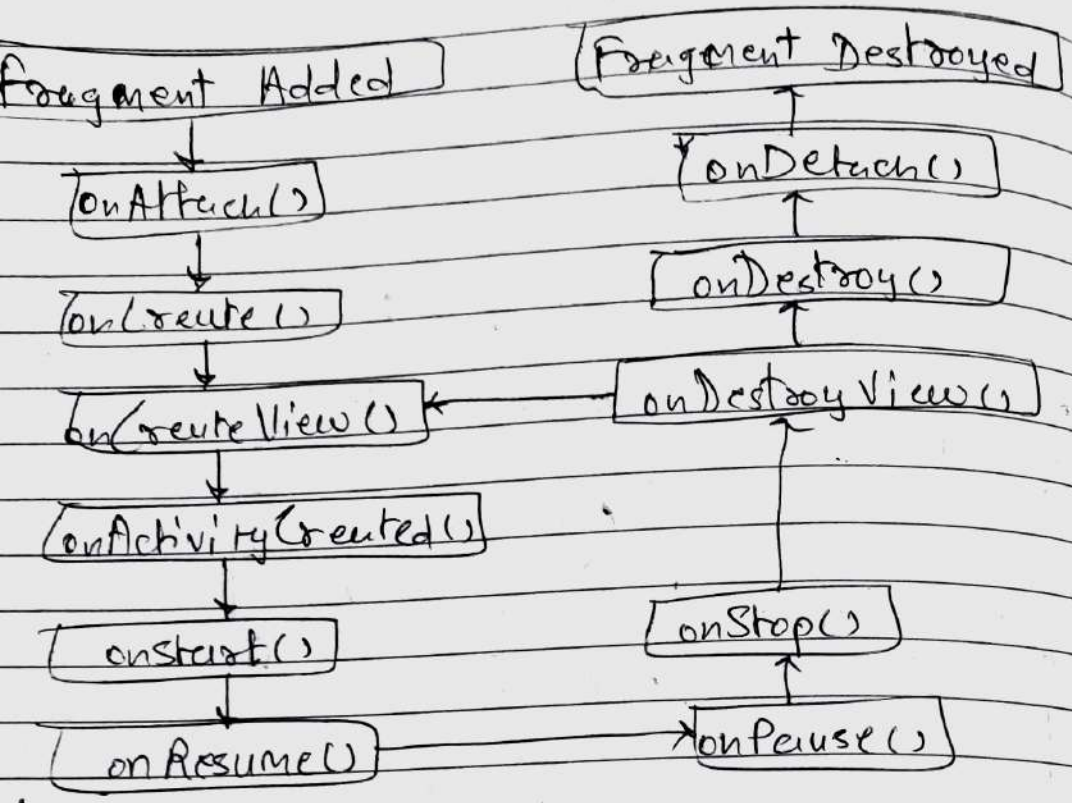
cId	cName	cPHone
1	Rohan	90000000
2	Own	80
3	Vaibhavi	2

Diagrams :-

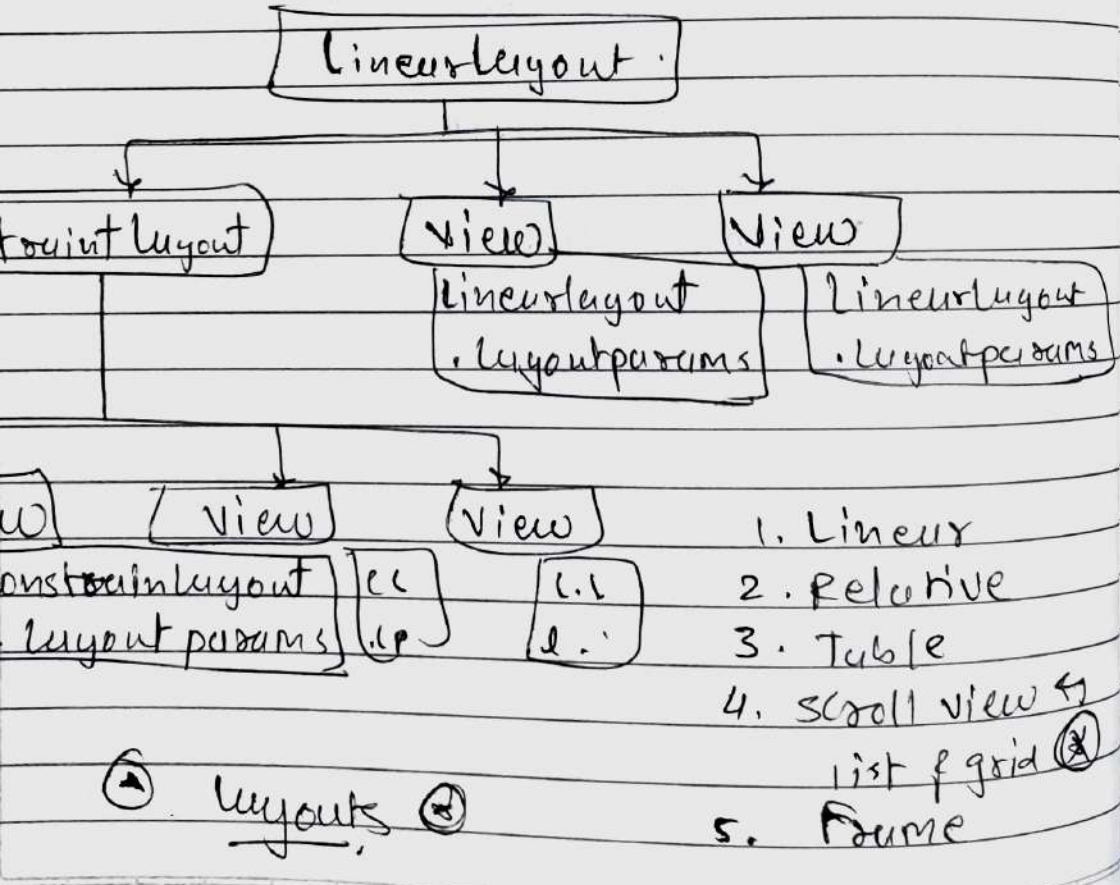
Android Architecture



Activity life cycle



④ Fragement Lifecycle ④



④ Layouts ④

② Android Components :-

(1) TextView
(2) EditText
(3) checkBox
(4) RadioButton
(5) Button
(6) RecyclerView
(7) list view
(8) card view

(1) ~~Text view~~ Button : android : text
android : background
android : id
android : onClicle

(2) TextView : android : id / android : gravity
android : fontFamily
android : text

(3) EditText : android : autoText (spelling)
android : drawableBotton
android : editable
android : background

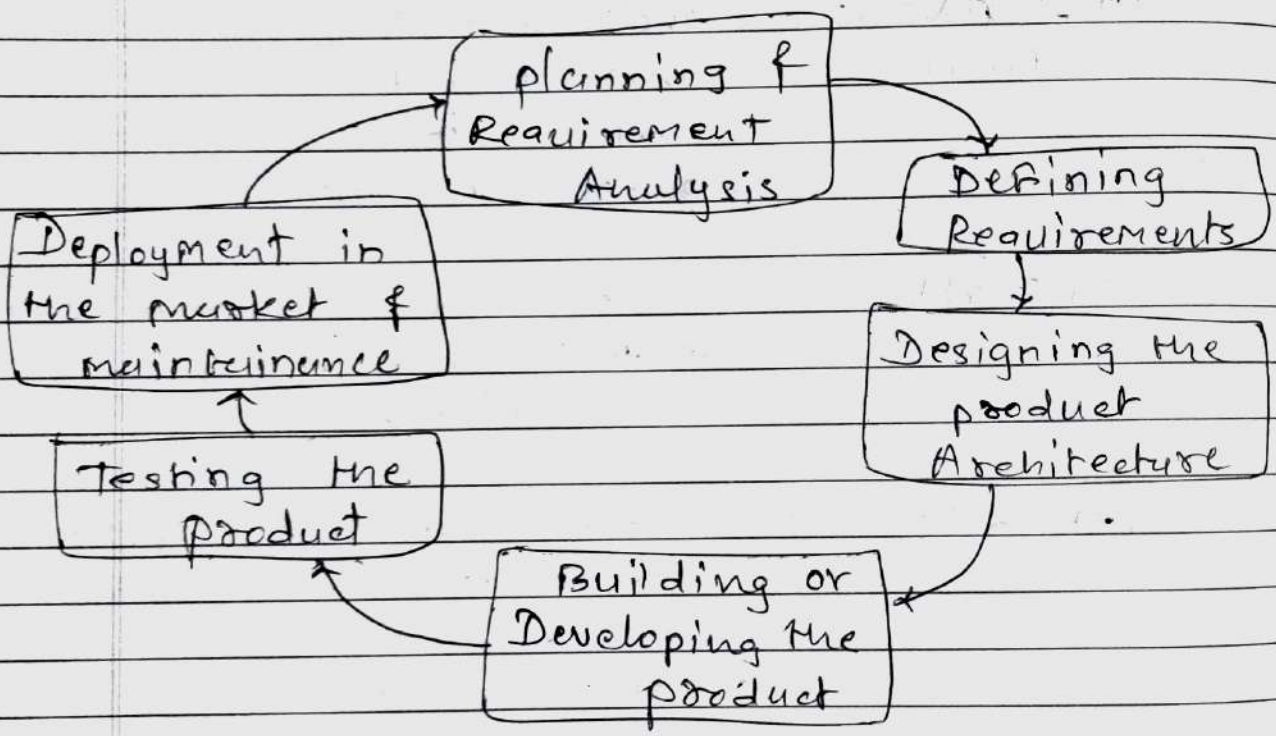
(4) CardView : cardview : cardBackgroundColor
c-v : card cornerRadius
c-v : card Elevation

(5) listview : android : divider
Android : dividerHeight
android : entries

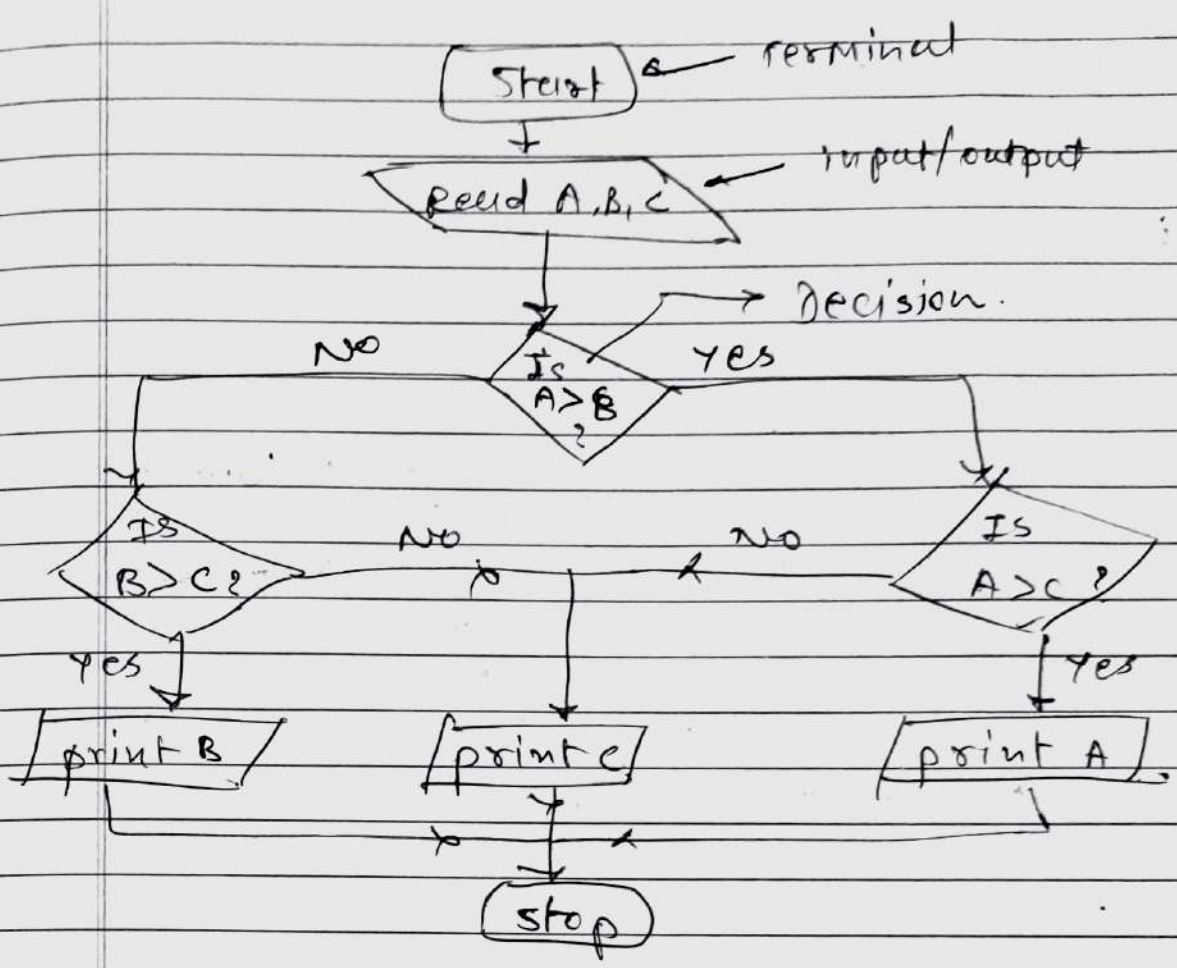
(6) Recycler View : 1. The card layout
2. The ViewHolder
3. The Data class.

(7) Menus : Options Menu
Contextual menu
Popup Menu

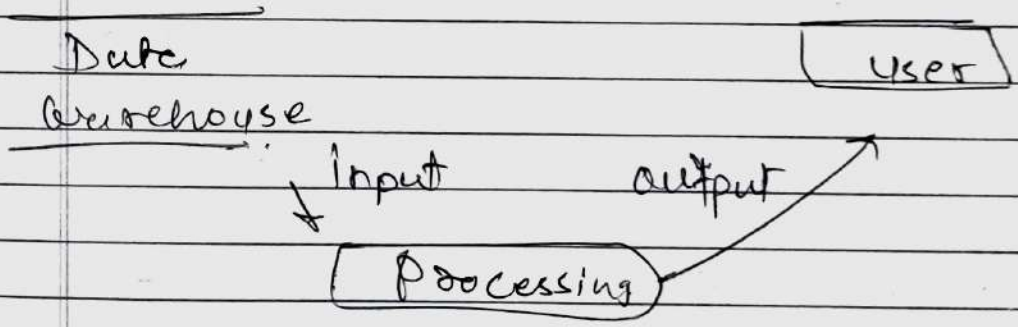
(8) SDLC :



(9) Flow chart : Terminal : (oval)
Input/output : (parallelogram)
processing : (box)
Decision : (Diamond)
Connectors : (circle)
Flow lines : → (Arrow).

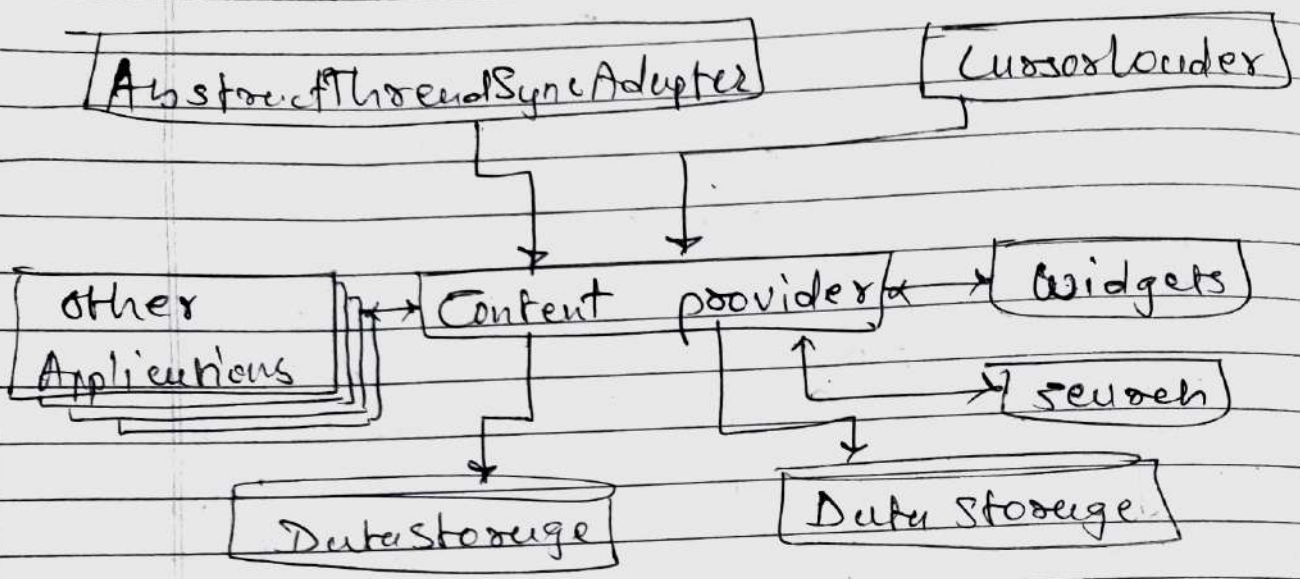


⑧ Data Flow Diagram :-



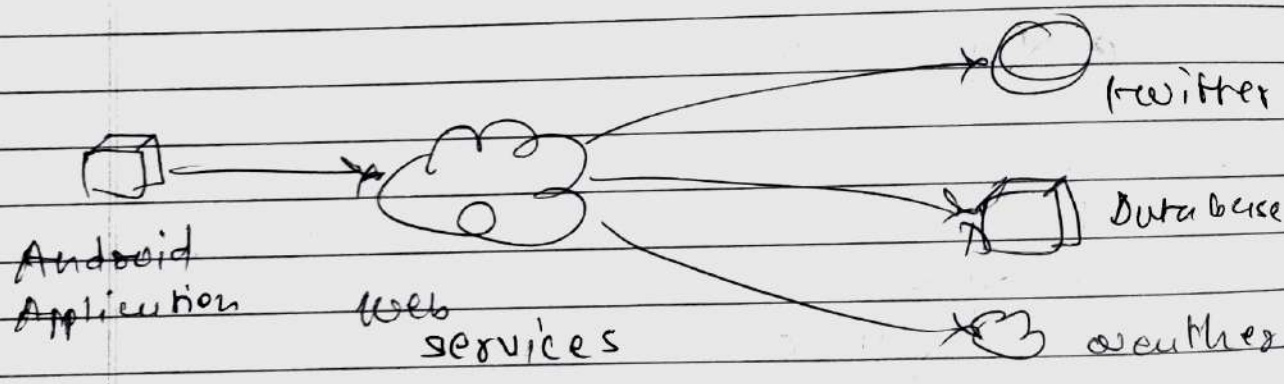
⑧ Structure of DFD ⑧

① Content providers :-



④ Content providers ⑦

② Web Services :-



③ types: Rest [RE Presentational state Transfer]
SOAP [simple object Access Protocol]
WSDL [web service Description Language]

④ Web Services ⑦

③ JSON :-

```
{ "title": "JSON Parser",
  "array": [
      {
        "company": "Facebook",
      },
      {
        "company": "Google"
      },
      {
        "company": "Microsoft"
      }],
  "nested": {
      "Flag": true,
      "random-number": 7
      }}
```

④ Animation :-

(1) Property Animation (ui components ⊗ in views)
(2) View Animation (in views)
(3) Drawable Animation (image over image).

⑤ Tween Animation :- motion tweening
shape tweening

⑥ Types: Alpha
Scale
Translate
Rotate

④ Methods :-

setDuration()
getDuration()
Start()
end()
cancel()

⑧ Media player

Component of media framework

⑧ Audio files :-

Audio track class (play)
Audio Record class (record)

⑧ playing Audio in Media Player :-

res/raw/sample_audio.mp3

instance of media player 7

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

⑧ video :-

< VideoView

android:id="
width

③ Media controller :-

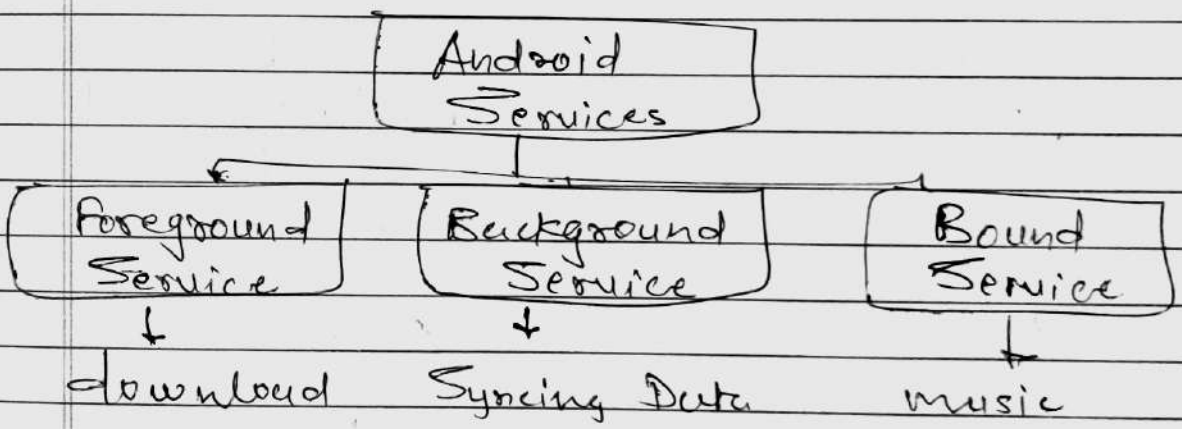
④ Record Video :-

(1) Intents to record video
↓
ACTION_VIDEO_CAPTURE

(2) Media Recording API
↓

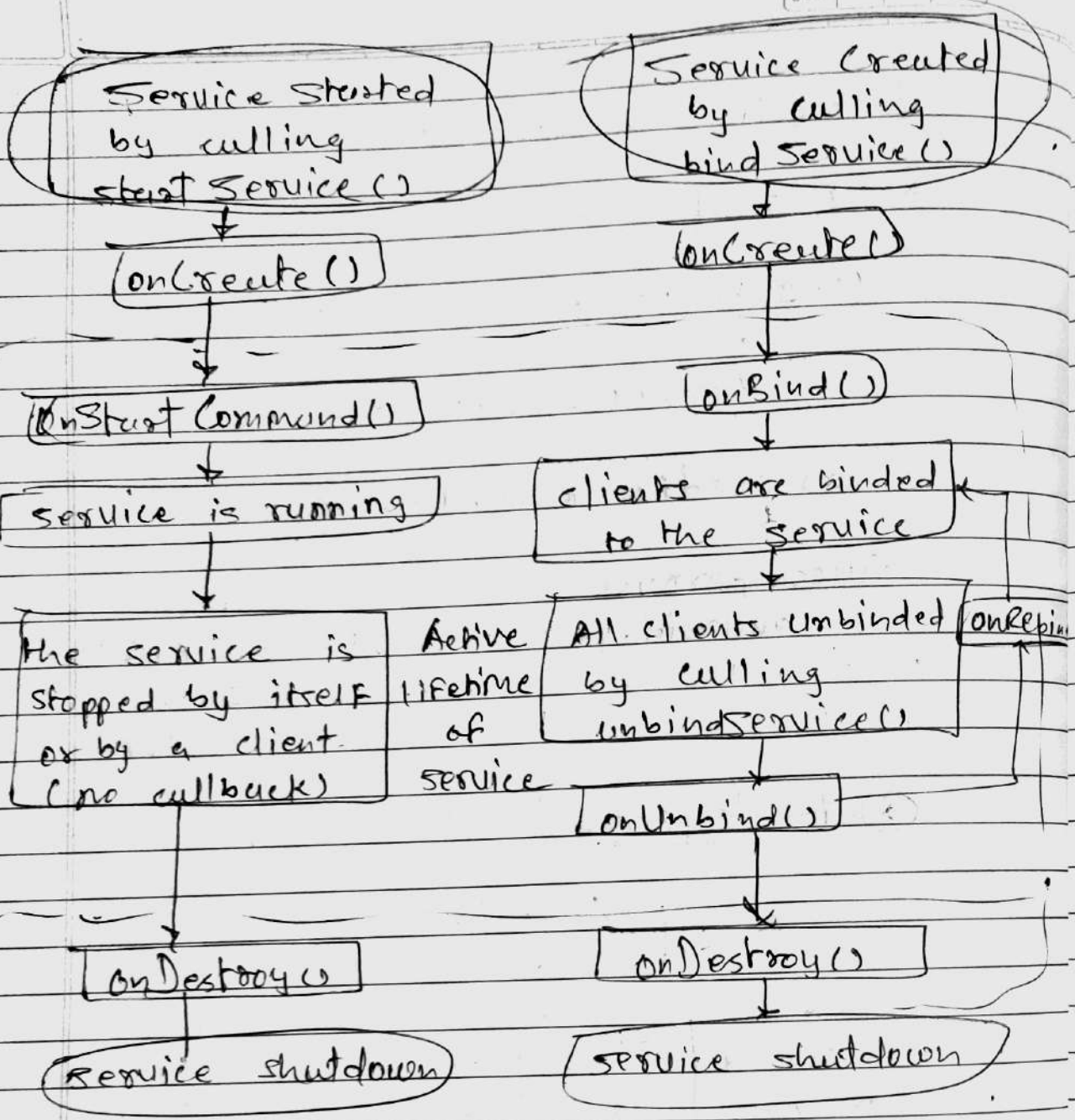
<uses-permission android:name
="android.permission.RECORD_AUDIO" />
" = " " . " _ VIDEO" />
" " " . " _ CAMERA

⑤ Background Services :-



⑥ Lifecycle :-

(1) Started Service
(Unbounded service)
(2) Bounded Service.



⊛ Lifecycle of Service ⊛

⑧ Notification :-

⑨ Types :- Status-bar - Not
Notification drawer Not.
Heads-up Not.
lock-screen Not.

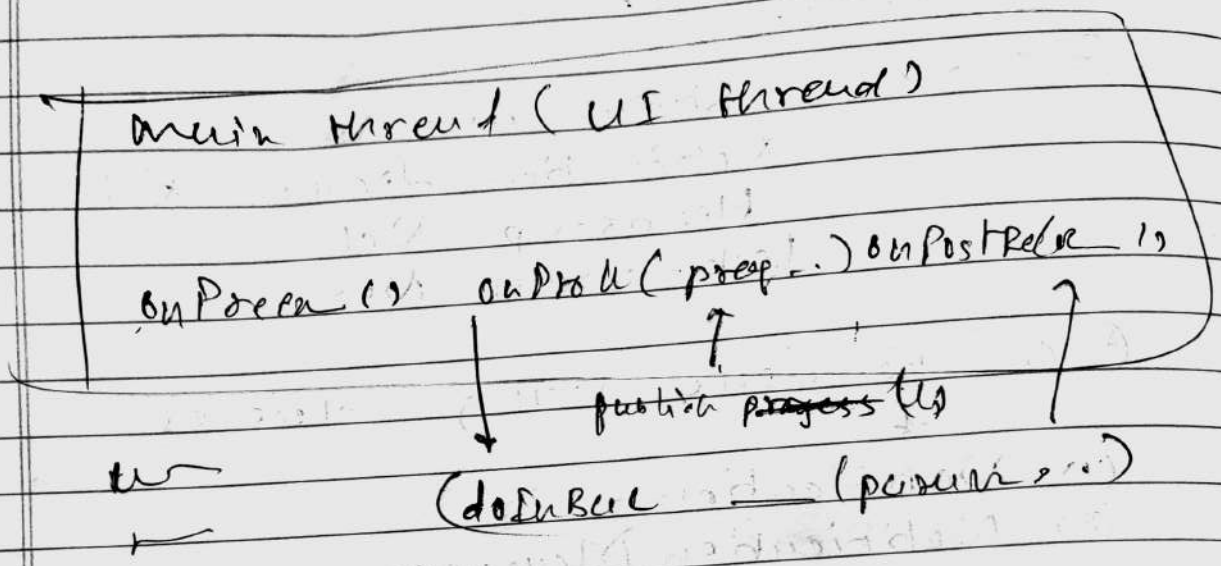
⑩ can be done using classes
*
(1) Notification
(2) Notification Manager

Eg. NotificationCompat.Builder builder
= new NotificationCompat.Builder
(this, CHANNEL_ID)

.setSmallIcon(R.drawable.notification
-icon);
.setContentTitle(textTitle)
.setContentText(textContent)
.setPriority(NotificationCompat.
PRIORITY_DEFAULT);

⑪ Sensors :-

(1) Motion Sensors.
(2) Position Sensors
(3) Environmental sensors


Main Thread (UI Thread)
onPreExecute()
onProgressUpdate(progress...)
onPostExecute()
publishProgress()
doInBackground(params...)