Android Development Short-Notes

CHAPTER 1

ANDROID : Android is a mobile operating system based on a modified version of the Linux kernel and other open source software, designed primarily for touchscreen mobile devices such as smartphones and tablets.

VERSIONS OF ANDROID :

1. donut
2. eclair
3. froyo
4. gingerbread
5. honeycomb
6. icecreamsandwich
7. jelly beans
8. kitkat
9. lolipop
10. marshmellow
11. nought
12. oreo
13. pie
14. os10

ANDROID 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.

The main components of android architecture are following:-

1) Applications : top layer - containing all applications like home, contact, gallary, camera

2) Application Framework : content several imporatant classes to develop applications

3) Android Runtime : Android Runtime environment is one of the most important part of Android. It contains components like core libraries and the Dalvik virtual machine(DVM)

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

5) Linux Kernel : this is like a heart of android architecture. it contains security, memory management, process managment, network stack, driver model

CHAPTER 2

SOFTWARE REQUIRED TO DEVELOP ANDROID APPLICATION :

1. java jdk5 or jdk 6
2. android sdk
3. eclips or android studio
4. Android Developemer tools
5. avd or android phone

ANDROID DEVELOPER TOOLS - Android developer tools create intractive and powerful application

1. SDK Tools **:** these are independent and no matter which android platform your working with

2. Platform Tools : platform tools are customize to support the feacture of the letest android platform

LIST OF PLATFORM TOOLS :

1. android debug bridge (ADB) : Android Debug Bridge (adb) is a versatile command-line tool that lets you communicate with a device.

2. Android Interface Defination language (AIDL)

3. adpt, dexdump and dex et

LIFECYCLE OF ANDORID APPLICATION :

(.java file - java complier(javac) - .class file - dex comploer (dx) -.dex file - packaging adpt- .apk file)

DALVIK VARTUAL MACHINE (DVM) :

the dalvik virtual machine is a register based varitual machine optimised for the mobile devices

it optimize the virtual machine for meemory battrylife and performance

ANDROID VIRTUAL DEVICE :

An andrid virtual device is a configration that defines the charctics of an android phone tables, were os, android tv and automative os

The AVD Manager is an interface you can launch from Android Studio that helps you create and manage AVDs

EMULATOR : In computing, an emulator is hardware or software that enables one computer system (called the host) to behave like another computer system

1) android emulator is a appliation that represents a virtual device

2) this provides all teh android applications in one device

ANDROID MANIFEST : The AndroidManifest.xml file contains information of your package, including components of the application such as activities, services, broadcast receivers, content providers etc.

CHAPTER 3

ACTIVITY : An activity represents a single screen with a user interface.

For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails

If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.

ACTIVITY LIFE CYCLE :

1. ACTIVITY LAUNCHED

2. onCreate() : the activity enters in a create state.

3. onStart() : this makes the activity visiable to user

4. onResume() : called when activity will start interacting with the user.

5. onPause() : called when activity is not visible to the user.

6. onStop() : called when activity is no longer visible to the user.

7. onDestroy() : called before the activity is destroyed.

SERVICES : A Service is an application component that can perform long-running operations in the background, and it does not provide a user interface.

Another application component can start a service, and it continues to run in the background even if the user switches to another application.

TYPES OF SERVICES :

1) FORGROUND : A foreground service performs someoperation that is noticeable to the user. EX) playing audio track in forground

2) BACKGROUND : A background service performs an operation that isn't directly noticed by the user. ex) app used a service to compact its storage

3) BOUND : A service is bound when an application component binds to it by calling bindService().

SERVICES STATES :

1. STARTED : A service is started when an application component, such as an activity, starts it by calling startService().

2. BOUND : A service is bound when an application component binds to it by calling bindService().

LIFE CYCLE OF SERVICES

UNBOUND SERVICE //bound

1) startService() : this is starts the sevice exrecuation

//bindService() : A service is bound when another component

2) onCreate() : The system invokes this method to perform one-time setup procedures when the service is initially created

//onbind : The system invokes this method by calling bindService() when another component wants to bind with the service

3) onStartCommand() : The system invokes this method by calling startService() when another component (such as an activity) requests that the service be started. If we implement this, we must call stopSelf() or stopService() to stop the service.

//onUnbind : The system calls this method when all clients have disconnected from a particular interface published by the service.

//onRebind() : The system calls this method when new clients have connected to the service, after it had previously been notified that all had disconnected in its onUnbind(Intent).

4) onDestroy() : The system invokes this method when the service is no longer used and is being destroyed.

CREATE A SERVCE : 1) startService(new

Intent(getBaseContext(),

MyService.class)); }

2) stopService(neW

Intent(getBaseContext(),

MyService.class));

ANDROID MANIFEST : <service android:name=".MyService" />

TOAST : Toast is used to display information for a period of time. It contains a message to be displayed quickly and disappears after specified period of time.

Toast is a subclass of Object class.

Toast notification in android always appears near the bottom of the screen.

Toast.makeText(Context context, CharSequence text, int duration): This method is used to initiate the Toast.

1. LENGTH_LONG : It is used to display the Toast for a long period of long time

2. LENGTH_SHORT : It is used to display the Toast for short period of Short time

BRODCAST RECIVER : • Android apps can send or receive broadcast messages from the Android system and other Android apps.

• These broadcasts are sent when an event of interest occurs.

• Broadcast Receivers simply respond to broadcast messages from other applications or from the system itself.

• For example, the Android system sends broadcasts when various system events occur, such as when the system boots up or the device starts charging.

HOW TO CREATE : public class MyReceiver extends BroadcastReceiver

{

@Override

public void onReceive(Context context, Intent intent)
{

Toast.makeText(context, "Intent Detected.", Toast.LENGTH_LONG).show();

```
}
```

BRODCAST RECIVERS :

• android.intent.action.BATTERY_LOW : Indicates low battery condition on the device.

• android.intent.action.BOOT_COMPLETED : This is broadcast once, after the system has finished booting

• android.intent.action.CALL : To perform a call to someone specified by the data

• android.intent.action.DATE_CHANGED : The date has changed

• android.intent.action.REBOOT : Have the device reboot

ANDROID MANIFIST : <receiver android:name="MyReceiver">

CONTENT PROVIDER :

• A Content Provider is used to store the data of applications.

• A content provider is only required if you need to share data between multiple applications. i.e Content Provider component supplies data from one application to others on request.

• Such requests are handled by the methods oN the ContentResolver class.

• A content provider can use different ways to store its data and the data can be stored in files, in a database or even over a network.

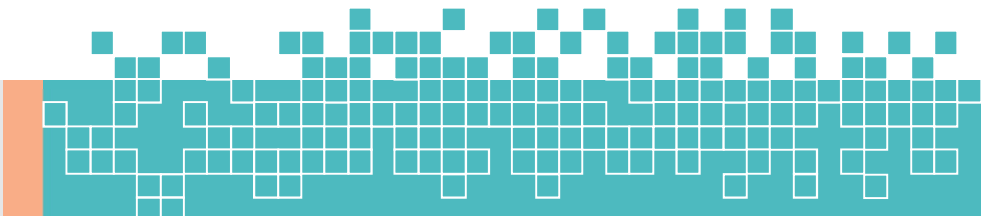
CONTENT URI : The most important concept to understand when dealing with content providers is the content URI(Uniform Resource Identifiers). Whenever you

want to access data from a content provider you have to specify a URI.

• Structure of a Content URI:
content://authority/optionalPath/optionalID

METHODS :

query() : A method that accepts arguments and fetches the data from the desired table. Data is retired as a cursor object.



insert() To insert a new row in the database of the content provider. It returns the content URI of the inserted row.

update() This method is used to update the fields of an existing row. It returns the number of rows updated.

delete() This method is used to delete the existing rows.\ It returns the number of rows deleted.

getType() This method returns the Multipurpose Internet Mail Extension(MIME) type of data to the given Content URI.

onCreate() As the content provider is created, the android system calls this method immediately to initialise the provider.

Fragments:

- Android Fragment is the part of activity, it is also known as sub-activity.
- There can be more than one fragment in an activity.
- Fragments represent multiple screen inside one activity.
- A fragment has its own layout and its own behavior with its own lifecycle callbacks.

• You can add or remove fragments in an activity while the activity is running.

LIFE CYCLE :

1)onAttach(Activity) it is called only once when it is atta-ched with activity.

2) onCreate(Bundle) It is used to initialize the fragment.

3) onCreateView(LayoutInflater, ViewGroup,Bundle) creates and returns view hierarchy.

4) onActivityCreated(Bundle) It is invoked after the completion of onCreate() method.

REMAINIBNG SAME AS ACTIVITY

INTENT :

• An Android Intent is an object carrying a message from one component to another component within the application or outside the application.

• The intents can communicate messages among any of the three core components of an application - activities, services, and broadcast receivers.

• Android intents are mainly used to:

1. Start the service

2. Launch an activity

3. Display a web page

4. Display a list of contacts

5. Broadcast a message

6. Dial a phone call etc.

- **INTENT OBJECT :**

1. Action: This is mandatory part of the Intent object and is a string naming the action to be performed.

2. Data : The URI of the data to be acted on and the MIME type of that data.

TYPES ON INTENT :

1. Explicit Intents: send data within the application

```
Intent i = new Intent(MainActivity.this,SecondActivity.class);
startActivity(i);
```

2. Implicit Intents: send data between two applications

```
Intent i = new Intent();
i.setAction(Intent.ACTION_VIEW);
i.setData(Uri.parse("www.google.com"));
startActivity(i);
```

INTENT FILTERS

- Intent Filters are used with implicit intents.
- Activities which want to handle implicit intents will associate themselves with Intent Filters.
- As the name itself suggests, an Intent Filter essentially signifies: what types of intents an activity would like to receive and which ones it would like to filter.

Creating Intent Filters

- Intent Filters are defined in the manifest file using the

<intent-filter> tag.

EX)

```
<intent-filter>
<action
android:name="com.learncomputer.Module2.SHOW_CURRENT" />
<category android:name="android.intent.category.BROWSABLE" />
<data android:mimeType="audio/mpeg" android:scheme="http"
/>
</intent-filter>
```

ANDROID LAYOUT :

- Layout basically refers to the arrangement of elements on a page.

• All elements in the layout are built with the help of Views and ViewGroups.

ANDROID LAYOUT TYPES

1. Linear Layout: LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally.

2. Relative Layout: RelativeLayout is a view group that displays child views in relative positions.

3. Table Layout : TableLayout is a view that groups views into rows and columns.

4. Absolute Layout : AbsoluteLayout enables you to specify the exact location of its children.

5. Frame Layout : The FrameLayout is a placeholder on screen that you can use to display a single view.

6. List View: ListView is a view group that displays a list of scrollable items.

7. Grid View: GridView is a ViewGroup that displays items in a two-dimensional,scrollable

LAYOUT ATTRIBUTE

• android:id: It uniquely identifies the Android Layout.

• android:hint: It shows the hint of what to fill inside the EditText.

• android:layout_height: It sets the height of the layout.

• android:layout_width: It sets the width of the layout.

- android:layout_gravity: It sets the position of the child view.
- android:layout_marginTop: It sets the margin of the from the top of the layout.
- android:layout_marginBottom: It sets the margin of the from the bottom of the layout.
- android:layout_marginLeft: It sets the margin of the from the left of the layout.
- android:layout_marginRight: It sets the margin of the from the right of the layout.
- android:layout_x: It specifies the x coordinates of the layout.
- android:layout_y: It specifies the y coordinates of the layout.

CHAPTER 4

AutoCompleteTextView : The AutoCompleteTextView is a view that is similar to EditText, except that it shows a list of completion suggestions automatically while the user is typing.

android.widget.AutoCompleteTextView;

Button :A push-button that can be pressed, or clicked, by the user to perform an action.

android.widget.Button;

CheckBox :An on/off switch that can be toggled by the user. You should use checkboxes when presenting users with a group of selectable options that are not mutually exclusive.

android.widget.CheckBox;

ToggleButton : An on/off button with a light indicator.

android.widget.ToggleButton;

RadioButton : The RadioButton has two states: either checked or unchecked.

android.widget.RadioButton;

RadioGroup: A RadioGroup is used to group together one or more RadioButtons.

android.widget.RadioGroup;

ProgressBar : The ProgressBar view provides visual feedback about some ongoing tasks, such as when you are performing a task in the background.

TYPES :

1. **DETERMINATE :** Use determinate mode for the progress bar when we want to show that a specific quantity of progress has occurred.

2. **INDETERMINATE :** Use indeterminate mode for the progress bar when we do not know how long an operation will take.

Ex) xml file : <ProgressBar
android:id="@+id/indeterminateBar"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />

java file : progress=new ProgressDialog(this);

progress.setMessage("Downloading Music");

progress.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);

METHODS : 1) getMax() : retuern maximun value of progress

2) incrementProgressBy(int diff)

3) setMax(int max)

4) setProgress(int value)

Spinner: A drop-down list that allows users to select one value from a set.

android.widget.Spinner;

TimePicker : The TimePicker view enable users to select a time of the day, in either 24-hour mode or AM/PM mode.

android.widget.TimePicker

1. setCurrentHour(Integer currentHour) / setHour(Integer hour):

2. setCurrentMinute(Integer currentMinute)/ setMinute(Integer minute):

3. getCurrentHour() / getHour():

4. getCurrentMinute() / getMinute ():

5. setIs24HourView(Boolean is24HourView):

DatePicker : The DatePicker view enable users to select a date of the day.

android.widget.DatePicker

METHODS : 1. setSpinnersShown(boolean shown):

2. getDayOfMonth():

3. getMonth():

4. getYear():

5. getFirstDayOfWeek():

ListView : android.widget.TextView;

GridView : android.widget.GridView;

Sending Email - In android, we can easily send an email from our android application using existing email clients such as GMAIL, Outlook, etc.

instead of building an email client from scratch.

- For this purpose, we need to write an Activity that launches an email client, using an implicit Intent with the right action and data

Intent it = new Intent(Intent.ACTION_SEND); use put extra method, Constants - EXTRA_BCC, EXTRA_CC, EXTRA_EMAIL, EXTRA_SUBJECT, EXTRA_TEXT, EXTRA_TITLE.

Sending Sms - we can send SMS from our android application in two ways either by using SMSManager API or Intents based on our requirements.

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

android.telephony.SmsManager; SmsManager smgr =SmsManager.getDefault(); smgr.sendTextMessage(sPhoneNo,null,sMessage,null,null);

Bluetooth - <uses-permission android:name="android.permission.BLUETOOTH ~ BLUETOOTH_ADMIN"/>

Constatnts - ACTION_REQUEST_ENABLE, ACTION_REQUEST_DISCOVERABLE, ACTION_STATE_CHANGED, ACTION_FOUND

methods - enable(), isEnabled(), disable(), getName(), setName(String name), getState(), startDiscovery()

Audio Capture - <uses-permission android:name="android.permission.RECORD_AUDIO" ~ "WRITE_EXTERNAL_STORAGE" ~ "STORAGE"/>

Methods - setAudioSource(), setAudioEncoder(), setOutputFormat(), setOutputFile(), stop(), start(), release().

android.media.MediaPlayer; import android.media.MediaRecorder;

Alert Dialog - import androidx.appcompat.app.AlertDialog; import android.content.DialogInterface;

CHAPTER 5

SQLITE : 1. SQLite is an open-ource relational database i.e. used to perform database operations on android devices such as storing ,manipulating or retriving

persistent data from the database.

2. Contains the methods for: creating, opening, closing, inserting, updating,deleting and quering an SQLite database.

3. SQLiteDatabase mydatabase= openOrCreateDatabase ("my_sqlite_database.db" ,SQLiteDatabase.CREATE_IF_NECESSARY , null);

why SQLITE : 1. it dose not required a seprate sever or system to oparte

2. it comes with 0 configration so their is no adminstration needed

3. it is self contained, which means no external dependancies

SQLITE OPEN HELPER : (android.database.sqlite.sqliteopenhelper) it is a class used for database creation and version managment for performing database operation.

syntax : sqliteopenhelper(context, name, database, cursorfactory, version);

EXECSQL : it is method which is used to execuate database quaries.

1. create : mydatabse.execSQL("create table student (enroll int, name string);

2. insert : mydatabse.execsql("insert into students values (1806109, "stduent1");

3. UPDATE : myDB.update(TableName, cv, "_id = ?", new String[]{id});

4. DELETE : myDatabase.delete("students" , "id=?" , new String[ ]{ id.toString( ) }) ;

CURSOR CLASS : we can retrive anything from database using object of curosr class.

RAWQUARY : it is method which is used to run raw quary

Cursor rawQuery(String sql, String[] selectionArgs)

Cursor resultSet = mydatabase.rawQuery("Select * from STUDENTS",null);

CURSOR CLASS METHODS :

getColumnCount() : returns the total number of columns.

getColumnIndex(String columnName) : returns index number of column by specifying name of column.

getColumnName(int columnIndex) : returns name of the column by specifying the index of the column.

getCount() : returns the total number of rows in the cursor.

getPosition() : returns the current position of the cursor in the table.

isClosed() : returns true if the cursor is closed and returns false otherwise.

Chapter 6

Permissions : An application can define its own permission variables and the permissions it needs inside the application's AndroidManifest.xml manifest file.

CUSTOM PERMISSON : This document describes how app developers can use the security features provided by Android to define their own permissions.

By defining custom permissions, an app can share its resources and capabilities with other app

DEPLOYING APP :

1. Register for Google Play : for frist time publishing app you have to register there is $25 charge for regristration

2. Are you going to Sell your App? : we can also add our apps in app purchase.

3. Prepare to Publish : The publishing process in Google Play can be a lot quicker if you have all of the store listing content complete and available before you start.

4. Add New Application : Your final Andromo application file (APK) to be uploaded. It must be under 50MB in size to be accepted.

5. Upload APK : The title for your app (max 30 characters).

A description of your app (max 4000 characters).

High-res icon, 512 x 512 32-bit PNG (with alpha).

Engage your users - understanding their needs, ratings and reviews

Understand your players - Player Analytics will help you understand how players are progressing, spending in your game

Grow your audience - how users find your app, run experiments on your store listing, compare performance to apps in the same category, and launch ad campaigns.

Earn more revenue - Tweak your pricing for individual markets, manage your in-app products and your subscription business, run promotions.