ANDROID with Kotlin

ANDROID :

- Android is a platform consist
- Operating System
- Middleware
- Key Applications

Android Components

- Activity
- Service
- Broadcast Receiver
- Content Provider

Activity :

- A single screen in the application with UI components, user will interact through Activity, it is a java/kotlin file.

Service :

- A long running background process with out any user interaction.

Broadcast Receiver :

- Broadcast Receivers are registered for system events. (eg : headset plugin, power connected/disconnected , screen on/off, making/receiving call…)

Content Provider :

- Content Provider is used to share the data between multiple applications.
- [In Android one of the security feature is which application is created the Database only that application can access the data,
  to share the data to other applications , application has to provide Content Provider.

In android following builtin applications are providing content provider. Contacts , Callog, settings, media, calendar, messages..]

Advanced Android :

- WebServices [JSON, GSON , REST API, Retrofit]
- Google APIs [Maps, Places , Place Picker , Directions]
- Firebase [Authentication, Database , Storage , Admob, Cloud Messaging , MLKit, Analytics, Crashlytics]
- Github
- Design Pattern [MVP / MVVM]
- Material Design [Recyler View & CardView, Floating Action Button , SnackBar, Toolbar & Menus,Navigation Drawer, View Pager & Tabbed Activity …]

Android Environment :

- Android Studio
- Java [Jdk]
- Android SDK

Android Manifest.xml :

- Manifest.xml provides complete (no activities , initial activity, services, broadcast receiver, permissions , features, app icon, app theme, API keys (maps, admit),etc) description about the application.
- Android platform before start the application it will read the Manifest.xml

R.java :

- R is termed as resource.

- R.java is an abstraction between res folder and kotlin file.
- for every resource in res folder it will create a static integer field in R.java, with help of integer field only we can access the resource in kotlin file.
- eg : R.layout.main
R.drawable.sample
R.raw.test

----------X----------X----------

[KOTLIN]

Kotlin:

-Modern object oriented programming language.

variable java syntax:

-> Datatype variable_name = value ;

variable kotlin syntax:

-> mutabletype variable_name = value ;

kotlin function syntax :

* In kotlin there are 2 types of functions:-
- Inline function (if function body contains only one statement)
- function

Inline function syntax :

fun functionName(variable_name:DataType…)=statement

eg :

fun sum(a:Int, b:Int) = a+b

function other:

```
     fun
functionName(varible_name:DataType…):ReturnType
     {
                    // function body
          return value
     }

     eg:   fun sum( ):Int {
          var a = readLine( ).toInt( )
          var b = readLine( ).toInt( )
          return a+b;
     }
      eg:   fun displayTodayInfo( ){
               var today = 27;
               println("Today is :$today");
                              }
      eg: fun displayTodayInfo( ):Unit{
          }
```

Constructor :

- constructor is a special named block with class name (in java).
- constructor is used to initialize the instance variable at the time object creation.
- in kotlin there are 2 types of constructors
- primary constructor
- secondary constructor

primary constructor :

```
class ClassName( ) {  }  // no-argument constructor
class ClassName(variableName:DataType)
//argumented constructor
```

- the input variable of above constructor we can access only with in the init block.

eg:

```
     class Vehicle(no_wheels:Int){
            init {
```

```
                // access constructor input variable
                }
    }
```
- to access the variable anywhere in the class use the following syntax.

```
    class ClassName(var variable_name:Type){  }

    class Vehicle(var no_wheels:Int){
         int no_wheels;
         Vehicle(int no_wheels) {
              this.no_wheels = no_wheels;
    }
```
 - In a single class we can create only one primary constructor.

- Object creation syntax :

```
         val   ref_name = ClassName( )
         val ref_name: ClassName = ClassName( )
```

eg :

```
              val   v  = Vehicle( )
              val v:Vehicle = Vehicle( )
```

Secondary Constructor :

- If you want to create more than one constructor in a class use secondary constructor.

syntax :

```
    constructor(variable_name:Type){
          // constructor body ….
    }
```

- In a single class we can create either primary or secondary constructor.

UpCasting :

holding subclass instance(object) into parent class reference is called as upcasting.

```
ParentClass p = new ChildClass( )
```

Downcasting :

converting parent class reference into child class is called as Downcasting. (down casting is possible only when it is upcasting)

```
ParentClass p = new ChildClass( ) //upcasting
ChildClass c = p; //error
ChildClass c = (ChildClass)p; //downcasting
```

Lamda Expression ( )-> :

Anonymous Inner Class :

- A class with out name is called as Anonymous Inner Class.
- Anonymous inner class .class file will create in the following format.

```
OuterClassName$1.class
```

syntax :

```
new ClassName/InterfaceName( ){
 // provide implementation for abstract function
                                                    }
```

- If you want to create a child class for abstract class or interface, instead of creating a separate class we can create Anonymous inner class.

- If anonymous inner class is providing implementation for functional interface, that anonymous inner class we can represent with LamdaExpression '( )->'.

Anonymous Inner class in Kotlin :

```
     object: ClassName( )/InterfaceName{
          // implementation for abstract functions
     }

Lamda Expressions :

     { param1,param2 ->                         }

Arrays :

var names:Array<String> = arrayOf(value1,value2,…)
var days:IntArray = intArrayOf(value1,value2,value2…)
var weights:FloatArray =
     floatArrayOf(value1,value2,value2…)

var amps:Array<Employee> = arrayOf(value1,value2…)

     when :
          when(value){
               value1 -> statement1
               value2 -> statement2
               value3 -> { statement3
                           statement4
                          }
               else-> statement5
          }

Collections :
     - we can use java collections in kotlin
          List
          Set
          Queue
          Map

          List :
          java :
                                                  L
                                                  L
       kotlin :
                                                  v

     - kotlin collections :
```

- MutableList
- List
- MutableSet
- Set
- MutableQueue
- Queue
- MutableMap
- Map

syntax :
Immutable Type :
var list: List = listOf(value1,value2….)

- we can perform only read operations on List(Immutable type).

Mutable Type :
var list : MutableList = mutableListOf( )
list.add(value)
list.remove( )
list.add(position, value)

----------X----------
----------X----------
--X----------

[XML]
XML :

- XML is termed as Extensible Markup Language.
- HTML is termed as Hypertext Markup Langauge.

Markup Language : Enclosing the data within tags is called as Markup Language. By using XML we can create our own (user defined) tags.

XML Features :

- XML is interoperable [platform independent , technology independent].

- Because of interoperability XML is used to transfer the data between multiple applications.
- XML is used as textual database.
- XML is used as deployment descriptor / configuration file.
- XML is used to design the UI.

sample XML :

```
<employees>
    <employee id="123"

    <employee id="124"

</employees>
```

XML Rules :

- every XML file must have only one root element.
- every xml element must be properly nested.
- XML is case sensitive.
- XML element name / attribute name shouldn't contain space, shouldn't begin with number/special character(_,-).
- attribute values must be placed inside double quotations(").
- use XML entities to represent the following characters.

<	<
>	>
'	'
"	"
&	&

- if any XML follows all the above rules that XML we called as well-formed XML, XML well-formness we can check by using browser.

Custom Rules :

- we can provide custom rules to the XML using DTD/XSD.
- DTD is termed as Document Type Definition.
- XSD is termed as XML Schema Definition.
- If any XML follows the rules of DTD / XSD that XML is called as Valid XML, XML valid ness we can check by using XML editors (Altova XML spy, XML buddy…)
- by default every valid XML is a well-formed XML but can't be vice-versa.

UI Group / Layouts in Android :

- UI group / Layout is used to specify how to arrange the UI components (button, textbox,lable…).
- following are the major UI groups in Android.

  - LinearLayout
  - TableLayout
  - RelativeLayout
  - ContraintLayout

LinearLayout :

- by using LinearLayout we can arrange the UI components in a vertical / horizontal format one after another.

syntax :

```
<LinearLayout
    syntax :
xmlns:prefix="xsd_location"
```

```
    xmlns:android="http://schemas.android.com/apk/res
/android"
               android:layout_width="match_parent"
               android:layout_height="match_parent"
               android:orientation="vertical |
horizontal">
                        // UI components
         </LinearLayout>
```

- for every UI component and UI group we have to specify width and height, following are the possible parameters to specify width and height.

- match_parent
- fill_parent
- wrap_content
- px [pixel]
- dp [density pixel]
- sp [scaled pixel] [only for text size]

Steps to create an Activity [kotlin / java file] :

1. create a class as a child of android.app.Activity[?]

Activity Life Cycle :

- Activity is having 4 states.

  - Doesn't Exist
  - Foreground
  - Pause
  - Background

- following are the major methods of Activity class.

  - onCreate( )
  - onPause( )

- onStart( ) - onStop( )
- onResume( ) - onRestart( )

- every Activity will maintain in Stack [Last In First Out] , when ever stack becomes empty application will close.

2. same like main( ) method in C/C++/Java in Android Activity onCreate( ) method will be invoke first so provide the implementation/override the onCreate( ) method.

```
override fun onCreate(savedInstanceState:Bundle?)
{

super.onCreate(savedInstanceState)
                // function body
}
```

3. we designed the UI in XML, use the following method to set the XML file to kotlin file.

```
setContentView(R.layout.xml_file)
```

- Button is having a click event, we can configure the click event in 2 ways.

- XML
- Kotlin

XML :

- configure the following attribute to the UI component to configure click event.

```
android:onClick="function_name"
```

- when the UI component is clicked it will invoke the specified method in kotlin file, if the method is not available in kotlin file it will give a MethodNotFoundException.

eg: xml: android:onClick="getText"

kotlin : fun getText(v:View) {
// code to execute when Ui comp is clicked..

- to get the UI component from XML to kotlin we have to configure an id for the UI component , use the following attribute to configure the Id.

android:id="@+id/id_name"

(Note : id name shouldn't contain space,special characters, capital letters and shouldn't begin with number)

use the following method to get the component from XML to Kotlin.

findViewById<UIComponent>(R.id.id_name)

- to configure the click event using kotlin configure the following listener to the UI component.

ui_comp.setOnClickListener(implementation_class_

```
/*  gt.setOnClickListener(object :
View.OnClickListener {
        override fun onClick(v: View) {
            tv1.text = et1.text
        }
    }) */
```

```
     gt.setOnClickListener {
         tv1.text = et1.text
     }
----------------------------------------X-------------
----------------------------------X---------------------
-----------------------------
```

Intent :

- Intent is used to provide the communication between Activity - Activity , Activity - Service and Activity - Broadcast Receiver.

- with respect to Activity there are 2 types intents.
  - Implicit Intent
  - Explicit Intent

Implicit Intent :

- Implicit Intent is used to call builtin Activities.
  (eg: Camera , browser , settings, messages …)

syntax : var i = Intent( )

startActivity(i)

Explicit Intent :

- Explicit Intent is used to call user defined Activity.

syntax :

val i = Intent(context,ActivityName::class.java)

- by using explicit intents we can invoke other application activities from our application.

syntax :

```
val i = getPackageManager( ).
getLaunchIntentForPackage("package_name")
startActivity(i)
```

- following are the major methods in Intent class.

setAction( )	putExtra( )
setData( )	getExtra( )
setType( )	

```
dial.setOnClickListener {
      val i = Intent( )
      i.action = Intent.ACTION_DIAL
      i.data =
Uri.parse("tel:${et1.text.toString()}")
      startActivity(i)
   }
```

----------X----------
----------X----------

AutoCompleteTextView :

- ACTV is a child of EditText which is used to provide an auto completion support.

xml :

```
<AutoCompleteTextView
    android:id="@+id/actv"
    ……. />
```

- for providing an auto completion first we have to configure the values, there are 2 ways to configure the values.

- XML
- kotlin

XML :

- configure the values in the below XML.

res >> values >> strings.xml

```
<string-array name="array_name">
        <item> value1 </item>
        <item> value2 </item>
        ……
</string-array>
```

- use the following code to get the XML configured values into kotlin file.

```
val values:Array<String>=

getResources( ).getStringArray(R.array.array_name)
```

kotlin :

```
val values:Array<String> = arrayOf(value1,value2…..)
```

(or)

```
val values:MutableList<String> = mutableListOf( )
values.add(value1)
values.add(value2)
```

……----------

----------X----------

- to present the values we have to create an adapter, there are 3 types of adapters.

- ArrayAdapter
- CustomAdapter

- CursorAdapter

ArrayAdapter :

- by using ArrayAdapter we can present string type of data.

syntax :

```
val adapter:ArrayAdapter<String>=ArrayAdapter<String>

     (context,xml_file,values)
  actv.adapter     = adapter
     actv.threshold = integer_value
```

Spinner :

- Spinner is one of the UI component in Android which is used to display the list of configured values in a dropdown menu.

xml :

```
              <Spinner
                                  android:id="@+id/sp1"
                                  ……. />
```

same like ACTV, to present the values we have configure the values in XML/Kotlin.

- if the values are configured in XML we can use the following attribute to set the values.

android:entries="@array/array_name"

- if the values are configured in kotlin we have to create an adapter to set the values.

- spinner is having onItemSelected event, configure the following listener to get the selected item from spinner.

```
     sp1.onItemSelectedListener =
     object: AdapterView.OnItemSelectedListener {
override fun onItemSelected(var1:AdapterView<?> ,var2:
View,

var3(position): Int , var4:Long ){ }
override fun
onNothingSelected(var1:AdapterView<?> ){    }
          }
```

Toast :

Toast is one of the notification method in Android which is used to display text for few seconds.

syntax :

```
 Toast.makeText(context, message , duration).show( )
```

eg :

```
          Toast.makeText(this@ActivityName, "Hello
World" ,
 Toast.LENGTH_LONG/LENGTH_SHORT).show( )
```

ListView :

ListView is one of the UI component in Android which is used to present the list of configured values.

xml :

```
                              <ListView

     android:id="@+id/lview"

     …….. />
```

same like ACTV & Spinner for presenting the values we have to configure the values in XML / Kotlin file.

- if the values are configured in XML we can use entries attribute to set the values, if the values are configured in kotlin use adapter to set the values.

Storage :

- Internal
- External
  - Phone

/storage/emulated/0

(or)

/storage/sdcard0/
  - sdcard

/storage/extSdCard/

/storage/sdcard1/

- to read the data from external storage , add the following permission in Manifest.xml

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

Runtime Permission (Android 6.0 and above versions) :

1. move the entire code (storage logic, location, contacts…) into a function.

2. get the permission status.

```
var status : Int  = ContextCompact.
  checkSelfPermission(context,
         Manifest.permission.PERMISSION_NAME)
```

3. check the permission is granted or not, if the permission is not granted request the user to grant a permission.

```
if(status == PackageManager.PERMISSION_GRANTED)
{
   readFiles( );
}else{

         ActivityCompact.requestPermissions(context,
arrayOf(Manifest.permission.PERMISSION_NAME1,…),
 request_code)
}
```

- to get the information (whether user selected allow or deny) override the following method in Activity.

```
override fun onRequestPermissionsResult(requestCode:
Int,
                    permissions: Array<String>,
grantResults: IntArray) {
super.onRequestPermissionsResult(requestCode,
permissions,
      grantResults)
if(grantResults[0] ==
PackageManager.PERMISSION_GRANTED)
{
   // logic…
}else{
               System.exit(0)    // to close the
application….
}
}
```

Custom Adapter :

- by using ArrayAdapter we can present only String type of data, to present your own UI on individual use custom adapter.

- to create custom adapter, create a class as a child of android.widget.BaseAdapter.

- it is an abstract class having following abstract functions.

- getCount( ) : Int
- getItem( ) : Any
- getItemId( ) : Long
- getView( ) : View

- LayoutInflater :

LayoutInflater class is used to convert the XML file into View object.

```
var lInflater : LayoutInflater = LayoutInflater.from(context)

var v:View = inflater.inflate(R.layout.xml_file, view_group_object)
```

Gallery :

- Gallery is one of the UI component in Android in Android, which is used to present the data in a horizontal format.

xml : <Gallery

android:id="@+id/gal"

……. />

public static void main(T… args)

WebView :

- WebView is one of the UI component in Android which is used to display webpages/html files in android application.

xml :

```
<WebView

    android:id="@+id/wview"
                                                    ……….
/>
```

kotlin :

- wview.loadUrl("url_address") method is used to display the specified webpage/html on webview.

permissions :

```
    <uses-permission android:name="android.permission.

INTERNET"/>
```

- by using WebView we can perform following operations.

- display webpages on browser.
- integrate the browser in application.
- display static HTML files.
- we can provide the communication between HTML UI and Android Activity.

- to display webpage on webview component set the following to webview.

```
    wview.setWebViewClient(WebViewClient())
                                                (or)
    wview.webviewClient = WebViewClient( )
```

- use the following methods to enable javascript and zoom controls.

```
wview.getSettings().setJavaScriptEnable(true)

wview.getSettings().setBuiltZoomControls(true)
```

- override the following methods of WebViewClient class to get the webview events like (page loading started , finished, loading…)

```
wview.webviewClient = object:WebViewClient( ){
                override fun onPageStartedLoading(……)
                override fun shouldOverrideUrlLoading(….)
                override fun onPageFinished(….)
        }
```

Progress Dialog :

```
val pDialog = ProgressDialog(context)
pDialog.setTitle(“Message”)
pDialog.setMessage(“please wait page is loading”)
pDialog.show( )
pDialog.dismiss( )
```

- to display .html file, place the .html file in assets folder.

right click on app >> new >> folder >> assets.

- use the following code to display .html file.

wview.loadUrl("file:///android_asset/filename.html")

- by using JavaScriptInterface we can provide the communication between HTML UI and Android Activity.

wview.addJavaScriptInterface(any_class_object, interface_name)

- by using interface name (second param) we can call the specified class (first param) methods from java script.

- which method is calling from javascript , that method has to declare with the following annotation.

@JavaScriptInterface

fragment :

- fragment is one of the UI component in Android.

- fragment is subtype of an Activity.
- in a single screen we can create multiple fragments.
- every fragment is having its own life cycle (kotlin file) and its own UI(xml).

Steps to create a Fragment :

- create a class as a child of android.support.v4.Fragment.
- same like onCreate( ) method in Activity , in fragment onCreateView( ) method will invoke first.

```
class MyFrag : android.support.v4.Fragment( )
{
override fun onCreateView(inflater:LayoutInflater,
```

```
    container:ViewGroup, b:Bundle) : View
  {

    var v:View =
inflater.inflate(R.layout.xml_file,container,

false)
    return v
  }
 }
```

- use the view object to get the UI component from fragment XML file (before return statement)
  eg : view_obj.id_name

- use the following code to manage (add, replace, remove…) fragments.

xml :

```
            <fragment
                        android:id="@+id/frag1"
                        ……. />
```

kotlin :

```
  var fManager:FragmentManager  =                                    g

    var tx:FragmentTansaction =

fManager.beginTransaction( )
    tx.add/replace/remove(R.id.fragment_id,

    object_of_fragment_class)
    tx.commit( )
```

- we can display fragment as a dialog by using DialogFragment class.

Fragment Life Cycle :

Shared Preferences :

- Shared Preferences is used to maintain the data in a key, value pair.
- SharedPreferences interface is used to manage shared preferences.

```
var spf: SharedPrefereces = getSharedPreferences(
        "preference_name", Context.MODE)
```

- above method is used to create the shared preferences / open the connection for existing shared preferences.

- to write the data into shared preferences we have to create an object for SharedPreferences.Editor.

```
var spe:SharedPreferences.Editor  = spf.edit( )
    spe.putX(key, value)           // X - Datatype
    spe.commit( )
```

- spf.getX("key",default_value) method is used to read the data from Shared Preferences, if the value is available with the specified key it will return the actual value otherwise it will return the default value.

- internally SPF will maintain the data in a XML file, we can explore the xml file using Device Explorer.

Device Explorer >> data >> data >> package name >> shared_prefernces >> spf_name.xml

- by using SPF we can maintain huge amount of data but for every value we have to specify a unique key it is difficult to assign and remember unique key thats why SPF is recommend to maintain small amount of data.

(eg : device pin number /pattern lock, keep me singin credentials, level highscore, etc…)

- to maintain huge amount of data Android is recommend to use SQLite Database.

SQLite Database :

- SQLiteDatabase is used to maintain structured data in Android.
- SQLiteDatabase interface is used to manage SQLite Database.
- openOrCreateDatabase(…) method is used to get the implementation object of SQLiteDatabase.

```
var dBase : SQLiteDatabase = openOrCreateDatabase
    (database_name,mode,cursor_factory(optional))
```

- SQLiteDatabase is providing predefined methods for performing CRUD operations.

```
dBase.insert(….)
dBase.query(….)      // Read
dBase.update(….)
dBase.delete(….)
```

- we can execute the native SQL queries by using the following methods.

```
dBase.execSQL("sql_query")   // C I U D
dBase.rawQuery("sql_query")   // R
```

- return type of dBase.query( ) (or) dBase.rawQuery( ) method is Cursor, if the data is available in a cursor we can use CursorAdapter for presenting the data.

```
var adapter:SimpleCursorAdapter  =
                    SimpleCursorAdapter(context, xml_file,
```

cursor_object, from, to, flag(int_value))

from - String Array ->arrayOf(db_column1,db_column2..)
to -intArray ->
intArrayOf(R.id.id_name1,R.id.id_name2.)

Steps to Achieve MVP :

- Create application, add the following packages under root package.

- view
- model
- presenter
- beans

- move Activity / Fragment files into view package.
- create data classes for SQLite tables under beans package.
- create the following interface under view package.
(create abstract function in the interface for Ouput operations)

```
interface ViewAPI
    {
        fun saveOutput(msg:String)
        fun readOutput(c:Cursor)
    }
```

- Activity / Fragment class should provide the implementation for ViewAPI interface.
- create the following interface under presenter package. (create abstract function in the interface for DB operations)

```
interface PresenterAPI
{
        fun  save(bean:IncExpBean)
        fun  read( )
 }
```

- create a class under model package, the class should provide the implementation for PresenterAPI interface and should contain a constructor which is taking ViewAPI interface as a input parameter.

Android Telephony :

- SMS
- CALL
- EMAIL
    - Builtin Activity
    - Java Mail API

SMS :

-

android.telephony.SmsManager class is used to send text messages in Android.

```
 var sManager : SmsManager =

SmsManager.getDefault( )

     sManager.sendTextMessage( receiver_num,
          sender_num, message, sendIntent ,
deliverIntent)
```

- In the above method sendIntent and deliverIntent are the Pending Intent objects.

Pending Intent :

Pending Intent is a child of Intent, which will execute later.

```
    var i:Intent = Intent( context,
ActivityName::class.java)
    var pIntent : PendingIntent =
PendingIntent.getActivity(
             context, req_code(int),intent_object,
req_flag(int))
```

permission : (runtime permission is required)

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

Call :

for making calls Android is providing a builtin Activity use implicit intents to call builtin Activity.

```
                            var i: Intent = Intent(  )
                            i.action =
Intent.ACTION_CALL
                            i.data =
Uri.parse("tel:"+number)
                            startActivity(i)
```

permission : (runtime permission is required)

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

Email :

- for sending email Android is providing a builtin Activity use implicit intents to call builtin Activity.

syntax :

```
    var i = Intent( )
    i.action = Intent.ACTION_SEND
    i.putExtra(Intent.EXTRA_EMAIL,arrayOf(mail1,mail2.
.))
```

```
    i.putExtra(Intent.EXTRA_SUBJECT, "subject here")
    i.putExtra(Intent.EXTRA_TEXT, "text here")
    i.putExtra(Intent.EXTRA_STREAM, uri_object)
    i.setType("message/rfc822")
    startActivity(i)
```

permission :

INTERNET

Attachment functionality :

Camera :

```
var i = Intent("android.media.action.IMAGE_CAPTURE")
startActivityForResult(i, request_code)
```

File Explorer :

```
    var i = Intent( )
    i.action = Intent.ACTION_GET_CONTENT
    i.type = "*/*"
  startActivityForResult(i,request_code)
```

if we use startActivityForResult( ) method, the result will get back from the next Activity and will invoke the following method.

```
    fun
onActivityResult(request_code,result_code,data)
  {
    ……..
    }
```

permissions :

CAMERA
WRITE_EXTERNAL_STORAGE

Java Mail API :

- By using builtin email activity we can't send email in the background, user has to select

any one of the email client application (gmail, outlook, etc..) for sending an email.

- To send a mail in the background with out any user interaction use Java Mail API.

Steps to work with Java Mail API :

- add the following .jar files as a modules. (right click on app >> new >> module >> jar >> select jar file)
  - activation.jar
  - additional.jar
  - mail.jar

-> goo.gl/mTjxFF

- add these modules as a dependency modules for the application.
  (right click on app >> module settings >> dependencies >>
  select + >> select module dependency >> select module)

- place the following .java files into package folder.
  - GmailSender.java
  - JSSEProvider.java
  - LongOperation.java

- Configure the from mail credentials in LongOperation.java
- LongOperation is a an AsyncTask , execute the AsyncTask from Activity.

```
var lop = LongOperation(to_mail, subject , message )
lop.execute( )
```

AsyncTask :

- In Android every application will run on a MainThread, if we execute long operations [network calls] on MainThread the Activity may will freeze (this state is called as Application Not Responding).
- If we execute long operations on MainThread, it will give NetworkOnMainThreadException.
- to execute long operations there are 2 options.
    - create a separate thread
    - execute the long operations on MainThread using AsyncTask.
- from separate thread we can't update the UI on MainActivity.

Steps to execute long operations(network calls) using AysncTask :

- create a class as a child of android.os.AsyncTask<Input, Progress, Output>
- It is an abstract class having an abstarct function called doInBackground( )
- following are the major methods in AsyncTask.
    - onPreExecute( )
    - doInBackground( )
    - onPostExecute( )
    - onProgressUpdate( )

- we can update the UI of MainThread only inside the onPostExecute( ) method.

- use the following code in Activity / Fragment to execute the AsyncTask.

```
var task = AsyncTaskName( )
task.execute( )
```

- we can’t execute multiple AsyncTask parlally, we can execute multiple async task’s sequentially.

- when we turn the device screen orientation , the Activity will re initialize and AsyncTask also will reinitialise.

Service :

- A long running background process with out any user interaction is called as Service.
- to create a service, create a class as a child of android.app.Service class.
- It is an abstract class having an abstract function called onBind( ).
- following are the major methods in Service class.
  - onCreate( )
  - onStartCommand( )
  - onDestroy( )
- when we start a service, if the service is not available it will invoke onCreate( ) and onStartCommand( ) methods, if service is already available it will execute only onStartCommand( ) method.
- when we stop service it will invoke onDestroy( ) method.
- Service doesn’t contain any UI, we will manage service from Activity using Intent.

```
var i : Intent =
Intent(context,ServiceName::class.java )
startService(i)
stopService(i)
```

- every service class has to configure in Manifest.xml with the following tag, inside <application> tag.

```
<service android:name="package_name.ServiceName"/>
```

- Service will run on application MainThread.
- we can't communicate with Activity UI from Service.

Intent Service :

- Intent Service is a child of Service which will stop by itself by calling selfStop( ) method.
- we can communicate with the Activity UI from Intent Service by using Broadcast Receiver.
- Intent Service will run on a separate Thread.
- To create Intent Service, create a class as a child of android.app.IntentService.
- It is an abstarct class having an abstarct function called onHandleIntent(i:Intent) method.
- same like service, intent service can start from Activity by using startService() method.
- same like service, IntentService has to be configure in Manifest.xml with <service> tag.
- IntentService is a replacement for AsyncTask.

Broadcast Receiver :

- Broadcast receivers are registered for system events.

eg: head set plugin, power connected , power disconnected, screen on/off, making/receiving call, battery low/full….

- By using broadcast receivers we can get the announcement for builtin events and we can

fire our own broadcast events using sendBroadcast(intent_obj) method.

- to create a broadcast receiver create a class as a child of android.content.BroadcastReceiver.

- It is an abstract class having an abstract function called onReceive(context,Intent)

- From Activity/Fragment for which events you want to get the broadcast announcements , configure the events using IntentFilter (group of intents is called as Intent Filter).

```
var filter:IntentFilter = IntentFilter( )
filter.addAction(Intent.ACTION_NAME1)
filter.addAction(Intent.ACTION_NAME2)
registerReceiver(broadcast_receiver_obj, intent_filter_obj)
```

- If any one of the configured event is happened it will invoke onReceive( ) method in Broadcast Receiver class.

Builtin Services :

- Location Service
- Notification Service
- Sensor Service
- Wifi Service
- Bluetooth Service
- Vibrator
- Telephony Service
- Connectivity Service

Location Service :

- getSystemService(Context.SERVICE_NAME) method is used to get the builtin services.

- by using location service we can capture the device current location (latitude , longitude).

- there are 2 location providers
- GPS
- Network
- getSystemService(Context.LOCATION_SERVICE) method is used to get the builtin location service, application framework is providing a class called LocationManager to manage location service.

```
val lManager = getSystemService(
      Context.LOCATION_SERIVCE) as
LocationManager
```

- use the following method to get the current location.

```
lManager.requestLocationUpdates(

LocationManager.Provider_name,

minimum_time_interval (milli_seconds),

minimum_distance (meters),

object_of_location_listener)
```

- location listener contains an abstract function called

```
fun
onLocationChanged(loc:Location) {

val lati : Double = loc.getLatitude( )

val longi : Double = loc.getLongitude( )
                                        }
```

permissions : [runtime permission is required…]

ACCESS_COARSE_LOCATION

ACCESS_FINE_LOCATION

- use the following method to stop getting the location updates (write this method inside Location Listener >> onLocationChanged( ) method).

```
lManager.removeUpdates(this)   // this -> location listener object
```

Notification Service :

- by using notification service we can display notification on notification / status bar.
- In Android there are 2 types of notifications.
  - Local notification (missed call , new message..)
  - Push notification (promotions messages …. which will send from Server)

- getSystemService(Context.NOTIFICATION_SERVICE) method is used to get the builtin notification service, application framework is providing a class called NotificationManager to manage notification service.

```
val nManager : NotificationManager =
getSystemService(Context.NOTIFICATION_SERVICE)
as NotificationManager
```

- use the following code to display notification on Notification bar.

Sensor Service :

- In Android there are types of Sensors

-

-

-

-

-

-

getSystemService(Context.SENSOR_SERVICE) method is used to get the builtin sensor service, application

framework is providing a class called SensorManager to manage sensor service.

```
var sManager = getSystemService(
   Context.SENSOR_SERVICE) as SensorManager
```

- use the following code to get the Sensor object.

```
var s:Sensor  = sManager.getDefaultSensor(
```

- use the following method to get the sensor updates.

```
sManager.registerListener(object:SensorEventListe
ner{
                                        fun
sensorChanged(….){  }
                                        fun
onAccuracyChanged(..){ }

},sensor_object,Sensor.SPEED)
```

add the following features in Manifest.xml :

```
<uses-feature

    android:name="android.hardware.sensor.proximity"
                android:required="true" />
<uses-feature

    android:name="android.hardware.sensor.gyroscope"
                android:required="true" />
```

Vibrator :

```
var vib:Vibrator  = getSystemService(
    Context.VIBRATOR_SERVICE)  as Vibrator
vib.vibrate(milli_seconds)
```

permission :

```
<uses-permission android:name=

“android.permission.VIBRATE” />
```

Telephony & Connectivity Service :

```
var tManager:TelephonyManager =
getSystemService(Context.TELEPHONY_SERVICE) as
                                    TelephonyManager

var cManager : ConnectivityManager =
getSystemService(CONNECTIVITY_SERVICE) as
                                    ConnectivityManager
```

permissions :

```
                            NETWORK_STATE (for
connectivity service)
                            READ_PHONE_STATE (for tel
service)
```

Wifi Service :

```
var wManager:WifiManager  =
              getApplicationContext().
           getSystemService(Context.WIFI_SERVICE) as
              WifiManager
```

- wManager.getWifiState( ) method is used to get the wifi state, return type of this method is Int (0 - disabled , 1 - disabling, 2 - enabling , 3 - enabled)

- wManager.setWifiEnabled(boolean) method is used to change the wifi state.

- wManager.getResults( ) method is used to get the list of wifi devices, return type of this function is List<ScanResult>.

```
                        var list =
wManager.getScanResults( )
```

```
                    for(device in list) {

    device.SSID             // Network name

device.frequency   // signal strength
                    }
```

- wManager.getConfiguredNetworks( ) method is used to get the list of paired wifi devices. return type of this function is List<WifiConfiguration>.

```
    var list = wManager.getConfiguredNetworks( )
                    for(device in list) {

    device.SSID             // Network name

device.status          // paired or not
                    }
```

permissions :

ACCESS_WIFI_STATE

CHANGE_WIFI_STATE

ACCESS_COARSE_LOCATION (runtime)

Bluetooth :

- In android there are different ways to manage bluetooth service, one of the best approach is manage bluetooth service using Bluetooth Adapter.

```
var bAdapter = BluetoothAdapter.getDefault( )
```

- bAdapter.isEanbled( ) method is used to get the bluetooth state, return type of this method is boolean.

- bAdapter.enable( ) / bAdapter.disable( ) methods is used to modify the bluetooth state.

- to get the list of available Bluetooth devices we have to configure a broadcast receiver.

bAdapter.startDiscovery( )

```
var filter : IntentFilter = IntentFilter( )
filter.addAction(BluetoothDevice.ACTION_FOUND)
registerReceiver(obj_of_broadcast_receiver , filter)
```

- when a bluetooth device is found it will invoke onReceive() method of BroadcastReceiver class.

```
class MyReceiver : BroadcastReceiver()
    {
override fun onReceive(c:Context, i:Intent) {
var device : BluetoothDevice =

    i.getParcableExtra(BluetoothDevice.EXTRA_DEVICE)
  device.name ;   device.address;
    }
}
```

permissions :

BLUETOOTH

BLUETOOTH_ADMIN

ACCESS_COARSE_LOCATION

Content Provider :

Content Provider is used to share the data between multiple applications.

[In android one of the security feature is which application is created the

DB only that application can access the data , to access the data from other applications application has to provide content provider.

In android following builtin applications are providing content provider

contacts , callog , settings, media, calendar, messages, etc.]

Steps to work with content provider :

- to get any content provider , we required an object for content resolver.

```
var cResolver:ContentResolver  = getContentResolver( )
```

- use the following method to access the data from content provider.

```
var c:Cursor  = resolver.query( CP_URI, projection,
```

- if the data is available in a cursor we use cursor adapter for presenting the data.

https://goo.gl/mCgjuN

Advanced Android :

- Web Services
- JSON
  - GSON
  - Rest API calls using Retrofit

Steps to work with REST API using Retrofit :

1. create a project, add the following libraries as a dependency libraries.
   (right click on app >> module settings >> dependencies >>

+ >>
library dependency)

1. GSON
('com.google.code.gson:gson:2.8.5’)
2. Retrofit
('com.squareup.retrofit2:retrofit:2.5.0’)
3. Retrofit-Gson converter

('com.squareup.retrofit2:converter-gson:2.5.0’)

2. create the equalant bean/pojo classes (in kotlin these classes are called as data classes) based on JSON response.

Article.kt :

```
data class Article(var title:String,var
description:String,
var url:String,var
urlToImage:String)
```

Articles.kt :

```
data class Atricles(var
articles:MutableList<Article>)
```

3. create an interface , that interface should contain an abstract function , the function return type should be Call<MainDataClass>

4. on top of abstract function specify the request type and configure the sub-url.

5. Initialize the retrofit object.

```
var r:Retrofit = Retrofit.Builder().

addConverterFactory(GsonConverterFactory.create()).
baseUrl("https://newsapi.org/").build()
```

base_url

6. create an implementation for interface using Retrofit.

```
var api =
r.create(NewsAPI::class.java)
```

7. call the abstract function from an interface implementation object.

```
var call:Call<Atricles> = api.getNews()
```

8. make a rest api web-service call using the following method.

```
call.enqueue(object:Callback<Atricles>{
  override fun onResponse(call: Call<Atricles>,

}
  override fun onFailure(call: Call<Atricles>, t:
Throwable) {

  }
})
```

9. configure the internet permission in Manifest.xml

Image Loading Libraries :

Picaso
Glide ('com.github.bumptech.glide:glide:4.8.0')

```
Glide.with(this@MainActivity).
  load("url_address").
  into(R.id.UI_component)
```

Google Maps :

1. create a project, add the following library as a dependency library.

play-service-maps
('com.google.android.gms:play-services-maps:16.0.0')

2. In activity xml file, create a fragment UI component with the following name.

```
<fragment
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:id="@+id/frag1"
   android:name=

"com.google.android.gms.maps.SupportMapFragment"/>
```

3. Get the SupportMapFragment from XML to kotlin.

```
var sFrag :SupportMapFragment =
 supportFragmentManager.findFragmentById(R.id.frag1) as
        SupportMapFragment
```

4. Get the GoogleMap object from

```
sFrag.getMapAsync(object : OnMapReadyCallback {
    override fun onMapReady(gMap: GoogleMap?) {

    }
})
```

5. to work with any Google-API we required an API key, go through the following URL to get an API key.

http://code.google.com/apis/console

AIzaSyDn0HBaBIxj7LJXdXNYwphWIPWbfwunY80

6. configure the API key in Manifest.xml with the following tag inside <application> tag.

```
<meta-data

android:name="com.google.android.geo.API_KEY"
        android:value="YOUR_API_KEY" />
```

7. use the following code to display a marker on GoogleMap.

```
var options = MarkerOptions( )
options.position(LatLng(latitude,longitude))
gMap.addMarker(options)
```

Google Places :

- create a project, add the following libraries as a dependency libraries.

- GSON
- Retrofit
- Retrofit-GSON converter
- Maps
- Place Picker ('com.google.android.gms:play-services-places:16.0.0')

Play store deployment Process :

1. generate
   - keystore
   - signed apk / app bundle
2. http://play.google.com/apps/publish

Firebase :

- Firebase will provide backend services to our applications.

Material Design :

- RecyclerView & CardView
- Floating Action Button (FAB)
- SnackBar
- View Pager / Tabbed Activity
- Navigation Drawer
- Toolbar & Menu

RecyclerView & CardView :

limitations of Custom Adapter :
item.
- because of this performance of custom adapter is slow.
- we have to create 3 separate components Listview for vertical direction, gallery for horizontal , grid view for grid format.
- we can’t use a single UI component for vertical , horizontal and grid format.

Above limitations are overcome in RecyclerView.
- we can present the UI with good look and feel by using RecyclerView and CardView.

Steps to work with RecyclerView and CardView :

1. create a project, add the following libraries as a dependency libraries.

RecyclerView 'com.android.support:recyclerview-v7:28.0.0'
CardView 'com.android.support:cardview-v7:28.0.0'

2. In activity XML file create RecyclerView UI component.

<RecyclerView

android:id=“@+id/rview”

…… />

3. get the RecyclerView component in kotlin file and specify the layout type.

```
var lManager =
LinearLayoutManager(this@MainActivity,
    LinearLayoutManager.VERTICAL,false)
    var gManager =
GridLayoutManager(this@MainActivity,2)
    rview.layoutManager = lManager // (or) gManager
```

4. create an xml file for individual item in which format you want to present the data , for better look and feel specify CardView as a root UI group.

5. create a class to hold the UI components of individual XML file, create a class as a child of RecyclerView.ViewHolder class.

6. create an adapter to present the data on Recyclerview, to create RecyclerView adapter create a class as a child of RecyclerView.Adapter<ViewHolder_object>.

Floating Action Button & SnackBar :

- to add FAB & snack bar add the following library.

Design Library ('com.android.support:design:28.0.0')

- to use FAB the root UI group must be Coordinator Layout.

Toolbar & Menu :

- to display menu options, configure the menu options in the following xml.

res >> menu >> file_name.xml

```
<menu>
```

<item title="Menu Title"

icon="@drawable/img_name"/>
<item title="Menu Title"

icon="@drawable/img_name"/> <

<menu>

</menu>
</item>
</menu> - override the following method in Activity to display menu options.

fun onCreateOptionsMenu(menu:Menu) {

getMenuInflater().inflate(R.menu.xml_file, menu)
}

- if we select any one of the menu option it will invoke the following method.

fun onOptionsItemSelected(item:MenuItem) {
}

- configure following attribute to display menu item in toolbar.

<item
……..

showAsAction="always|ifroom" />

Toolbar :

- if we create an Activity be default a tool bar will add, configure the following attribute in styles.xml if you don't want toolbar.

<style name="AppTheme"
parent="Theme.AppCompat.NoActionBar">

- use the following method in Activity to change toolbar title.

supportActionBar?.setTitle("Welcome 2 NIT...")

- use the following code to customize toolbar.

- configure <Toolbar> component in Activity XML.

- configure the following attributes in styles. xml.

<item name="windowActionBar">false</item>

<item name="windowNoTitle">true</item>

- set the custom toolbar to activity using following method.

setSupportActionBar(tBar)

Navigation Drawer :

View Pager / SwipeView / Tabbed Activity :

Flutter :

-> X-Platform technology given by google.