**SYNTAX OF ANDROID STUDIO**

1 . Toasting a message

Syntax:-

* Toast.makeText(this.applicationContex,”Hello Programers”,Toast.LENGTH\_SHORT).show()

If we want that message to show for a *second* you must use this LENGTH\_SHORT if you want it for few second you must use this LENGTH\_LONG

2. setOnClickLisner () and setOnLongClickLisner()

Syntax:- setOnClickLisner

* Btn.setOnClickLIsner{
* // here that code will execute if we press that view
* }

Syntax:- setOnLongClickLisner()

* Btn.setOnLongClickLisner{
* // put your code which will execute if you press that view for a long time
* true // it is compulsory for longLongClickLisner
* }

3.Animation for splash screen in kotlin,XMl and android studio

\*right click resource file create a directory called anim , Right click anim directory and create a Animation resource file (if the name of the directory is anim you will get AnimationResource file option when you right click that anim Directory press new) if you are in a different directory you should set the XML tag to “set”

XML : Code

<?xml version="1.0" encoding="utf-8"?>  
<set xmlns:android="http://schemas.android.com/apk/res/android">  
 <translate  
 android:duration="2000"  
 android:fromYDelta="-50%p"  
 android:fromXDelta="0%p"  
 android:interpolator="@android:interpolator/accelerate\_decelerate"  
 android:toYDelta="0%p"  
 android:toXDelta="0%p"/>  
  
</set>

If you run your code now it will not execute this animation

So you should implement animation in MainActivity.kt

Kotlin Code:

class Splash\_Screen : AppCompatActivity() {  
  
 private lateinit var animationForTitle : Animation  
 private lateinit var animatedView : TextView  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_splash\_\_screen*)  
  
 // hinding status bar and Action bar  
  
 *supportActionBar*?.hide()  
 *window*.setFlags(WindowManager.LayoutParams.*FLAG\_FULLSCREEN*,  
 WindowManager.LayoutParams.*FLAG\_FULLSCREEN*)  
  
 // animating here  
  
 animatedView = findViewById(R.id.*TV\_splash\_screen*)  
 animationForTitle = AnimationUtils.loadAnimation(this,R.anim.*anim\_top*)  
 animatedView.*animation* = animationForTitle  
  
 // using handler to set timer and intent to go to home fragmet.kt  
 Handler().postDelayed(**{** Toast.makeText(this,"Welcome To Fav Dish",Toast.*LENGTH\_SHORT*).show()  
 val intenting = Intent(this,MainActivity :: class.*java*)  
 startActivity(intenting)  
 finish()  
  
  
 **}**,4000)  
  
 }  
}

4.Customizing tool Bar (Action Bar)

\* Create a menu Directory in resource file that tag should be <menu>

In it you should use

XML Menu Code : -

<?xml version="1.0" encoding="utf-8"?>  
<menu xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto">  
  
 <item  
 android:id="@+id/add\_Plus"  
 android:icon="@drawable/ic\_baseline\_add\_24"  
 android:title="addingButton"  
 app:showAsAction="ifRoom"/>

</menu>

If you want to axcess the id of the view in the MainActivity.kt you should use this code

Kotlin Code : -

override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)

setAsOptionMenu(true) // it is included if you are using it in a fragment

}

override fun onCreateOptionsMenu(menu: Menu, inflater: MenuInflater) {  
 super.onCreateOptionsMenu(menu, inflater) // inflating a menu  
 inflater.inflate(R.menu.*menu\_actionbar*, menu)  
  
}  
  
override fun onOptionsItemSelected(item: MenuItem): Boolean {  
 if(item.itemId == R.id.app\_plus){  
 val intentinghe = Intent(this.*context*,Update\_Dish\_Activity :: class.*java*)  
 startActivity(intentinghe)

}  
  
 return true  
}

5. Rounded corners

\* Right Click Drawable folder,Click new ,select Drawable Resource file in the message popup select –

Root element As a shape

Xml Code : - in drawables

<?xml version="1.0" encoding="utf-8"?>  
<shape xmlns:android="http://schemas.android.com/apk/res/android">  
  
 <!-- it is called as solid it is used to set the colors -->  
  
 <solid android:color="@color/white"/>  
  
 <corners android:radius="25dp"/> <!-- Radius to set corners for all corners-->  
  
 <!-- it is to set radius for each corners -->  
  
 <corners android:bottomLeftRadius="4dp"/>  
 <corners android:bottomRightRadius="4dp"/>  
 <corners android:topLeftRadius="4dp"/>  
 <corners android:topRightRadius="4dp"/>  
  
</shape>

XML Code in Activity main Activity.xml : -

Down-

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app=<http://schemas.android.com/apk/res-auto>  
 xmlns:tools=<http://schemas.android.com/tools>  
 android:layout\_width="match\_parent"

// add this line of code in main the activity Xml

android:background="@drawable/rounded\_corners" // it is the line

android:layout\_height="match\_parent">

6. usefull module in kotlin and android and Xml file

\* this UI will be same in all the devices all the pixel sizes

\* Go to gradle build (.app) file ,

\* Gradle code

Dependencies {

.......

/\* it is the module for intuit \*/  
implementation 'com.intuit.sdp:sdp-android:1.0.6'

.......

}

\* just put it here and it will work properly in all the devices

7. Ask Run time Permission in all the devices

\* I am using the Dexter module it is the easy, to ask run time Permission in kotlin android development

\* in AndroidManifest in user Permission

AndroidManifest Xml Code :-

In Manifest Xml Tag ->

<uses-permission android:name="android.permission.CAMERA"></uses-permission>  
<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE"></uses-permission>

* It is to ask Permission in app
* In Kotlin File ->

Dexter.withContext(this).withPermission(  
 android.Manifest.permission.*READ\_EXTERNAL\_STORAGE*).withListener(object : PermissionListener {  
 override fun onPermissionGranted(p0: PermissionGrantedResponse?) {

// Code which should be executer when the use axcepted

// The Permission

// here you should put intent open the camera or the

// permission which is asked   
 }  
  
 override fun onPermissionDenied(p0: PermissionDeniedResponse?) {  
 // execute the code when the user Denied the permission

// we can express it with the Alert Dialog or Dialog

// alert Dialog will be teached soon   
 }  
  
 override fun onPermissionRationaleShouldBeShown(p0: PermissionRequest?, token: PermissionToken?) {  
   
 }  
  
})

if there are multiple permission to ask the user use object : MultiplePermissionLisner in the place of Permission Lisner

8 . Alert Dialog