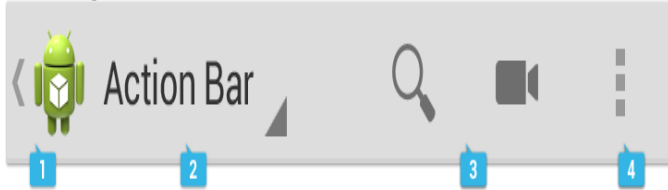**What is ActionBar? Explain with example of DrawerLayout and ActionBarDrawerToggle.**

- Action Bar is one of the important part of any application, whether it's a web application or a mobile app.
- Android ActionBar is a menu bar that runs across the top of the activity screen in android.
- In general an ActionBar consists of the following four components:

1. **App Icon**: App branding logo or icon will be displayed here.
2. **View Control**: A dedicated space to display Application title. Also provides option to switch between views by adding spinner or tabbed navigation.
3. **Action Buttons**: Some important actions of the app can be added here.
4. **Action Overflow**: All unimportant action will be shown as a menu.
- An ActionBar have following attributes:
    - android:title : Attribute value contains the title of the menu item
    - android:icon : Attribute references an icon in the drawable directories
    - android:showAsAction : This attribute indicates how the given item should be portrayed in the action bar.

Example of ActionBar with Drawer Layout:-

activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout 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"
    android:layout_height="match_parent"
    android:id="@+id/drawer"
    tools:context=".MainActivity">

    <FrameLayout
        android:id="@+id/frame"
        android:layout_width="match_parent"
        android:layout_height="match_parent"></FrameLayout>

    <ListView
        android:id="@+id/listview"
        android:layout_gravity = "start"
        android:layout_width="150dp"
        android:background="#FFF"
        android:layout_height="match_parent"></ListView>

</androidx.drawerlayout.widget.DrawerLayout>
```

MainActivity.kt

```kotlin
class MainActivity : AppCompatActivity() {
    lateinit var toggle : ActionBarDrawerToggle
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        supportFragmentManager
            .beginTransaction()
            .replace(R.id.frame, Unit1())
            .commit()

        var drawer = findViewById<DrawerLayout>(R.id.drawer)
        supportActionBar?.setDisplayHomeAsUpEnabled(true)
        toggle = ActionBarDrawerToggle( activity: this, drawer, "Open Drawer", "Close Drawer")
        drawer.setDrawerListener(toggle)
        toggle.syncState()

        var lv = findViewById<ListView>(R.id.listview)
        var units = arrayOf("Unit-1","Unit-2","Unit-3","Unit-4","Unit-5")
        var adapter = ArrayAdapter( context: this,
            android.R.layout.simple_dropdown_item_1line, units)
        lv.adapter = adapter

        listview.setOnItemClickListener { adapterView, view, i, l ->
            drawer.closeDrawers()
            when(i){
                0-> {supportFragmentManager.
                    beginTransaction().
                    replace(R.id.frame,Unit1()).
                    commit()
                }
                1-> {supportFragmentManager.
                    beginTransaction().
                    replace(R.id.frame,Unit2()).
                    commit()
                }
                2-> {supportFragmentManager.beginTransaction()
                    .replace(R.id.frame,Unit3()).
                    commit()
                }
                3-> {supportFragmentManager.beginTransaction()
                    .replace(R.id.frame,Unit4()).
                    commit()
                }
                4-> {supportFragmentManager.beginTransaction()
                    .replace(R.id.frame,Unit5()).
                    commit()
                }
            }
        }
    }
}
```
.

Fragment : Unit-1

```xml
<androidx.constraintlayout.widget.ConstraintLayout xmlns:androi
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Unit1">
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:text="Kotlin Basics"
        android:textColor="@color/colorPrimaryDark"
        android:textSize="30sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="70dp"
        android:text="Click Me"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Unit1.kt

```kotlin
class Unit1 : Fragment() {

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        var v : View! = inflater.inflate(R.layout.fragment_unit1,
            container, attachToRoot: false)
        v.button.setOnClickListener { it: View!
            Toast.makeText(v.context,
                text: "Android is really interesting",
                Toast.LENGTH_LONG).show()
        }
        return v
    }

}
```

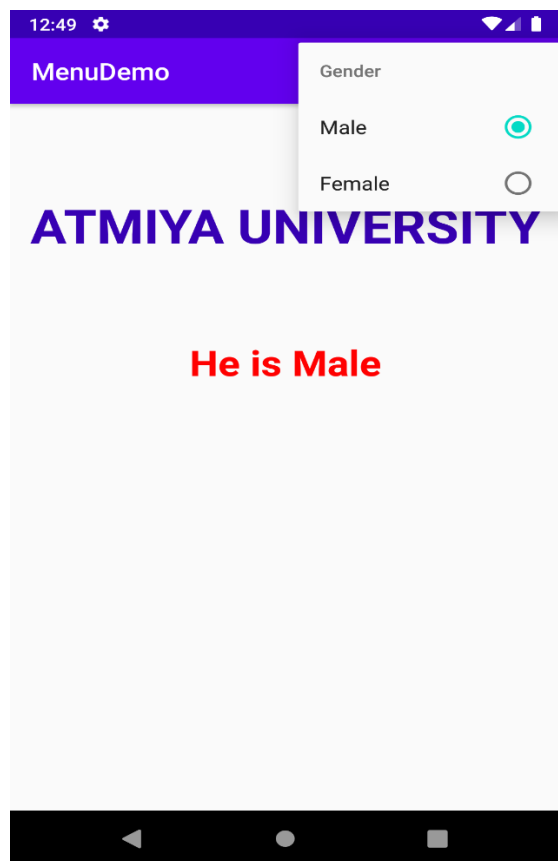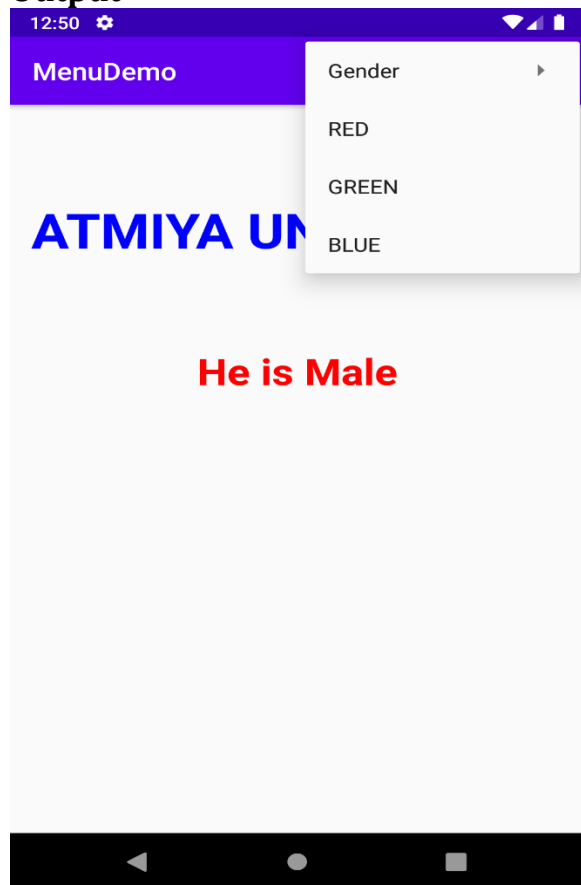**Same as above take four other fragments for other 4 units and four other kotlin file.**

Output
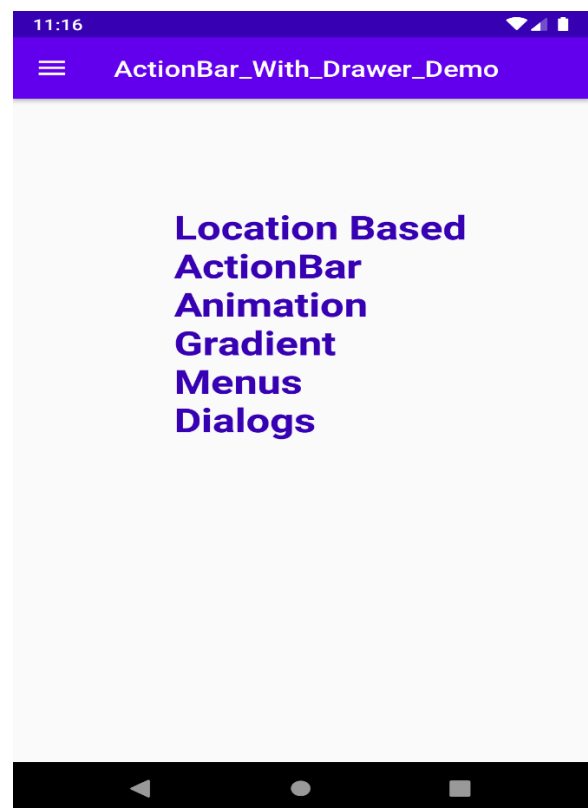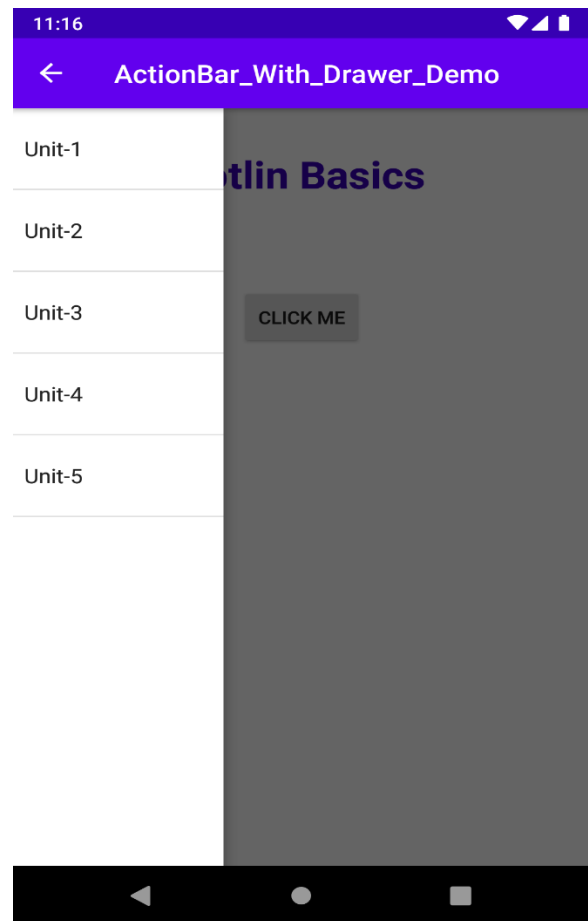
Example : ActionBar with OptionMenu

```kotlin
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
    override fun onCreateOptionsMenu(menu: Menu?): Boolean {
        menu?.add(1, 101, 1, "RED")
        menu?.add(1, 102, 1, "GREEN")
        menu?.add(1, 103, 1, "BLUE")
        var smenu : SubMenu? = menu?.addSubMenu("Gender")
        smenu?.add(2, 11, 1, "Male")?.setChecked(true)
        smenu?.add(2, 12, 1, "Female")?.setChecked(false)
        smenu?.setGroupCheckable(2, true, true)
        return super.onCreateOptionsMenu(menu)
    }
    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        when (item.itemId) {
            101 -> {
                textView.setTextColor(Color.RED)
            }
            102 -> {
                textView.setTextColor(Color.GREEN)
            }
            103 -> {
                textView.setTextColor(Color.BLUE)
            }
            11 -> {
                textView2.setText("He is Male")
            }

            12 -> {
                textView2.setText("She is Female")
            }
        }
        return super.onOptionsItemSelected(item)
    }
}
```

What is the use of Dialog? Explain key methods required for managing the dialog lifecycle.

Dialog

- Android supports following seven types of dialog
  - Dialog, AlertDialog, CharacterPickerDialog, DatePickerDialog, ProgressDialog, TimePickerDialog and custom dialog.
- Dialog may be launched once / repeatedly.

Key Methods to manage dialog lifecycle

- **showDialog()**
  - used to display a dialog.
- **dismissDialog ()**
  - used to stop showing a dialog.
  - It is kept in Activity's dialog pool,
  - If showDialog() is used after it cached version is displayed.
- **removeDialog()**
  - used to remove dialog from Activity's Object Dialog pool.
  - Dialog is not kept for future use.
  - If showDialog() is used after it then dialog must be recreated.

Adding Dialog to an Activity

- Define unique id for the dialog (within activity)
- Implement **onCreateDialog()** method → To return a dialog when unique id is passed.
- Implement **onPreapraeDialog()** method → To initialize dialog appropriately.
- Launch the dialog using **showDialog()** method. → Pass unique id.

Defining a dialog
- **onCreateDialog()** method is used to define dialog.
- Dialog must be defined in advance.
- Each dialog will have special unique identifier (integer)
- It is called only once for initial dialog creation.

Initializing a dialog

- To re-initialize dialog when dialog is shown from activity's dialog pool.
- **onPreapreDialog()** method is used.
- It is called each time **showDialog()** method is called.
- it is used to modify the dialog before it is again shown to the user.

Launching a dialog

- **showDialog()** method is used to show a dialog.
- Unique if of dialog is passed as a argument to display appropriate dialog.

Dismiss dialog

- Most dialogs have automatic dismissial circumstances.
- To forcefully dismiss a dialog **dissmissDialog()** method is used and unique id is passed as argument.

Removing dialog

- dismissDialog() doesn't destroy a dialog, dialog can be shown again that is it's cached content is displayed.

To force activity to remove dialog from Activity pool **removeDialog()** method with unique id of dialog as argument is used.

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Alert Dialog

```
var ad = AlertDialog.Builder( context: this)
ad.setMessage("Are you enjoying??")
ad.setTitle("Enjoying")
ad.setPositiveButton( text: "Yes",
    DialogInterface.OnClickListener { dialogInterface, i ->
        Toast.makeText(applicationContext,
            text: "Thank You for your feedback",
            Toast.LENGTH_LONG).show()
})
ad.setNegativeButton( text: "No",
    DialogInterface.OnClickListener { dialogInterface, i ->
    Toast.makeText(applicationContext,
        text: "Oh God! I will definitely teach you again",
        Toast.LENGTH_LONG).show()
})
ad.show()
```

Progress Dialog

```
var pd = ProgressDialog( context: this)
pd.setTitle("Downloading")
pd.setMessage("File Downloading")
pd.max = 100
pd.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL)
pd.show()

Thread(Runnable {
    var count=0
        while(count<=100) {
            try {
                pd.progress = count
                count += 10
                Thread.sleep( millis: 1000)
            } catch (i: InterruptedException) {

            }
        }
    pd.dismiss()
}).start()
```

DatePicker Dialog

```
var c : Calendar = Calendar.getInstance()
var dp = DatePickerDialog( context: this,
    DatePickerDialog.OnDateSetListener { dp, yy,mm, dd ->
                dt = "$dd/${mm+1}/$yy"
                editText.setText(dt)
    }, c.get(Calendar.YEAR),
        c.get(Calendar.MONTH),
        c.get(Calendar.DAY_OF_MONTH)
)
dp.show()
```

TimePicker Dialog

```
var c : Calendar = Calendar.getInstance()
var tp = TimePickerDialog( context: this,
    TimePickerDialog.OnTimeSetListener { tp, hh, mi ->
        dt = dt + " $hh : $mi"
        editText.setText(dt)
}, c.get(Calendar.HOUR),
    c.get(Calendar.MINUTE),
    is24HourView: false)
tp.show()
```

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Explain various methods of Canvas and Paint class with example.

Canvas Class
- The Canvas class holds the "draw" calls.
- To draw something, you need 4 basic components:
    1. A Bitmap to hold the pixels,
    2. A Canvas to host the draw calls (writing into the bitmap),
    3. A drawing primitive (e.g. Rect, Path, text, Bitmap), and
    4. A paint (to describe the colors and styles for the drawing).

Methods of Canvas

- **drawColor(int color)**
  Fill the entire canvas' bitmap (restricted to the current clip) with the specified color,

- **drawBitmap(Bitmap bitmap, float left, float top, Paint paint)**
  Draw the specified bitmap, with its top/left corner at (x,y), using the specified paint, transformed by the current matrix.

- **drawText(String text, float x, float y, Paint paint)**
  Draw the text, with origin at (x,y), using the specified paint.

- **getHeight()**
  Returns the height of the current drawing layer.

- **getWidth()**
  Returns the width of the current drawing layer.

- **drawCircle(float cx, float cy, float radius, Paint p)**
  Draw the specified circle using the specified paint
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Paint Class
- The Paint class holds the style and color information about how to draw geometries, text and bitmaps.

Constructor
- **Paint()**
  Create a new paint with default settings.
- **Paint(int flags)**
 Create a new paint with the specified flags.

Methods of Paint
- **setColor(int color)** : It is used to set paints color.
- **setTextSize(float size):** Set the paint's text size.
- **setTypeface(Typeface t)**: Set or clear the typeface object.
  setTypeface(null) will assign default font settings to paint.
- **setStyle(Paint.Style):** Paint style controls how an object is filled with color.
  For example, the following code instantiates a Paint object and sets the Style to STROKE, which signifies that the object should be painted as a line drawing and not filled (the default):
    o   Paint p = new Paint();
    o   p.setStyle(Paint.Style.STROKE);

- **setAntiAlias(boolean aa)**
- Antialiasing makes many graphics—whether they are shapes or typefaces—look smoother on the screen.
- This property is set within the Paint of an object. or by using setFlags(), setting or clearing the ANTI_ALIAS_FLAG bit AntiAliasing smooths out the edges of what is being drawn, but is has no impact on the interior of the shape.

  Paint p = new Paint(Paint.ANTI_ALIAS_FLAG)
  p.setAntiAlias(false)

- **setShader()**
  We can set the paint gradient using setShader() method.
_____
_____
_____
_____
_____
_____
_____
_____
_____

Working with Paint Gradients

- You can create a gradient of colors using one of the gradient subclasses.
- The different gradient classes, including
    - LinearGradient,
    - RadialGradient, and
    - SweepGradient

  are available under the superclass android.graphics. Shader.
- All gradients need at least two colors—a start color and an end color—but might contain any number of colors in an array.
- The different types of gradients are differentiated by the direction in which the gradient "flows." Gradients can be set to mirror and repeat as necessary.
- You can set the Paint gradient using the setShader() method.
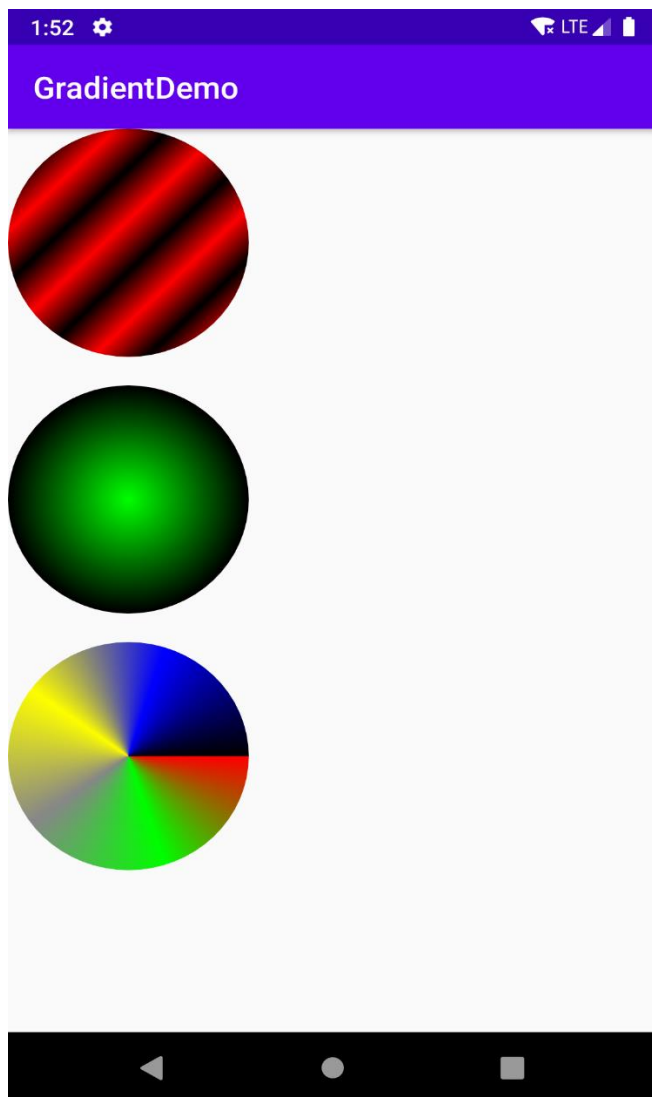
Working with Linear Gradients

- A linear gradient is one that changes colors along a single straight line.
- You can achieve this by creating a LinearGradient and setting the Paint method
- setShader() before drawing on a Canvas, as follows:

var p = Paint(Paint.*ANTI_ALIAS_FLAG*)
var lg = LinearGradient(0f,0f,45f,45f,
Color.*RED*,Color.*BLACK*,Shader.TileMode.**MIRROR**)
p.*shader* = lg
canvas?.drawCircle(200f,200f,200f,**p**)

Working with Radial Gradients

- A radial gradient is one that changes colors starting at a single point and radiating outward in a circle.
- The second circle in diagram is a radial gradient between green and black.
- You can achieve this by creating a RadialGradient and setting the Paint method.

var p = Paint(Paint.*ANTI_ALIAS_FLAG*)
var rg = RadialGradient(200f, 650f, 200f, Color.*GREEN*,
Color.*BLACK*, Shader.TileMode.**MIRROR**)
p.*shader* = rg
canvas?.drawCircle(200f,650f,200f,**p**)

Working with Sweep Gradients

- A sweep gradient is one that changes colors using slices of a pie.
- This type of gradient is often used for a color *chooser*. The last circle at the bottom of diagram is a sweep gradient between red, yellow, green, blue, and magenta.

var p = Paint(Paint.*ANTI_ALIAS_FLAG*)
var sg = SweepGradient(200f, 1100f,
intArrayOf (Color.*RED*, Color.*GREEN*, Color.*GRAY*,
Color.*YELLOW*, Color.*BLUE*, Color.*BLACK*), **null**)
p.*shader* = sg
canvas?.drawCircle(200f, 1100f, 200f,**p**)

_____
_____
_____
_____
_____
_____

Gradient using XML

- Create the following shape drawable file named linear_gradient.xml, radial_gradient and linear_gradient in drawable folder

linear_gradient.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns: android =
"http://schemas.android.com/apk/res/android"
  android:shape="oval">
  <gradient
    android:type="linear"
    android:angle="45"
    android:startColor="#F00"
    android:centerColor="#000"
    android:endColor="#0f0"
    ></gradient>
</shape>
```

radial_gradient.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android =
"http://schemas.android.com/apk/res/android"
  android:shape="oval">

  <gradient
    android:type="radial"
    android:gradientRadius="200"
    android:startColor="#F00"
    android:endColor="#0F0"
    android:centerColor="#000"
    ></gradient>
</shape>
```

sweep_gradient.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android=
"http://schemas.android.com/apk/res/android"
  android:shape="oval">

  <gradient
    android:type="sweep"
    android:startColor="#F00"
    android:centerColor="#0F0"
    android:endColor="#00F"
    ></gradient>
</shape>
```

activity_main.xml

```xml
<ImageView
  android:id="@+id/imageView"
  android:layout_width="200dp"
  android:layout_height="0dp"
  app:srcCompat="@drawable/linear_gradient" />

<ImageView
  android:id="@+id/imageView2"
  android:layout_width="200dp"
  android:layout_height="200dp"
  app:srcCompat="@drawable/radial_gradient" />

<ImageView
  android:id="@+id/imageView3"
  android:layout_width="0dp"
  android:layout_height="200dp"
  app:srcCompat="@drawable/sweep_gradient" />
```

Output
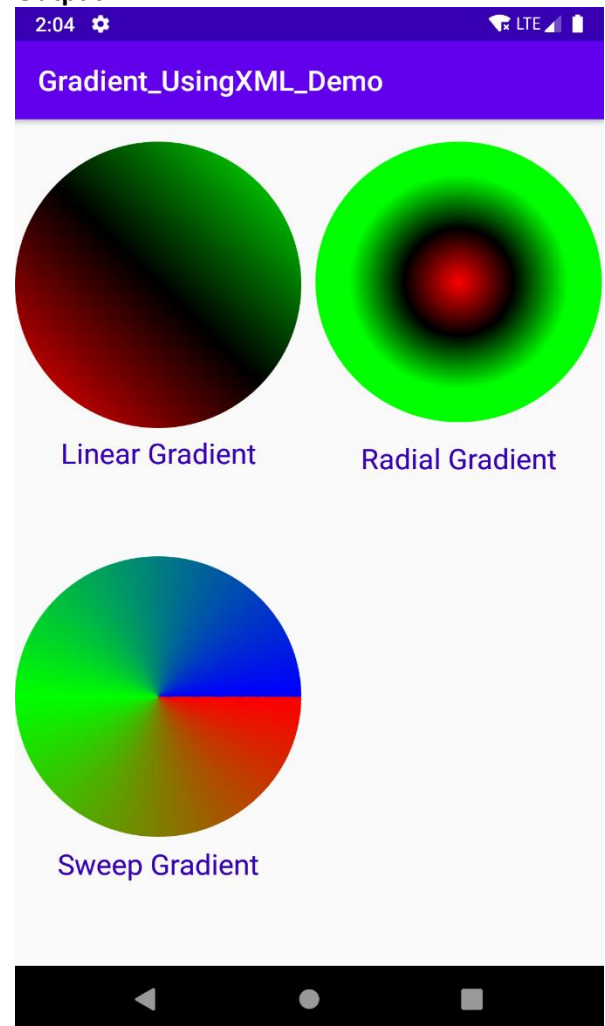
What is Frame-by-Frame Animation?Explain in detail?
OR
What is Tweened Animation? Explain in detail.
OR
Differentiate Frame-by-frame animation and Tweened animation.

Frame-by-Frame Animation
* It is like a digital flipbook
* Series of similar images with a subtle (slight) differences display on the screen in a sequence.
* When these images are displayed quickly it gives illusion of a movement (animated). This technique is called Frame-by-Frame animation.
* Animation smoothness depends on
* Adequate no. of frames
* Choosing appropriate speed for swapping (changing) images.

When should be used?
Best used for complicated graphics transformation which can not be implemented programmatically.

Example

```
var ad = AnimationDrawable()
var frame1 = resources.getDrawable
              (R.drawable.image1, null)
var frame2 = resources.getDrawable
              (R.drawable.image2, null)
var frame3 = resources.getDrawable
              (R.drawable.image3, null)

ad.addFrame(frame1, 200)
ad.addFrame(frame2, 300)
ad.addFrame(frame3, 200)

imageView.background = ad

//To Start Animation
   ad.start()

//To Stop Animation
   ad.stop()
```
_____
_____
_____
_____
_____
_____
_____

Tweened Animation
* This animation is applied on single drawable resource which can be
  * Bitmap graphics, ShapeDrawable, TextView, Any other view
* Intermediate frames will be generated (rendered) by system.
* Android supports following tweening transformations which can be defined in xml or in java.
* alpha, rotate, scale, translate
* Common properties of these transformation are
* When to start (startOffset)
* How long to animate (duration)
* Whether to return to starting state when animation completes. (fillAfter)

Tweening transformation
For xml way create xml file under **res->anim** folder.
Rotate (Spin)
* Rotate is defined as degrees.
* To spin object clockwise (0 to 360 degree)or counterclockwise (0 to -360 degree) around pivot point (within object boundary)
* Pivot point can be → fixed coordinate / percentage

spin.xml
```
<rotate
  android:fromDegrees="0"
  android:toDegrees="360"
  android:duration = "5000"
  android:pivotX="50%"
  android:pivotY="50%">
</rotate>

var an =
AnimationUtils.loadAnimation(this,R.anim.spin)
imageView.startAnimation(an)
```

Scale (Zoom)
* Used to stretch object vertically / horizontally.
* Starting scale (fromXScale, fromYScale)
* Target scale (toXScale, toYScale)
zoom.xml
```
<scale
  android:fromXScale="1.0"
  android:fromYScale="2.0"
  android:toXScale="4.0"
  android:toYScale="4.0"
  android:pivotY="50%"
```

```xml
    android:pivotX="50%"
    android:duration = "5000"
></scale>
```

//To Apply zoom animation
var an =
AnimationUtils.loadAnimation(**this**,R.anim.*zoom*)
imageView.startAnimation(an)

Translate (Movement)
- Moves object from one coordinate to other.
- Starting position (fromXDelta, fromYDelta)
- Relative target location (toXDelta, toYDelta)

move.xml
```xml
  <translate
   android:fromYDelta="-100"
   android:duration = "2500">
  </translate>
  <translate
   android:toYDelta="100"
   android:duration = "2500">
 </translate>
```

var an =
AnimationUtils.loadAnimation(**this**,R.anim.*move*)
imageView.startAnimation(an)

Alpha (Blink)
- It is used to blink the object.

blink.xml
```xml
<alpha
   android:fromAlpha="0.0"
   android:toAlpha="1.0"
   android:repeatMode = "reverse"
   android:duration = "500"
></alpha>
```

var an =
AnimationUtils.loadAnimation(**this**,R.anim.*blink*)
imageView.startAnimation(an)

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Frame By Frame	Tweened Animation
It is performed on a set of images.	It can perform series of simple transformation.
Series of Similar Images with slight difference displays on the screen in a sequence.	An animation that performs transitions such as transparency, zooming, moving, and spinning on a graphic.
It is like a digital Flip Book. It is work only on image related control	It works on almost each view like TextView, ImageView etc.
When these images are displayed quickly it gives illusion of a movement (animated).	Intermediate frames for twinning transformations like alpha, rotate, translate and scales are generated by the system.
Animation smoothness depends upon no of frames and appropriate speed of changing the images.	Animation depends upon the when to start and how long to animation i.e. duration.

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Location Based Services

- Location-based services" is an umbrella term that describes the different technologies you can use to find a device's current location.
- The two main LBS elements are
  - **Location Manager** — Provides hooks to the location-based services.
  - **Location Providers** — Each of these represents a different location finding technology used to determine the device's current location.
- Using the Location Manager, you can do the following
  - Obtain your current location
  - Follow movement
  - Set proximity alerts for detecting movement into and out of a specified area
  - Find available Location Providers
  - Monitor the status of the GPS receiver
- Access to the location-based services is provided by the Location Manager.
- To access the Location Manager, request an instance of the LOCATION_SERVICE using the getSystemService() method.

Finding Last Known Location

```
var lm = getSystemService(Context.LOCATION_SERVICE)
                as LocationManager
var loc = lm.getLastKnownLocation(
LocationManager.GPS_PROVIDER )

if(loc!=null) {
   tv.setText("Longitude = " + loc.getLongitude()
+        "\nLatitude = " + loc.getLatitude());
}
```

Refreshing the Current Location

```
var ll = object:LocationListener{
   override fun onLocationChanged(p0: Location?) {
   tv.setText("Longitude = " + p0.getLongitude()
        +"\nLatitude = "+p0.getLatitude());
   }
   override fun onStatusChanged(p0: String?, p1: Int,
p2: Bundle?) {}
   override fun onProviderEnabled(p0: String?) {}
   override fun onProviderDisabled(p0: String?) {}
}
```

```
lm.requestLocationUpdates(
LocationManager.GPS_PROVIDER,
1000, // milliseconds
100.2f, // distance in meter
ll)
```

Finding a Location Provider

- The LocationManager class includes static string constants that return the provider name for three Location Providers:
- LocationManager.GPS_PROVIDER
- LocationManager.NETWORK_PROVIDER
- LocationManager.PASSIVE_PROVIDER

- To get a list of the names of all the providers available call getProviders, using a Boolean to indicate if you want all, or only the enabled, providers to be returned:

```
var enabledOnly = true;
var providers =
     locationManager.getProviders(enabledOnly);
```

Permission Required

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

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Using Geocoder

- Geocoding enables you to translate between street addresses and longitude/latitude map coordinates.
- This can give you a recognizable context for the locations and coordinates used in location based services and map-based Activities.
- The Geocoder classes are included as part of the Google Maps library, so to use them you need to import it into your application by adding a uses-library node within the application node as shown here:
- As the geocoding lookups are done on the server, your applications also requires the Internet usespermission in your manifest:
  <uses-permission android:name= "android.permission.INTERNET"/>
- The Geocoder class provides access to two geocoding functions:
- **Forward geocoding** — Finds the latitude and longitude of an address
- **Reverse geocoding** — Finds the street address for a given latitude and longitude

```
private fun forwardGeocode(city: String?){

var gc = Geocoder(this, Locale.getDefault())
var addresses = gc.getFromLocationName(city,2)
var address = addresses.get(0)

textView2.visibility = View.VISIBLE
textView2.setText("${address.latitude} \n
${address.longitude}\n ${address.locality}")
}
```
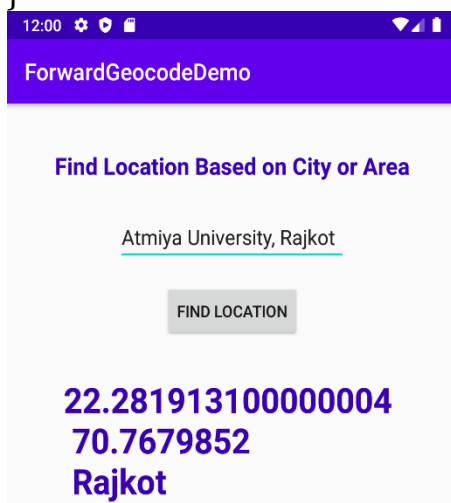


```
private fun reverseGeocode(loc: Location?) {
  var gc = Geocoder(this, Locale.getDefault())

  var addresses =
gc.getFromLocation(loc!!.latitude,loc.longitude,2)

  var address = addresses.get(0)
  textView2.setText("Current Location of Your Device
is\n${address.getAddressLine(0)} \n ${address.locality}
")
}
```
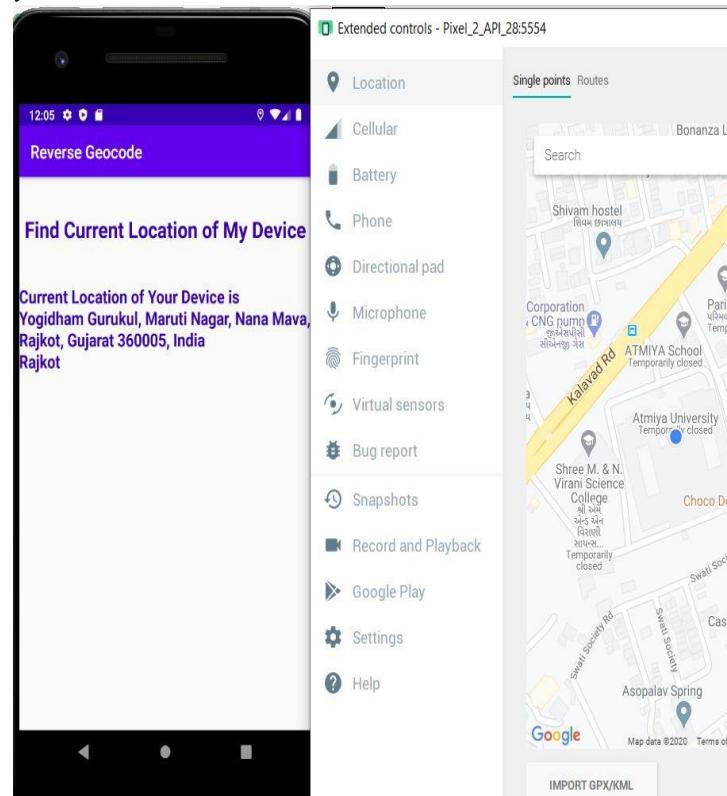

- Reverse geocoding returns street addresses for physical locations specified by latitude / longitude pairs.
- It's a useful way to get a recognizable context for the locations returned by location-based services.
- Forward geocoding (or just geocoding) determines map coordinates for a given location
- To geocode an address, call getFromLocationName on a Geocoder object.
- Pass in a string that describes the address you want the coordinates for, the maximum number of results to return, and optionally provide a geographic bounding box within which to restrict your search results:

CREATING MAP-BASED ACTIVITIES
- One of the most intuitive ways to provide context for a physical location or address is to use a map.
- Using a MapView, you can create Activities that include an interactive map.
- Map Views support annotation using Overlays and by pinning Views to geographical locations.
- Map Views offer full programmatic control of the map display, letting you control the zoom, location, and display modes — including the option to display a satellite view.

Introducing Map View and Map Activity
This section introduces several classes used to support Android maps.
- MapView is the user interface element that displays the map.
- MapActivity is the base class you extend to create an Activity that can include a Map View. The MapActivity class handles the application life cycle and background service management required for displaying maps. You can only use Map Views within MapActivity-derived Activities.
- Overlay is the class used to annotate your maps. Using Overlays, you can use a Canvas to draw onto any number of layers displayed on top of a Map View.
- MapController is used to control the map, enabling you to set the center location and zoom levels.
- MyLocationOverlay is a special Overlay that can be used to display the current position and orientation of the device.
- ItemizedOverlays and OverlayItems are used together to let you create a layer of map markers, displayed using Drawables and associated text.

Getting Your Maps API Key
To use a Map View in your application, you must first obtain an API key from the Android developer website at
http://code.google.com/android/maps-api-signup.html

- Without an API key the Map View cannot download the tiles used to display the map.
- To obtain a key, you need to specify the MD5 fingerprint of the certificate used to sign your application.
- Generally, you sign your application using two certificates: a default debug certificate and a production certificate.

- To use maps in your applications, you need to extend MapActivity.
- The layout for the new class must then include a MapView to display a Google Maps interface element.
- The Android maps library is not a standard Android package; as an optional API, it must be explicitly included in the application manifest before it can be used. Add the library to your manifest using a uses-library tag within the application node, as shown in the following XML snippet:

```
<uses-library
android:name="com.google.android.maps"/>
```

- To do this, you need to add a uses-permission tag to your application manifest for INTERNET, as shown here:

```
<uses-permission
android:name="android.permission.INTERNET"/>
```
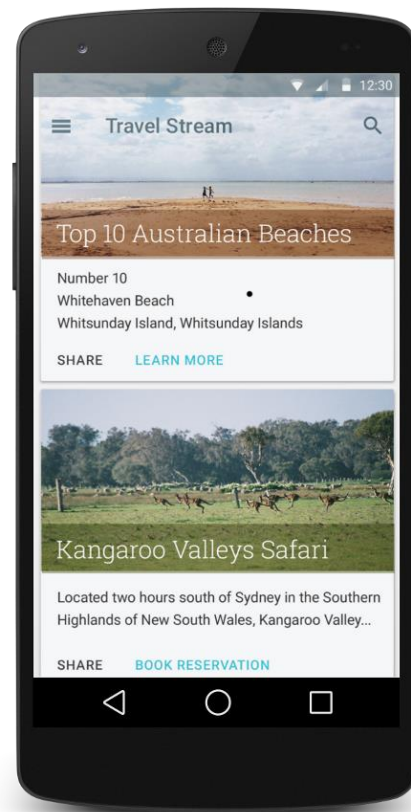
SKELETON MapActivity

```
class MapsActivity : AppCompatActivity(),
OnMapReadyCallback {
    private lateinit var mMap: GoogleMap
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_maps)
        val mapFragment = supportFragmentManager
    .findFragmentById(R.id.map) as SupportMapFragment
        mapFragment.getMapAsync(this)
    }
    override fun onMapReady(googleMap: GoogleMap) {
        mMap = googleMap
        val rajkot = LatLng(-22.308155, 70.800705)
mMap.addMarker(MarkerOptions().position(rajkot).title("Marker in Rajkot"))
mMap.moveCamera(CameraUpdateFactory.newLatLng(rajkot))   }
}
```

RecyclerView & CardView

- RecyclerView is flexible and efficient version of ListView.
- It is a container for rendering larger data set of views that can be recycled and scrolled very efficiently.
- RecyclerView is like traditional ListView widget, and with more flexibility to customizes and optimized to work with larger datasets.
- If your app needs to display a scrolling list of elements based on large data sets (or data that frequently changes), you should use RecyclerView.
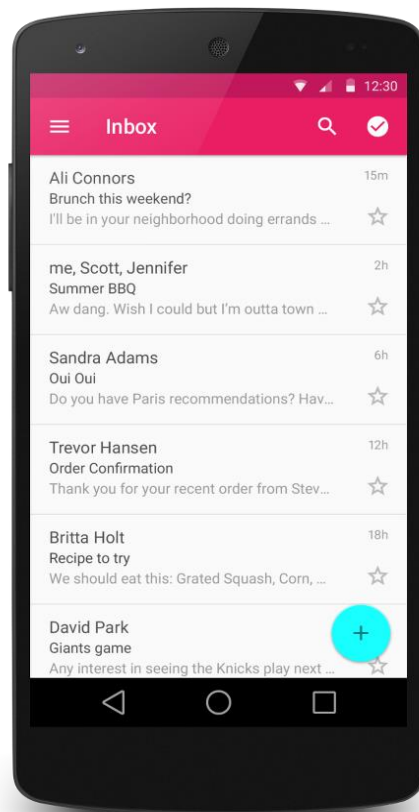
A List using RecyclerView	**A List using CardView**

RecyclerView Overview

- In the RecyclerView model, several different components work together to display your data. The overall container for your user interface is a RecyclerView object that you add to your layout.
- The RecyclerView fills itself with views provided by a layout manager that you provide.
- You can use one of standard layout managers (such as LinearLayoutManager or GridLayoutManager), or implement your own.
- The views in the list are represented by view holder objects. These objects are instances of a class you define by extending RecyclerView.ViewHolder.
- Each view holder is in charge of displaying a single item with a view. For example, if your list shows music collection, each view holder might represent a single album.
- The RecyclerView creates only as many view holders as are needed to display the on-screen portion of the dynamic content, plus a few extra.
- As the user scrolls through the list, the RecyclerView takes the off-screen views and rebinds them to the data which is scrolling onto the screen.

- The view holder objects are managed by an adapter, which you create by extending RecyclerView.Adapter.
- The adapter creates view holders as needed. The adapter also binds the view holders to their data.
- It does this by assigning the view holder to a position, and calling the adapter's onBindViewHolder() method. That method uses the view holder's position to determine what the contents should be, based on its list position.

Add the support library
To access the RecyclerView widget, you need to add the v7 Support Libraries to your project as follows:

1. Open the `build.gradle` file for your app module.

2. Add the support library to the `dependencies` section.

```
dependencies {
    implementation 'com.android.support:recyclerview-v7:28.0.0'
}
```

CardView Overview
- Apps often need to display data in similarly styled containers. These containers are often used in lists to hold each item's information.
- The system provides the CardView API as an easy way for you show information inside cards that have a consistent look across the platform.
- These cards have a default elevation above their containing view group, so the system draws shadows below them.
- Cards provide an easy way to contain a group of views while providing a consistent style for the container.

Add the dependency
The CardView widget is part of the v7 Support Libraries. To use it in your project, add the following dependency to your app module's build.gradle file:

```
dependencies {
    implementation 'com.android.support:cardview-v7:28.0.0'
}
```

Use these properties to customize the appearance of the CardView widget:
✓ To set the corner radius in your layouts, use the card_view:cardCornerRadius attribute.
✓ To set the corner radius in your code, use the CardView.setRadius method.
✓ To set the background color of a card, use the card_view:cardBackgroundColor attribute.

Example of RecyclerView and CardView

Step-1) Add following dependencies in app build.gradle file (If using android studio 3.6 or before)

```
dependencies {
    implementation 'androidx.cardview:cardview:1.0.0'
    implementation 'androidx.recyclerview:recyclerview:1.1.0'
}
```

Step-2) Add recyclerview control in activity_main.xml

```xml
<?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"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginStart="1dp"
        android:layout_marginTop="1dp"
        android:layout_marginEnd="1dp"
        android:layout_marginBottom="1dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Step-3 Add CardView Design in res -> Layout -> mylayout.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    app:cardUseCompatPadding="true"
    app:cardCornerRadius="25dp"
    app:contentPadding="5dp"
    app:cardBackgroundColor="#BCDAFD"
    android:layout_height="wrap_content">
```

```xml
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:layout_weight="1"
        app:srcCompat="@drawable/ic_launcher_background" />
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:orientation="vertical">
        <TextView
            android:id="@+id/textView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textStyle="bold"
            android:gravity="center"
            style="@style/TextAppearance.AppCompat.Large"
            android:text="TextView" />

        <TextView
            android:id="@+id/textView2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            style="@style/TextAppearance.AppCompat.Large"
            android:text="TextView" />
    </LinearLayout>
</LinearLayout>
</androidx.cardview.widget.CardView>
```

Design Sample of CardView
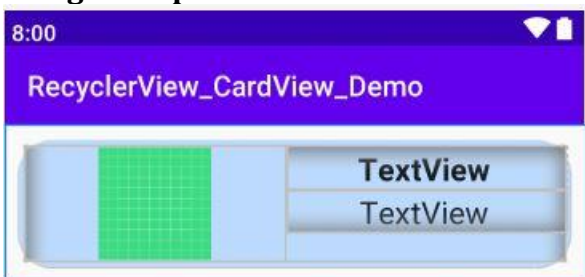
Step-4 Create the Adapter class for Recycler

```kotlin
class RecyclerAdapter : RecyclerView.Adapter<RecyclerAdapter.ViewHolder>() {
    val fname : Array<String> = arrayOf("Dr. Parag C. Shukla",
                    "Dr. Hetal Thaker","(Dr.) Kalpesh Popat",
                    "Prof. Shrey Shah","(Dr.) Ankit Faldu",
                    "Niraj Bhagchandani")
    val desg : Array<String> = arrayOf("Associate Professor\nHead-MCA",
                        "Associate Professor","Assistant Professor",
                        "Assistant Professor","Assistant Professor",
                    "Assistant Professor")
    val pics : IntArray = intArrayOf(R.drawable.pcs,R.drawable.hrt,
                    R.drawable.kmp,R.drawable.sms,R.drawable.ajf,
                    R.drawable.ndb)

    inner class ViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        var image : ImageView
        var fname : TextView
        var desg : TextView
        init {
            image = itemView.findViewById(R.id.imageView)
            fname = itemView.findViewById(R.id.textView)
            desg = itemView.findViewById(R.id.textView2)
        }
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
        val v : View! = LayoutInflater.from(parent.context)
                    .inflate(R.layout.mylayout,parent, attachToRoot: false)
        return ViewHolder(v)
    }

    override fun getItemCount(): Int {
        return fname.size
    }

    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        holder.fname.text = fname[position]
        holder.desg.text = desg[position]
        holder.image.setImageResource(pics[position])
    }
}
```

Step-5 bind data to recyclerview in MainActivity.kt

```kotlin
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        var recyclerview = findViewById<RecyclerView>(R.id.recyclerView)
        recyclerview.adapter = MyAdapter()

        var layout = LinearLayoutManager( context: this)
        recyclerview.layoutManager = layout
    }
}
```

Output