



The
University
Of
Sheffield.



COM4510/6510

Software Development for Mobile Devices

Lab 3: Designing sensible layouts

Dr Po Yang

The University of Sheffield

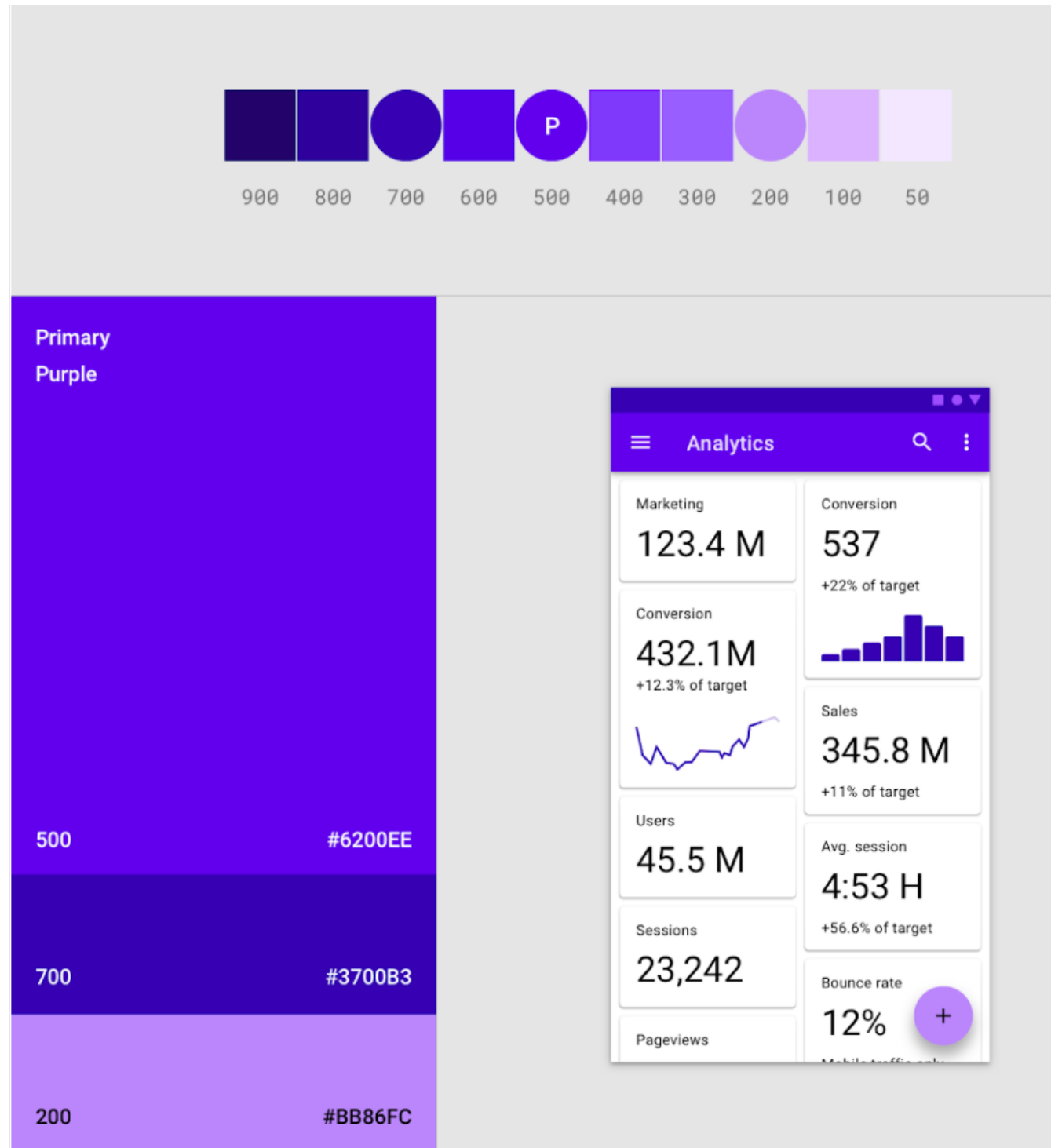
po.yang@sheffield.ac.uk

Colours

- Apps must be consistent and use **fixed limited range of colours**
- **Primary colour**
 - the color of large blocks
 - branding
- **Accent colour**
 - the colour of things that must stand up
 - e.g. exceptional actions
- **Primary color will also use a palette of variations**
 - use a couple of variations of the primary color
 - typically one darker and one lighter



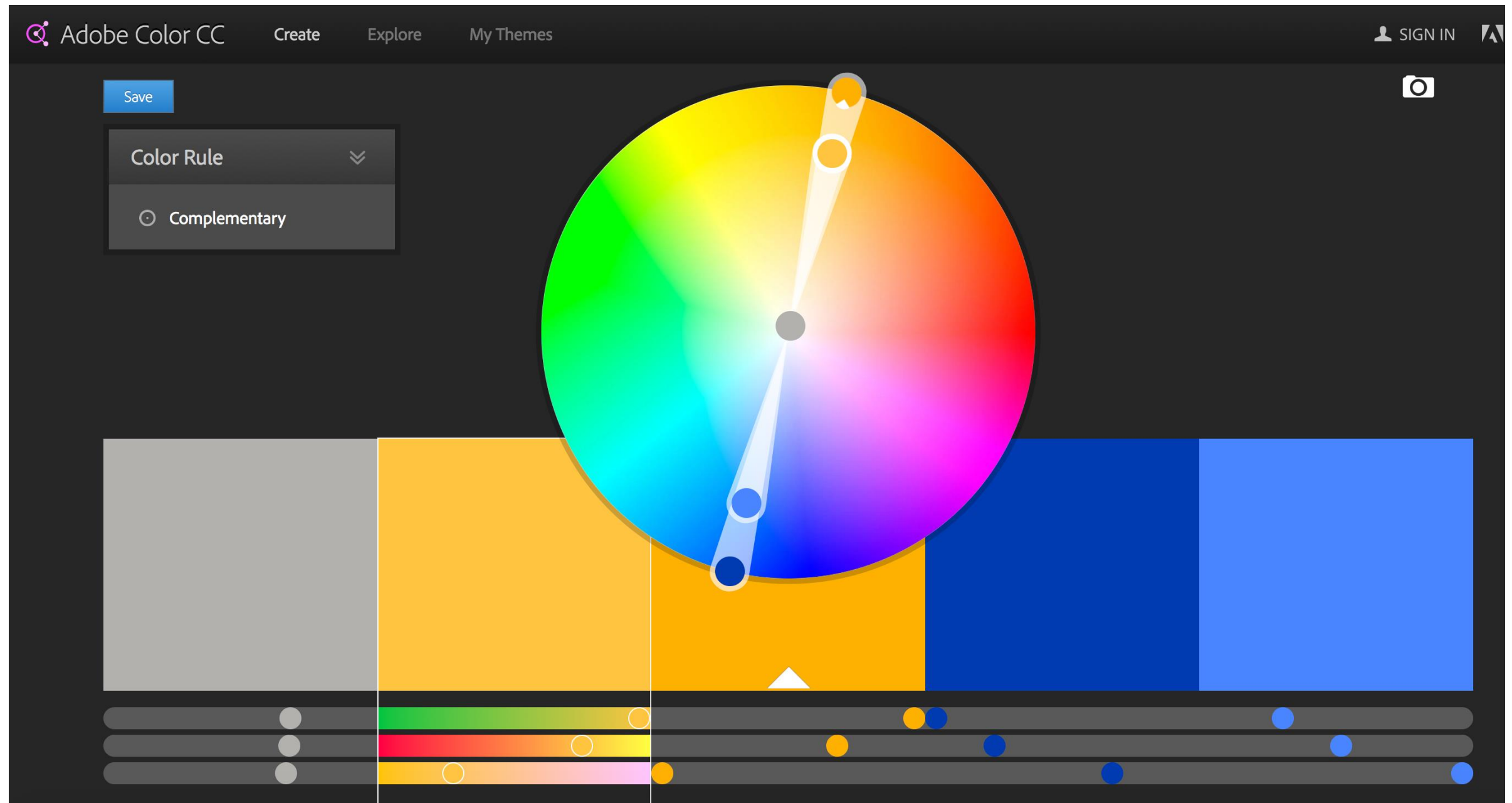
The
University
Of
Sheffield.



Primary color: purple

This UI uses a primary color and two primary variants

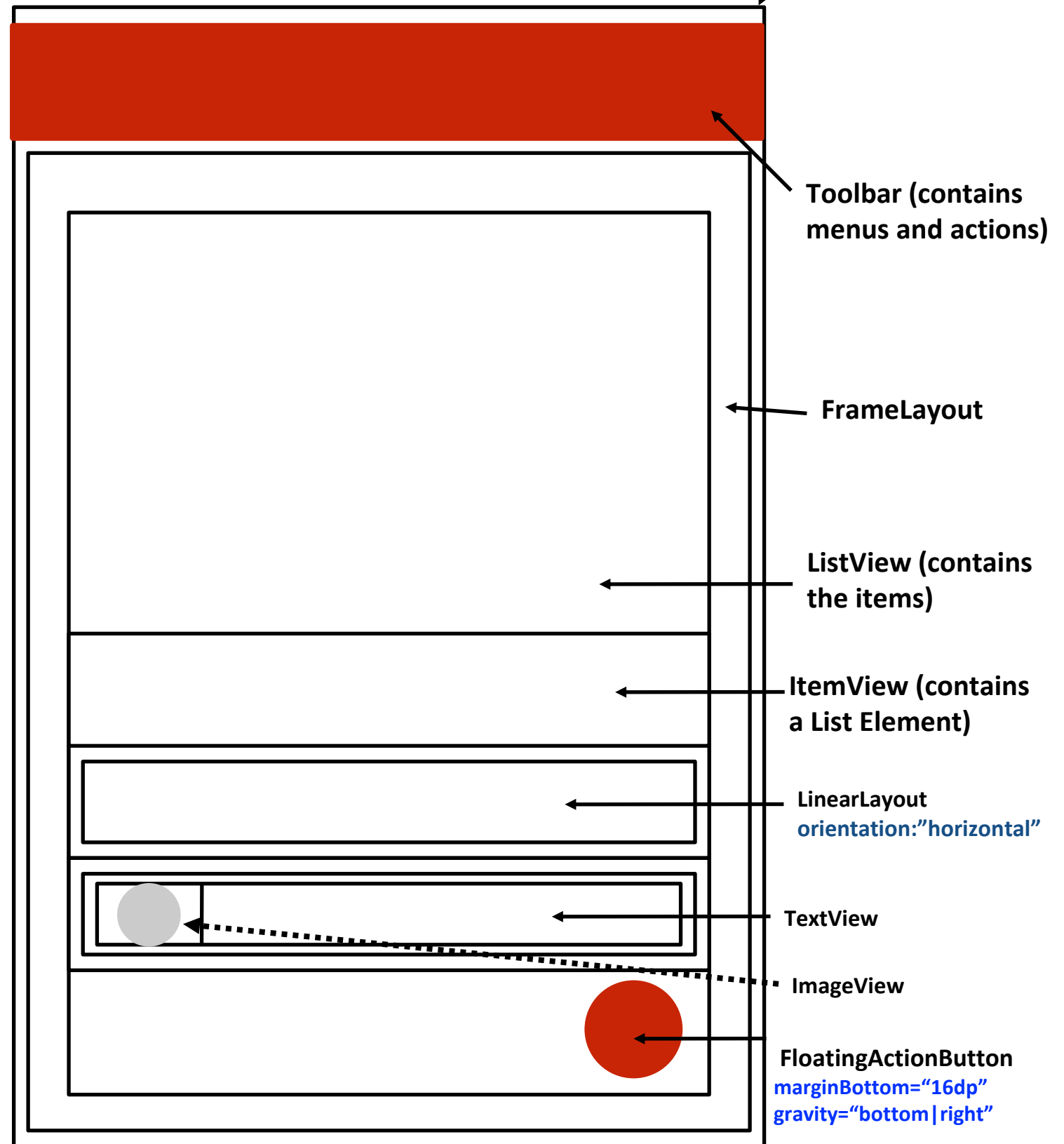
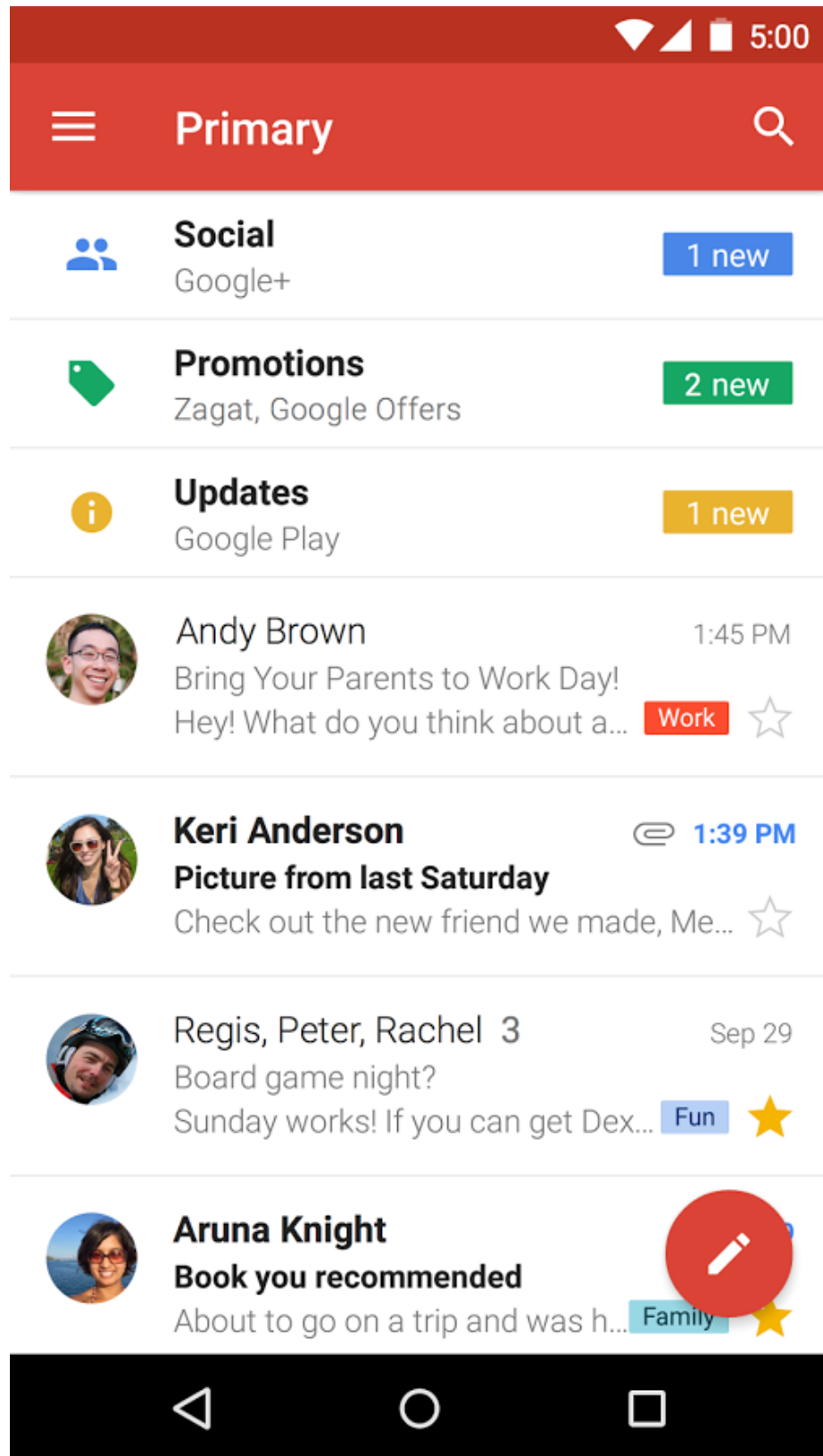
- To choose similar and complementary colours





The University Of Sheffield.

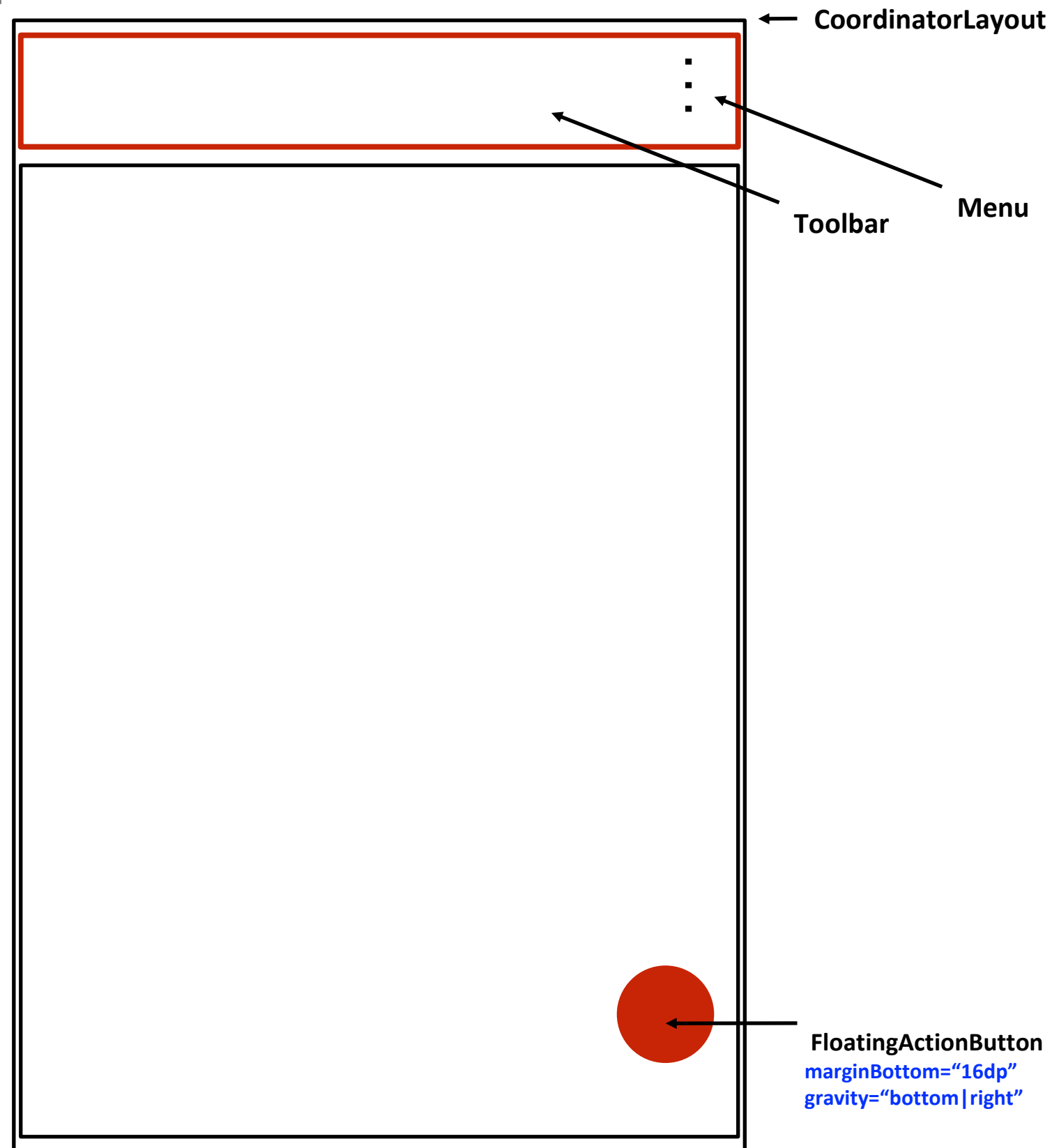
Designing GMail





Let's start with the basics

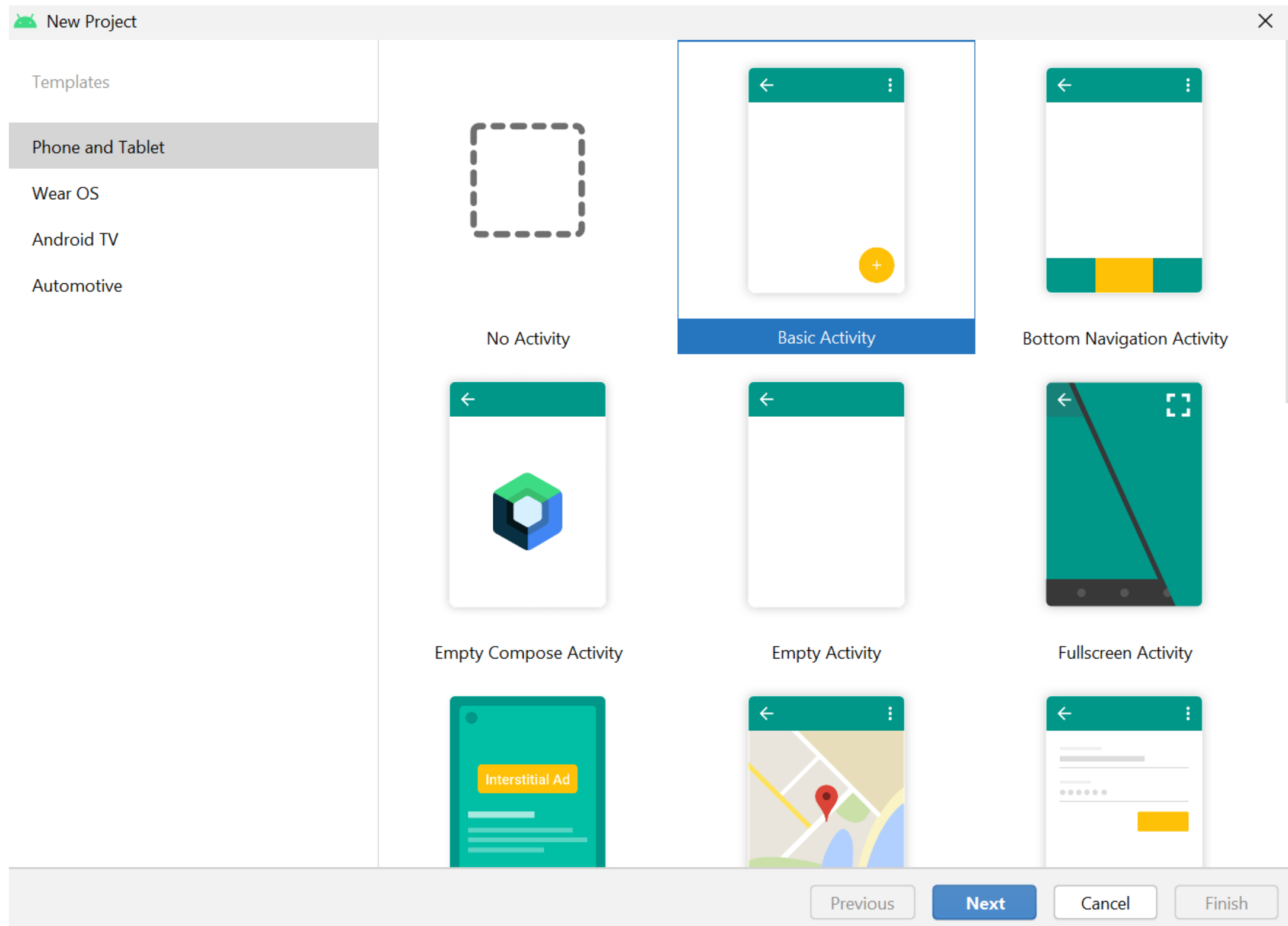
Note: in the figures
I am omitting the
android: Or **app:**
packages.
marginBottom
should be
android:marginBottom



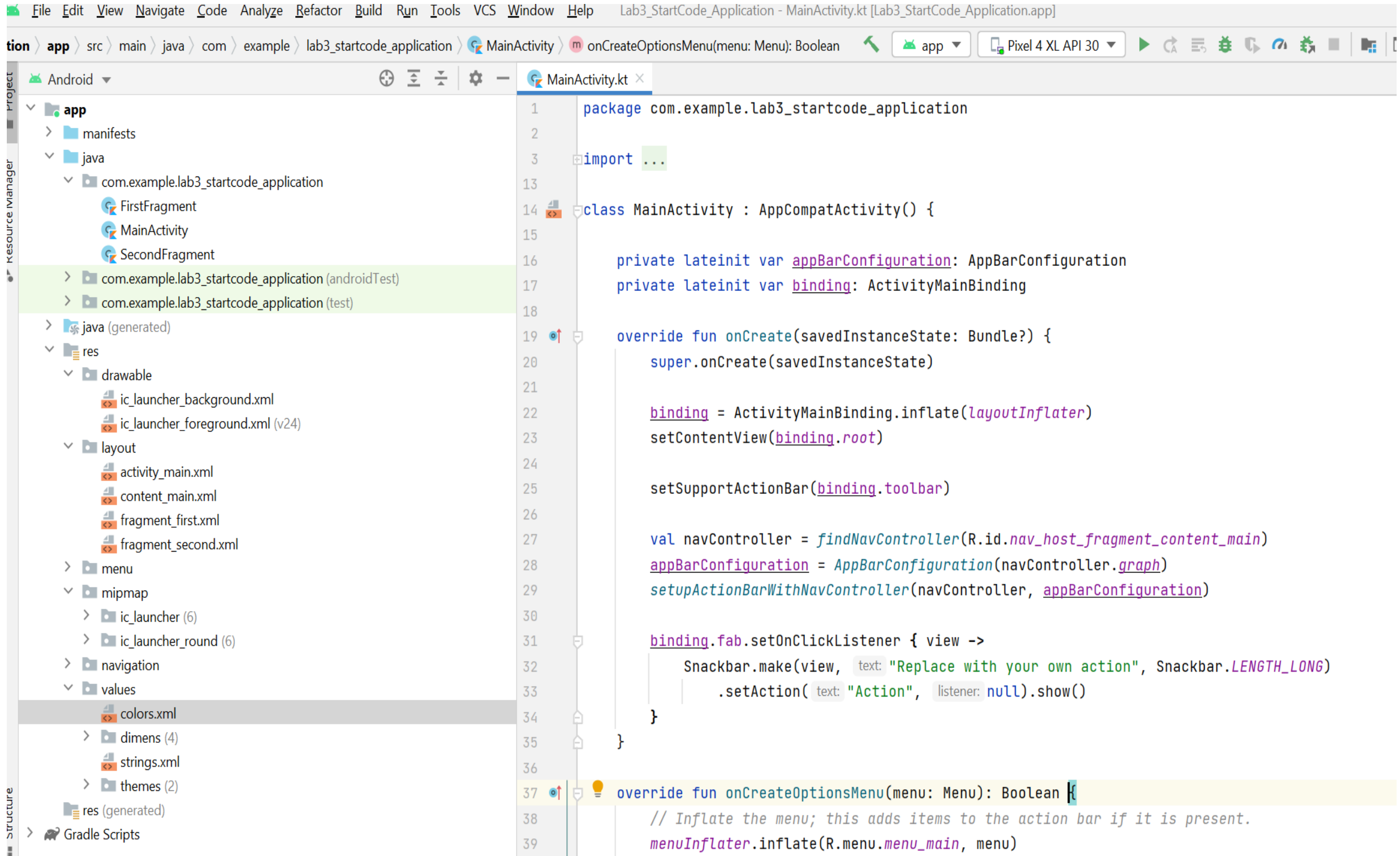


The
University
Of
Sheffield.

Exercise: Start a New Project



Download the code from Mole



```

1  package com.example.lab3_startcode_application
2
3  import ...
13
14  class MainActivity : AppCompatActivity() {
15
16      private lateinit var appBarConfiguration: AppBarConfiguration
17      private lateinit var binding: ActivityMainBinding
18
19      override fun onCreate(savedInstanceState: Bundle?) {
20          super.onCreate(savedInstanceState)
21
22          binding = ActivityMainBinding.inflate(layoutInflater)
23          setContentView(binding.root)
24
25          setSupportActionBar(binding.toolbar)
26
27          val navController = findNavController(R.id.nav_host_fragment_content_main)
28          appBarConfiguration = AppBarConfiguration(navController.graph)
29          setupActionBarWithNavController(navController, appBarConfiguration)
30
31          binding.fab.setOnClickListener { view ->
32              Snackbar.make(view, text: "Replace with your own action", Snackbar.LENGTH_LONG)
33                  .setAction(text: "Action", listener: null).show()
34          }
35      }
36
37      override fun onCreateOptionsMenu(menu: Menu): Boolean {
38          // Inflate the menu; this adds items to the action bar if it is present.
39          menuInflater.inflate(R.menu.menu_main, menu)
  
```




activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout 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">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/Theme.Lab3_StartCode_Application.AppBarOverlay">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/Theme.Lab3_StartCode_Application.PopupOverlay" />

    </com.google.android.material.appbar.AppBarLayout>

    <include layout="@layout/content_main" />

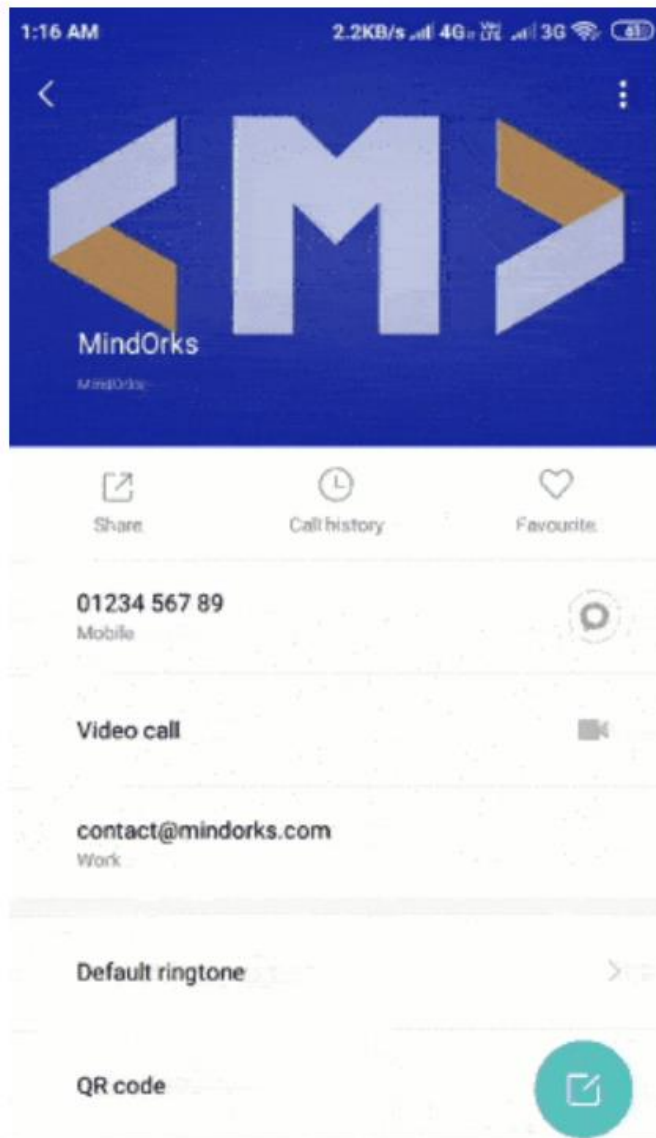
    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_marginEnd="16dp"
        android:layout_marginBottom="16dp"
        app:srcCompat="@android:drawable/ic_dialog_email" />

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

CoordinatorLayout

- CoordinatorLayout is a **super-powered FrameLayout**.
- CoordinatorLayout is intended for two primary use cases:
 - **As a top-level application decor or chrome layout**
 - **As a container for a specific interaction with one or more child views**
- By specifying behaviours for child views of a CoordinatorLayout you can provide many different **interactions within a single parent** and those views can also interact with one another.
- View classes can specify a default behavior when used as a child of a CoordinatorLayout using the DefaultBehavior annotation

Using coordinator layout



<https://blog.mindorks.com/using-coordinator-layout-in-android>

Typical pattern: include

```
<include layout="@layout/content_main" />
```

- Useful to simplify the layout
- Useful to reuse elements

```
</com.google.android.material.appbar.AppBarLayout>
```

```
<include layout="@layout/content_main" />
```

```
<com.google.android.material.floatingactionbutton.FloatingActionButton
```

```
    android:id="@+id/fab"
```

```
    android:layout_width="wrap_content"
```

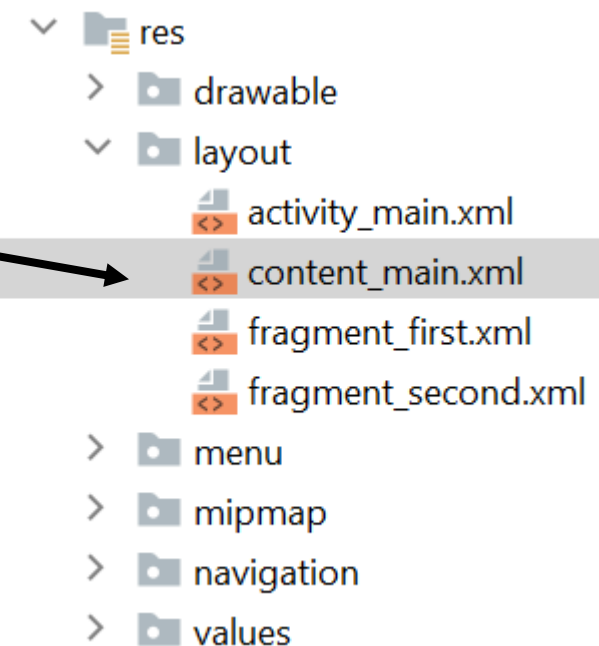
```
    android:layout_height="wrap_content"
```

```
    android:layout_gravity="bottom|end"
```

```
    android:layout_marginEnd="16dp"
```

```
    android:layout_marginBottom="16dp"
```

```
    app:srcCompat="@android:drawable/ic_dialog_email" />
```





content_main.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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="com.google.android.material.appbar.AppBarLayout$Scrolli...">

    <fragment
        android:id="@+id/nav_host_fragment_content_main"
        android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:defaultNavHost="true"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:navGraph="@navigation/nav_graph" />
</androidx.constraintlayout.widget.ConstraintLayout>

```

Absolute
Positioning
taking the whole parent

Relative Positioning in ConstraintLayout
Positioning
based on "parent" and relative positioning
to others



MainActivity.kt

```
class MainActivity : AppCompatActivity() {
```

```
    private lateinit var appBarConfiguration: AppBarConfiguration
```

```
    private lateinit var binding: ActivityMainBinding
```

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

```
        super.onCreate(savedInstanceState)
```

```
        binding = ActivityMainBinding.inflate(layoutInflater)
```

```
        setContentView(binding.root)
```

Set Layout

```
        setSupportActionBar(binding.toolbar)
```

Set ToolBar

```
        val navController = findNavController(R.id.nav_host_fragment_content_main)
```

```
        appBarConfiguration = AppBarConfiguration(navController.graph)
```

Get hold of Controller
Button

```
        setupActionBarWithNavController(navController, appBarConfiguration)
```

```
        binding.fab.setOnClickListener { view -> Assign onClick
```

```
            Snackbar.make(view, text: "Replace with your own action", Snackbar.LENGTH_LONG)
```

```
                .setAction(text: "Action", listener: null).show()
```

```
        }
```

```
    }
```


MainActivity.kt

```
import android.os.Bundle
import com.google.android.material.snackbar.Snackbar
import androidx.appcompat.app.AppCompatActivity
import androidx.navigation.findNavController
import androidx.navigation.ui.AppBarConfiguration
import androidx.navigation.ui.navigateUp
import androidx.navigation.ui.setupActionBarWithNavController
import android.view.Menu
import android.view.MenuItem
import com.example.lab3_startcode_application.databinding.ActivityMainBinding
```

You need import Android support library

Menu

It has a menu

```
override fun onCreateOptionsMenu(menu: Menu): Boolean {  
    // Inflate the menu; this adds items to the action bar if it is present.  
    menuInflater.inflate(R.menu.menu_main, menu)  
    return true  
}
```

**inflate menu layout
declared under /res/menu/menu_main.xml**

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {  
    // Handle action bar item clicks here. The action bar will  
    // automatically handle clicks on the Home/Up button, so long  
    // as you specify a parent activity in AndroidManifest.xml.  
    return when (item.itemId) {  
        R.id.action_settings -> true  
        else -> super.onOptionsItemSelected(item)  
    }  
}
```

**when menu
item selected**

**id of element selected
(an integer)**



menu_main.xml

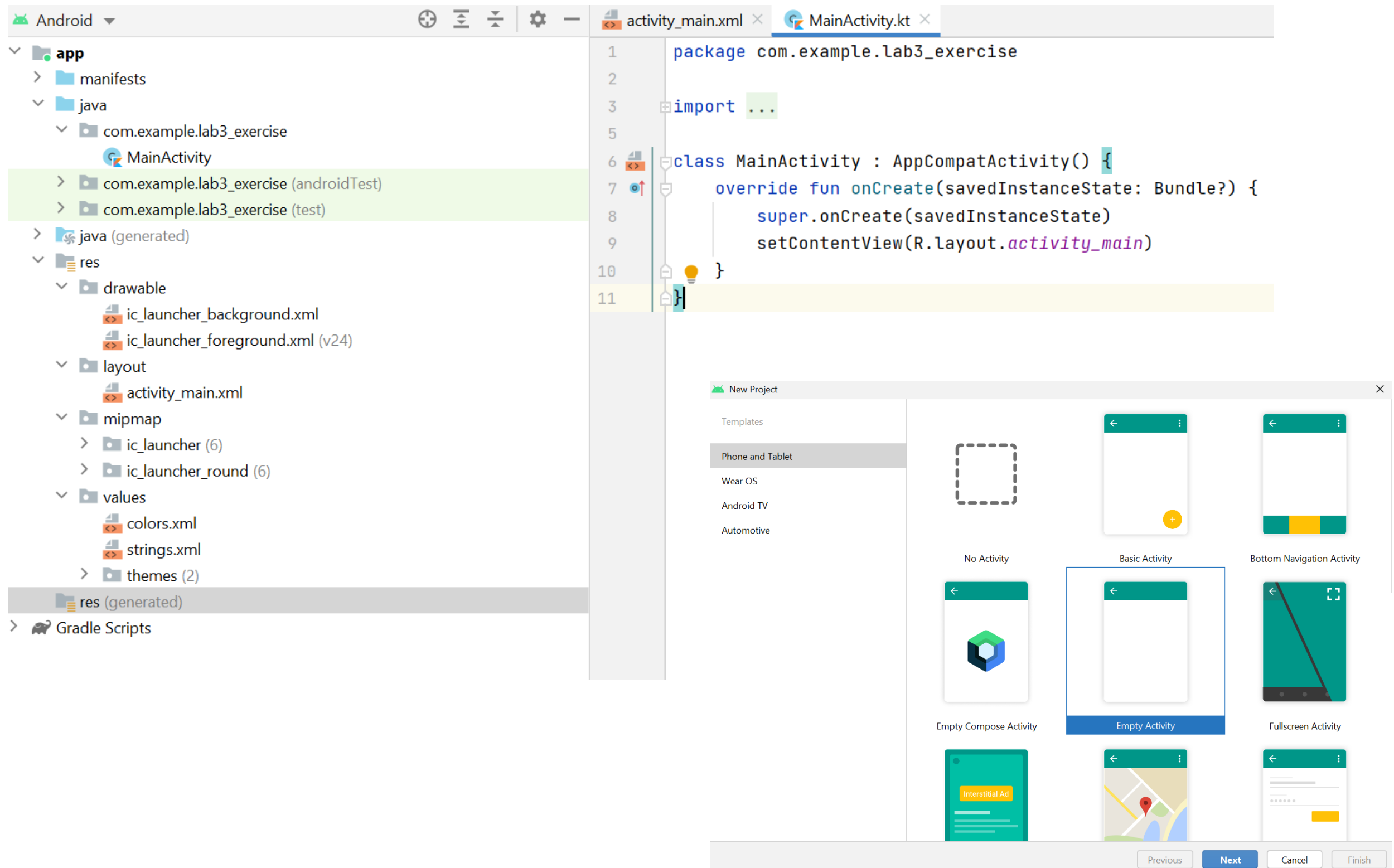
```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="com.example.lab3_startcode_application.MainActivity">
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:title="Settings"
        app:showAsAction="never" />
</menu>
```

Menu in the code

```
override fun onOptionsItemSelected(item: MenuItem): Boolean
{
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so
    long
    // as you specify a parent activity in AndroidManifest.xml.
    return when (item.itemId) {
        R.id.action_settings -> true
        else -> super.onOptionsItemSelected(item)
    }
}
```

Kotlin Code

Exercise: starting from Empty Activity



Android

app

- manifests
- java
 - com.example.lab3_exercise
 - MainActivity
 - com.example.lab3_exercise (androidTest)
 - com.example.lab3_exercise (test)
 - java (generated)
- res
 - drawable
 - ic_launcher_background.xml
 - ic_launcher_foreground.xml (v24)
 - layout
 - activity_main.xml
 - mipmap
 - ic_launcher (6)
 - ic_launcher_round (6)
 - values
 - colors.xml
 - strings.xml
 - themes (2)
- res (generated)
- Gradle Scripts

```

1 package com.example.lab3_exercise
2
3 import ...
4
5
6 class MainActivity : AppCompatActivity() {
7     override fun onCreate(savedInstanceState: Bundle?) {
8         super.onCreate(savedInstanceState)
9         setContentView(R.layout.activity_main)
10    }
11 }
  
```

New Project

Templates

- Phone and Tablet
- Wear OS
- Android TV
- Automotive

No Activity

Basic Activity

Bottom Navigation Activity

Empty Compose Activity

Empty Activity

Fullscreen Activity

Interstitial Ad

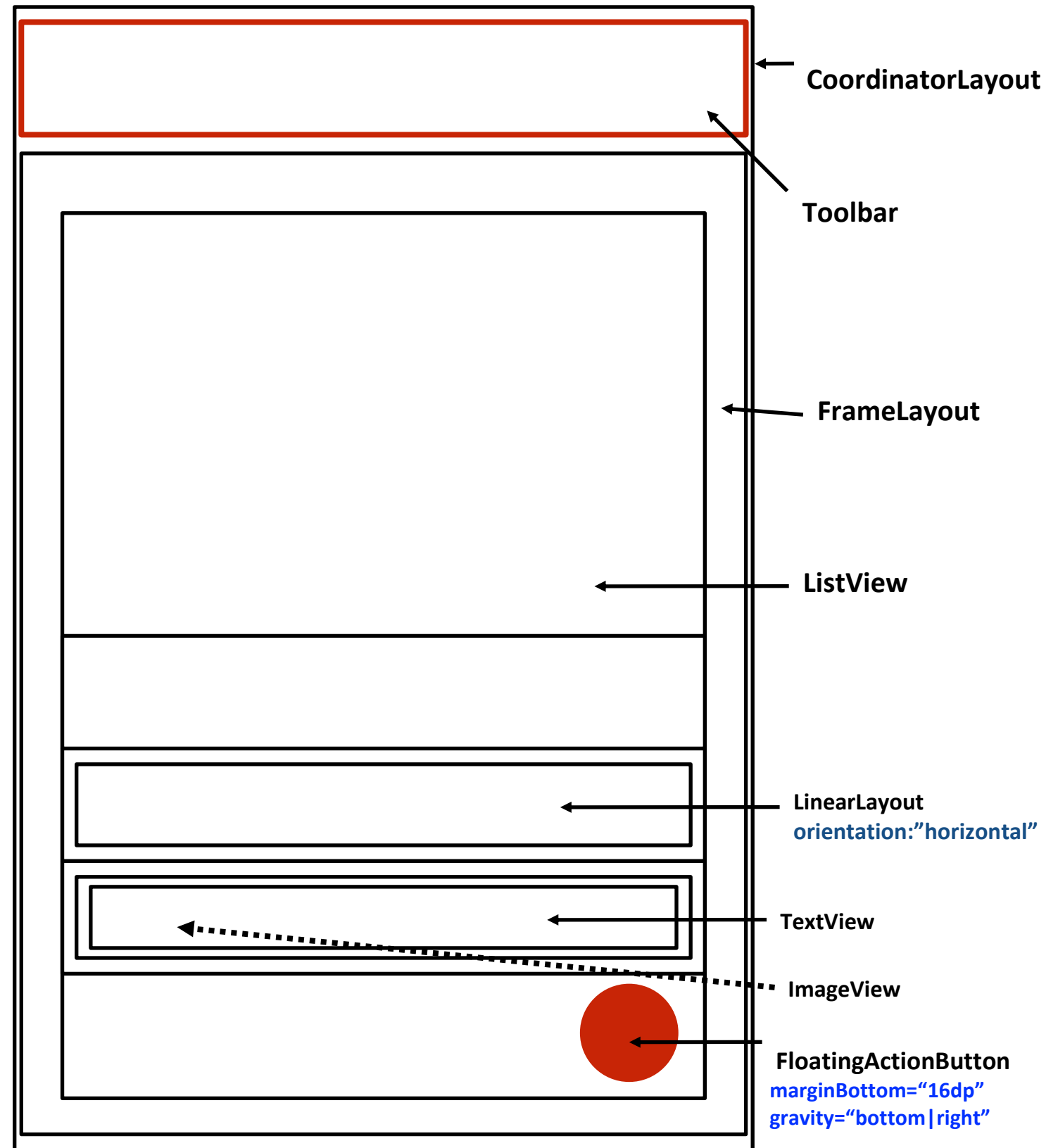
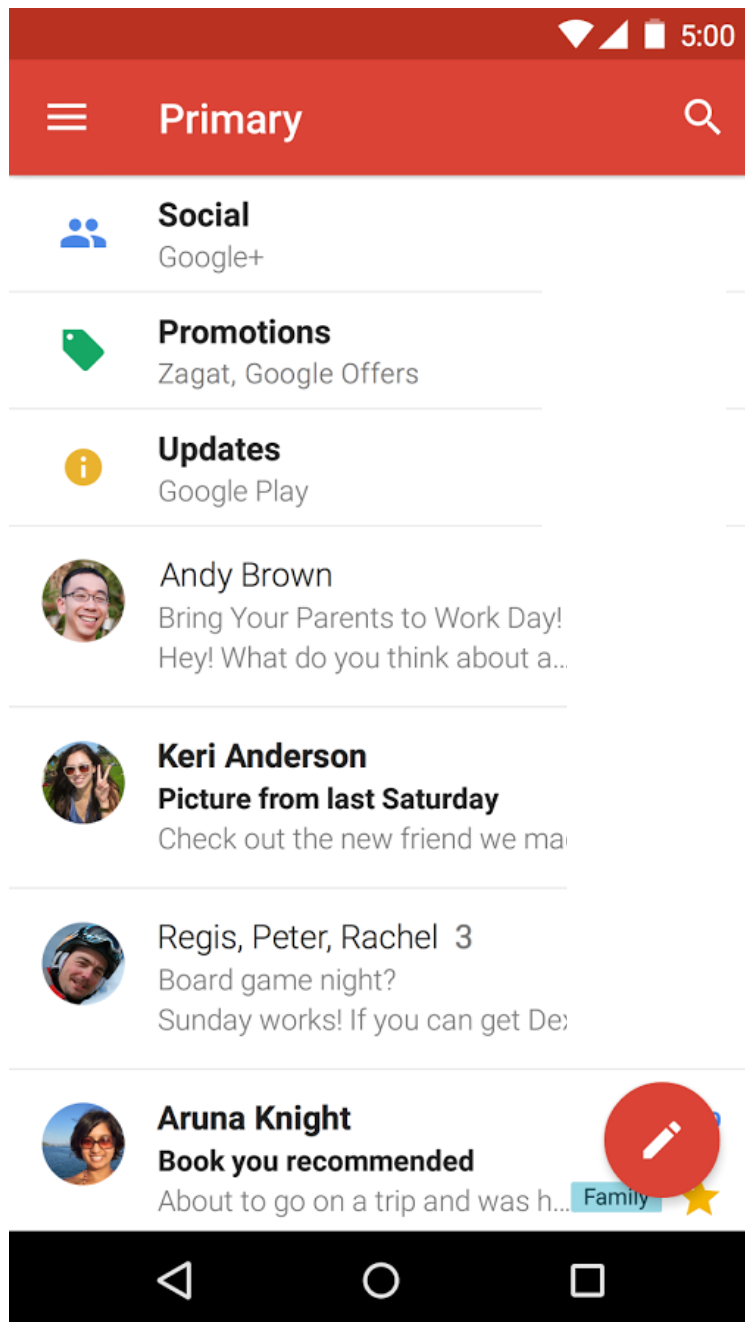
Previous Next Cancel Finish

Look at the code

- And make sure to understand it
- Note what happens when you press the button
- Find the code that does that

This is the target

Like in a simplified
Gmail



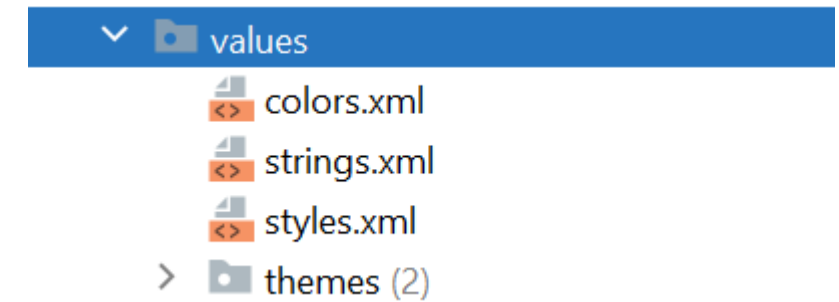
How?

- We will use a list of items
- This is implemented using two elements:
 - A recyclerView class containing the list of items
 - An adapter class describing the elements of the list
- The list of elements to display are passed to the recycler class which will display them



Preparation of Resource

- Colors.xml



```
<?xml version="1.0" encoding="utf-8"?>
```

```
<resources>
```

```
    <color name="purple_200">#FFBB86FC</color>
```

```
</resources>
```

```
<!-- Base application theme. -->
```

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
```

```
    <!-- Customize your theme here. -->
```

```
    <item name="colorPrimary">@color/colorPrimary</item>
```

```
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
```

```
    <item name="colorAccent">@color/colorAccent</item>
```

```
</style>
```

```
<style name="AppTheme.NoActionBar">
```

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

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

```
</style>
```

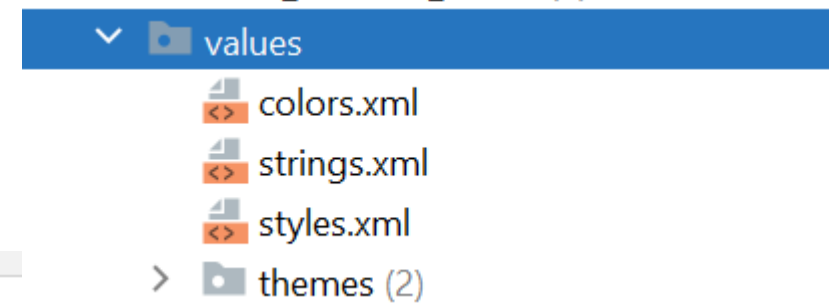
```
<style name="AppTheme.AppBarOverlay" parent="ThemeOverlay.AppCompat.Dark.ActionBar" />
```

```
<style name="AppTheme.PopupOverlay" parent="ThemeOverlay.AppCompat.Light" />
```

```
</resources>
```

Preparation of Resource

- styles.xml

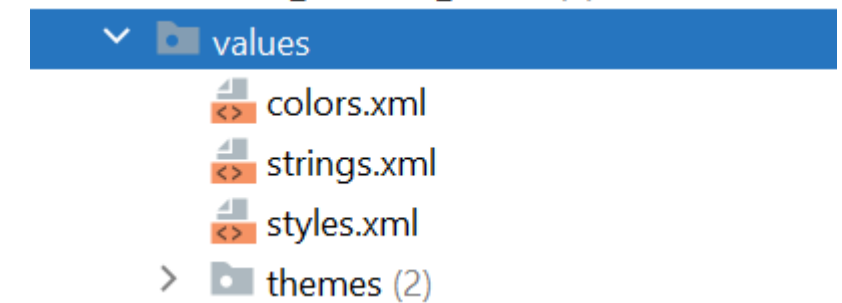


```
<resources>

  <!-- Base application theme. -->
  <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
  </style>
  <style name="AppTheme.NoActionBar">
    <item name="windowActionBar">false</item>
    <item name="windowNoTitle">true</item>
  </style>
  <style name="AppTheme.AppBarOverlay" parent="ThemeOverlay.AppCompat.Dark.ActionBar" />
  <style name="AppTheme.PopupOverlay" parent="ThemeOverlay.AppCompat.Light" />
</resources>
```

Preparation of Resource

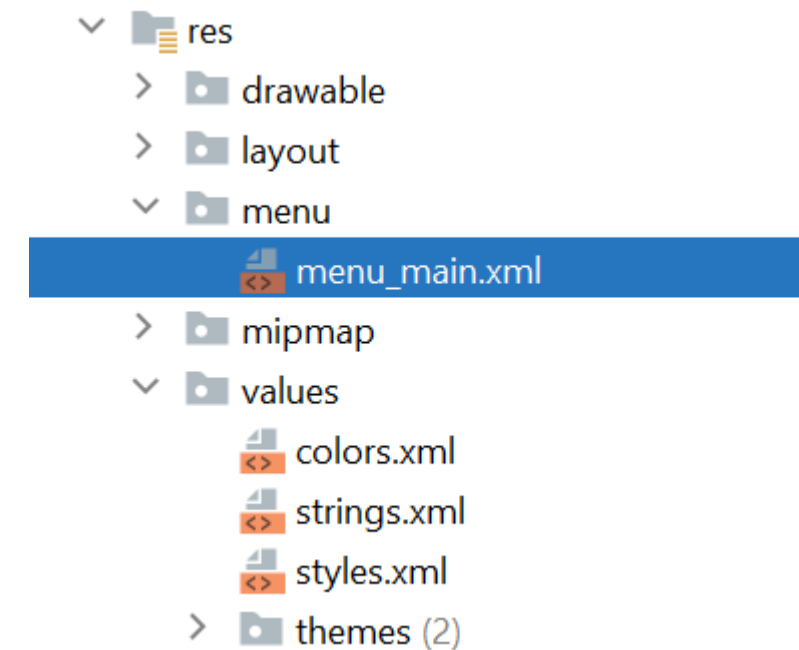
- strings.xml



```
<resources>
    <string name="app_name">week3_exercise_kt</string>
    <string name="action_settings">Settings</string>
</resources>
```

Preparation of Resource

- menu_main.xml



```
<?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"
      xmlns:tools="http://schemas.android.com/tools"
      tools:context="com.lab.week3_exercise_kt.MainActivity">
    <item android:id="@+id/action_settings" android:orderInCategory="100"
          android:title="@string/action_settings" app:showAsAction="never" />
</menu>
```


Imports

- MainActivity.kt

```
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

import android.widget.Toolbar
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.google.android.material.floatingactionbutton.FloatingActionButton
import com.google.android.material.snackbar.Snackbar
```



activity_main.xml

- Change the layout to coordinatorlayout

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
    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">

    <com.google.android.material.appbar.AppBarLayout android:layout_width="match_parent"
        android:layout_height="wrap_content" android:theme="@style/AppTheme.AppBarOverlay">

        <androidx.appcompat.widget.Toolbar android:id="@+id/toolbar"
            android:layout_width="match_parent" android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary" app:popupTheme="@style/AppTheme.PopupOverlay" />

    </com.google.android.material.appbar.AppBarLayout>

    <include layout="@layout/content_main" />

    <com.google.android.material.floatingactionbutton.FloatingActionButton android:id="@+id/fab"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_gravity="bottom|end" android:layout_margin="16dp"
        app:srcCompat="@android:drawable/ic_dialog_email" />

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

Change to coordinatorlaout

Including another content_main layout

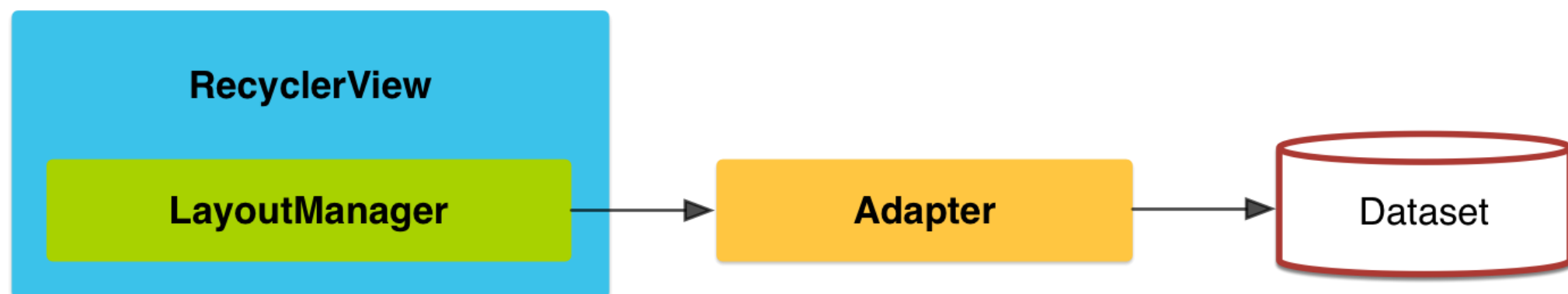
RecyclerView

The **RecyclerView class** simplifies the display and handling of large data sets by providing:

- Layout managers for positioning items
- Default animations for common item operations, such as removal or addition of items

You also have the flexibility to define custom layout managers and animations for RecyclerView widgets.

Figure 1. The RecyclerView widget.



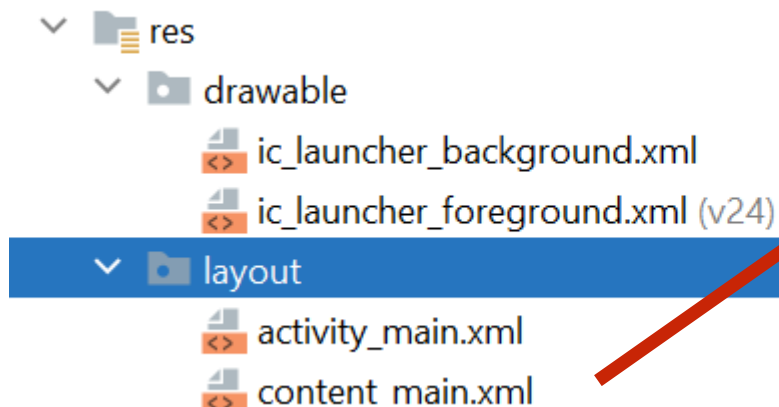
content_main.xml

```
<?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"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context=".MainActivity"
    tools:showIn="@layout/activity_main">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/my_list"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        tools:ignore="MissingConstraints" />

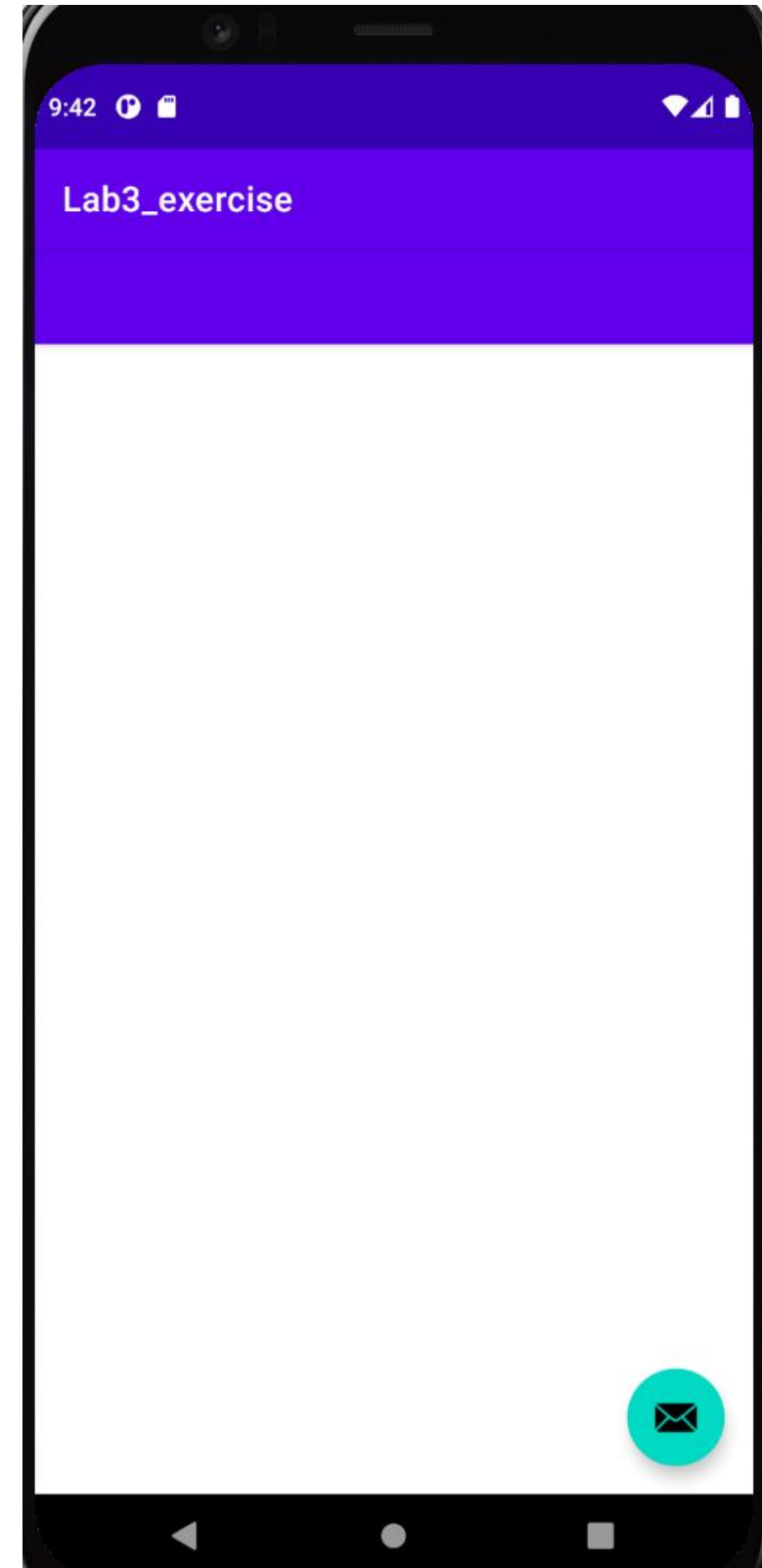
</androidx.constraintlayout.widget.ConstraintLayout>
```

Creating a
content_main xml file
and inserting the
RecyclerView



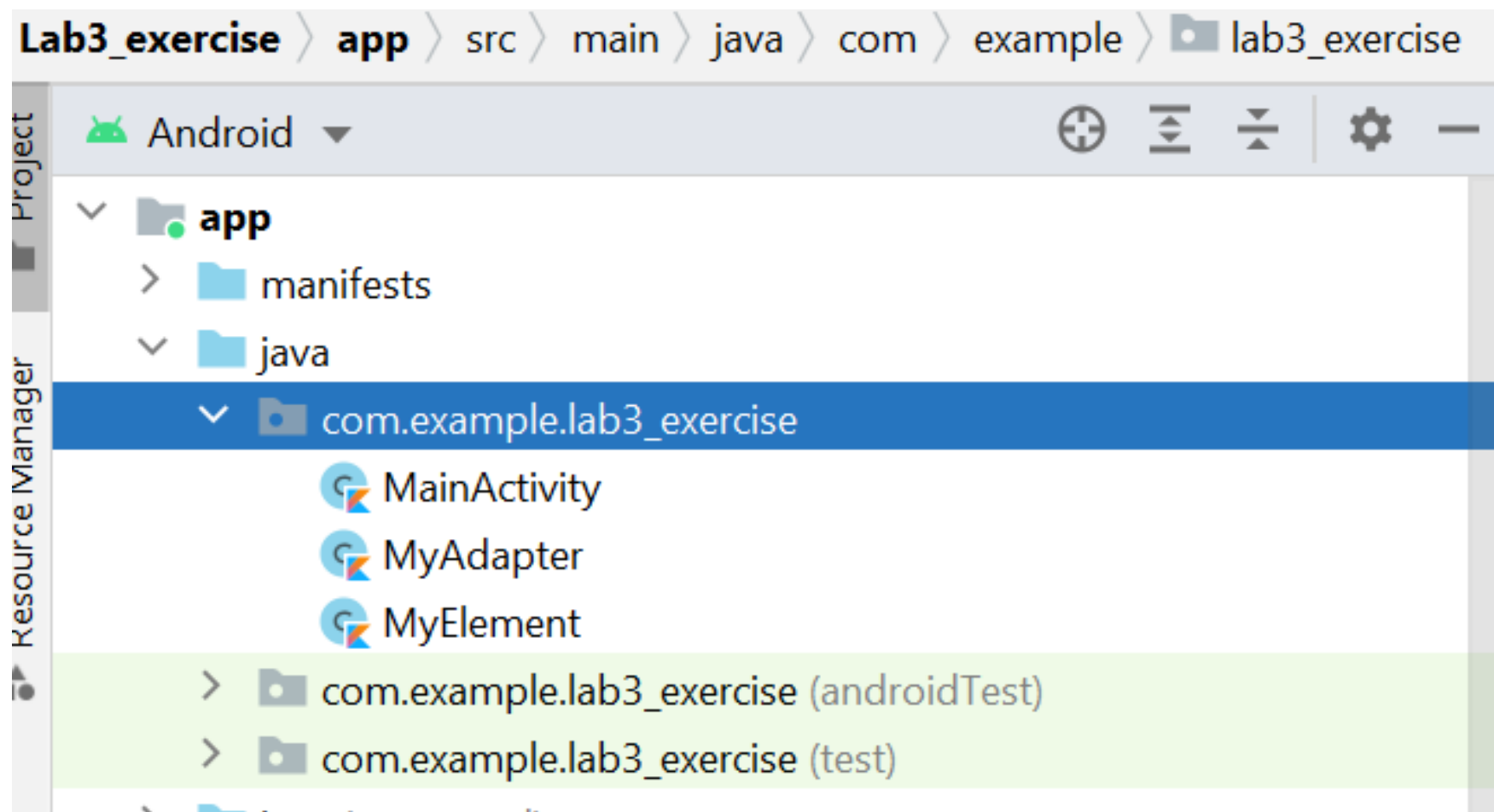
Run

- You will see the right interface



In Kotlin (on Create)

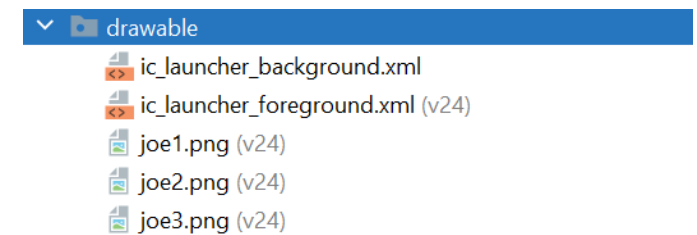
- We need create another two classes
 - **MyElement** ---- **ViewHolder**
 - **MyAdapter** ---- **ViewHolder**



MyElement-ViewHolder

```
package com.example.lab3_exercise

class MyElement(var image: Int, var title: String, var preview: String) {
    init {
    }
}
```



- Image will be stored in e.g. **res/drawable/filename.png**
- The folder is under
~/**AndroidStudioProjects/<your project name>/app/src/main/res/drawable-24**
- **Please download them from the week 3 tutorial.**
- So it will be referable as **R.drawable.filename**
- To create an element for testing
- **MyElement myElement=new MyElement(R.drawable.Joe, "Good Morning",
"Just wanted to say hello");**

In MainActivity

- Complete the imports

```
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.Menu
import android.view.MenuItem
import android.view.View

import android.widget.Toolbar
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.google.android.material.floatingactionbutton.FloatingActionButton
import com.google.android.material.snackbar.Snackbar
```

In MainActivity

- Declare classes

```
class MainActivity : AppCompatActivity() {  
    private lateinit var mRecyclerView: RecyclerView  
    private lateinit var mAdapter: RecyclerView.Adapter<RecyclerView.ViewHolder>  
    private lateinit var layoutManager: RecyclerView.LayoutManager  
    private val myDataset: Array<MyElement> = arrayOf<MyElement>(  
        MyElement(  
            R.drawable.joe1, title: "Hello",  
            preview: "Would like to say hello 1"  
        ),  
        MyElement(  
            R.drawable.joe2, title: "Hello",  
            preview: "Would like to say hello 2"  
        ),  
        MyElement(  
            R.drawable.joe3, title: "Good Evening",  
            preview: "JWould like to say hello 3"  
        )  
    )  
}
```

In MainActivity

- onCreate()

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
  
    val toolbar: androidx.appcompat.widget.Toolbar = findViewById(R.id.toolbar)  
  
    mRecyclerView = findViewById<RecyclerView>(R.id.my_list)  
  
    // use this setting to improve performance if you know that changes  
    // in content do not change the layout size of the RecyclerView  
    //mRecyclerView.setHasFixedSize(true);  
  
    // use a linear layout manager  
    layoutManager = LinearLayoutManager(context, this)  
    mRecyclerView.layoutManager = layoutManager  
  
    // specify an adapter (see also next example)  
    mAdapter = MyAdapter(myDataset) as RecyclerView.Adapter<RecyclerView.ViewHolder>  
    mRecyclerView.adapter = mAdapter  
    val fab: FloatingActionButton = findViewById<FloatingActionButton>(R.id.fab)  
    fab.setOnClickListener(View.OnClickListener { view ->  
        Snackbar.make(view, text: "Replace with your own action", Snackbar.LENGTH_LONG)  
            .setAction(text: "Action", listener: null).show()  
    })  
}
```

Declare the RecyclerView

Create a new layout manager
for it

MyAdapter Class needs to be
completed

An Example of Adapter

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk
/res/android"
    android:layout_width="fill_parent"

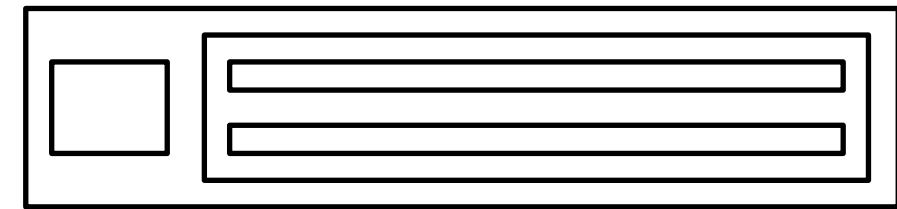
    android:layout_height="?android:attr/listPref
erredItemHeight"
    android:padding="6dip" >

    <ImageView
        android:id="@+id/icon"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:layout_alignParentBottom="true"
        android:layout_alignParentTop="true"
        android:layout_marginRight="6dip"
        android:contentDescription="TODO"
        android:src="@drawable/ic_launcher"

    />

    <TextView
        android:id="@+id/secondLine"
        android:layout_width="fill_parent"
        android:layout_height="26dip"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_toRightOf="@id/icon"
        android:ellipsize="marquee"
        android:maxLines="1"
        android:text="Description"
        android:textSize="12sp" />
```

The elements are positioned horizontally:
All of them are aligned to the bottom of the parent and then declared to be to the right of some others



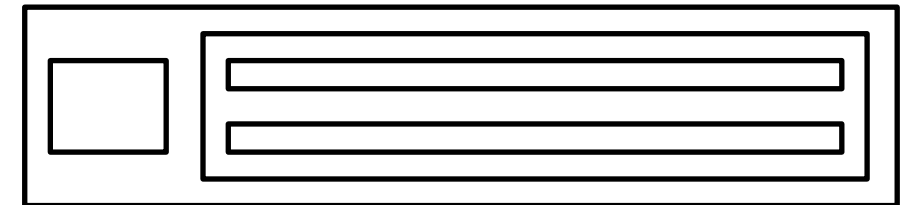
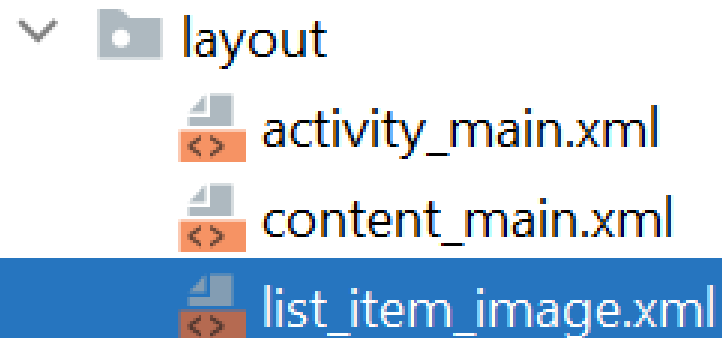
```
<TextView
    android:id="@+id/firstLine"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_above="@id/secondLine"
    android:layout_alignParentRight="true"
    android:layout_alignParentTop="true"
    android:layout_alignWithParentIfMissing="true"

    android:layout_toRightOf="@id/icon"
    android:gravity="center_vertical"
    android:text="Example application"
    android:textSize="16sp" />

</RelativeLayout>
```

List_item_image.xml

- To create res/layout/list_item_image.xml



Here we create the same layout as before but instead of relative layout we use a linear layout with a horizontal orientation



List_item_image.xml

- To create res/layout/list_item_image.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="8dp"
    android:orientation="horizontal">
```

```
<ImageView
```

```
    android:id="@+id/image_item"
    android:layout_width="0dp"
    android:layout_height="70dp"
    android:layout_weight="1" />
```

```
<LinearLayout
```

```
    android:layout_width="0dp"
    android:layout_height="match_parent"
    android:layout_weight="4"
    android:gravity="center"
    android:orientation="vertical">
```

```
<TextView
```

```
    android:id="@+id/title"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:gravity="bottom|left"
    android:textStyle="bold"
/>
```

```
<LinearLayout
```

```
    android:layout_width="0dp"
    android:layout_height="match_parent"
    android:layout_weight="4"
    android:gravity="center"
    android:orientation="vertical">
```

```
<TextView
```

```
    android:id="@+id/title"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:gravity="bottom|left"
    android:textStyle="bold"
/>
```

```
<TextView
```

```
    android:id="@+id/preview"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:gravity="top|left"
/>
```

```
</LinearLayout>
```

```
</LinearLayout>
```

ListItem (or Adapter)

- **It represents an item of the list**
- **The input of a list (items in the list) can be arbitrary Kotlin objects.**
 - The adapter extracts the correct data from the data object passed by the recyclerView and assigns this data to the views in the row of the ListView.
 - These items are typically called the data model of the list.
 - An adapter can receive data as input
 - An adapter manages the data model and adapts it to the individual entries in the widget
 - Every line in the widget displaying the data consists of a layout which can be as complex as you want. A typical line in a list has an image on the left side and two text lines in the middle as depicted in the following graphic.

MyAdapter Class

- Imports

```
import android.content.Context  
import android.media.Image  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.ImageView  
import android.widget.TextView  
import androidx.recyclerview.widget.RecyclerView
```



MyAdapter Class

```
class MyAdapter : RecyclerView.Adapter<MyAdapter.ViewHolder> {  
    private lateinit var context: Context  
    private var items: Array<MyElement>  
  
    constructor(items: Array<MyElement>) {  
        this.items = items  
    }  
  
    constructor(cont: Context, items: Array<MyElement>) : super() {  
        this.items = items  
        context = cont  
    }  
  
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {...}  
    override fun onBindViewHolder(holder: ViewHolder, position: Int) {...}  
  
    override fun getItemCount(): Int {...}  
  
    public class ViewHolder constructor(itemView: View) : RecyclerView.ViewHolder(itemView) {...}
```

It needs a variable for the data

The Adapter needs a class telling it how to map the data into the layout

These two methods create the view and bind a new ViewHolder to the view

called when the data is to be displayed for the nth element



MyAdapter Class

These two methods create the view and bind a new ViewHolder to the view

```
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {  
    //Inflate the layout, initialize the View Holder  
    val v: View = LayoutInflater.from(parent.context).inflate(  
        R.layout.list_item_image,  
        parent, attachToRoot: false  
    )  
    return ViewHolder(v)  
}  
  
override fun onBindViewHolder(holder: ViewHolder, position: Int) {  
    //Use the provided View Holder on the onCreateViewHolder method to populate the  
    // current row on the RecyclerView  
    if (items[position] != null) {  
        holder.title.text = items[position].title  
        holder.preview.text = items[position].preview  
        holder.imageView.setImageResource(items[position].image)  
    }  
    //animate(holder);  
}
```

Inner class ViewHolder

called when the data is to be displayed
for the nth element

The Adapter needs a class
telling it how
to map the data into the layout

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

```
public class ViewHolder constructor(itemView: View) : RecyclerView.ViewHolder(itemView) {  
    lateinit var image: Image  
    var title: TextView = itemView.findViewById<View>(R.id.title) as TextView  
    var preview: TextView = itemView.findViewById<View>(R.id.preview) as TextView  
    var imageView: ImageView = itemView.findViewById<View>(R.id.image_item) as ImageView  
}
```


In MainActivity

- onCreateOptionsMenu
- onOptionsItemSelected

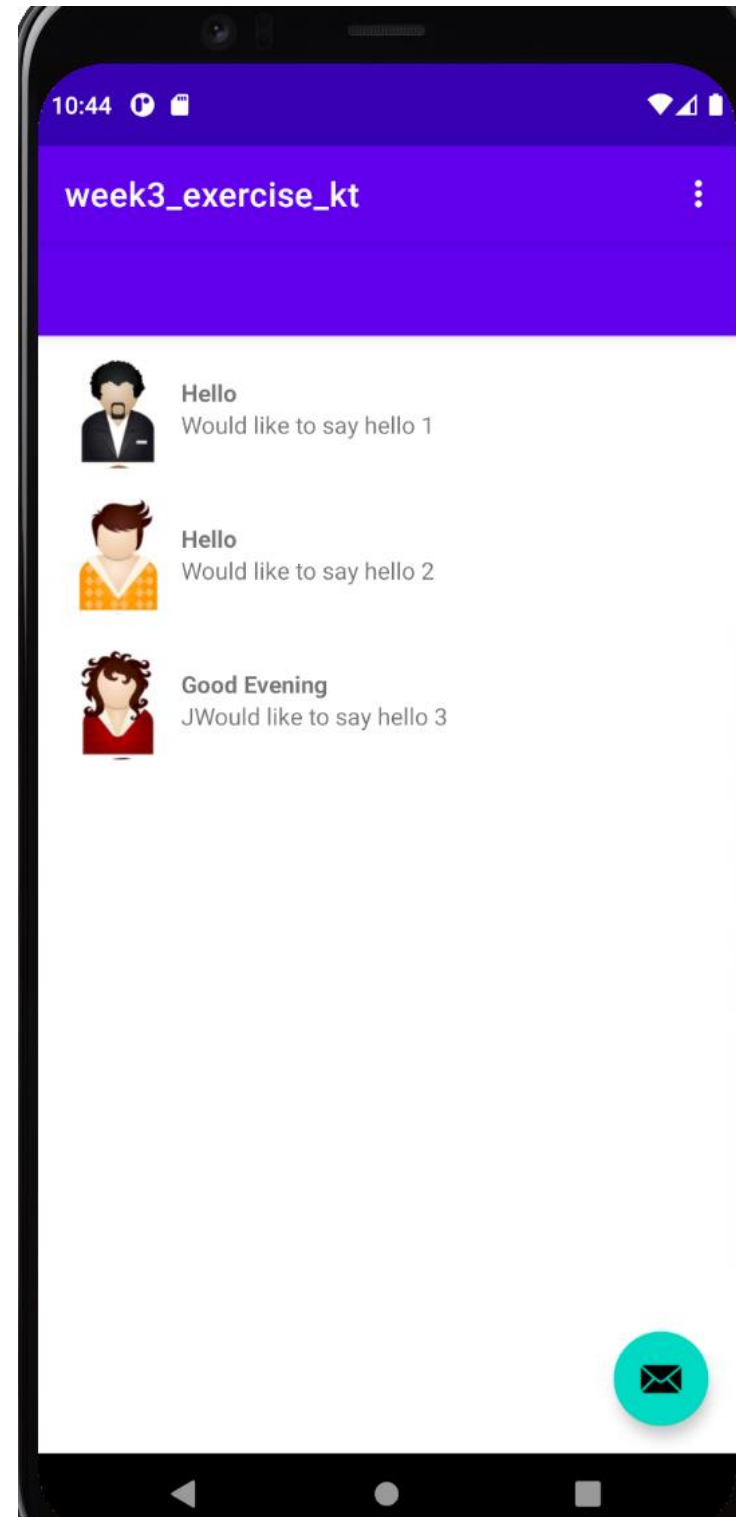
```
override fun onCreateOptionsMenu(menu: Menu?): Boolean {  
    // Inflate the menu; this adds items to the action bar if it is present.  
    menuInflater.inflate(R.menu.menu_main, menu)  
    return true  
}
```

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {  
    // Handle action bar item clicks here. The action bar will  
    // automatically handle clicks on the Home/Up button, so long  
    // as you specify a parent activity in AndroidManifest.xml.  
    val id = item.itemId  
    return if (id == R.id.action_settings) {  
        true  
    } else super.onOptionsItemSelected(item)  
}
```



The
University
Of
Sheffield.

Try it



Try it

- The solution is already on Mole but do not open it until you have finished or are stuck
 - First ask one of us, though
- If you still have time, try to add more images under Drawables and more text
- Try to modify the margins and the size of the images