Practical class # 10 – RecyclerView 2

1) Download the IMAGE.zip file form the course website.

Copy the images from your computer into your project's res > drawable folder. You can refer to these images with R.drawable.image1. (folder \app\src\main\res\drawable).

2) Add support for images in the Affirmation class

1. Open the Affirmation.kt file within the model package.
2. Modify the constructor of the Affirmation class by adding another Int parameter named imageResourceId

Both stringResourceId and imageResourceId are integer values. Although this looks okay, the caller could accidentally pass in the arguments in the wrong order (imageResourceId first instead of stringResourceId).

To avoid this, you can use Resource annotations. Annotations are useful because they add additional info to classes, methods, or parameters. Annotations are always declared with an @ symbol. In this case, add the @StringRes annotation to your string resource ID property and @DrawableRes annotation to your drawable resource ID property. Then you will get a warning if you supply the wrong type of resource ID.

Add the @StringRes annotation to stringResourceId.
Add the @DrawableRes annotation to imageResourceId.
Make sure the imports androidx.annotation.DrawableRes and androidx.annotation.StringRes are added at the top of your file after the package declaration.

```
data class Affirmation(
        @StringRes val stringResourceId: Int,
        @DrawableRes val imageResourceId: Int
)
```

3) Initialize list of affirmations with images

1. Open Datasource.kt. You should see an error for each instantiation of Affirmation.
2. For each Affirmation, add the resource ID of an image as an argument, such as R.drawable.image1

```
Affirmation(R.string.affirmation1, R.drawable.image1),
```

4) Add an ImageView to the list item layout

1. Open res > layout > list_item.xml. Add a LinearLayout around the existing TextView and set the orientation property to vertical.

2. Move the xmlns schema declaration line from the TextView element to the LinearLayout element to get rid of the error.

```
<?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:orientation="vertical">

   <TextView
      android:id="@+id/item_title"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />

</LinearLayout>
```

3. Inside the LinearLayout, before the TextView, add an ImageView with a resource ID of item_image.

```
<ImageView
      android:layout_width="match_parent"
      android:layout_height="194dp"
      android:id="@+id/item_image"
      android:importantForAccessibility="no"
      android:scaleType="centerCrop" />
```

5) Update the ItemAdapter to set the image

1. Update the ItemViewHolder constructor

```
class ItemViewHolder(private val view: View): RecyclerView.ViewHolder(view) {
   val textView: TextView = view.findViewById(R.id.item_title)
   val imageView: ImageView = view.findViewById(R.id.item_image)
}
```

2. Update the onBindViewHolder

```
override fun onBindViewHolder(holder: ItemViewHolder, position: Int) {
      val item = dataset[position]
      holder.textView.text = context.resources.getString(item.stringResourceId)
      holder.imageView.setImageResource(item.imageResourceId)
   }
```

6) Polish the UI

1. Open item_list.xml (app > res > layout > item_list.xml) and add 16dp padding to the existing LinearLayout.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:orientation="vertical"
  android:padding="16dp">
```

2. change text appearance

More details about text appearances:
https://medium.com/androiddevelopers/android-styling-common-theme-attributes-8f7c50c9eaba

```
<TextView
    android:id="@+id/item_title"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="16dp"
    android:textAppearance="?attr/textAppearanceHeadline6" />
```

3. Use cards

```
<?xml version="1.0" encoding="utf-8"?>
<com.google.android.material.card.MaterialCardView
xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:layout_margin="8dp">

  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <ImageView
      android:id="@+id/item_image"
      android:layout_width="match_parent"
      android:layout_height="194dp"
      android:importantForAccessibility="no"
      android:scaleType="centerCrop" />

    <TextView
      android:id="@+id/item_title"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:padding="16dp"
      android:textAppearance="?attr/textAppearanceHeadline6" />

  </LinearLayout>

</com.google.android.material.card.MaterialCardView>
```

4. Change the app theme colors

More colors can be created at

https://m2.material.io/resources/color/#!/?view.left=0&view.right=0

Open colors.xml (res > values > colors.xml).

Add new color resources to the file for the blue colors defined below

```
<color name="blue_200">#FF90CAF9</color>
<color name="blue_500">#FF2196F3</color>
<color name="blue_700">#FF1976D2</color>
```

Open themes.xml (res > values > themes > themes.xml).

Find the <!-- Primary brand color. --> section.

Add or change colorPrimary to use @color/blue_500.

Add or change colorPrimaryVariant to use @color/blue_700

Use blu2_200 and blue_500 for the dark theme (themes.xml (night))