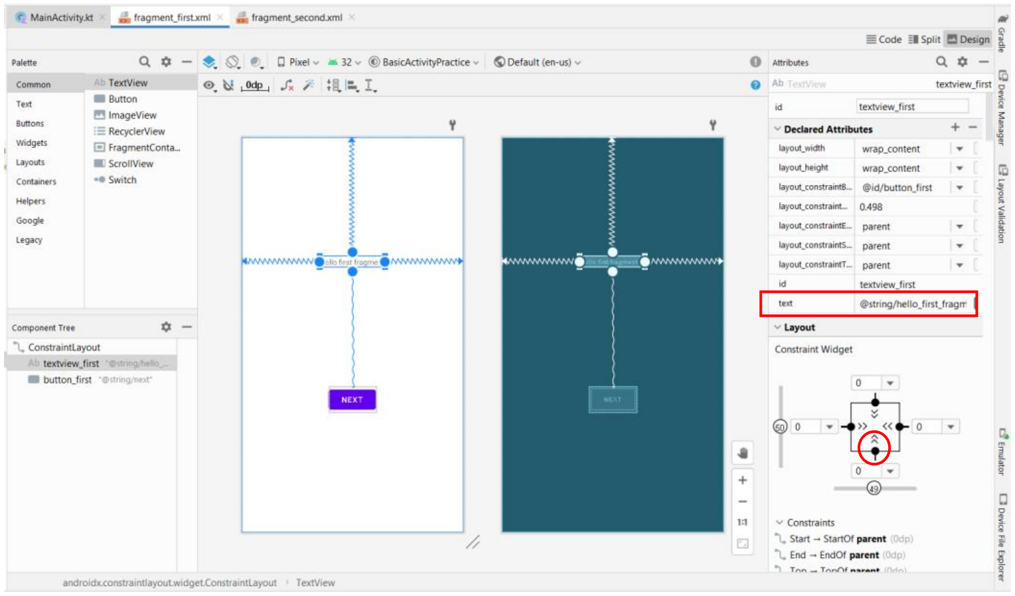
Practice – Guide 04

- Open Android Studio and create a new project with a “Basic Activity”.
- Give a name for the project, package name and save it in an appropriate folder. After the project has successfully build, open the “Android” project view.

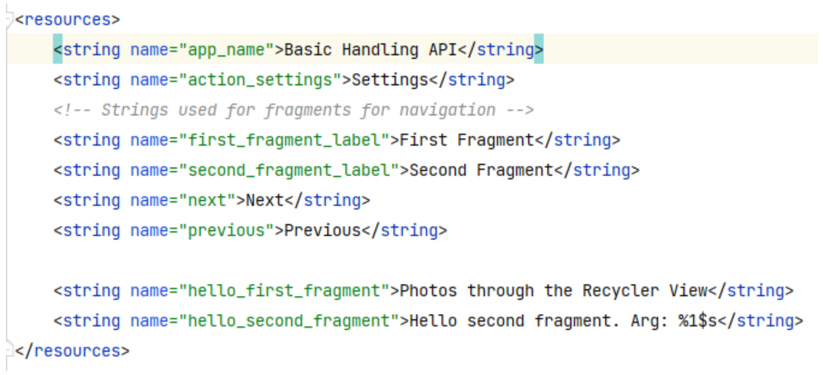
Design fragment_first.xml

- Go to fragment_first.xml and let’s adjust the text view. To adjust we have to remove any constraint bond with another. Here let’s we remove the bottom constraint of the text view. Then select the text view and click on the point (indicated in red colored circle in the figure below).



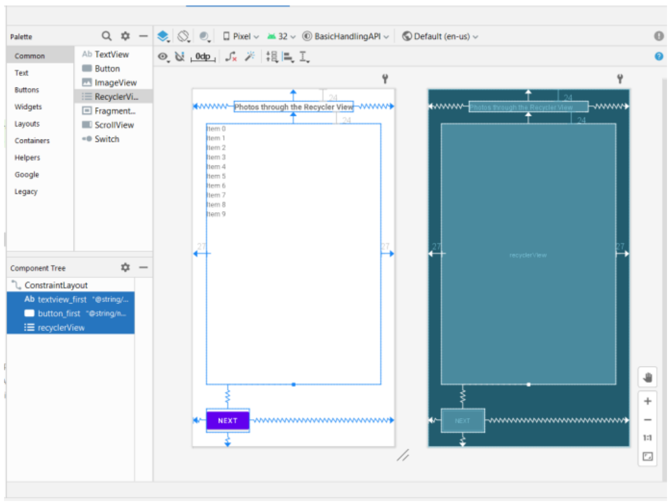
id	textview_first
Declared Attributes	
layout_width	wrap_content
layout_height	wrap_content
layout_constraintB...	@id/button_first
layout_constraint...	0.498
layout_constraintE...	parent
layout_constraintS...	parent
layout_constraintT...	parent
id	textview_first
text	@string/hello_first_fragm

- Now you can simply adjust textView.
- Next, to change the already extracted string resource of the textView, you have to go to **res/values/string.xml** file. To do that press Ctrl key and click in the text attribute’s value (Red coloured rectangle in the above figure). Then, change the String to **“Photos through Recycler View”**.



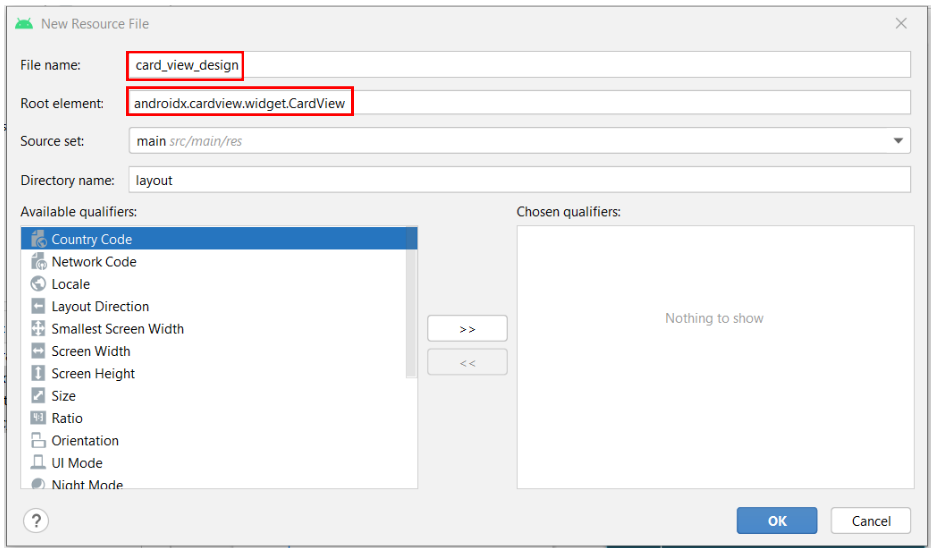
```xml
<resources>
    <string name="app_name">Basic Handling API</string>
    <string name="action_settings">Settings</string>
    <!-- Strings used for fragments for navigation -->
    <string name="first_fragment_label">First Fragment</string>
    <string name="second_fragment_label">Second Fragment</string>
    <string name="next">Next</string>
    <string name="previous">Previous</string>

    <string name="hello_first_fragment">Photos through the Recycler View</string>
    <string name="hello_second_fragment">Hello second fragment. Arg: %1$s</string>
</resources>
```

- Now add a recyclerView on to the “fragment_first” and set the constraints.
- Adjust just as below.



<u>Now we are going to create the redundant layout that should load into the recyclerView.</u>

- Go to **res/layout** folder and right click on it. Then, choose **New->Layout Resource File**
- Add a new layout file as “card_view_design” and design as below.


New Resource File

File name: card_view_design

Root element: androidx.cardview.widget.CardView

Source set: main src/main/res

Directory name: layout

Available qualifiers:

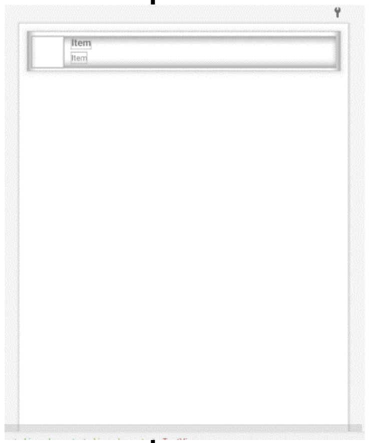
Country Code
Network Code
Locale
Layout Direction
Smallest Screen Width
Screen Width
Screen Height
Size
Ratio
Orientation
UI Mode
Night Mode

Chosen qualifiers:

Nothing to show

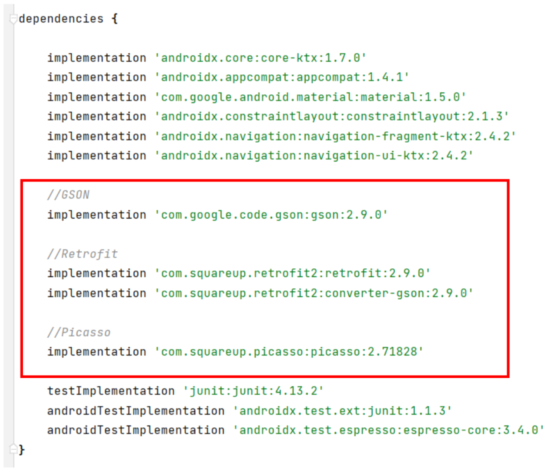
- For the convenience of you during this study leave time, the code for designing UI **card_view_design.xml** attractively given below.

```xml
<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"
    android:layout_height="50dp"
    android:layout_margin="10dp"
    app:cardElevation="6dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:padding="5dp">

        <ImageView
            android:id="@+id/imageview"
            android:layout_width="40dp"
            android:layout_height="40dp" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical">
            <TextView
                android:id="@+id/textView_title"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginStart="10dp"
                android:layout_marginLeft="15dp"
                android:text="Item"
                android:textSize="12dp"
                android:textStyle="bold" />

            <TextView
                android:id="@+id/textview_url"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginStart="10dp"
                android:layout_marginLeft="15dp"
                android:layout_marginTop="4dp"
                android:text="Item"

                android:textSize="10dp"/>
        </LinearLayout>

    </LinearLayout>

</androidx.cardview.widget.CardView>
```

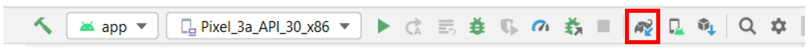


Configuration of the project

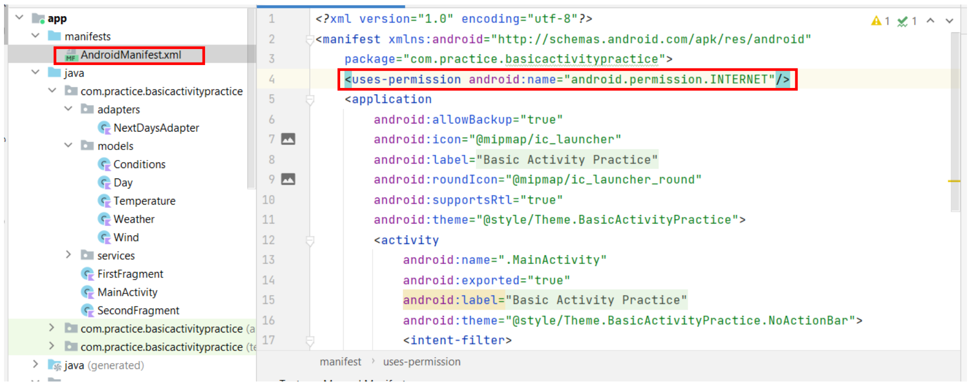
- Go to **build.gradle** file and add the dependencies for GSON, Retrofit and RetrofitGSON Convertor Factory and Piccasso under dependencies. Then, **gradle sync** to sync dependencies. NOTE: Remember to sync after adding any dependency



```
dependencies {

    implementation 'androidx.core:core-ktx:1.7.0'
    implementation 'androidx.appcompat:appcompat:1.4.1'
    implementation 'com.google.android.material:material:1.5.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.3'
    implementation 'androidx.navigation:navigation-fragment-ktx:2.4.2'
    implementation 'androidx.navigation:navigation-ui-ktx:2.4.2'

    //GSON
    implementation 'com.google.code.gson:gson:2.9.0'

    //Retrofit
    implementation 'com.squareup.retrofit2:retrofit:2.9.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'

    //Picasso
    implementation 'com.squareup.picasso:picasso:2.71828'

    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
}
```



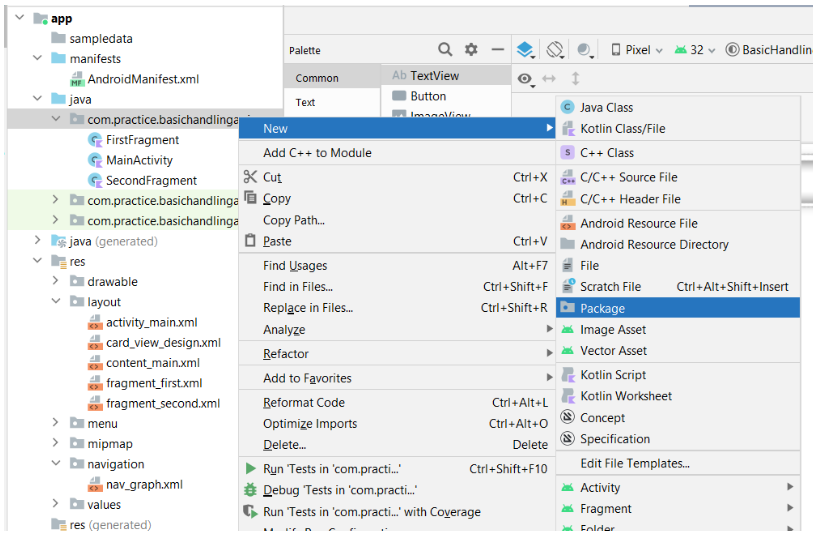
- Now since we are using an online resource, we need to give permission to the app to access Internet. Therefore, following line of code should put in Manifest.xml file after **<manifest>** tag and before starting **<application>** tag.

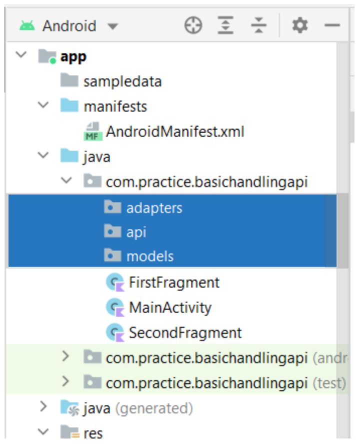


```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.practice.basicactivitypractice">
    <uses-permission android:name="android.permission.INTERNET"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Basic Activity Practice"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.BasicActivityPractice">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="Basic Activity Practice"
            android:theme="@style/Theme.BasicActivityPractice.NoActionBar">
            <intent-filter>
```

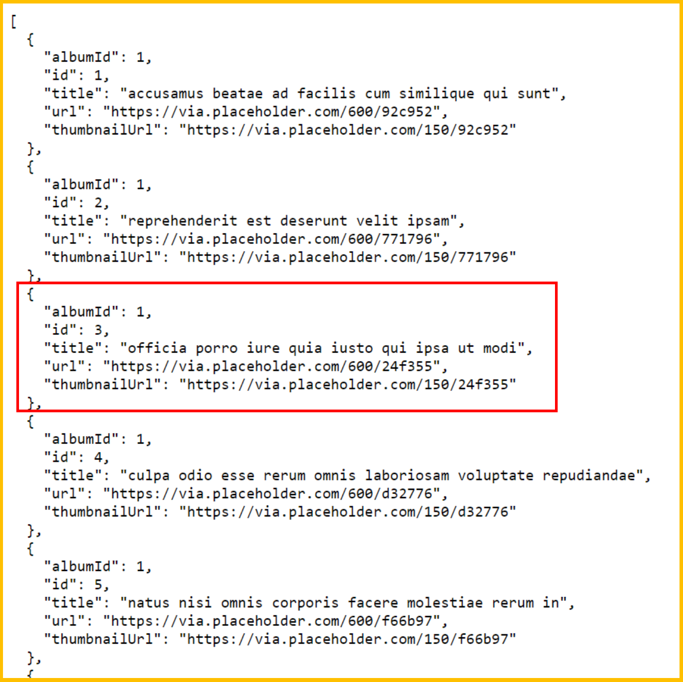
Programming part

- Let's create three new packages as "adapters", "api" & "models".
    - **adapters**: include adapter classes to handle data within recyclerView
    - **api**: include API configuration class to access API and manipulate them
    - **models**: include model data classes so that the type of data in API we access.



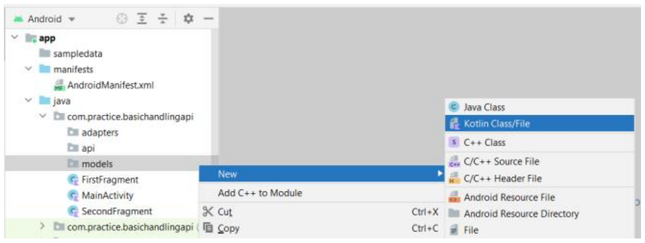


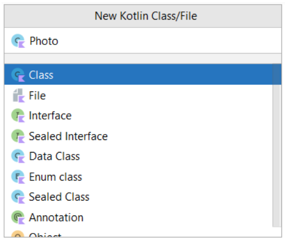
- Let's create **data classes** in "**models**" package.
- This is part of the API data that we are accessing.



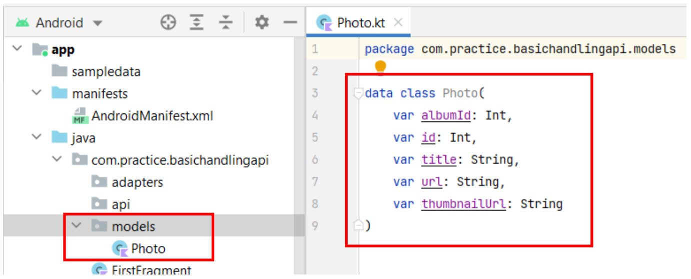
```
[
  {
    "albumId": 1,
    "id": 1,
    "title": "accusamus beatae ad facilis cum similique qui sunt",
    "url": "https://via.placeholder.com/600/92c952",
    "thumbnailUrl": "https://via.placeholder.com/150/92c952"
  },
  {
    "albumId": 1,
    "id": 2,
    "title": "reprehenderit est deserunt velit ipsam",
    "url": "https://via.placeholder.com/600/771796",
    "thumbnailUrl": "https://via.placeholder.com/150/771796"
  },
  {
    "albumId": 1,
    "id": 3,
    "title": "officia porro iure quia iusto qui ipsa ut modi",
    "url": "https://via.placeholder.com/600/24f355",
    "thumbnailUrl": "https://via.placeholder.com/150/24f355"
  },
  {
    "albumId": 1,
    "id": 4,
    "title": "culpa odio esse rerum omnis laboriosam voluptate repudiandae",
    "url": "https://via.placeholder.com/600/d32776",
    "thumbnailUrl": "https://via.placeholder.com/150/d32776"
  },
  {
    "albumId": 1,
    "id": 5,
    "title": "natus nisi omnis corporis facere molestiae rerum in",
    "url": "https://via.placeholder.com/600/f66b97",
    "thumbnailUrl": "https://via.placeholder.com/150/f66b97"
  },
  {
```

- Since we are accessing "albumId", "id", "title", "url" & "thumbnailUrl" as one object, then, we can create just one data class as "Photo".



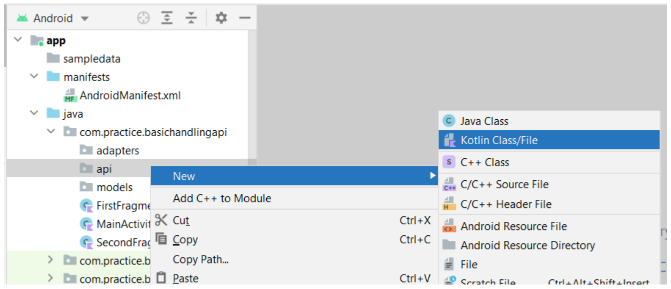


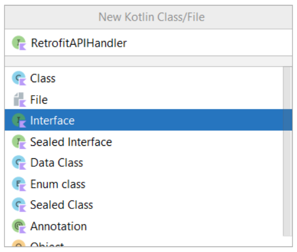
NOTE:
Here the variable names and data types must be same with the properties of the API data object in the API.

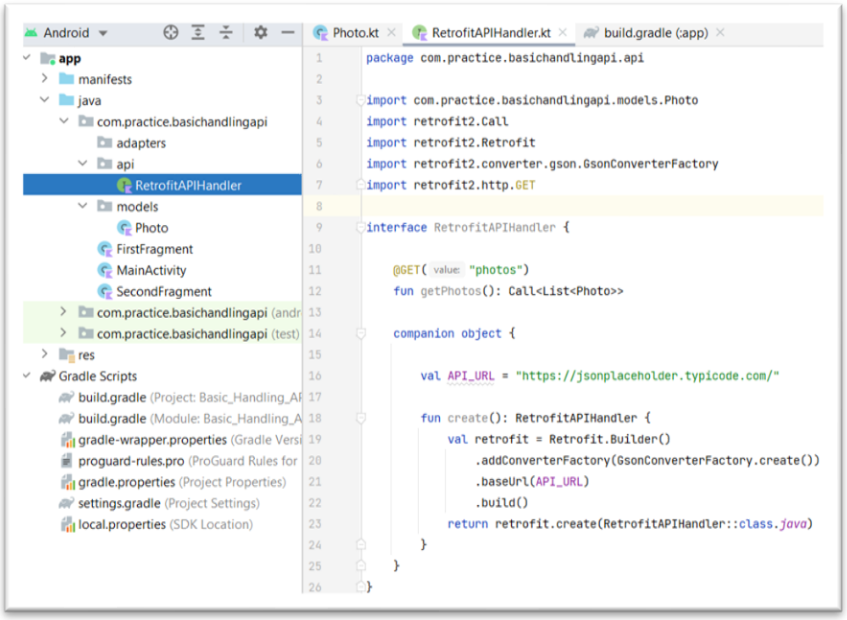


```kotlin
package com.practice.basichandlingapi.models

data class Photo(
    var albumId: Int,
    var id: Int,
    var title: String,
    var url: String,
    var thumbnailUrl: String
)
```

- Let's create the interface to deal with API in the "api" package.






Android
app
manifests
java
com.practice.basichandlingapi
adapters
api
RetrofitAPIHandler
models
Photo
FirstFragment
MainActivity
SecondFragment
com.practice.basichandlingapi (andr
com.practice.basichandlingapi (test)
res
Gradle Scripts
build.gradle (Project: Basic_Handling_AF
build.gradle (Module: Basic_Handling_A
gradle-wrapper.properties (Gradle Versi
proguard-rules.pro (ProGuard Rules for
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
Photo.kt
RetrofitAPIHandler.kt
build.gradle (:app)

```kotlin
package com.practice.basichandlingapi.api

import com.practice.basichandlingapi.models.Photo
import retrofit2.Call
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory
import retrofit2.http.GET

interface RetrofitAPIHandler {

    @GET( value: "photos")
    fun getPhotos(): Call<List<Photo>>

    companion object {

        val API_URL = "https://jsonplaceholder.typicode.com/"

        fun create(): RetrofitAPIHandler {
            val retrofit = Retrofit.Builder()
                .addConverterFactory(GsonConverterFactory.create())
                .baseUrl(API_URL)
                .build()
            return retrofit.create(RetrofitAPIHandler::class.java)
        }
    }
}
```

NOTE:

1) **If the endpoint is just like,**

   https://jsonplaceholder.typicode.com/**photos**

   Then,

```
@GET( value: "photos")
fun getPhotos(): Call<List<Photo>>
```

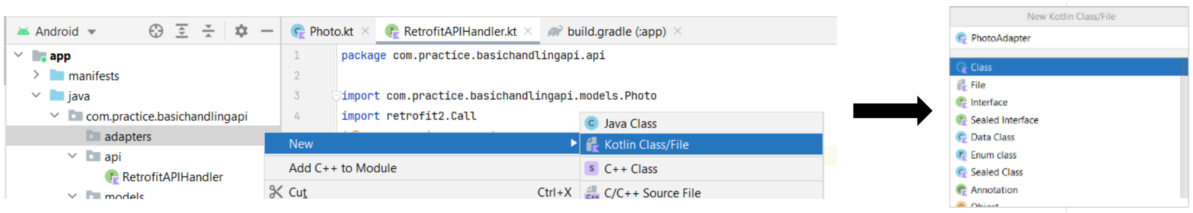
2) **If the endpoint is just like,**

   https://jsonplaceholder.typicode.com/**posts/1/comment**

   Then,

```
@GET( value: "posts/{postId}/comments")
fun getCommentsForPostId(@Path( value: "postId")id:String):Call<List<Comments>>
```

- Let's create the adapter class, **PhotoAdapter** in the "**adapter**" package for the recyclerView.



```
package com.practice.basichandlingapi.api

import com.practice.basichandlingapi.models.Photo
import retrofit2.Call
```

Step 01

- In PhotoAdapter class constructor parameters we include,
  - private val photoList : List<Photo>
  - val context: Callback<List<Photo>>

  If we apply a click event for the recyclerView then, include this also as a parameter.

  - private val onItemClicked: (position: Int) -> Unit

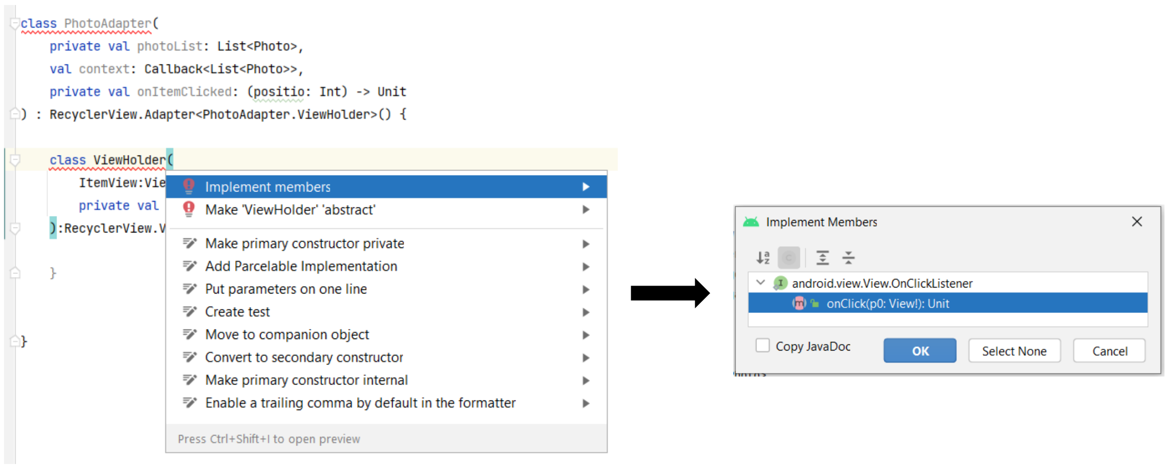
  (Keep in mind, if we do this we should create a function to handle clickEvent logic in the caller Fragment)

Step 02

- Create a class called as "**ViewHolder**" inside **PhotoAdapter** class. Include following parameters for its constructor.
  - ItemView: View
  - private val onItemIsClicked: (position: Int) -> Unit

- Now we need to do implementation
  - ➢ RecyclerView.ViewHolder(ItemView)
  - ➢ View.OnClickListener

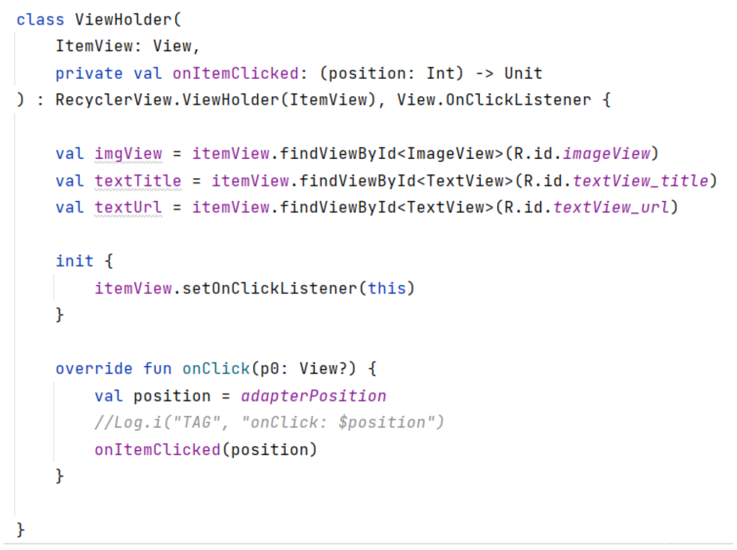
- Implement members
  - ▪ Go to the redline appear with ViewHolder. Click there and press Alt+Enter
  - ▪ Then implement members



```
class PhotoAdapter(
    private val photoList: List<Photo>,
    val context: Callback<List<Photo>>,
    private val onItemClicked: (positio: Int) -> Unit
) : RecyclerView.Adapter<PhotoAdapter.ViewHolder>() {

    class ViewHolder(
        ItemView:Vie
        private val
    ):RecyclerView.V

    }

}
```

Implement members
Make 'ViewHolder' 'abstract'
Make primary constructor private
Add Parcelable Implementation
Put parameters on one line
Create test
Move to companion object
Convert to secondary constructor
Make primary constructor internal
Enable a trailing comma by default in the formatter
Press Ctrl+Shift+I to open preview
Implement Members
android.view.View.OnClickListener
onClick(p0: View!): Unit
Copy JavaDoc
OK
Select None
Cancel

- Finally, class ViewHolder will look like this.



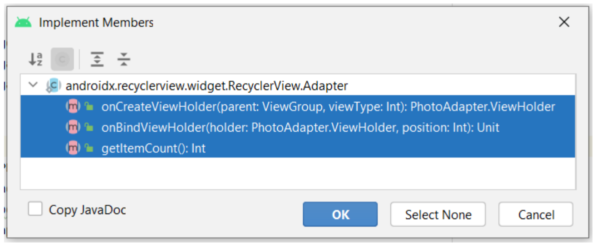
```
class ViewHolder(
    ItemView: View,
    private val onItemClicked: (position: Int) -> Unit
) : RecyclerView.ViewHolder(ItemView), View.OnClickListener {

    val imgView = itemView.findViewById<ImageView>(R.id.imageView)
    val textTitle = itemView.findViewById<TextView>(R.id.textView_title)
    val textUrl = itemView.findViewById<TextView>(R.id.textView_url)

    init {
        itemView.setOnClickListener(this)
    }

    override fun onClick(p0: View?) {
        val position = adapterPosition
        //Log.i("TAG", "onClick: $position")
        onItemClicked(position)
    }

}
```

Step 03 – continue step 01

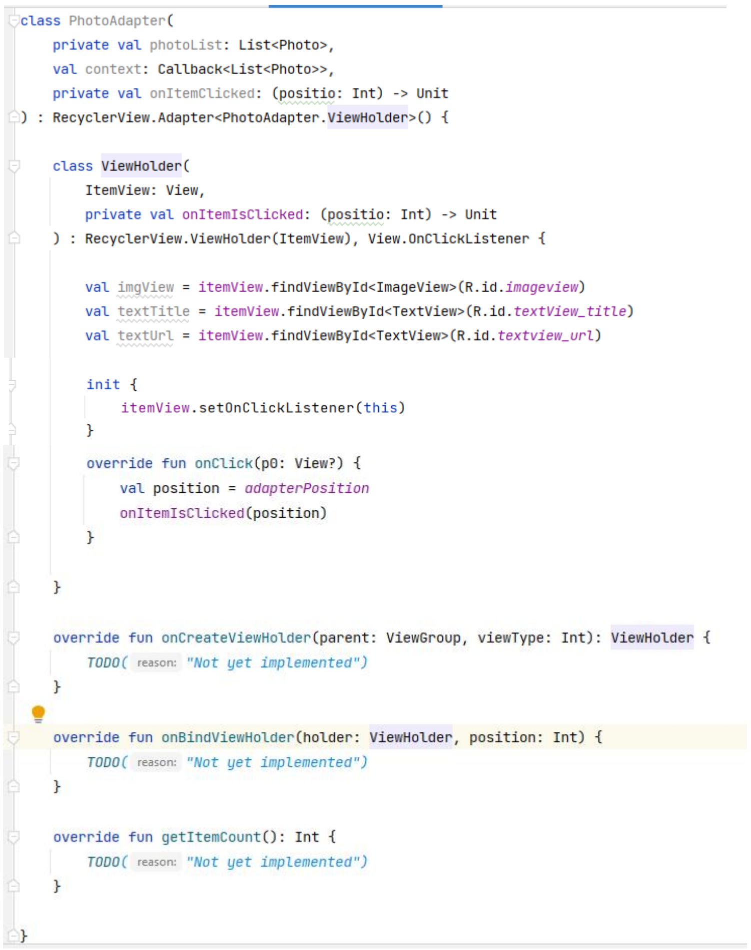
- Since this **PhotoAdapter class** is an adapter for the **recyclerView** then, we **implement RecyclerView.Adapter** class while **initializing its constructor and pass the current class' ViewHolder into its generic type**.

  - ➢ RecyclerView.Adapter<PhotoAdapter.ViewHolder>()

- Implement members
    - Go to the red-line appear with **PhotoAdapter**. Click there and press **Alt+Enter**
    - Then implement members
    - Select all and click OK.


Implement Members
androidx.recyclerview.widget.RecyclerView.Adapter
onCreateViewHolder(parent: ViewGroup, viewType: Int): PhotoAdapter.ViewHolder
onBindViewHolder(holder: PhotoAdapter.ViewHolder, position: Int): Unit
getItemCount(): Int
Copy JavaDoc
OK
Select None
Cancel

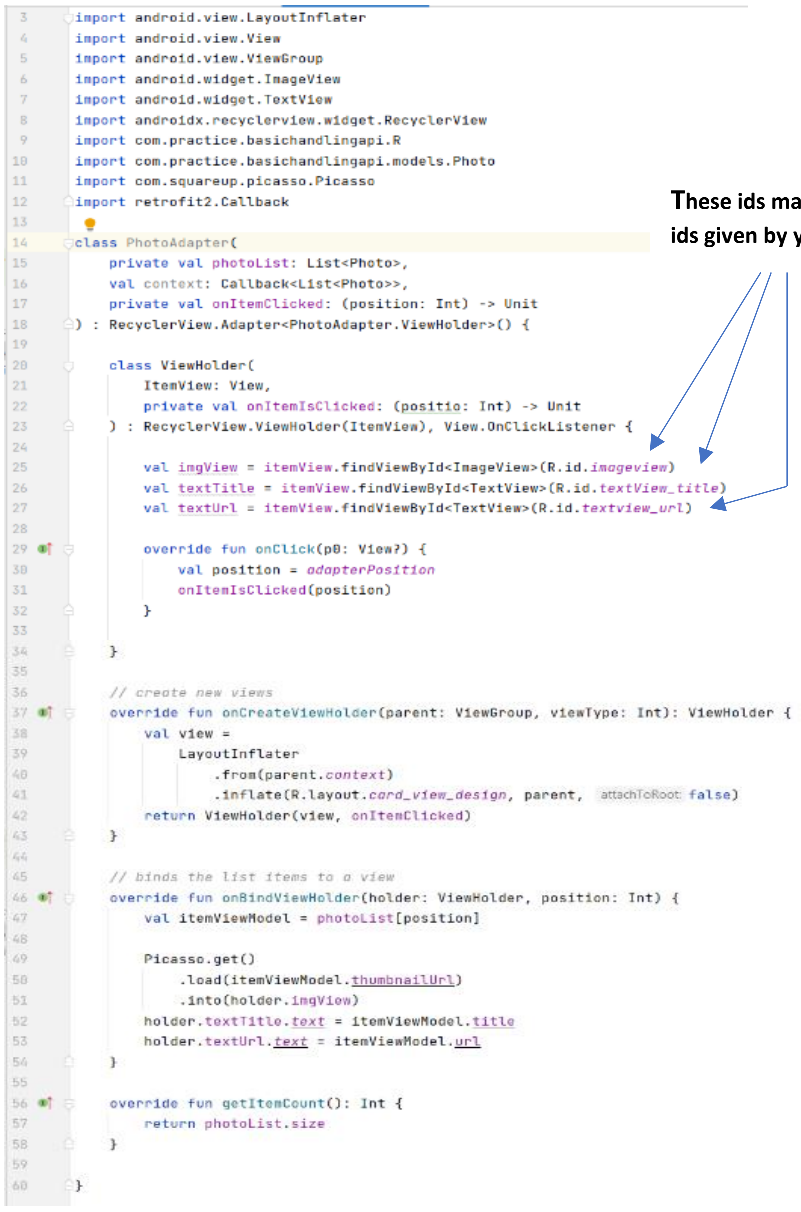
- Now it may look like this.



```
class PhotoAdapter(
    private val photoList: List<Photo>,
    val context: Callback<List<Photo>>,
    private val onItemClicked: (positio: Int) -> Unit
) : RecyclerView.Adapter<PhotoAdapter.ViewHolder>() {

    class ViewHolder(
        ItemView: View,
        private val onItemIsClicked: (positio: Int) -> Unit
    ) : RecyclerView.ViewHolder(ItemView), View.OnClickListener {

        val imgView = itemView.findViewById<ImageView>(R.id.imageview)
        val textTitle = itemView.findViewById<TextView>(R.id.textView_title)
        val textUrl = itemView.findViewById<TextView>(R.id.textview_url)

        init {
            itemView.setOnClickListener(this)
        }

        override fun onClick(p0: View?) {
            val position = adapterPosition
            onItemIsClicked(position)
        }

    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
        TODO( reason: "Not yet implemented")
    }

    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        TODO( reason: "Not yet implemented")
    }

    override fun getItemCount(): Int {
        TODO( reason: "Not yet implemented")
    }

}
```

- In the overridden method onCreateViewHolder, create new views including the **card_view_design** and return a ViewHolder object.
- In the overridden method onBindViewHolder, bind the data in the list to the ViewHolder objects just returned.
  (Such a similar way we can understand how each cardView is appeared in the recyclerView).
- In the overridden method getItemCount(), return the size of the list we passed to the constructor of the PhotoAdapter class.

- Finally, **PhotoAdapter** looks like below.



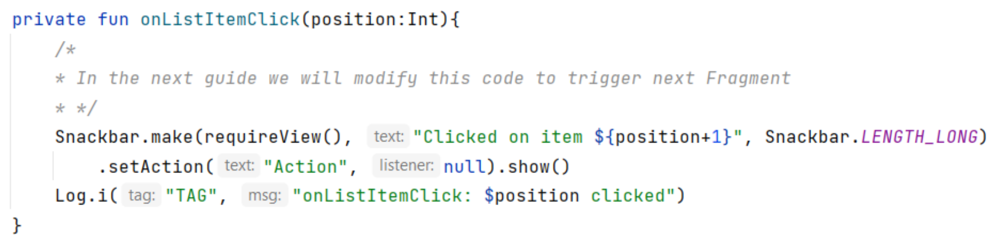
```kotlin
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import com.practice.basichandlingapi.R
import com.practice.basichandlingapi.models.Photo
import com.squareup.picasso.Picasso
import retrofit2.Callback

class PhotoAdapter(
    private val photoList: List<Photo>,
    val context: Callback<List<Photo>>,
    private val onItemClicked: (position: Int) -> Unit
) : RecyclerView.Adapter<PhotoAdapter.ViewHolder>() {

    class ViewHolder(
        ItemView: View,
        private val onItemIsClicked: (positio: Int) -> Unit
    ) : RecyclerView.ViewHolder(ItemView), View.OnClickListener {

        val imgView = itemView.findViewById<ImageView>(R.id.imageview)
        val textTitle = itemView.findViewById<TextView>(R.id.textView_title)
        val textUrl = itemView.findViewById<TextView>(R.id.textview_url)

        override fun onClick(p0: View?) {
            val position = adapterPosition
            onItemIsClicked(position)
        }

    }

    // create new views
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
        val view =
            LayoutInflater
                .from(parent.context)
                .inflate(R.layout.card_view_design, parent,  attachToRoot: false)
        return ViewHolder(view, onItemClicked)
    }

    // binds the list items to a view
    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        val itemViewModel = photoList[position]

        Picasso.get()
            .load(itemViewModel.thumbnailUrl)
            .into(holder.imgView)
        holder.textTitle.text = itemViewModel.title
        holder.textUrl.text = itemViewModel.url
    }

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

}
```

These ids may differ according to ids given by you for those widgets

Code what should happen on the First Fragment:

Calling the API and response to the API Callback functions

- Go to FirstFragment.kt file.
- Create a private function to what to happen when the click event is triggered on an item in the recyclerView.

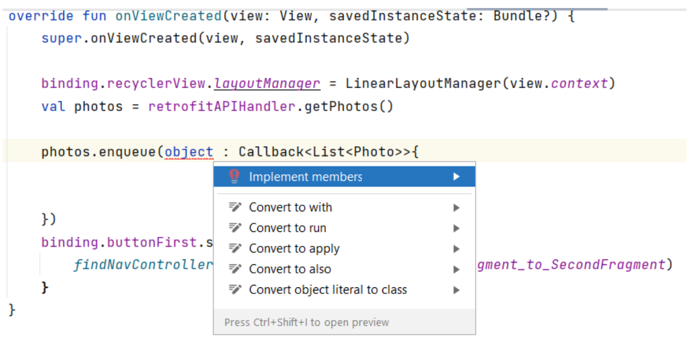


```kotlin
private fun onListItemClick(position:Int){
    /*
    * In the next guide we will modify this code to trigger next Fragment
    * */
    Snackbar.make(requireView(), text: "Clicked on item ${position+1}", Snackbar.LENGTH_LONG)
        .setAction( text: "Action", listener: null).show()
    Log.i( tag: "TAG", msg: "onListItemClick: $position clicked")
}
```

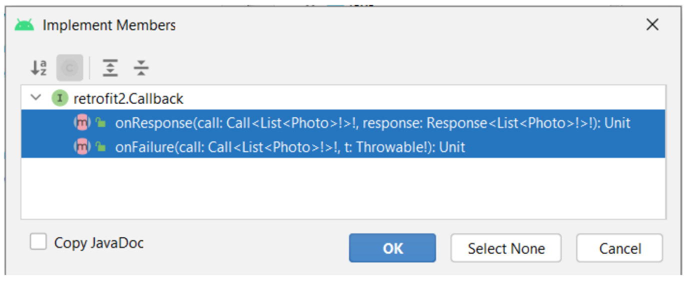
- Assign RetrofitAPIHandler instance to a private variable.

```kotlin
private val retrofitAPIHandler = RetrofitAPIHandler.create()
```

- Now within onVIewCreated() method we do,
  - ➢ Set the vertical LinearLayoutManager to the layoutManager of the recyclerView using binding.
  - ➢ Using the previous reference (**retrofitAPIHandler**), call the **getPhotos()** method and assign the result to another variable.
  - ➢ Then using that reference call **enqueue** method and implement members (**onResponse() and onFailure()**) just as in previous steps.

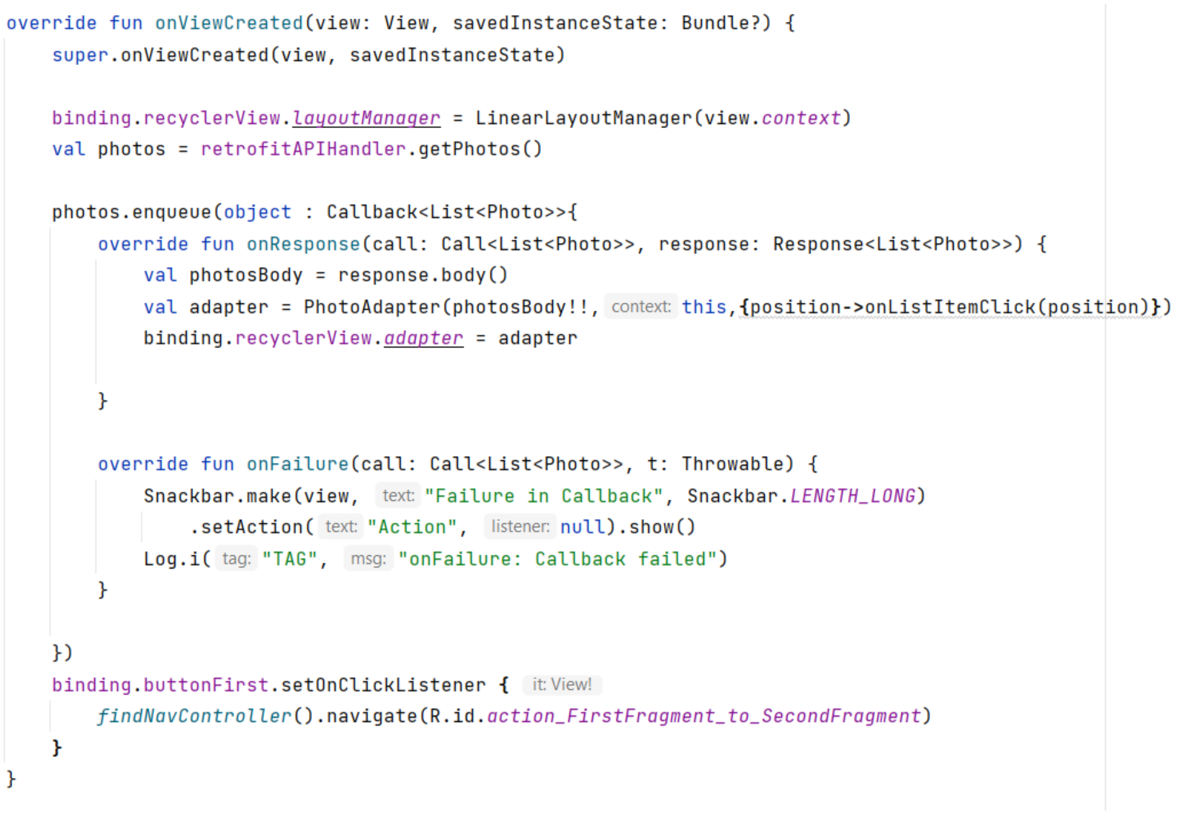


```kotlin
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    binding.recyclerView.layoutManager = LinearLayoutManager(view.context)
    val photos = retrofitAPIHandler.getPhotos()

    photos.enqueue(object : Callback<List<Photo>>{


    })
    binding.buttonFirst.s                    gment_to_SecondFragment)
        findNavController
    }
}
```



onResponse(call: Call<List<Photo>!>!, response: Response<List<Photo>!>!): Unit
onFailure(call: Call<List<Photo>!>!, t: Throwable!): Unit

- Finally, **onViewCreated()** method will look like this.



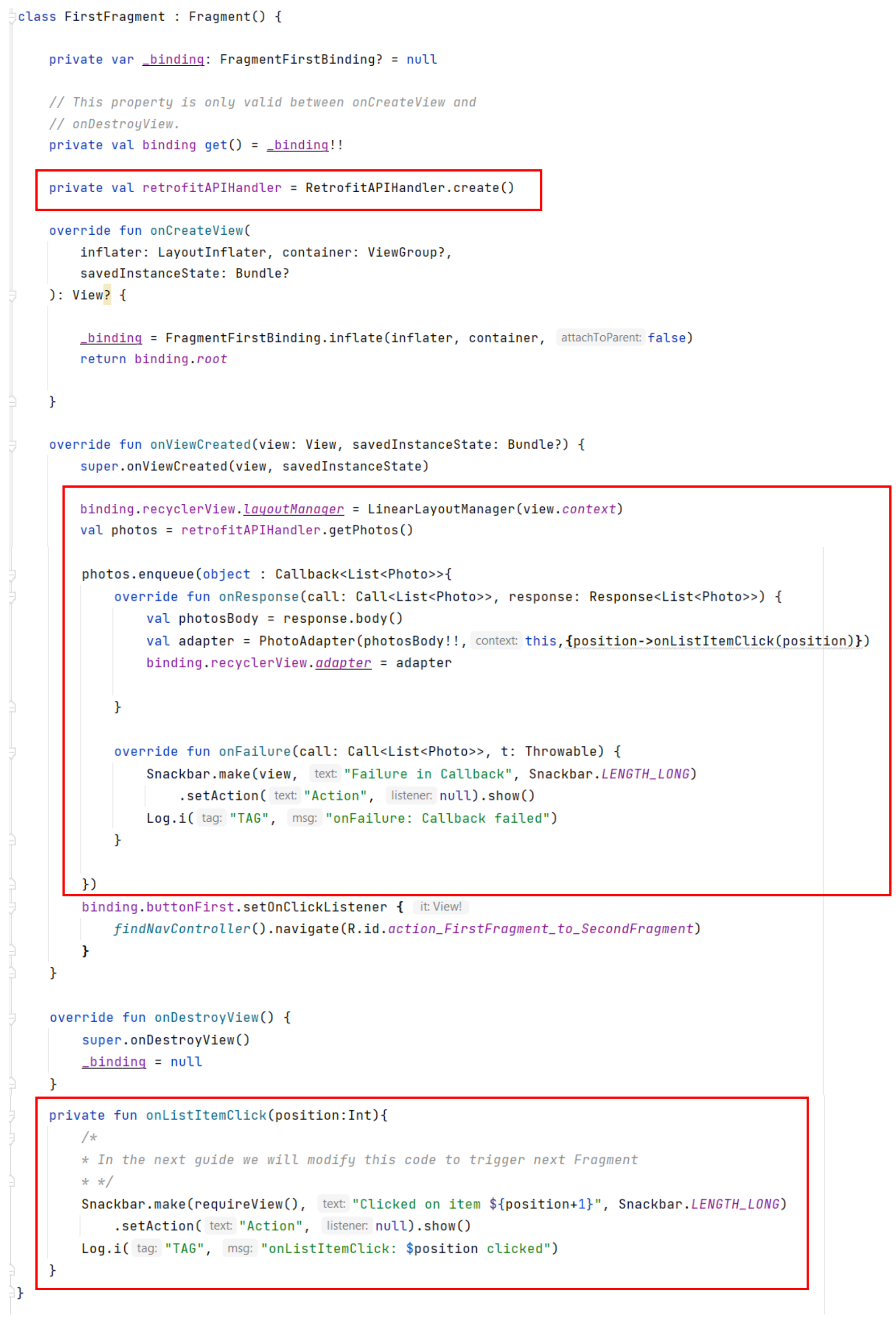
```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    binding.recyclerView.layoutManager = LinearLayoutManager(view.context)
    val photos = retrofitAPIHandler.getPhotos()

    photos.enqueue(object : Callback<List<Photo>>{
        override fun onResponse(call: Call<List<Photo>>, response: Response<List<Photo>>) {
            val photosBody = response.body()
            val adapter = PhotoAdapter(photosBody!!, context: this,{position->onListItemClick(position)})
            binding.recyclerView.adapter = adapter

        }

        override fun onFailure(call: Call<List<Photo>>, t: Throwable) {
            Snackbar.make(view,  text: "Failure in Callback", Snackbar.LENGTH_LONG)
                .setAction( text: "Action",  listener: null).show()
            Log.i( tag: "TAG",  msg: "onFailure: Callback failed")
        }

    })
    binding.buttonFirst.setOnClickListener {  it: View!
        findNavController().navigate(R.id.action_FirstFragment_to_SecondFragment)
    }
}
```

NOTE:

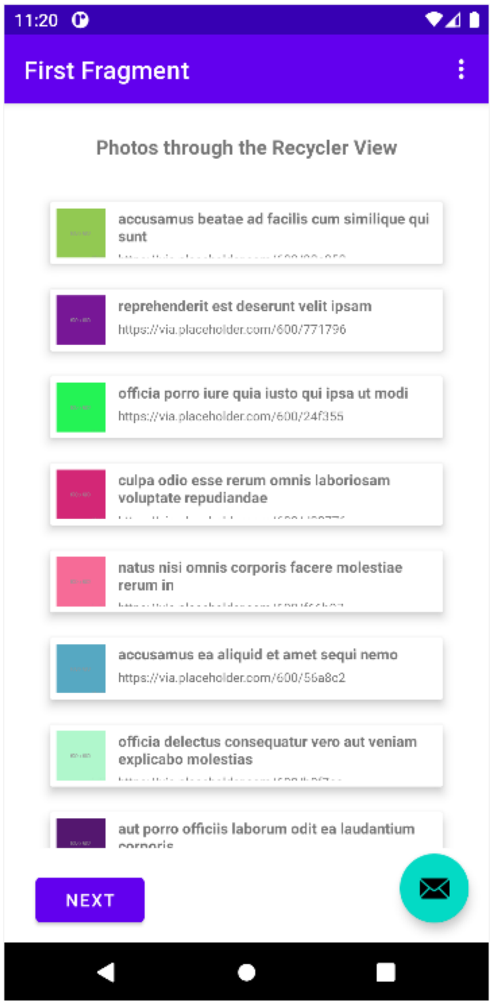
- Previously created **onListItemClicked** method will pass into the constructor of **PhotoAdapter** class.

- Extra Knowledge for future in Android Development:
    - Different types of functions in Kotlin.
    - Different types of Adapters in android.

➢ Finally, your code of **class FirstFragment** will look like this.



```
class FirstFragment : Fragment() {

    private var _binding: FragmentFirstBinding? = null

    // This property is only valid between onCreateView and
    // onDestroyView.
    private val binding get() = _binding!!

    private val retrofitAPIHandler = RetrofitAPIHandler.create()

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {

        _binding = FragmentFirstBinding.inflate(inflater, container,  attachToParent: false)
        return binding.root

    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        binding.recyclerView.layoutManager = LinearLayoutManager(view.context)
        val photos = retrofitAPIHandler.getPhotos()

        photos.enqueue(object : Callback<List<Photo>>{
            override fun onResponse(call: Call<List<Photo>>, response: Response<List<Photo>>) {
                val photosBody = response.body()
                val adapter = PhotoAdapter(photosBody!!,  context: this,{position->onListItemClick(position)})
                binding.recyclerView.adapter = adapter

            }

            override fun onFailure(call: Call<List<Photo>>, t: Throwable) {
                Snackbar.make(view,  text: "Failure in Callback", Snackbar.LENGTH_LONG)
                    .setAction( text: "Action",  listener: null).show()
                Log.i( tag: "TAG",  msg: "onFailure: Callback failed")
            }

        })
        binding.buttonFirst.setOnClickListener {  it: View!
            findNavController().navigate(R.id.action_FirstFragment_to_SecondFragment)
        }
    }

    override fun onDestroyView() {
        super.onDestroyView()
        _binding = null
    }
    private fun onListItemClick(position:Int){
        /*
        * In the next guide we will modify this code to trigger next Fragment
        * */
        Snackbar.make(requireView(),  text: "Clicked on item ${position+1}", Snackbar.LENGTH_LONG)
            .setAction( text: "Action",  listener: null).show()
        Log.i( tag: "TAG",  msg: "onListItemClick: $position clicked")
    }
}
```



Photos through the Recycler View

accusamus beatae ad facilis cum similique qui sunt

reprehenderit est deserunt velit ipsam
https://via.placeholder.com/600/771796

officia porro iure quia iusto qui ipsa ut modi
https://via.placeholder.com/600/24f355

culpa odio esse rerum omnis laboriosam voluptate repudiandae

natus nisi omnis corporis facere molestiae rerum in

accusamus ea aliquid et amet sequi nemo
https://via.placeholder.com/600/56a8c2

officia delectus consequatur vero aut veniam explicabo molestias

aut porro officiis laborum odit ea laudantium

- Here we just used FirstFragment only.
- When an item is clicked then a snackbar message just appeared indicating what item you have clicked.
- In the next guide we will pass current fragment data to the next fragment.
- To do that we will modify the **onListItemClicked** method in the **FirstFragment** here.

Now you know how to,

- Design the UI
- Configuration of project.
- How to use and apply simple APIs.
- Handling logic to manipulate and bind data to the application.
- Implement RecyclerView along with another layout.
- Embedding clickListeners to the redundant layouts in the recyclerView.

Resources:

- **GSON** related:

```
implementation 'com.google.code.gson:gson:2.9.0'
```

- **Retrofit** related :

```
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
```

- **Picasso** related (for Images):

```
implementation 'com.squareup.picasso:picasso:2.71828'
```

Let’s continue this in the next practice – guide.