CONSUMING PUBLIC API TUTORIAL

In this tutorial I will be using the Jokes API. This will display a list of 10 jokes.

Link: https://official-joke-api.appspot.com/random_ten

We will use the Volley library.

A volley library is a HTTP library that provides easier and faster networking for Android apps. It provides support for raw strings, images and JSON. The built-in features make it easier to use, therefore, you don't have to write boilerplate code. However, it is no suitable for large download operations.

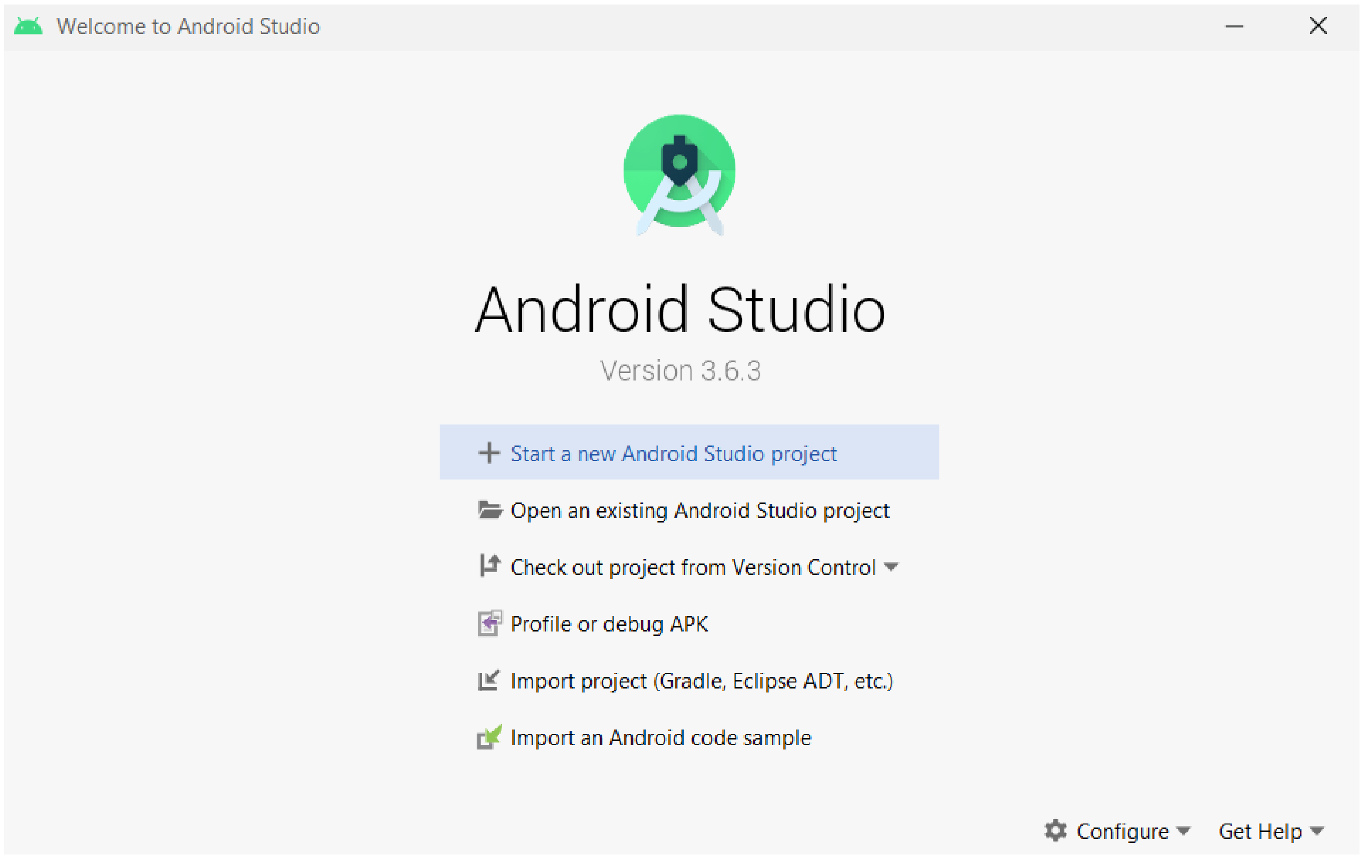
Some of the benefits are:

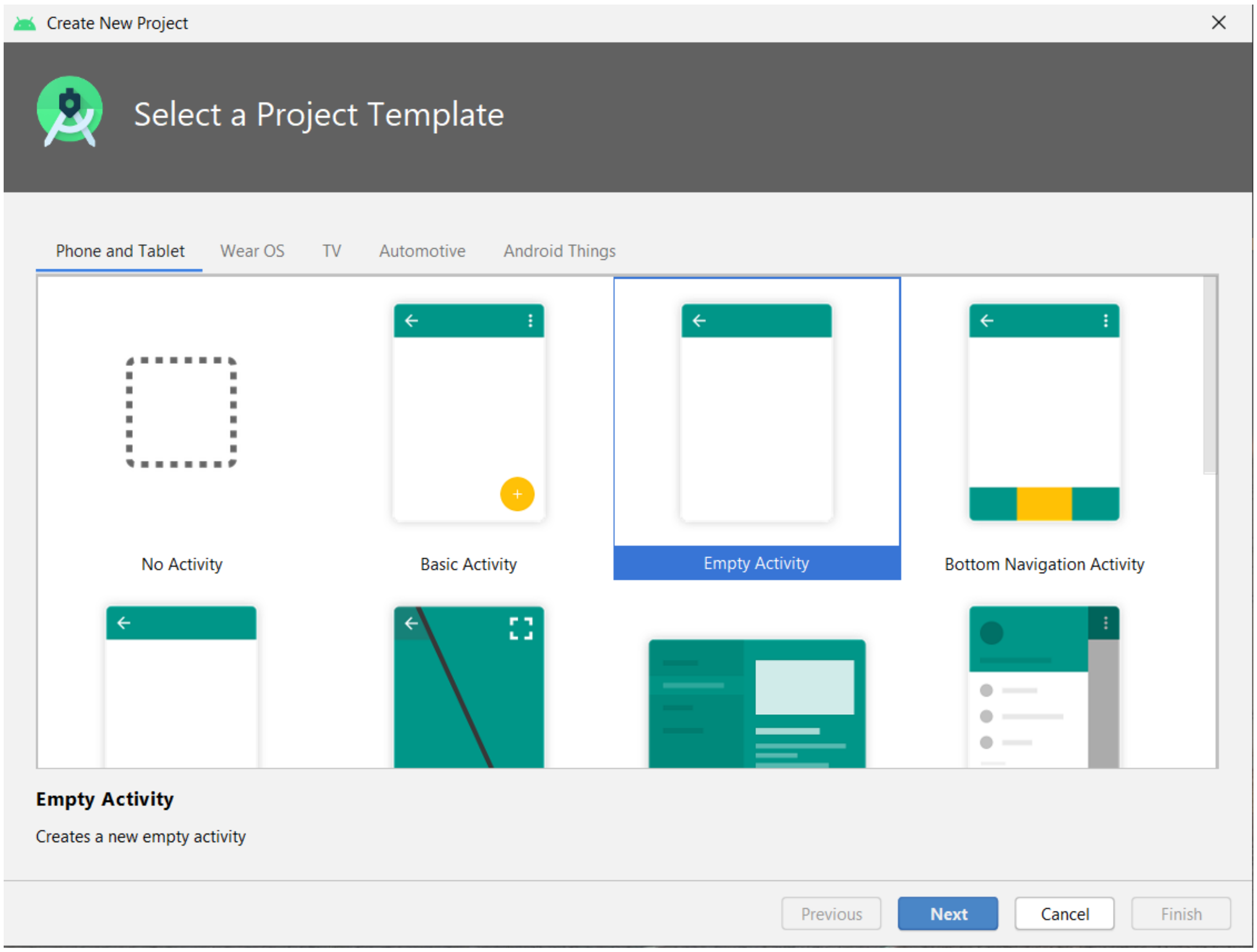
- It provides automatic scheduling of network request.
- It can provide multiple network connections done at the same.
- It can prioritize requests.
- There is ease of customization.
- It has a strong architecture that correctly populates your UI with data from the network.
- It has debugging and tracing tools.

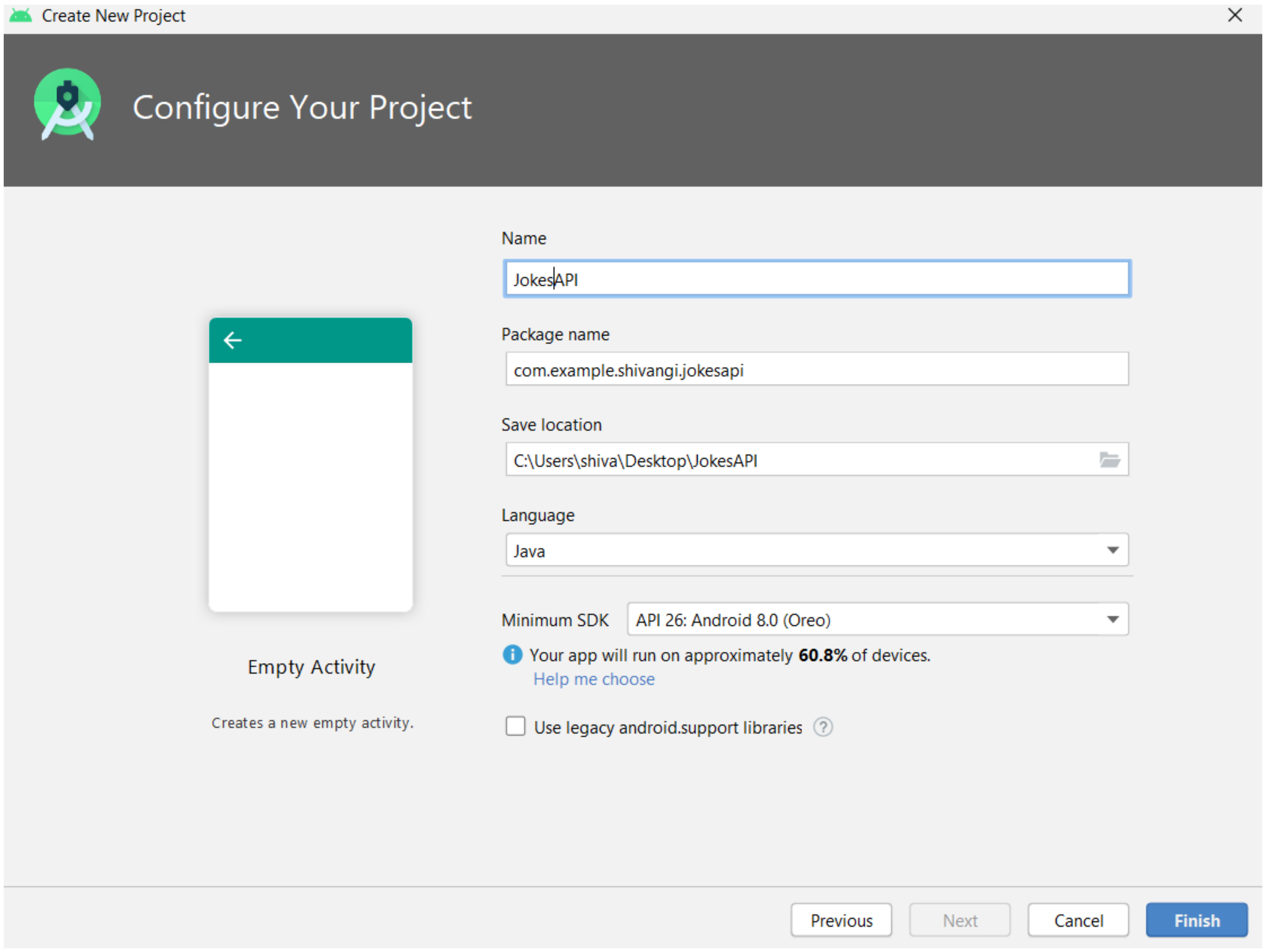
To add Volley to your project, you have to add the volley dependency to your build.gradle file.

Let's Begin!

Create an app and name it JokesAPI.


Welcome to Android Studio
Android Studio
Version 3.6.3
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
Configure
Get Help


Create New Project
Select a Project Template
Phone and Tablet
Wear OS
TV
Automotive
Android Things
No Activity
Basic Activity
Empty Activity
Bottom Navigation Activity
Empty Activity
Creates a new empty activity
Previous
Next
Cancel
Finish


Create New Project

Configure Your Project

Name: JokesAPI

Package name: com.example.shivangi.jokesapi

Save location: C:\Users\shiva\Desktop\JokesAPI

Language: Java

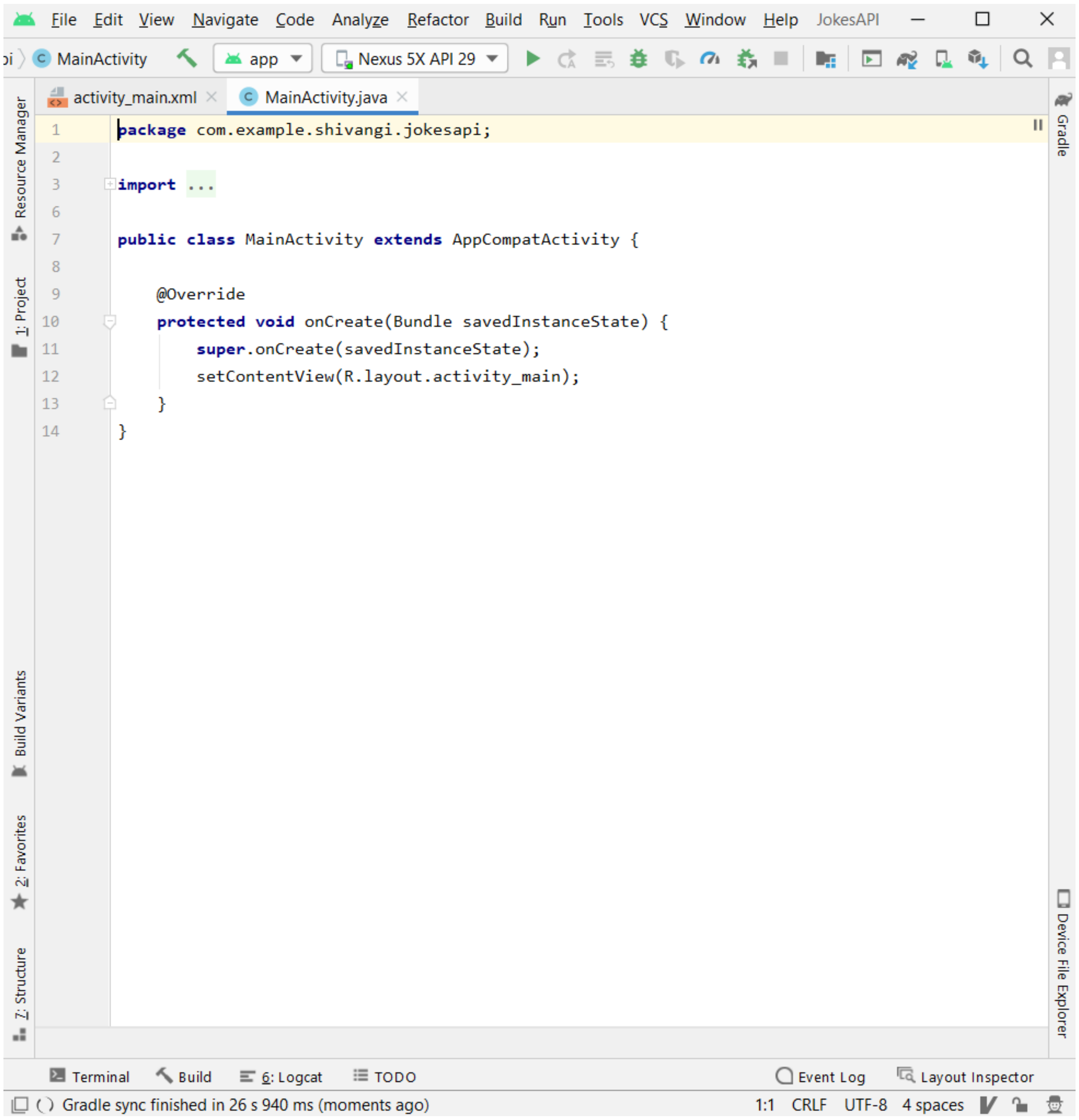
Minimum SDK: API 26: Android 8.0 (Oreo)

Your app will run on approximately **60.8%** of devices.

Help me choose

Use legacy android.support libraries [ ]

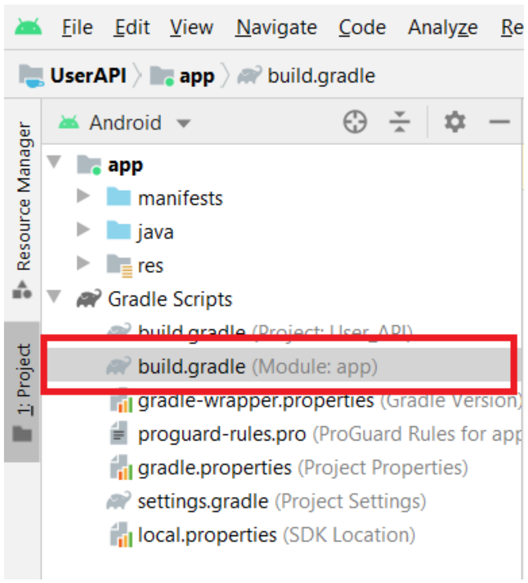
After the project syncs, the MainActivity.java looks like this.

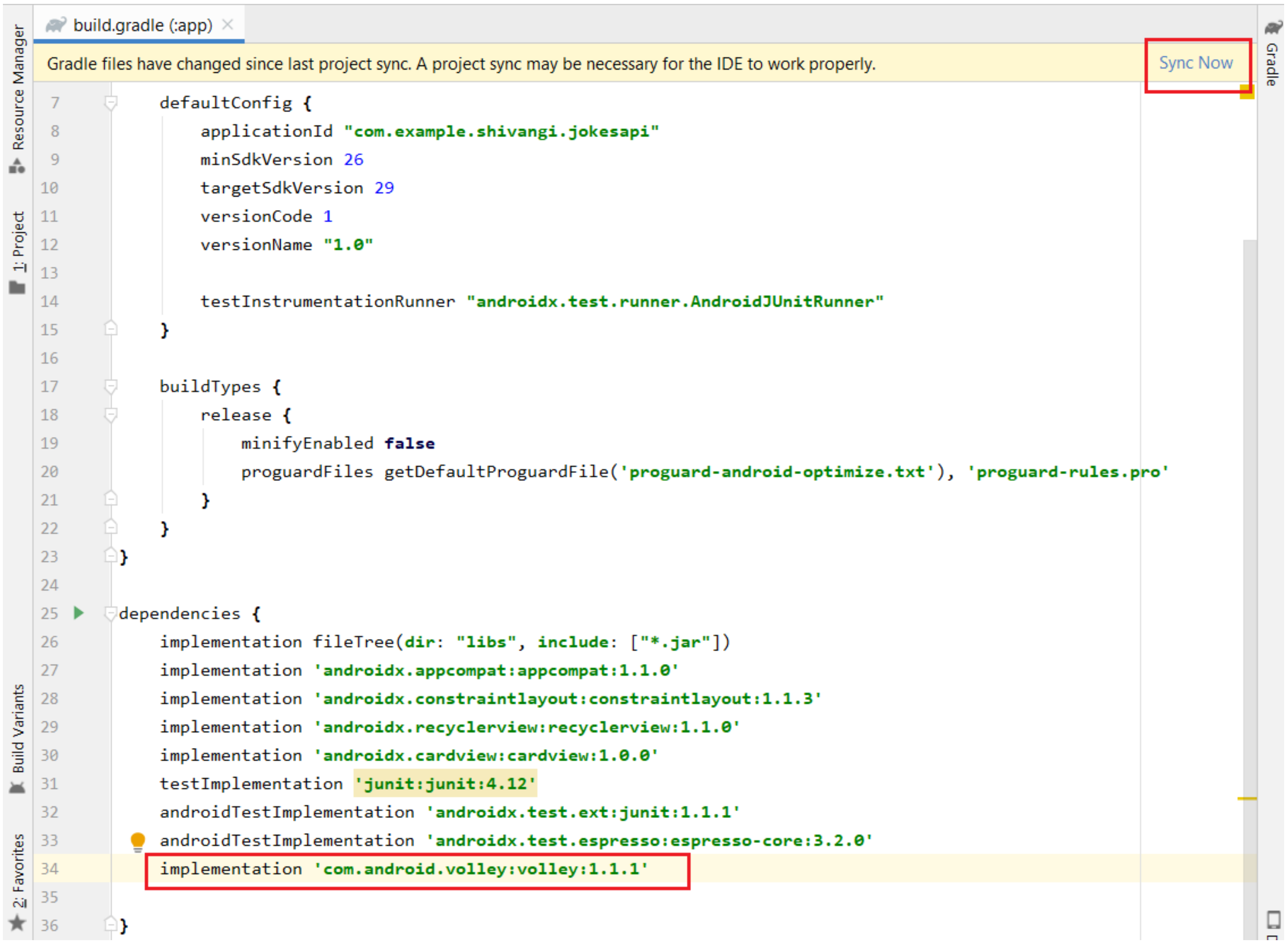


```java
package com.example.shivangi.jokesapi;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Gradle sync finished in 26 s 940 ms (moments ago)   1:1 CRLF UTF-8 4 spaces

Now go to the build.gradle(Module: app) file and add the volley dependency under the dependencies method. The click on the SYNC NOW link provided at the top right of the file.

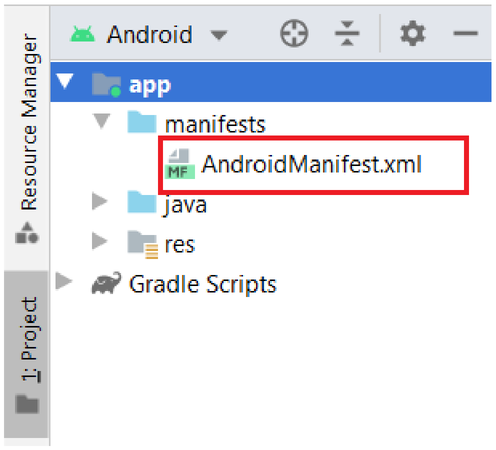

File Edit View Navigate Code Analyze Re
UserAPI › app › build.gradle
Android
Resource Manager
1: Project
app
manifests
java
res
Gradle Scripts
build.gradle (Project: User_API)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)

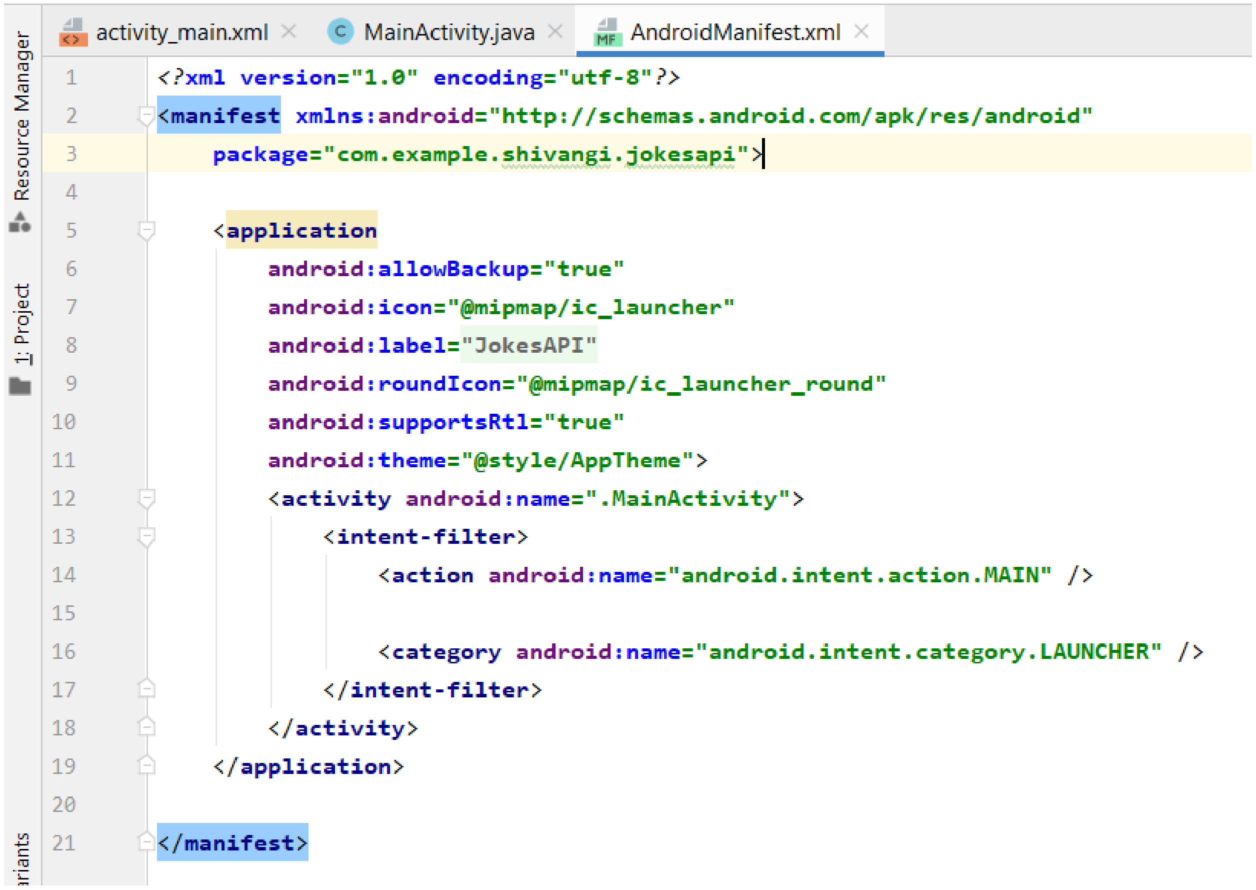


Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

```
    defaultConfig {
        applicationId "com.example.shivangi.jokesapi"
        minSdkVersion 26
        targetSdkVersion 29
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: "libs", include: ["*.jar"])
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'androidx.recyclerview:recyclerview:1.1.0'
    implementation 'androidx.cardview:cardview:1.0.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
    implementation 'com.android.volley:volley:1.1.1'

}
```

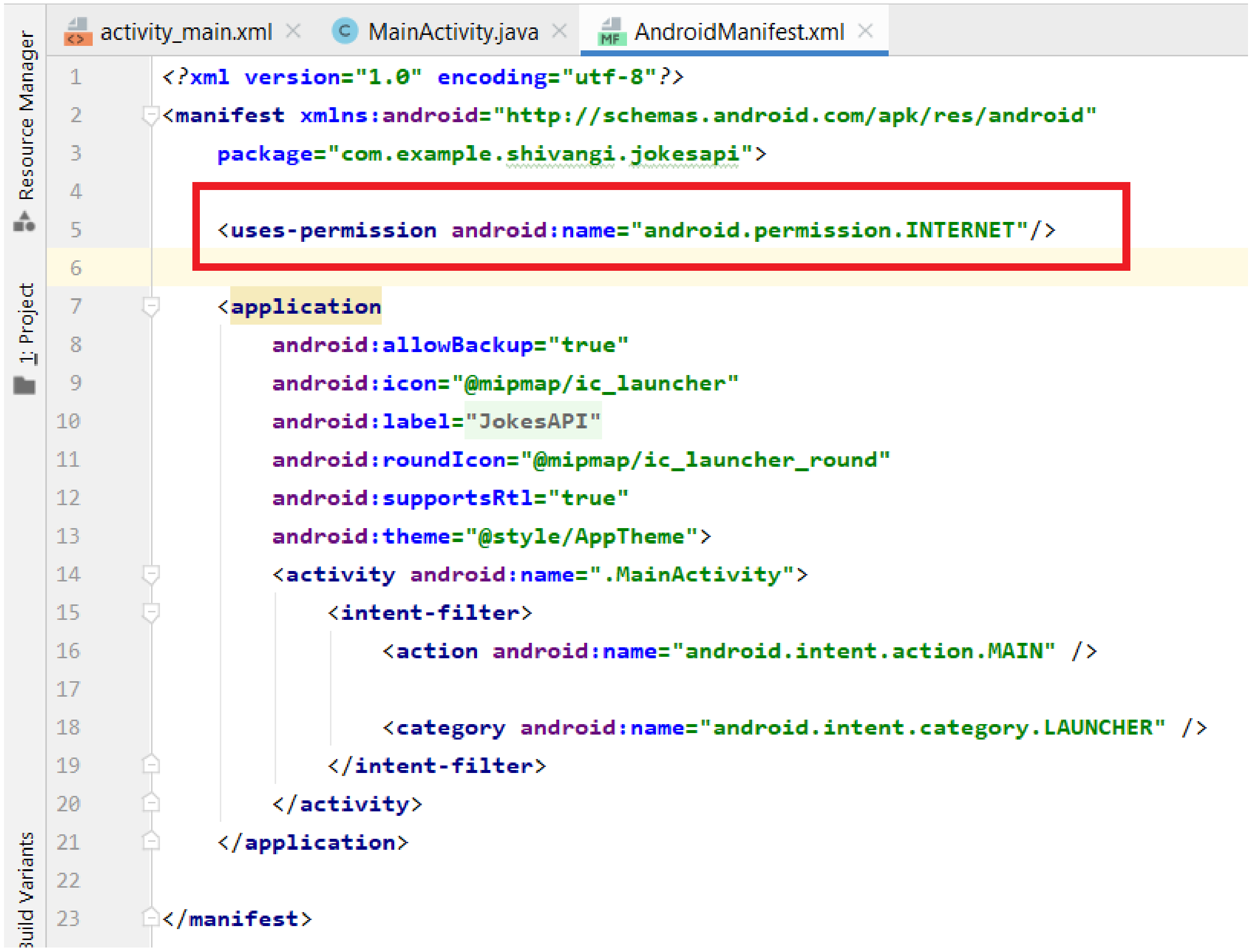
In order to access the API link, we have to grant permission to access the internet. This is done in the manifest file. Go to your manifest.xml file provided under the manifest folder on the left pane.





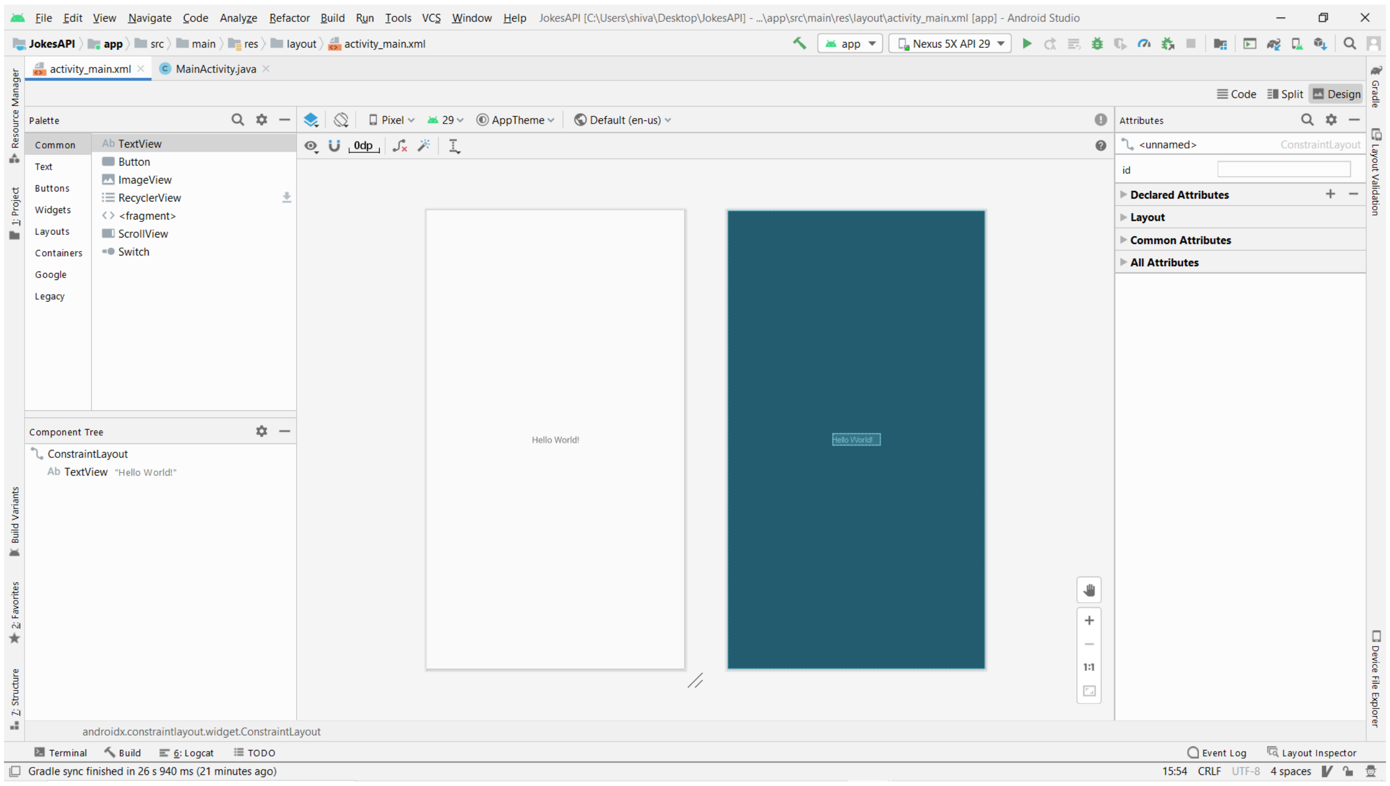
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.shivangi.jokesapi">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="JokesAPI"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Now add the code to grant Internet permission.

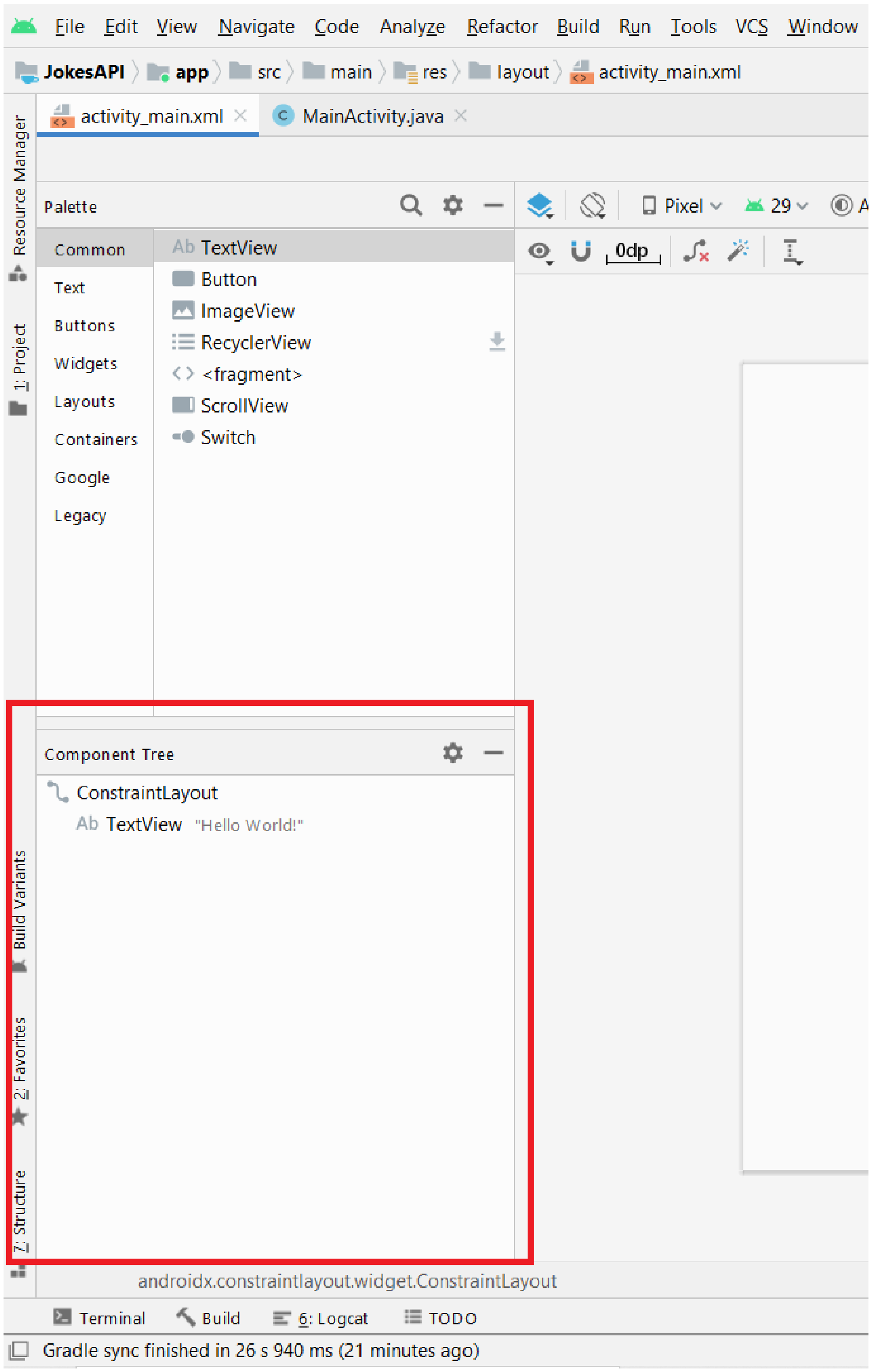


```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.shivangi.jokesapi">

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="JokesAPI"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Now go to your activity_main.xml file. On the top right corner, you will have the option to switch between the layout views.

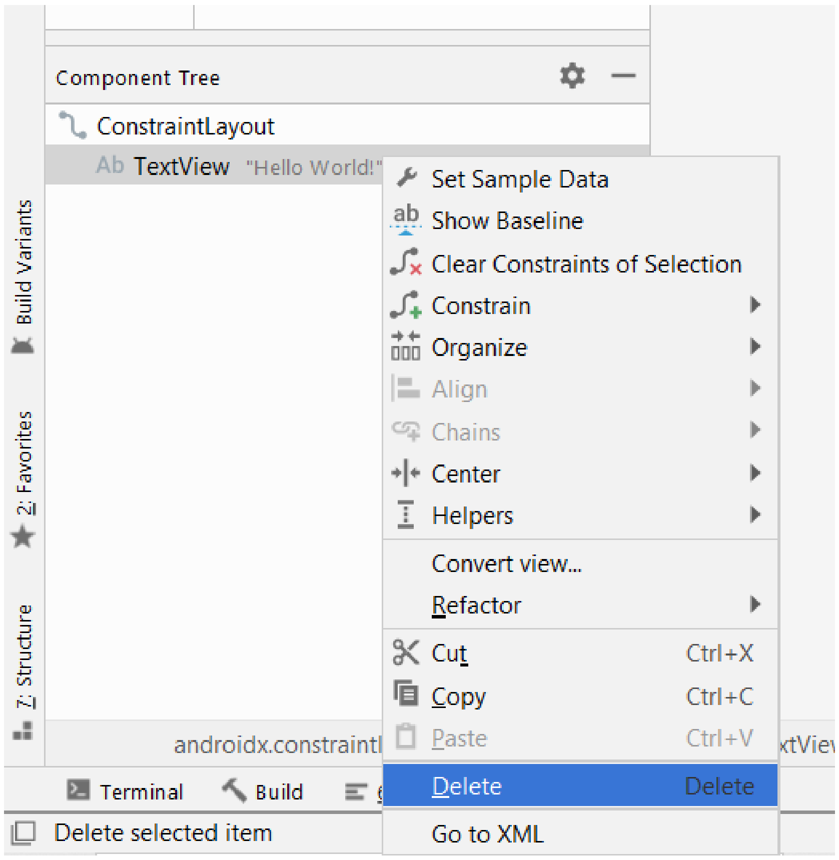

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help JokesAPI [C:\Users\shiva\Desktop\JokesAPI] - ...\app\src\main\res\layout\activity_main.xml [app] - Android Studio
JokesAPI app src main res layout activity_main.xml
activity_main.xml MainActivity.java
Palette
Common Text Buttons Widgets Layouts Containers Google Legacy
TextView Button ImageView RecyclerView <fragment> ScrollView Switch
Component Tree
ConstraintLayout
TextView "Hello World!"
Hello World!
Attributes
<unnamed> ConstraintLayout
id
Declared Attributes
Layout
Common Attributes
All Attributes
androidx.constraintlayout.widget.ConstraintLayout
Terminal Build 6: Logcat TODO Event Log Layout Inspector
Gradle sync finished in 26 s 940 ms (21 minutes ago)
15:54 CRLF UTF-8 4 spaces

On the left bottom corner, you will see a tab named component tree. Right click on the TextView and delete it.



androidx.constraintlayout.widget.ConstraintLayout

Gradle sync finished in 26 s 940 ms (21 minutes ago)

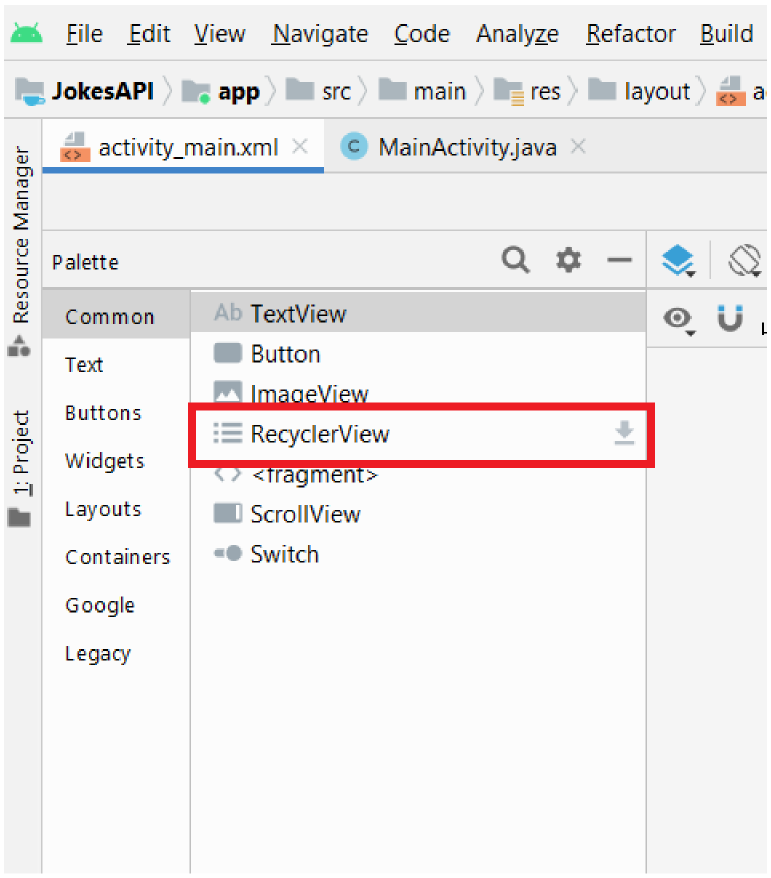


We will add a RecyclerView to the layout. A RecyclerView allows the user to display a list of items and provide an option to scroll if the items exceed the length of the screen layout. The word Recycle means to reuse. Therefore, RecyclerView provides the ability to reuse a layout for a list of items making it efficient.

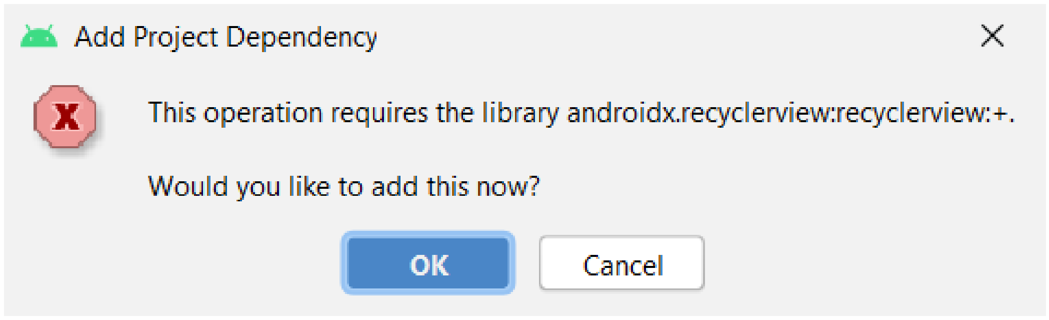
In a RecyclerView, we have a LayoutManager that handles the layout together with the LinearLayoutManager. The LinearLayout could be vertically or the horizontal oriented. We also require an Adapter that connects your data to the RecyclerView. The adapter uses the ViewHolder to prepare the data before it is presented. It acts as the bridge between the AdapterView and the data. The adapter is responsible for creating a view for each item in the data set and also provides access to the data items. The ViewHolder contains information for the item that will be displayed.

Let's begin by adding the RecyclerView dependency. An easier way to add the dependency is to add through your activity_main.xml file.

Click on the download icon next to the RecyclerView in the Palette.



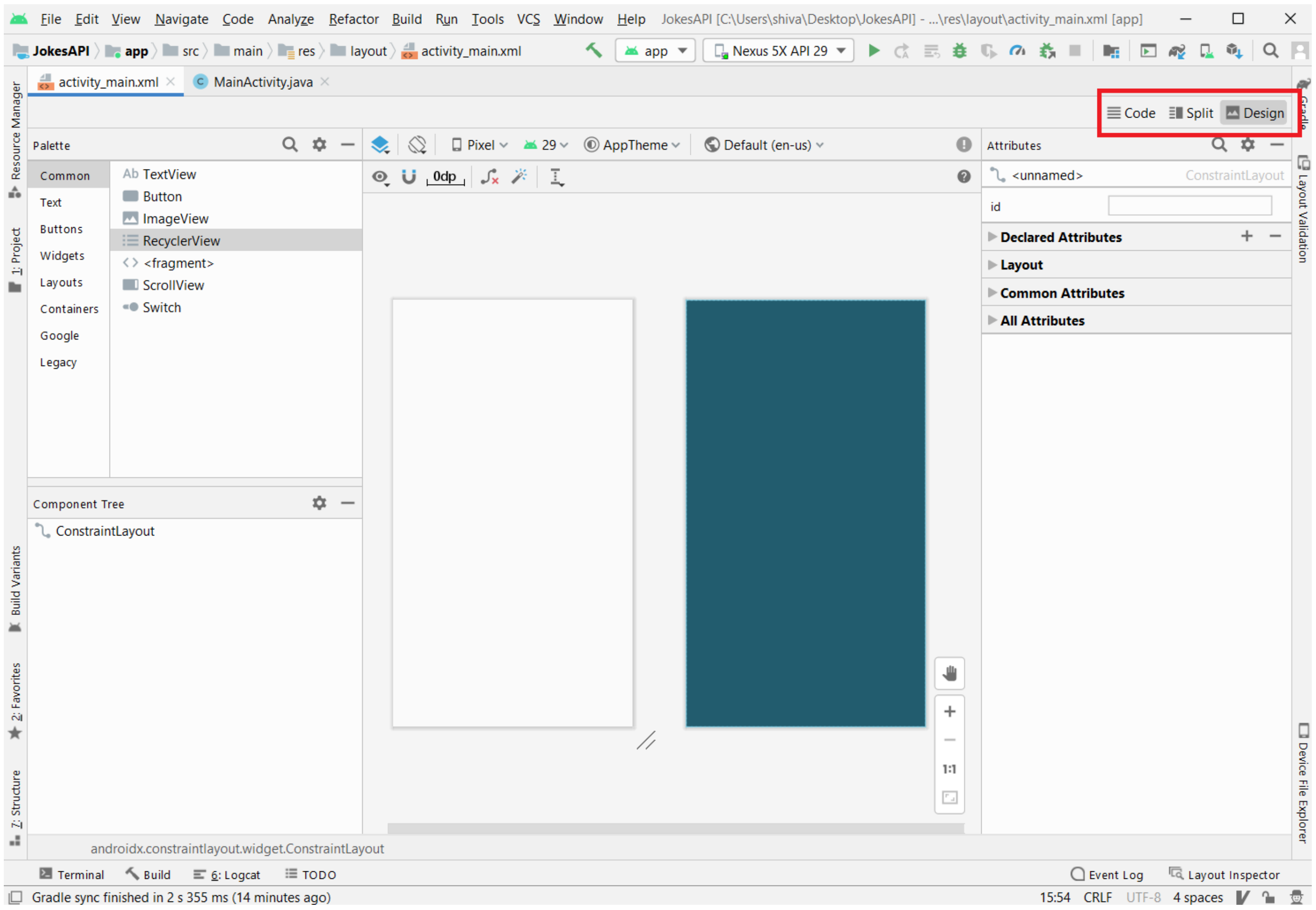
A dialog box like below will appear. Click ok.



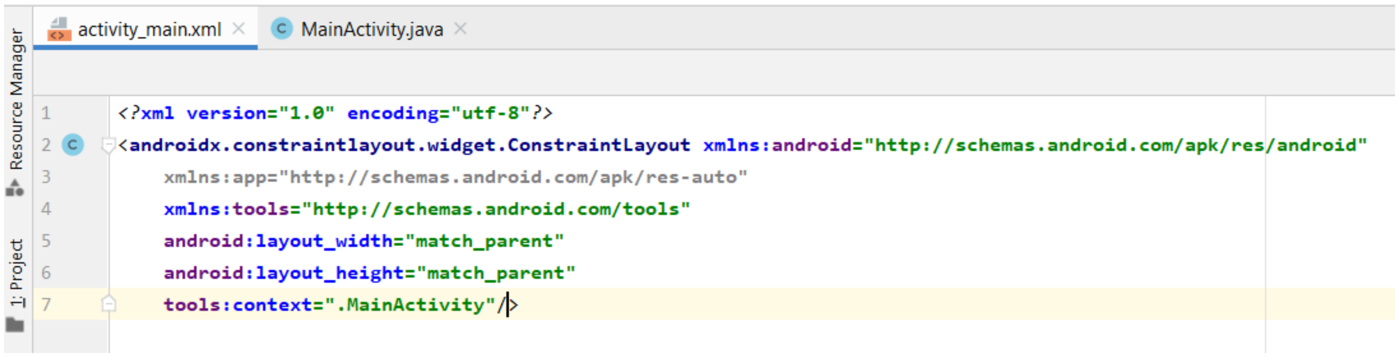
This operation requires the library androidx.recyclerview:recyclerview:+.

Would you like to add this now?

Once the dependency is added, switch to code view. Three icons at the top-right corner are provided that lets you switch between code, split and design view. Click on the code to switch to code view.

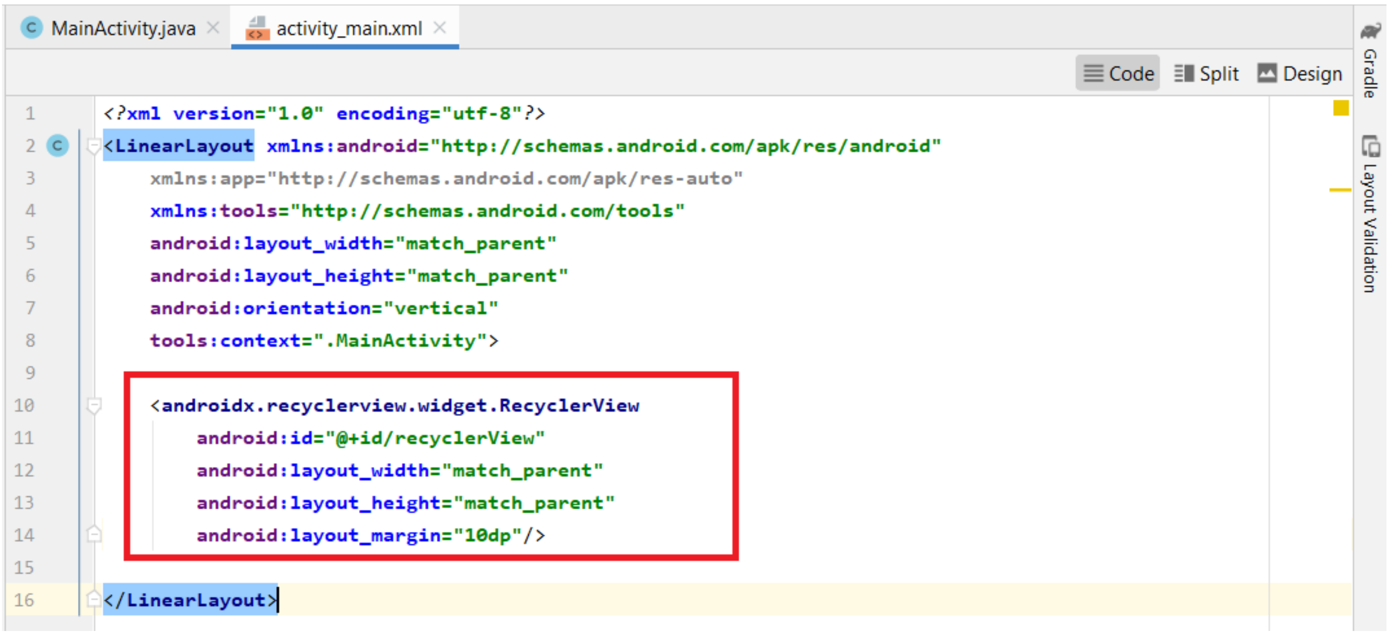


We are currently in design view, click on the code option to change the view to code view. Your activity_main.xml file looks like below before any changes are made.



```
<?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"/>
```

Add the following code to your activity_main.xml file.

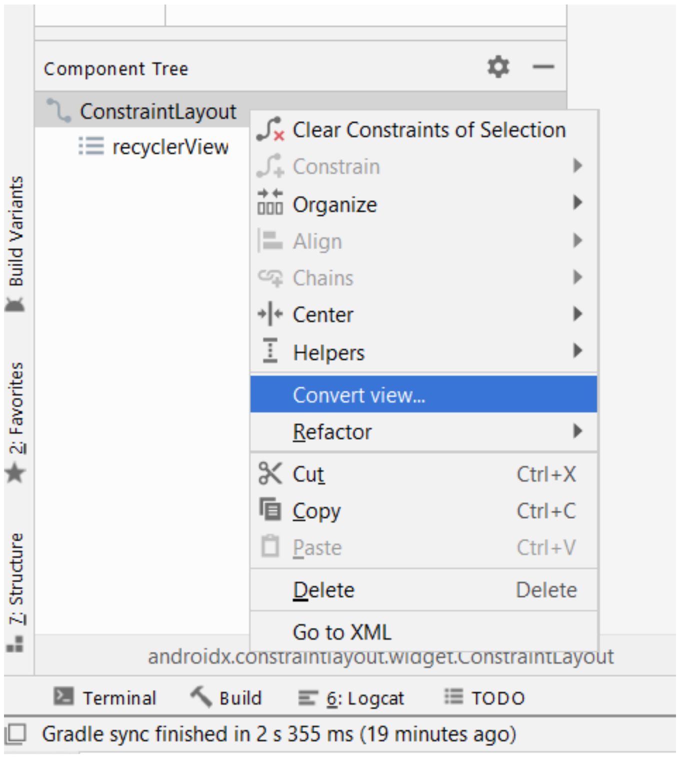


MainActivity.java | activity_main.xml

Code | Split | Design

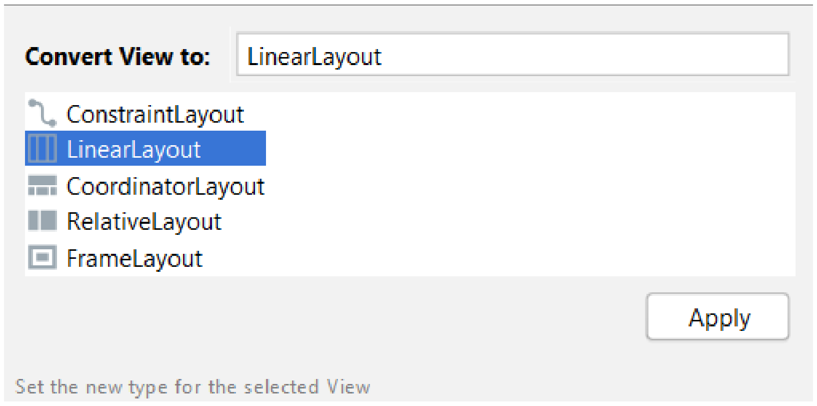
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="10dp"/>

</LinearLayout>
```

Switch back to design view. We will change the layout of the file to a vertical LinearLayout. In the component tree provided on the left bottom corner, you will see a ConstraintLayout. Right click on it and select Convert view…

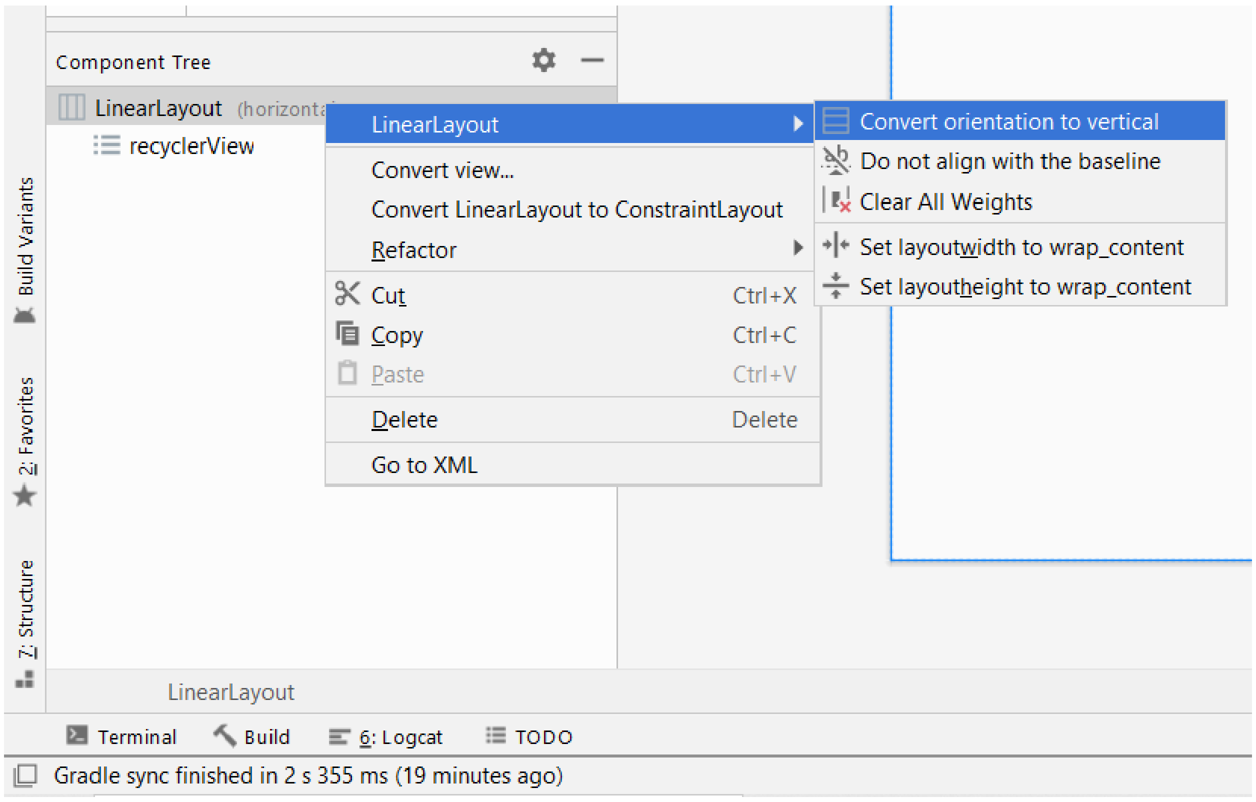


Gradle sync finished in 2 s 355 ms (19 minutes ago)

Select LinearLayout from the dialog box that appears.



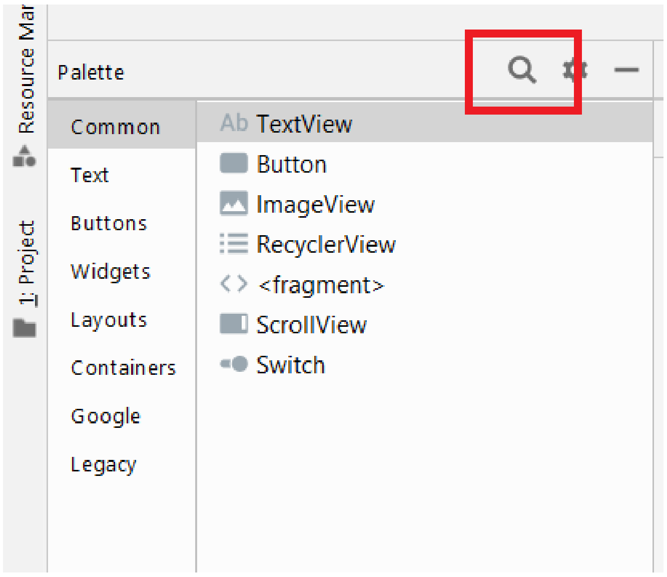
Once again right-click on the LinearLayout(horizontal) under the component tree. Go to **LinearLayout** > **Convert orientation to vertical**.



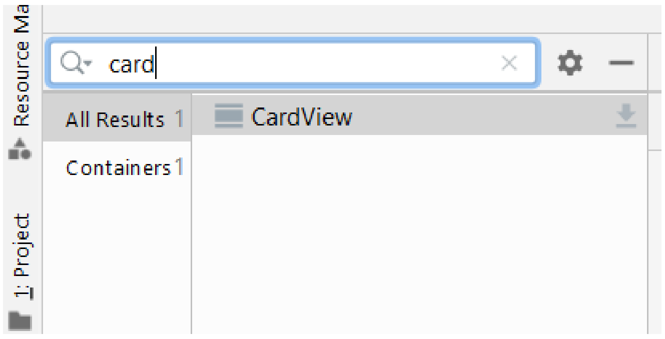
Gradle sync finished in 2 s 355 ms (19 minutes ago)

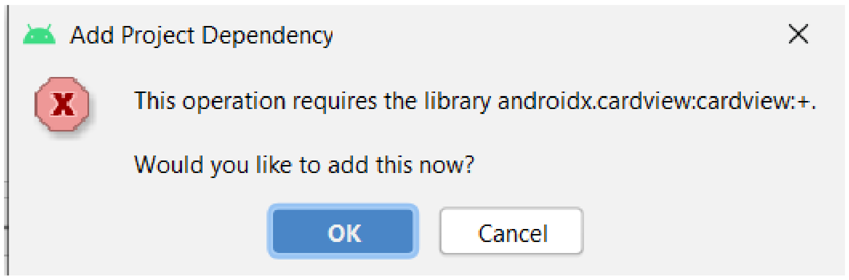
We created the layout for all the list items. Now we will create a layout for one list item.

First, we will add the CardView dependency. To do this, search for CardView in the palette.



Click on the download icon next to the CardView. Click OK on the dialog box that appears.

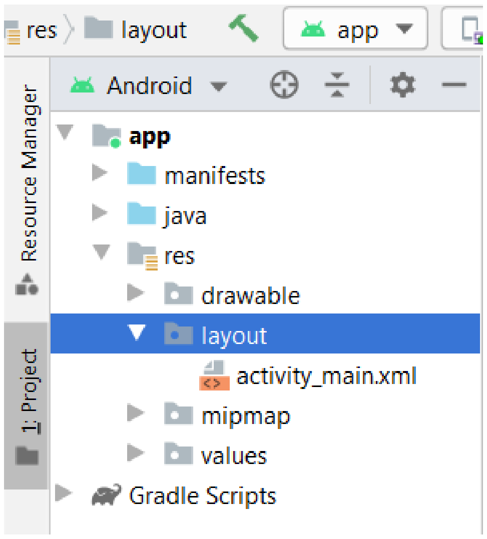



Add Project Dependency

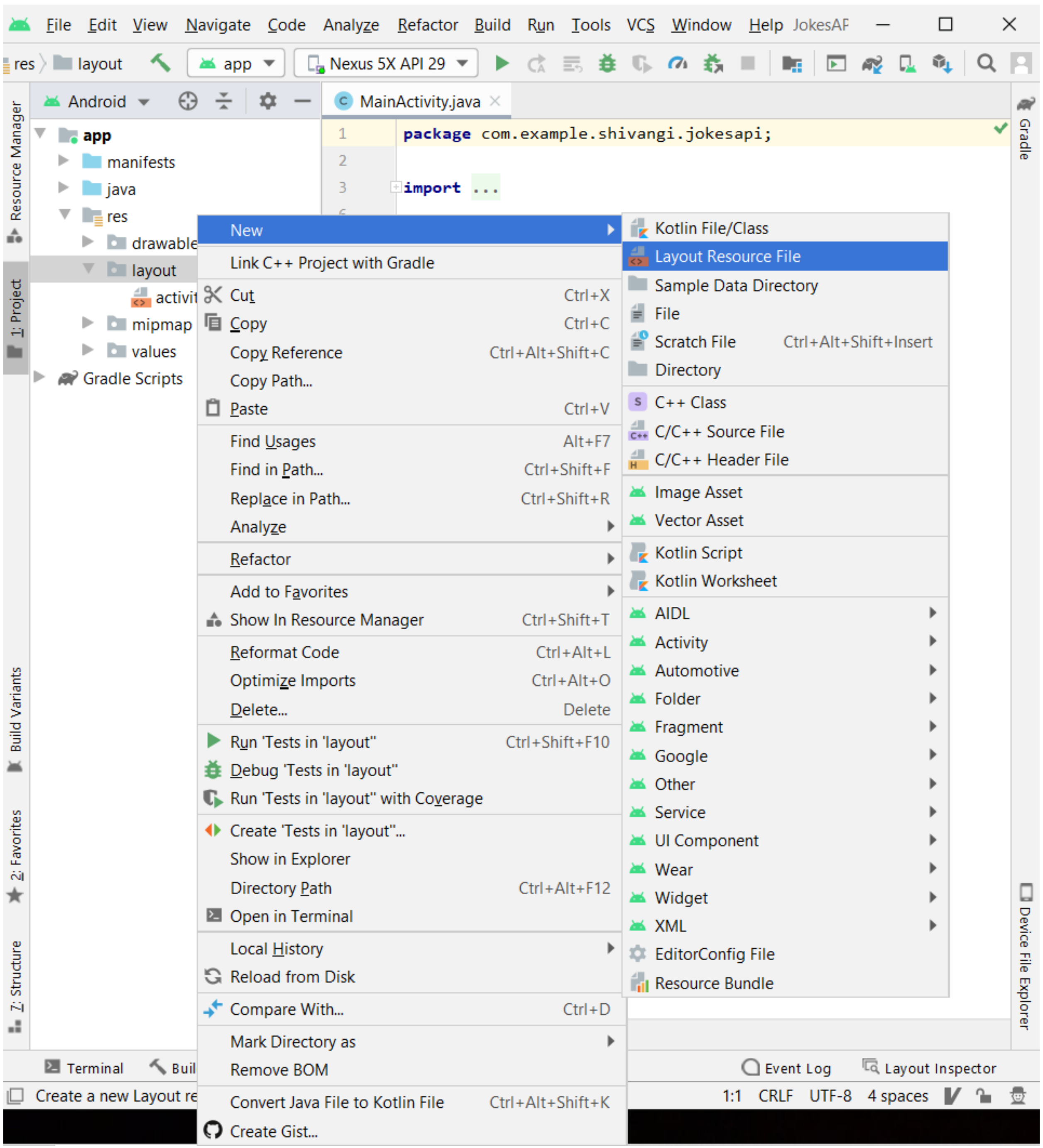
This operation requires the library androidx.cardview:cardview:+.

Would you like to add this now?

Now let’s create a new layout resource file, right-click on the layout folder provided under the res folder.



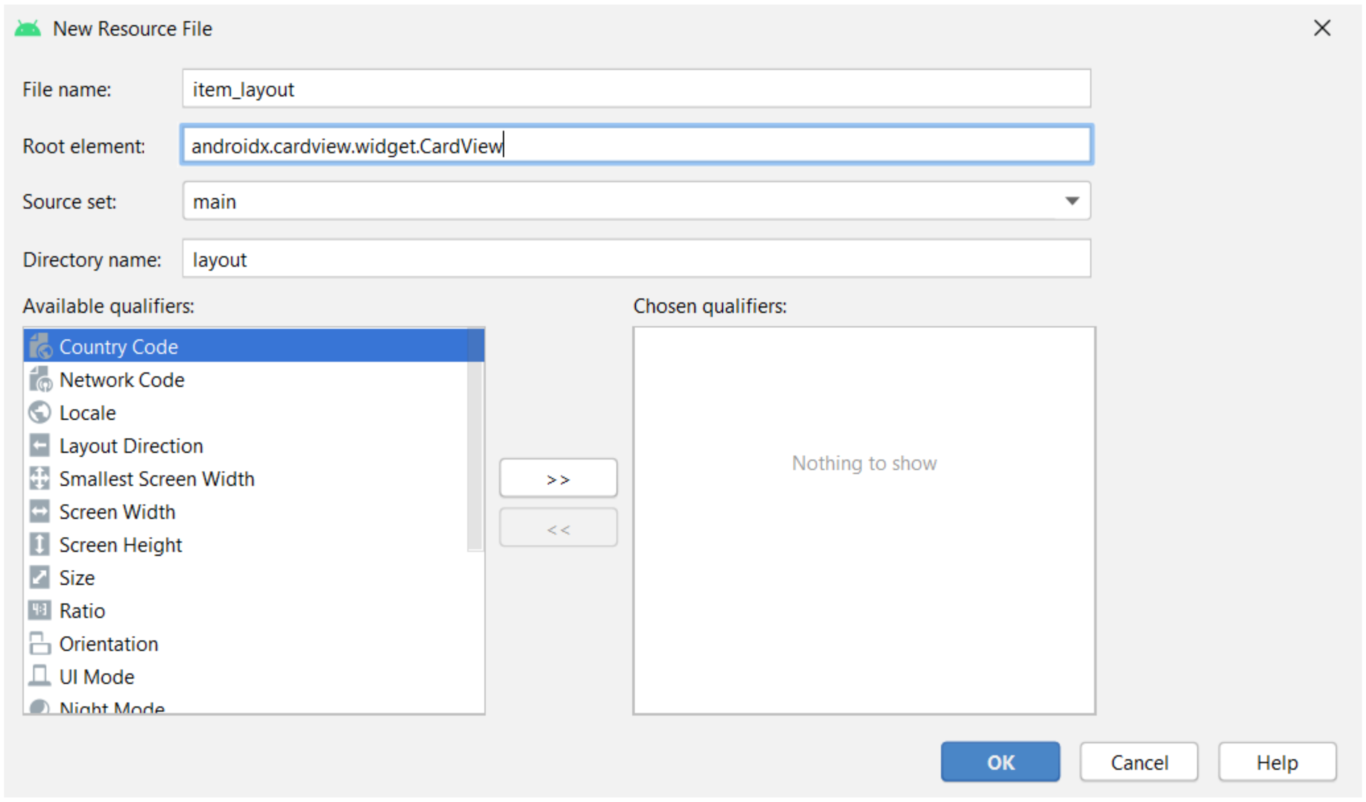
Select **New** > **Layout Resource File**.



```
package com.example.shivangi.jokesapi;

import ...
```

Shivangi Patel

Set the file name as item_layout and the root element as CardView in the dialog box that appears.



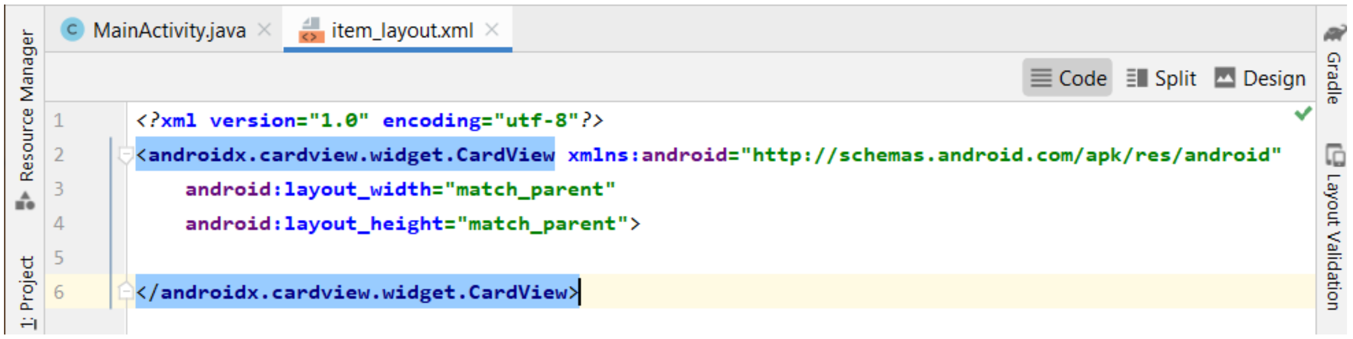
File name: item_layout

Root element: androidx.cardview.widget.CardView

Source set: main

Directory name: layout

Switch to code view. Below is an image of your item_layout.xml file before any changes are made.

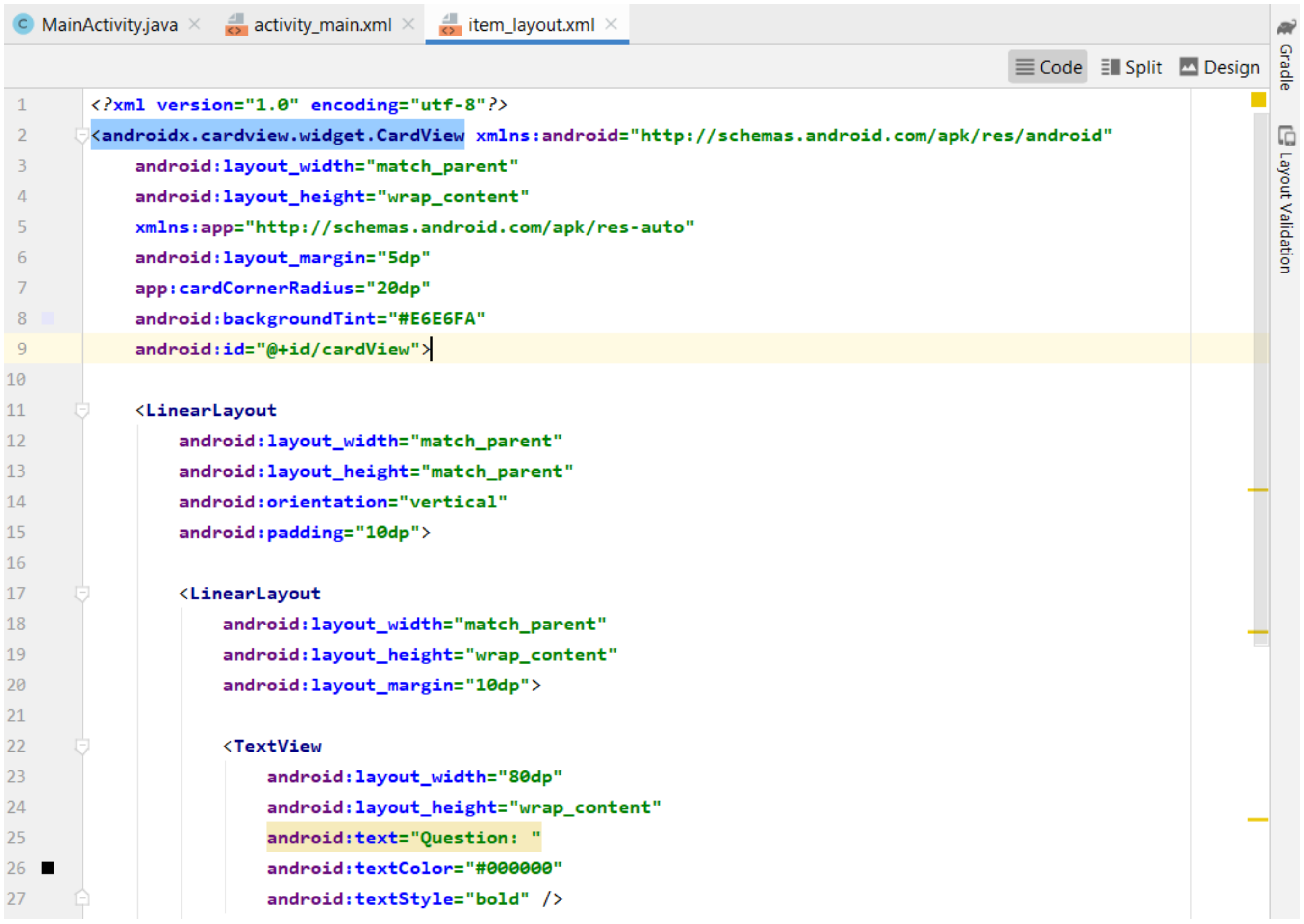


```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

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

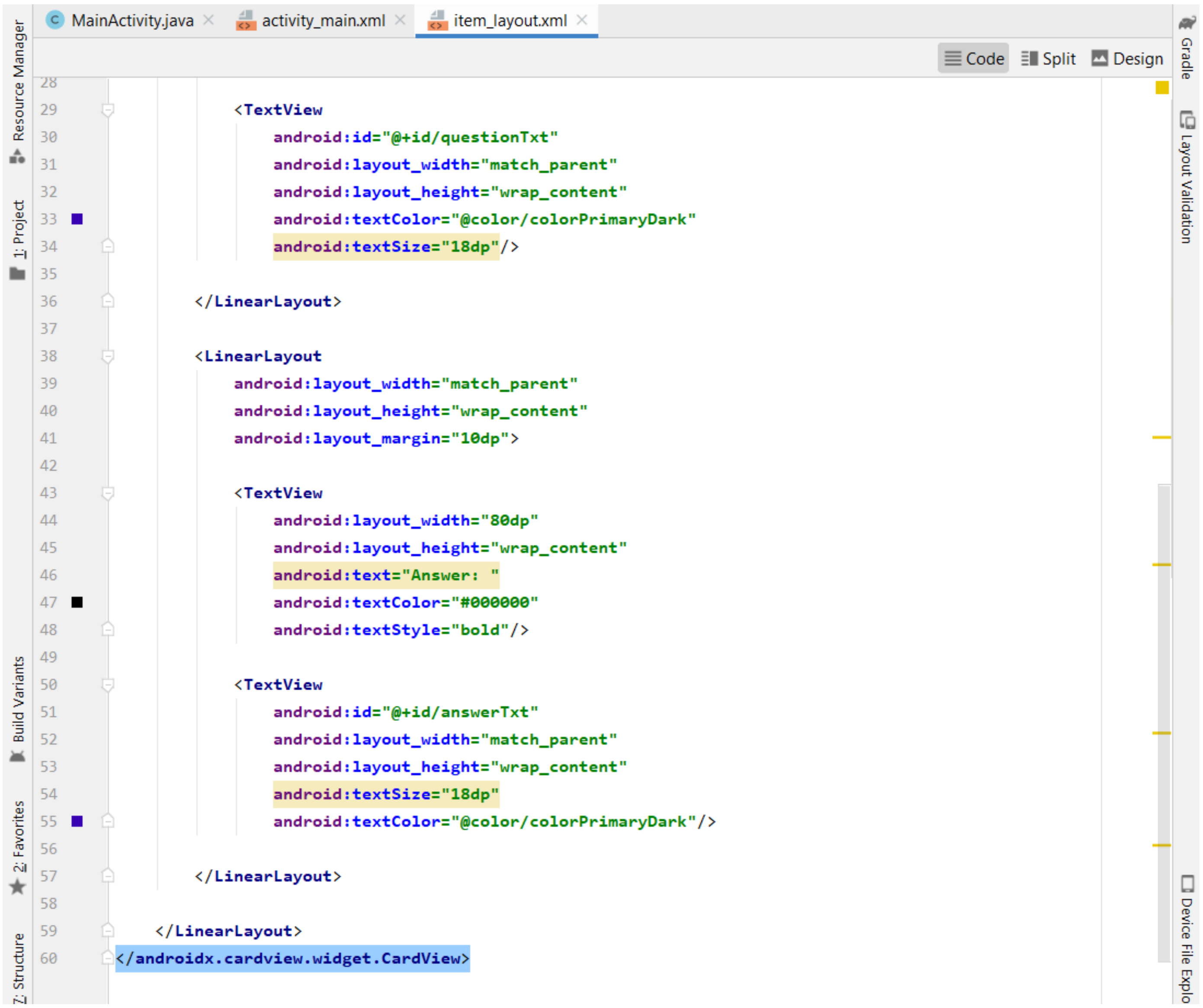
Now add the following code to your item_layout.xml file. The code below represents a CardView that contains a vertical LinearLayout. Inside the vertical LinearLayout, there are two horizontal LinearLayout containing two TextView components in each.

One TextView holds text that will not change, the other will display text retrieved from the API.

Notice, we changed the layout_height of the CardView to wrap_content and added a few more properties for the CardView.



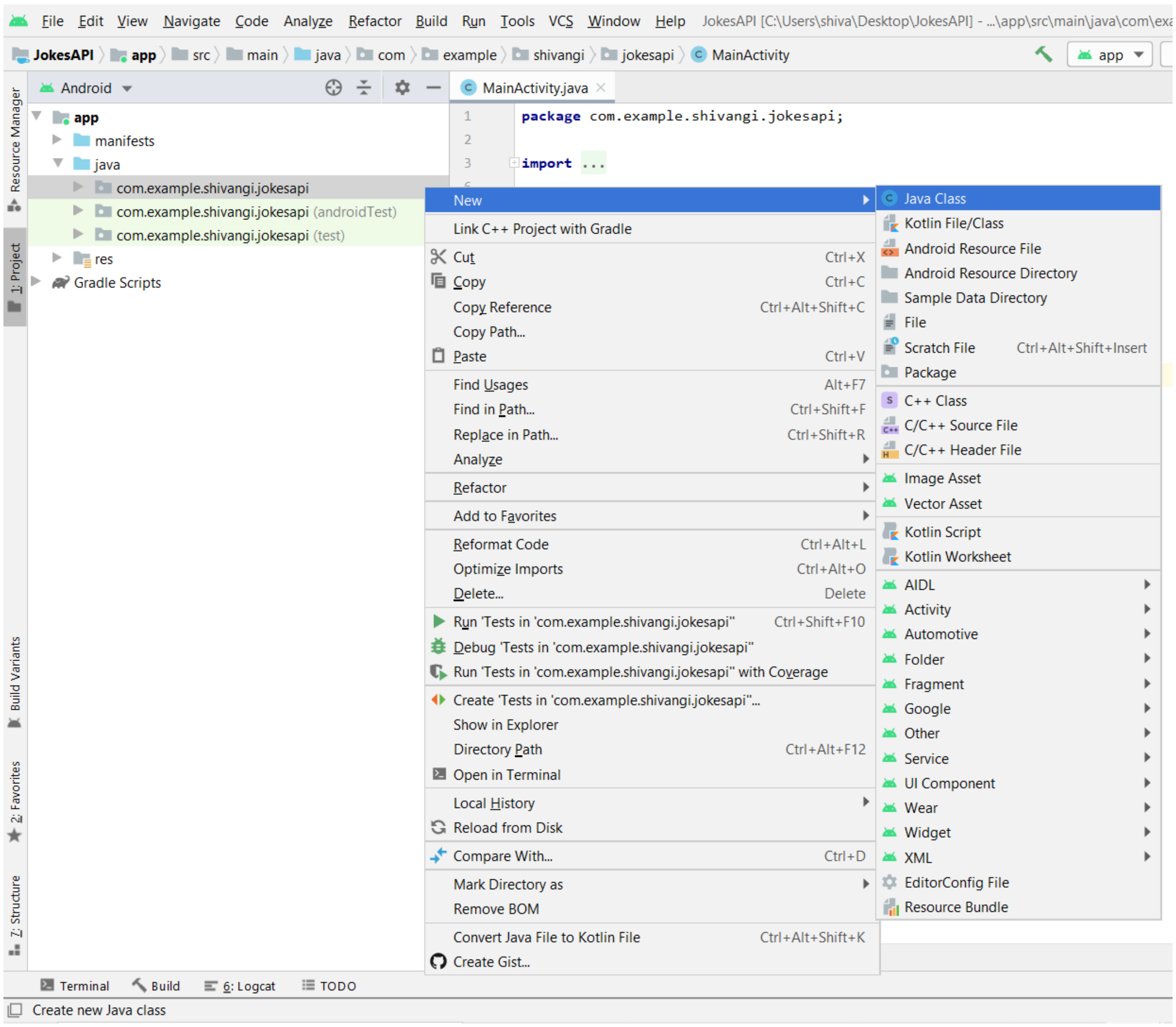
```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_margin="5dp"
    app:cardCornerRadius="20dp"
    android:backgroundTint="#E6E6FA"
    android:id="@+id/cardView">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:padding="10dp">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="10dp">

            <TextView
                android:layout_width="80dp"
                android:layout_height="wrap_content"
                android:text="Question: "
                android:textColor="#000000"
                android:textStyle="bold" />
```



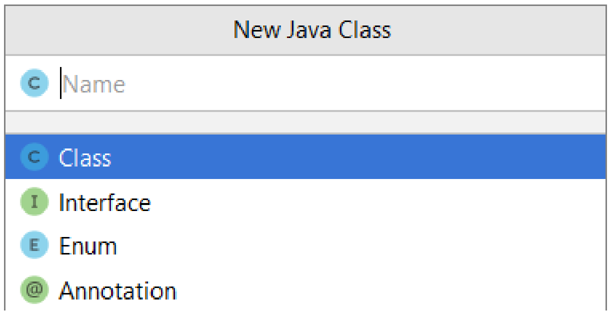
```
            <TextView
                android:id="@+id/questionTxt"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:textColor="@color/colorPrimaryDark"
                android:textSize="18dp"/>

        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="10dp">

            <TextView
                android:layout_width="80dp"
                android:layout_height="wrap_content"
                android:text="Answer: "
                android:textColor="#000000"
                android:textStyle="bold"/>

            <TextView
                android:id="@+id/answerTxt"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:textSize="18dp"
                android:textColor="@color/colorPrimaryDark"/>

        </LinearLayout>

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

Now we will create a class that will contain all the information in a layout.

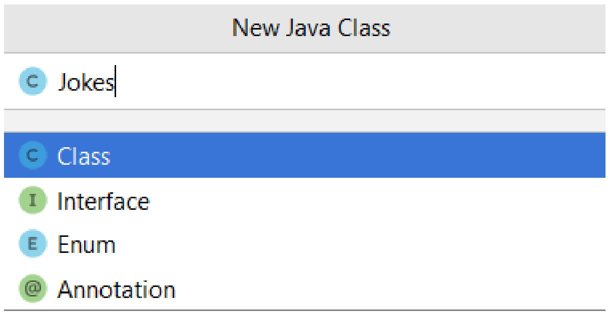
To create a new Java class, right click on your package, select **New** > **Java Class**.


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
JokesAPI [C:\Users\shiva\Desktop\JokesAPI] - ...\app\src\main\java\com\exa
JokesAPI app src main java com example shivangi jokesapi MainActivity
Android
app
manifests
java
com.example.shivangi.jokesapi
com.example.shivangi.jokesapi (androidTest)
com.example.shivangi.jokesapi (test)
res
Gradle Scripts
MainActivity.java
New
Link C++ Project with Gradle
Cut Ctrl+X
Copy Ctrl+C
Copy Reference Ctrl+Alt+Shift+C
Copy Path...
Paste Ctrl+V
Find Usages Alt+F7
Find in Path... Ctrl+Shift+F
Replace in Path... Ctrl+Shift+R
Analyze
Refactor
Add to Favorites
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Delete... Delete
Run 'Tests in 'com.example.shivangi.jokesapi'' Ctrl+Shift+F10
Debug 'Tests in 'com.example.shivangi.jokesapi''
Run 'Tests in 'com.example.shivangi.jokesapi'' with Coverage
Create 'Tests in 'com.example.shivangi.jokesapi''...
Show in Explorer
Directory Path Ctrl+Alt+F12
Open in Terminal
Local History
Reload from Disk
Compare With... Ctrl+D
Mark Directory as
Remove BOM
Convert Java File to Kotlin File Ctrl+Alt+Shift+K
Create Gist...
Java Class
Kotlin File/Class
Android Resource File
Android Resource Directory
Sample Data Directory
File
Scratch File Ctrl+Alt+Shift+Insert
Package
C++ Class
C/C++ Source File
C/C++ Header File
Image Asset
Vector Asset
Kotlin Script
Kotlin Worksheet
AIDL
Activity
Automotive
Folder
Fragment
Google
Other
Service
UI Component
Wear
Widget
XML
EditorConfig File
Resource Bundle
Resource Manager
1: Project
Build Variants
2: Favorites
7: Structure
Terminal
Build
6: Logcat
TODO
Create new Java class

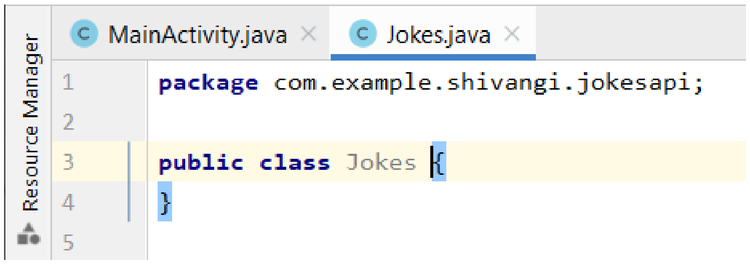
```
package com.example.shivangi.jokesapi;

import ...
```



Name the class Jokes.



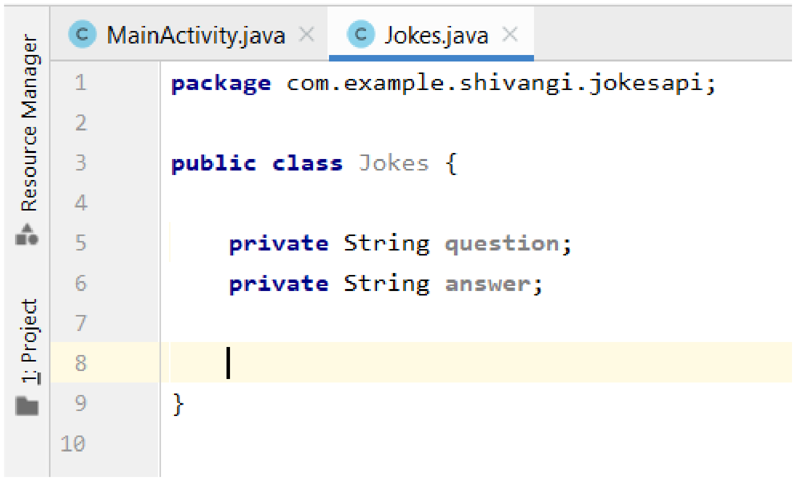
Below is a screenshot of how your java class will look like once you've created it.



```java
package com.example.shivangi.jokesapi;

public class Jokes {
}
```

Now we will declare class variables. These variables will store the data that we will receive from the API. Since we are receiving two strings, one is a question and the other is the answer, we will create two class variables.

Add the following code to your Jokes.java class.



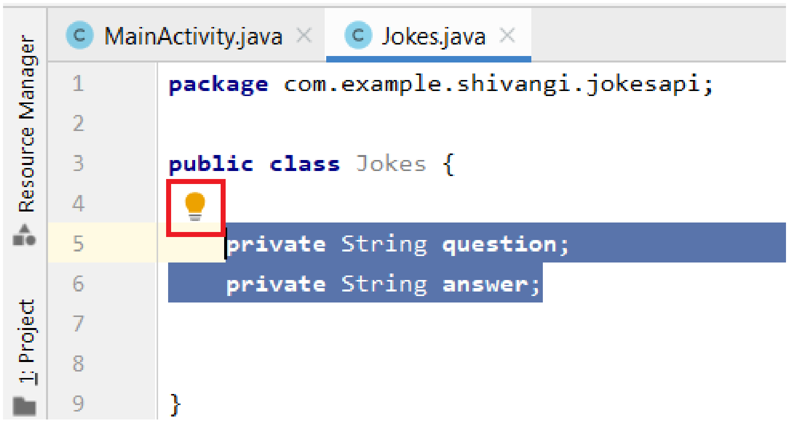
```java
package com.example.shivangi.jokesapi;

public class Jokes {

    private String question;
    private String answer;


}
```

We have to declare a constructor that will initialize the object of a class. This method will be used to perform a task.

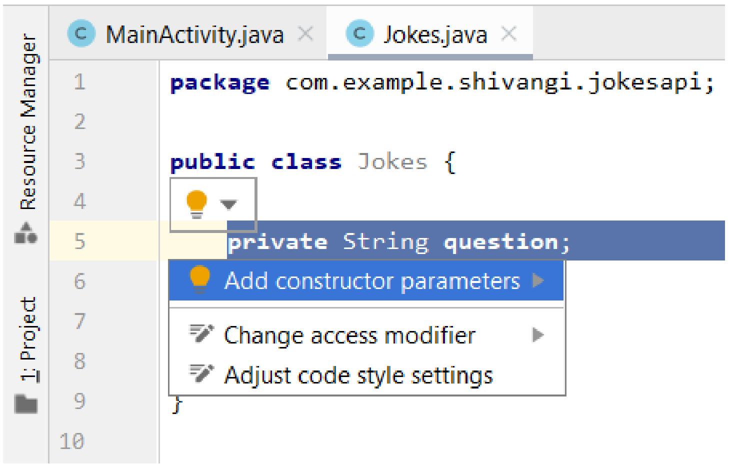
To create a constructor, we type the following code in our Jokes.java class:

```
public Jokes() {}
```

However, we need to add two parameters for the constructor which will represent the two strings we declared, an easier way to do this would be selecting the two variables. Click on the yellow bulb that will appear on the left side once you select the two variables and select Add constructor parameters. This will create the constructor with the parameters and add all the necessary code needed in the constructor.

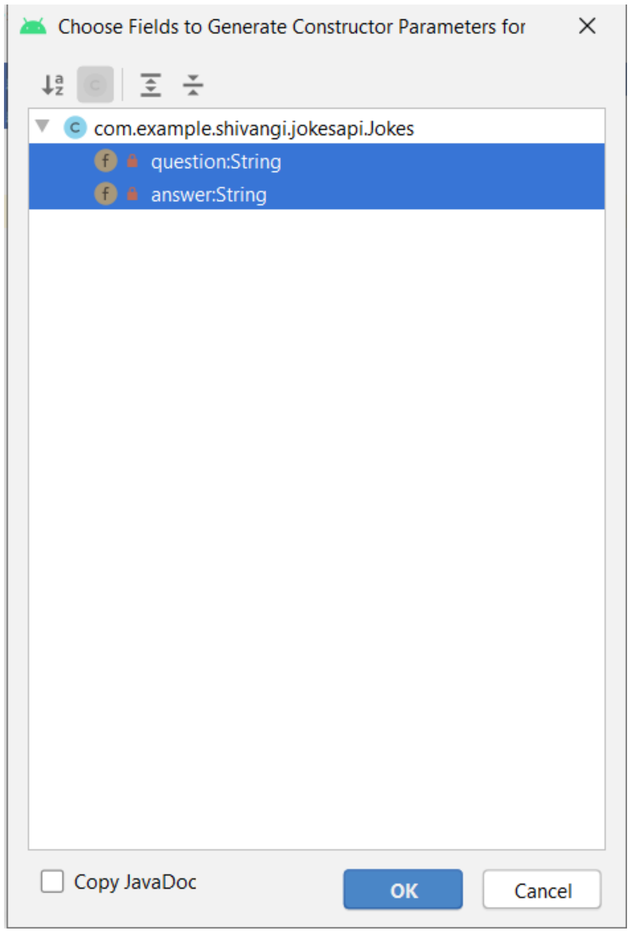


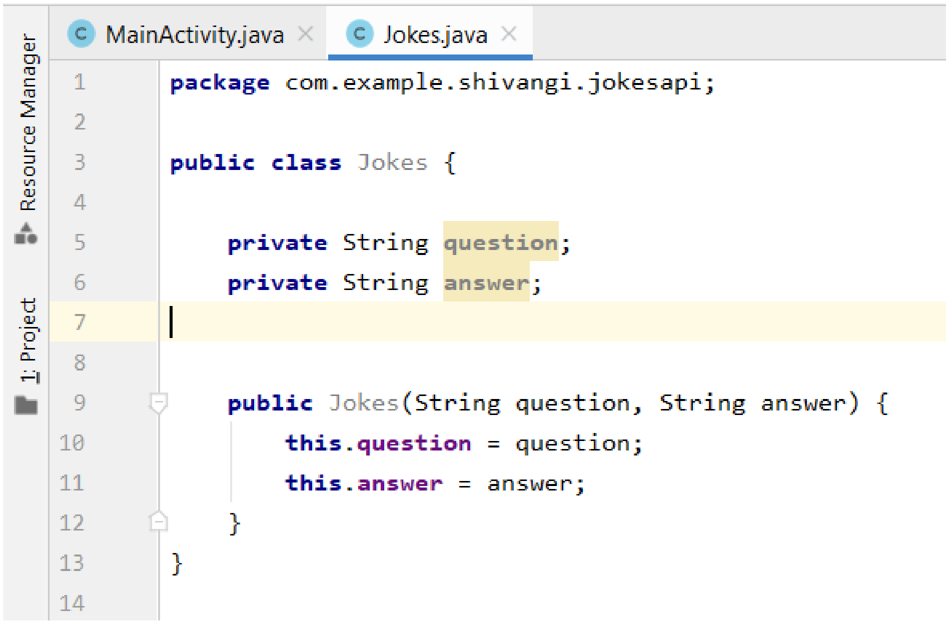
```
package com.example.shivangi.jokesapi;

public class Jokes {

    private String question;
    private String answer;


}
```



```
package com.example.shivangi.jokesapi;

public class Jokes {

    private String question;
```

A dialog box will appear, select both strings and click ok.


Choose Fields to Generate Constructor Parameters for
com.example.shivangi.jokesapi.Jokes
question:String
answer:String
Copy JavaDoc
OK
Cancel

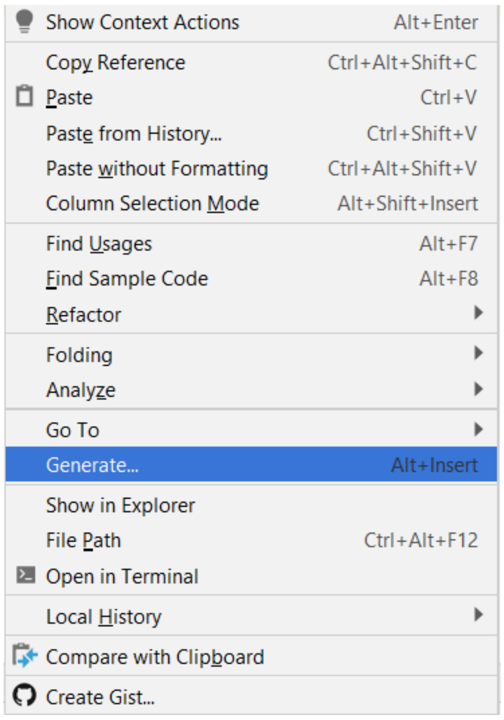


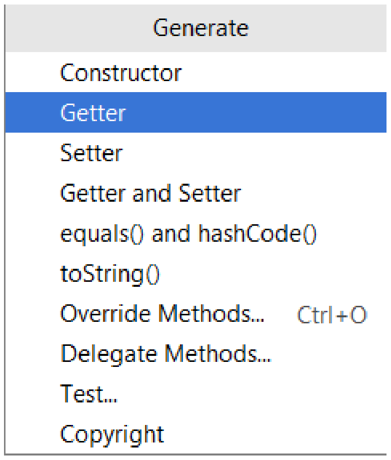
```java
package com.example.shivangi.jokesapi;

public class Jokes {

    private String question;
    private String answer;


    public Jokes(String question, String answer) {
        this.question = question;
        this.answer = answer;
    }
}
```

The next step will be creating the getter methods. Getters and Setters are used to protect your data. For each variable in your class, there will be a getter and a setter method. A getter method returns a value while a setter method sets the value.

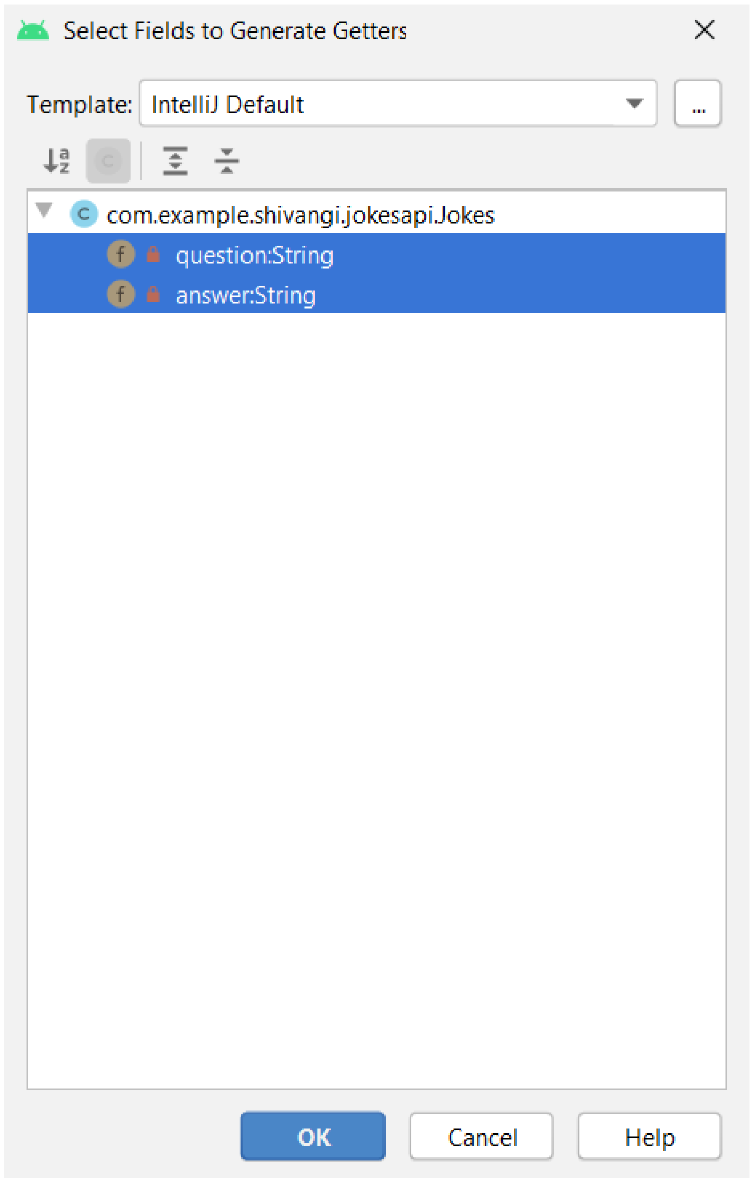
For our app, we will only use getter method as we are just retrieving the data.

To create getter methods for the two variables, right click anywhere in the class and select generate from the list. A Generate dialog box as below will appear with a list of options. Select Getter from the list.





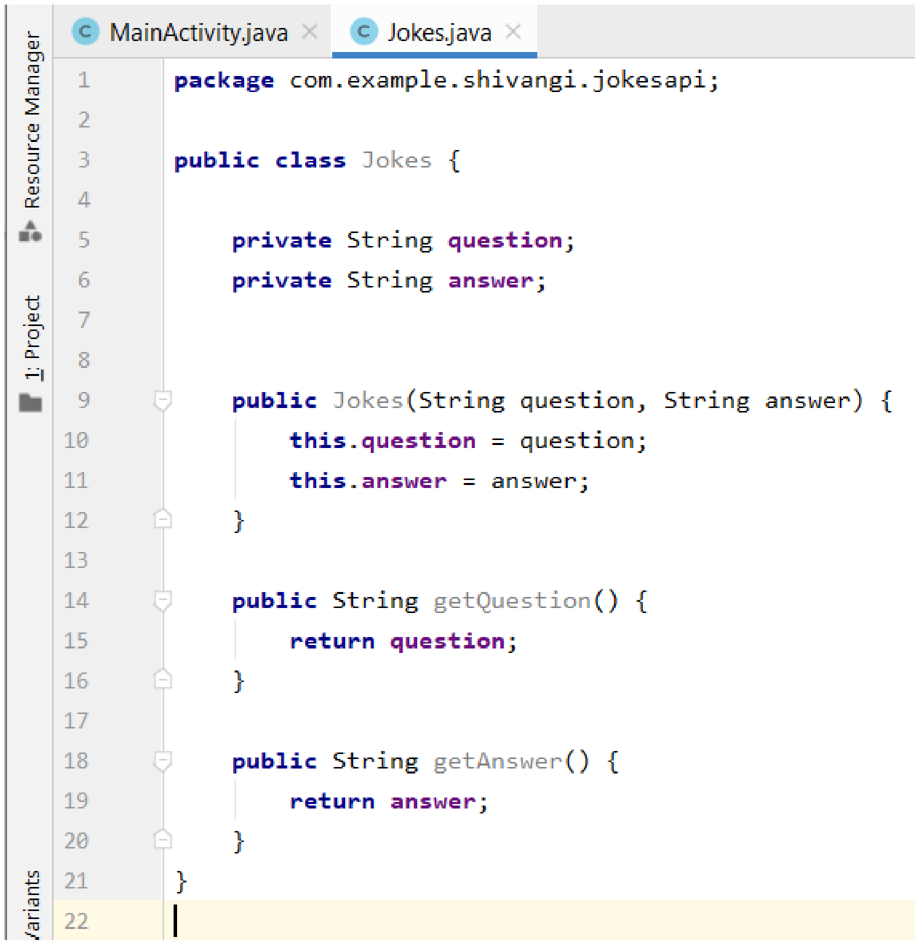
When you select getter from the list, another dialog box will appear. Select both strings and click ok.


Select Fields to Generate Getters

Template: IntelliJ Default

com.example.shivangi.jokesapi.Jokes
question:String
answer:String

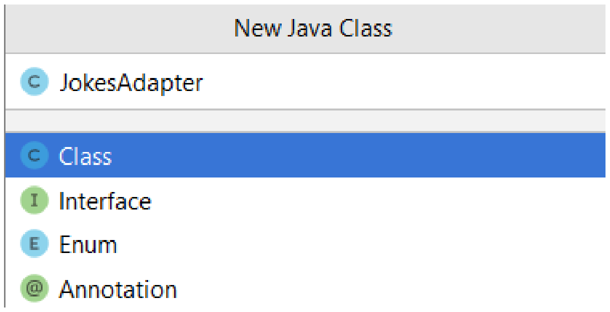
Your final Jokes.java file will look like the picture below.

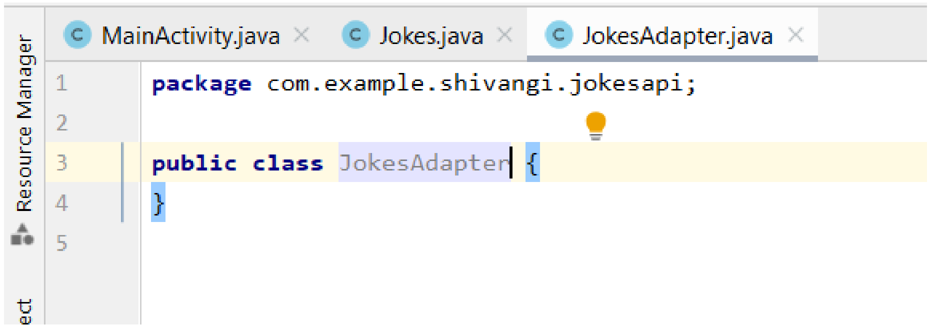


```
package com.example.shivangi.jokesapi;

public class Jokes {

    private String question;
    private String answer;


    public Jokes(String question, String answer) {
        this.question = question;
        this.answer = answer;
    }

    public String getQuestion() {
        return question;
    }

    public String getAnswer() {
        return answer;
    }
}
```

We will now create an adapter class for the RecyclerView. As we have mentioned earlier, an adapter is the bridge between the data and the RecyclerView. Also it provides access to the data items and is responsible for creating a view for each data item.

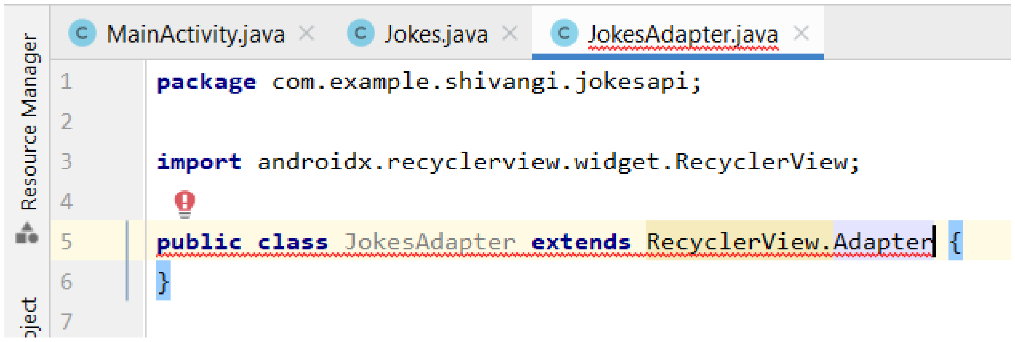
Create a new java class and name it JokesAdapter.





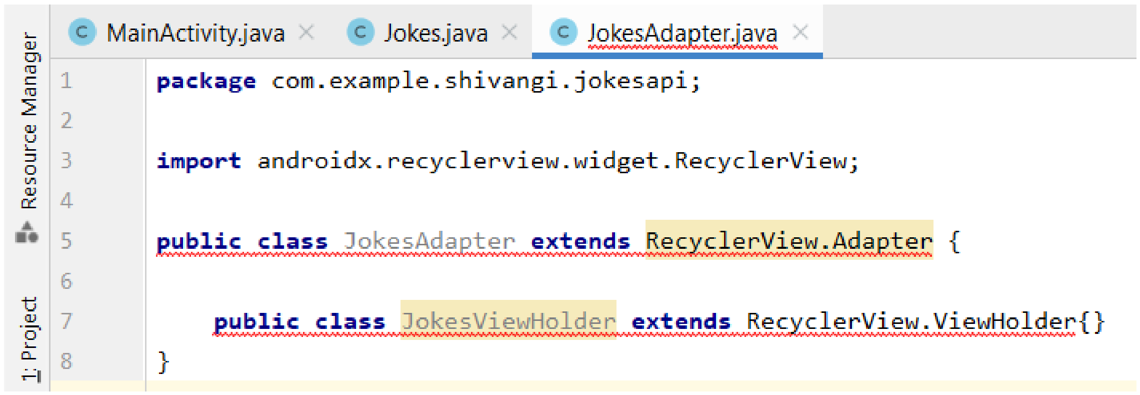
```
package com.example.shivangi.jokesapi;

public class JokesAdapter {
}
```

We will extend our JokesAdapter class to extend RecyclerView.Adapter.



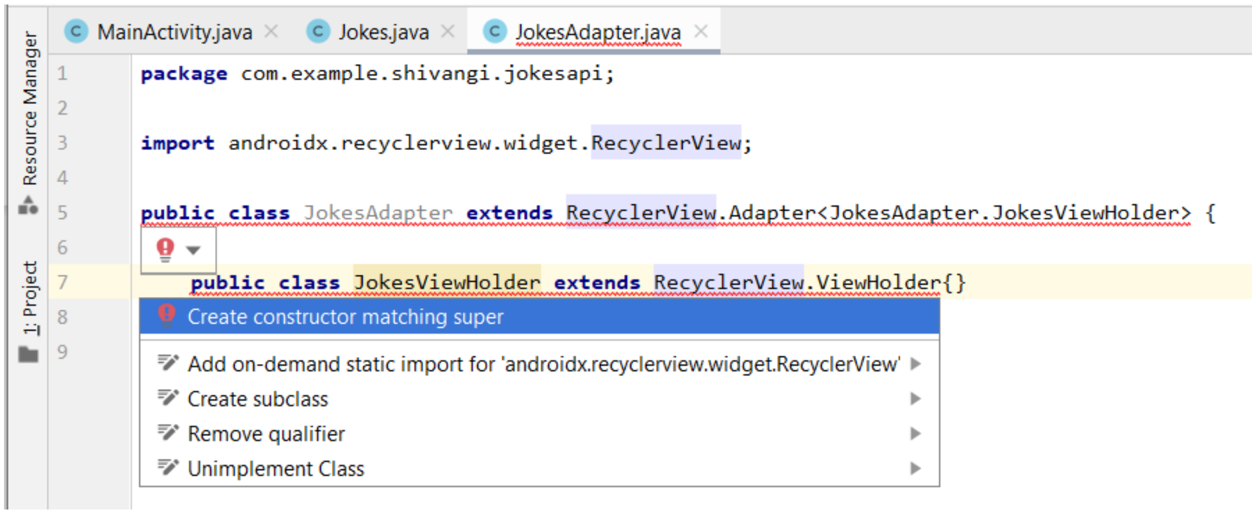
```
package com.example.shivangi.jokesapi;

import androidx.recyclerview.widget.RecyclerView;

public class JokesAdapter extends RecyclerView.Adapter {
}
```

We will now create an inner ViewHolder class and extend it RecyclerView.ViewHolder.



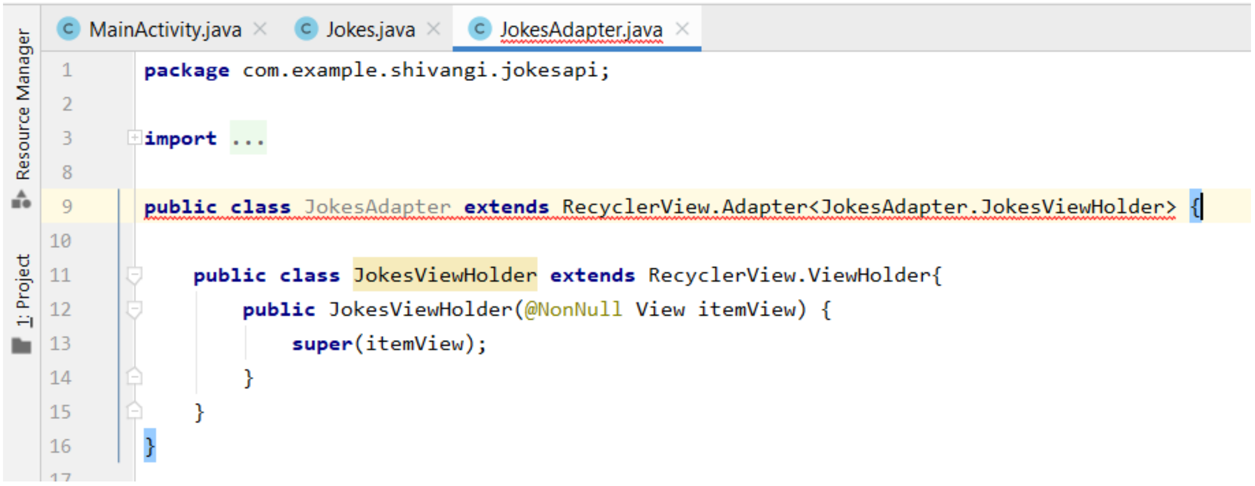
```
package com.example.shivangi.jokesapi;

import androidx.recyclerview.widget.RecyclerView;

public class JokesAdapter extends RecyclerView.Adapter {

    public class JokesViewHolder extends RecyclerView.ViewHolder{}
}
```

The red lines are due to methods that are not implemented. To implement these methods and constructors, click on the JokesViewHolder class, a red bulb icon will appear on the left. Click on it and select Create constructor matching super.



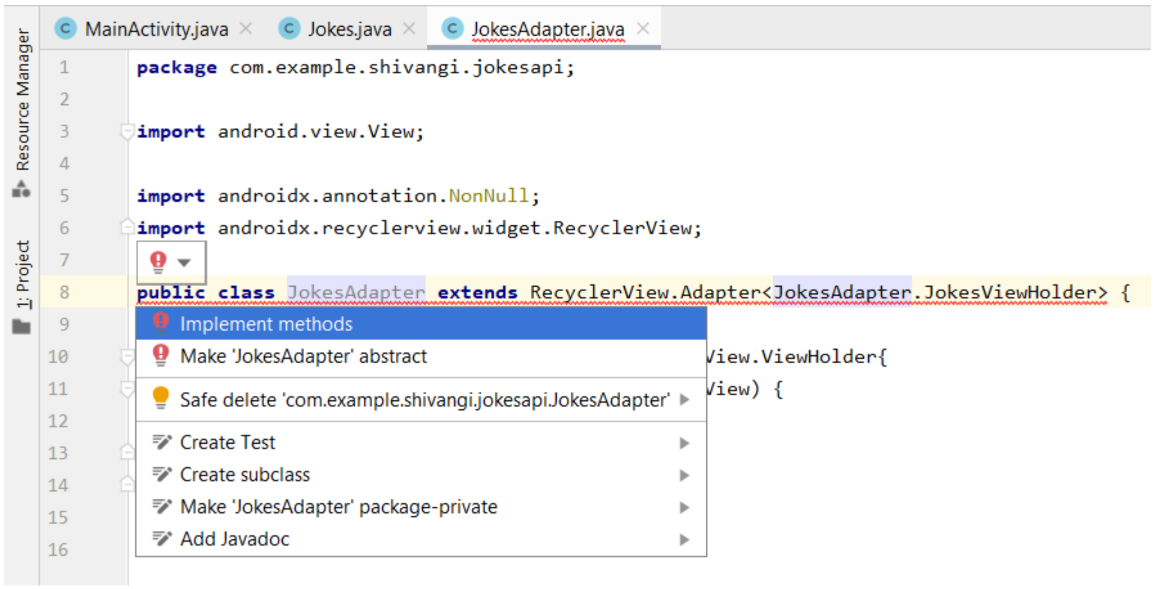
```
package com.example.shivangi.jokesapi;

import androidx.recyclerview.widget.RecyclerView;

public class JokesAdapter extends RecyclerView.Adapter<JokesAdapter.JokesViewHolder> {

    public class JokesViewHolder extends RecyclerView.ViewHolder{}
```

The constructor will be created for you.



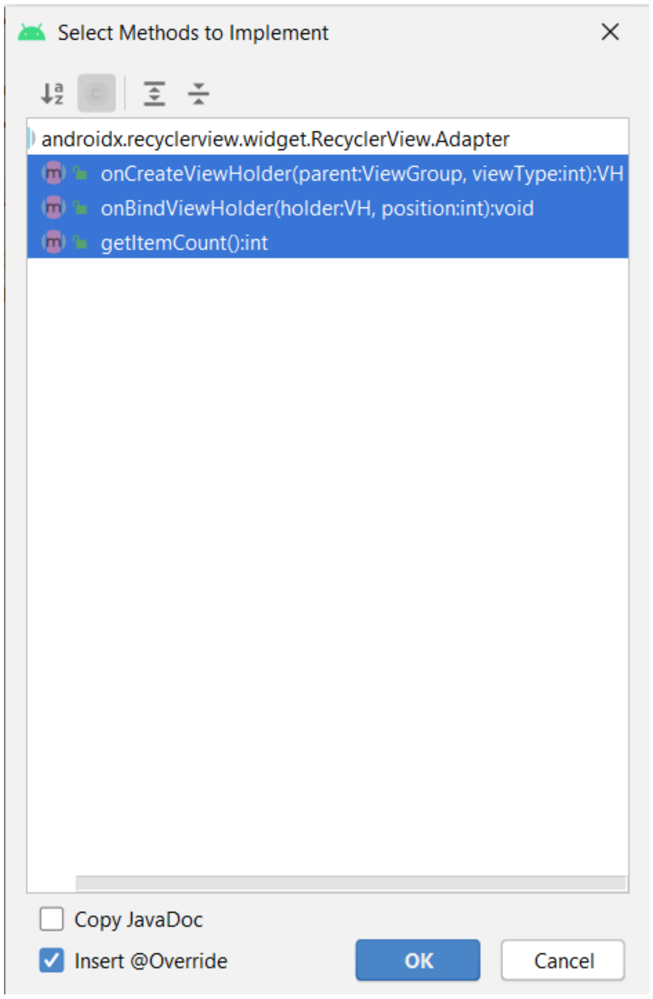
```
package com.example.shivangi.jokesapi;

import ...

public class JokesAdapter extends RecyclerView.Adapter<JokesAdapter.JokesViewHolder> {

    public class JokesViewHolder extends RecyclerView.ViewHolder{
        public JokesViewHolder(@NonNull View itemView) {
            super(itemView);
        }
    }
}
```

Now we will implement the JokesAdapter methods. Click on the JokesAdapter, a red bulb will appear on the left. From the dropdown list, select Implement methods. A dialog box will appear. You don’t have to make any changes, just click OK.



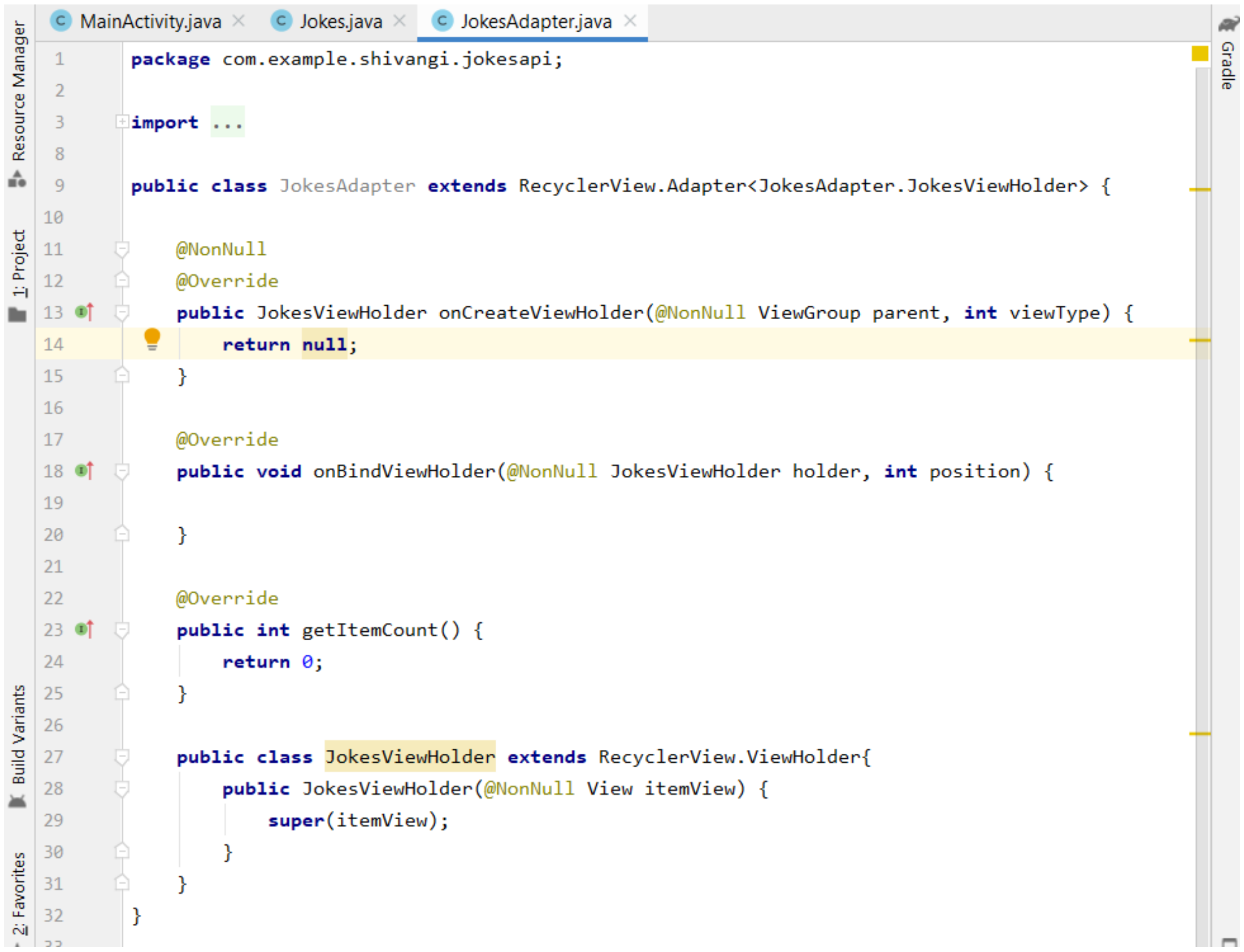
```
package com.example.shivangi.jokesapi;

import android.view.View;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

public class JokesAdapter extends RecyclerView.Adapter<JokesAdapter.JokesViewHolder> {

                                                             View.ViewHolder{
                                                             View) {
```

Implement methods
Make 'JokesAdapter' abstract
Safe delete 'com.example.shivangi.jokesapi.JokesAdapter'
Create Test
Create subclass
Make 'JokesAdapter' package-private
Add Javadoc



androidx.recyclerview.widget.RecyclerView.Adapter
- onCreateViewHolder(parent:ViewGroup, viewType:int):VH
- onBindViewHolder(holder:VH, position:int):void
- getItemCount():int

Your JokesAdapter class will look as follows.

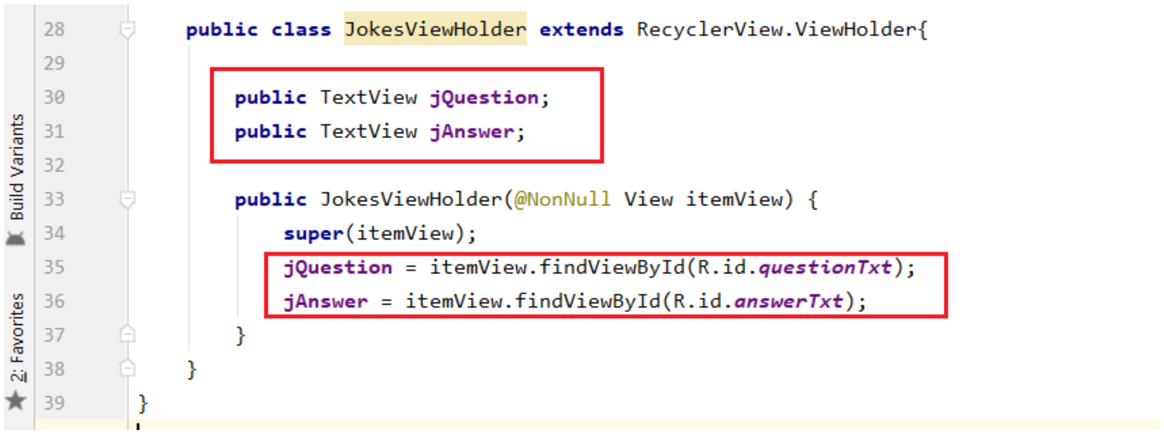


MainActivity.java Jokes.java JokesAdapter.java

```java
package com.example.shivangi.jokesapi;

import ...

public class JokesAdapter extends RecyclerView.Adapter<JokesAdapter.JokesViewHolder> {

    @NonNull
    @Override
    public JokesViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        return null;
    }

    @Override
    public void onBindViewHolder(@NonNull JokesViewHolder holder, int position) {

    }

    @Override
    public int getItemCount() {
        return 0;
    }

    public class JokesViewHolder extends RecyclerView.ViewHolder{
        public JokesViewHolder(@NonNull View itemView) {
            super(itemView);
        }
    }
}
```

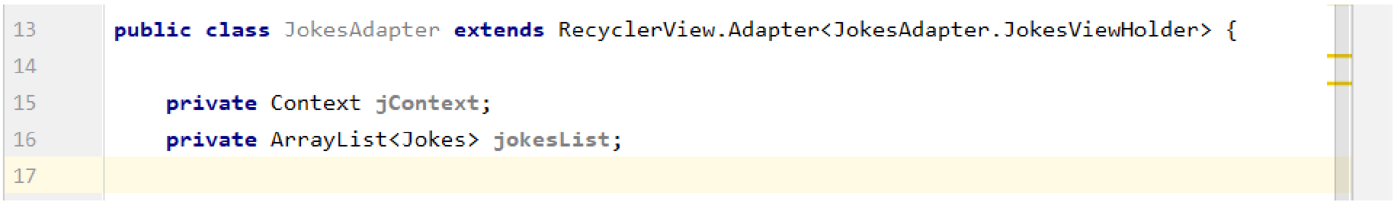
In our ViewHolder, we will add variables. These variables will be assigned to refer to the components in our item_layout.xml. We use the itemView parameter provided in the JokesViewHolder constructor to access the components.

Add the following code in your JokesViewHolder.



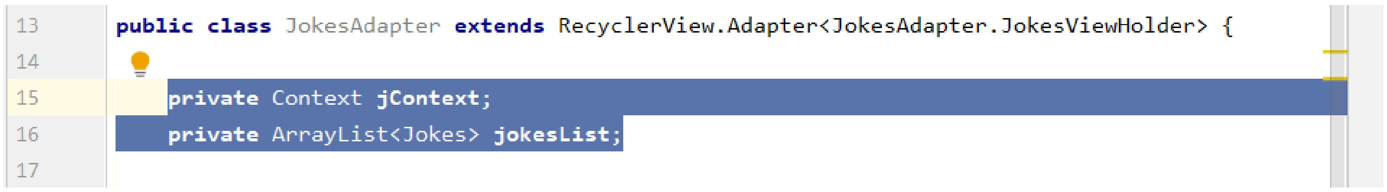
```java
    public class JokesViewHolder extends RecyclerView.ViewHolder{

        public TextView jQuestion;
        public TextView jAnswer;

        public JokesViewHolder(@NonNull View itemView) {
            super(itemView);
            jQuestion = itemView.findViewById(R.id.questionTxt);
            jAnswer = itemView.findViewById(R.id.answerTxt);
        }
    }
}
```

We will add two class variables in our JokesAdapter class, one referring to the context of the Adapter and the other referring to the data set which is presented in ArrayList.



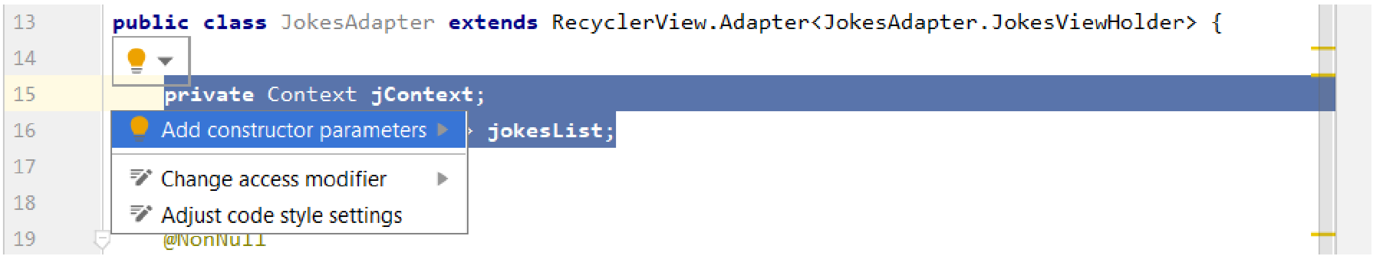
```
public class JokesAdapter extends RecyclerView.Adapter<JokesAdapter.JokesViewHolder> {

    private Context jContext;
    private ArrayList<Jokes> jokesList;

```

We also need a constructor for JokesAdapter class. Select the two variables. A yellow bulb icon will appear on the left.



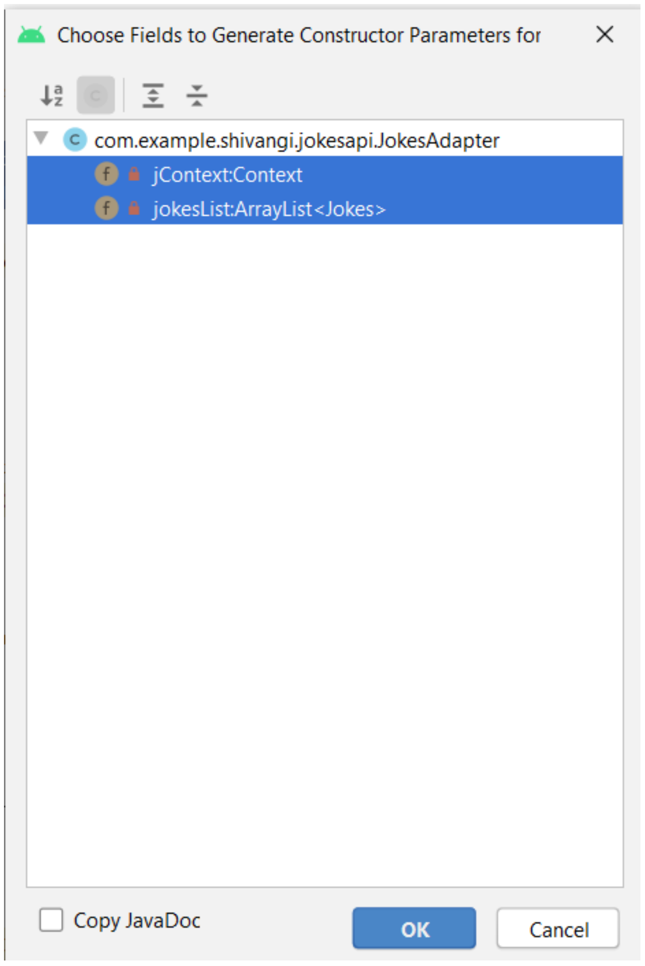
```
public class JokesAdapter extends RecyclerView.Adapter<JokesAdapter.JokesViewHolder> {

    private Context jContext;
    private ArrayList<Jokes> jokesList;

```

From the dropdown, select Add constructor parameter.

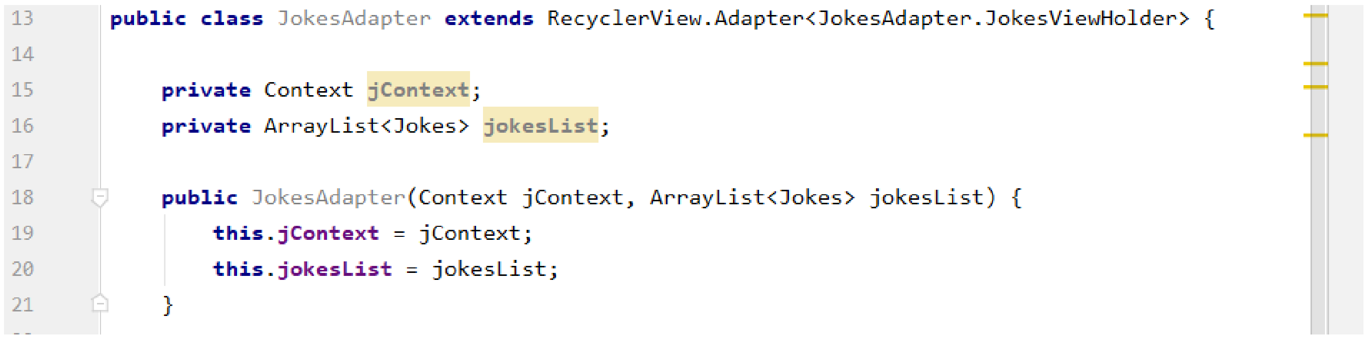


```
public class JokesAdapter extends RecyclerView.Adapter<JokesAdapter.JokesViewHolder> {

    private Context jContext;
                                  jokesList;



    @NonNull
```

A dialog box will appear. Select both variables and click ok.


Choose Fields to Generate Constructor Parameters for
com.example.shivangi.jokesapi.JokesAdapter
jContext:Context
jokesList:ArrayList<Jokes>
Copy JavaDoc
OK
Cancel

Your constructor looks as follows:

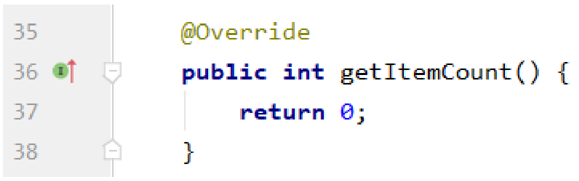


```
public class JokesAdapter extends RecyclerView.Adapter<JokesAdapter.JokesViewHolder> {

    private Context jContext;
    private ArrayList<Jokes> jokesList;

    public JokesAdapter(Context jContext, ArrayList<Jokes> jokesList) {
        this.jContext = jContext;
        this.jokesList = jokesList;
    }
```

Now let’s look at the three methods we implemented in our JokesAdapter: onCreateViewHolder, onBindViewHolder and getItemCount.

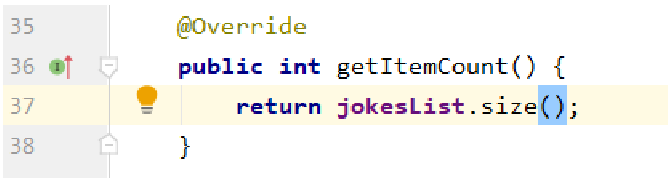
Let’s start with getItemCount method. This method simply gets the size of the array.

The default returns a zero.



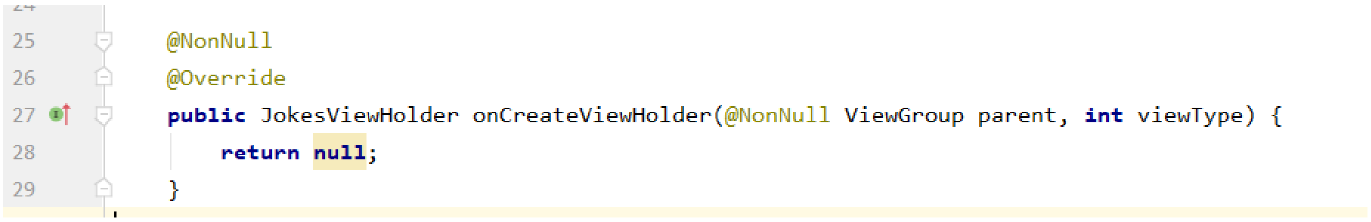
```
    @Override
    public int getItemCount() {
        return 0;
    }
```

Change the return value to return the size of the ArrayList.



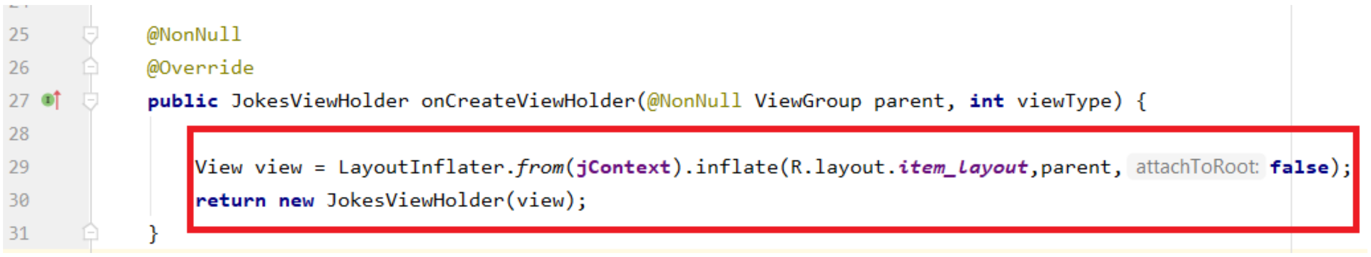
```
    @Override
    public int getItemCount() {
        return jokesList.size();
    }
```

onCreateViewHolder method will inflate the item_layout for each data in the ArrayList. Your method currently looks as below.



```
    @NonNull
    @Override
    public JokesViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        return null;
    }
```

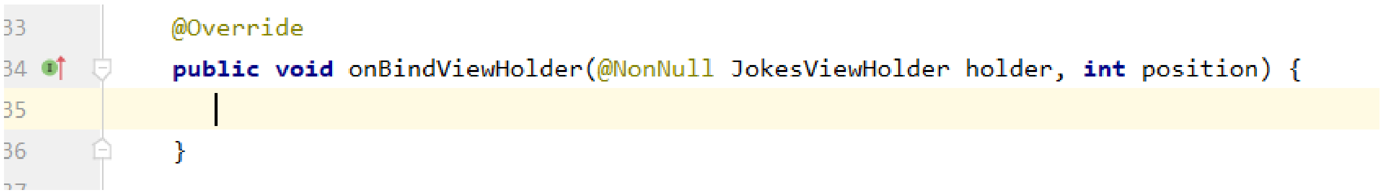
Add the following code.



```
    @NonNull
    @Override
    public JokesViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

        View view = LayoutInflater.from(jContext).inflate(R.layout.item_layout,parent, attachToRoot: false);
        return new JokesViewHolder(view);
    }
```

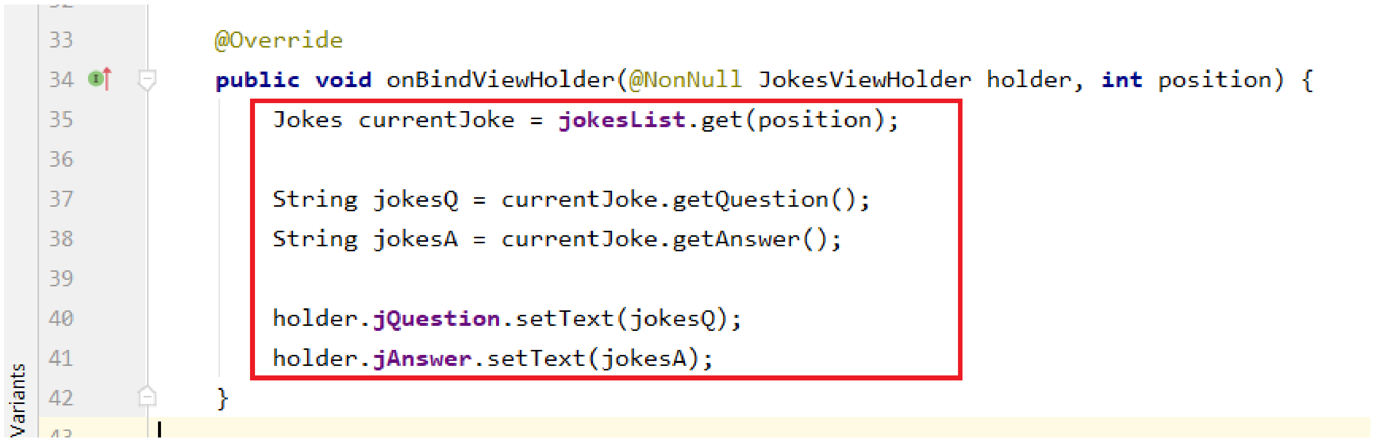
The last method in our JokesAdapter class is the onBindViewHolder method. This will get the position and value of the current item. The values of the current item will be stored in variables. We then set the views with the help of the JokesViewHolder.

Currently your onBindViewHolder method looks as follows:



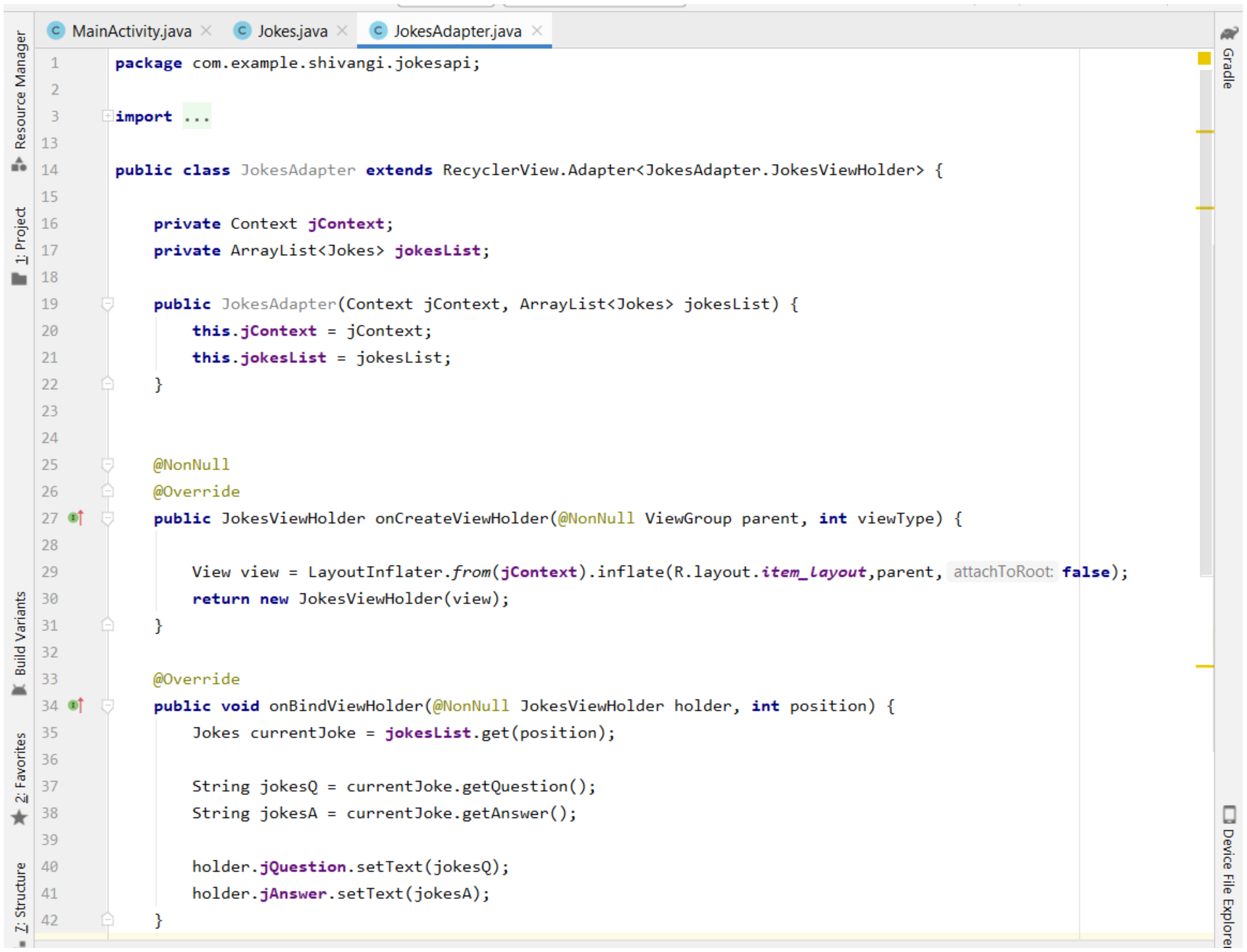
```
    @Override
    public void onBindViewHolder(@NonNull JokesViewHolder holder, int position) {
        
    }
```

Add the following code to the method:

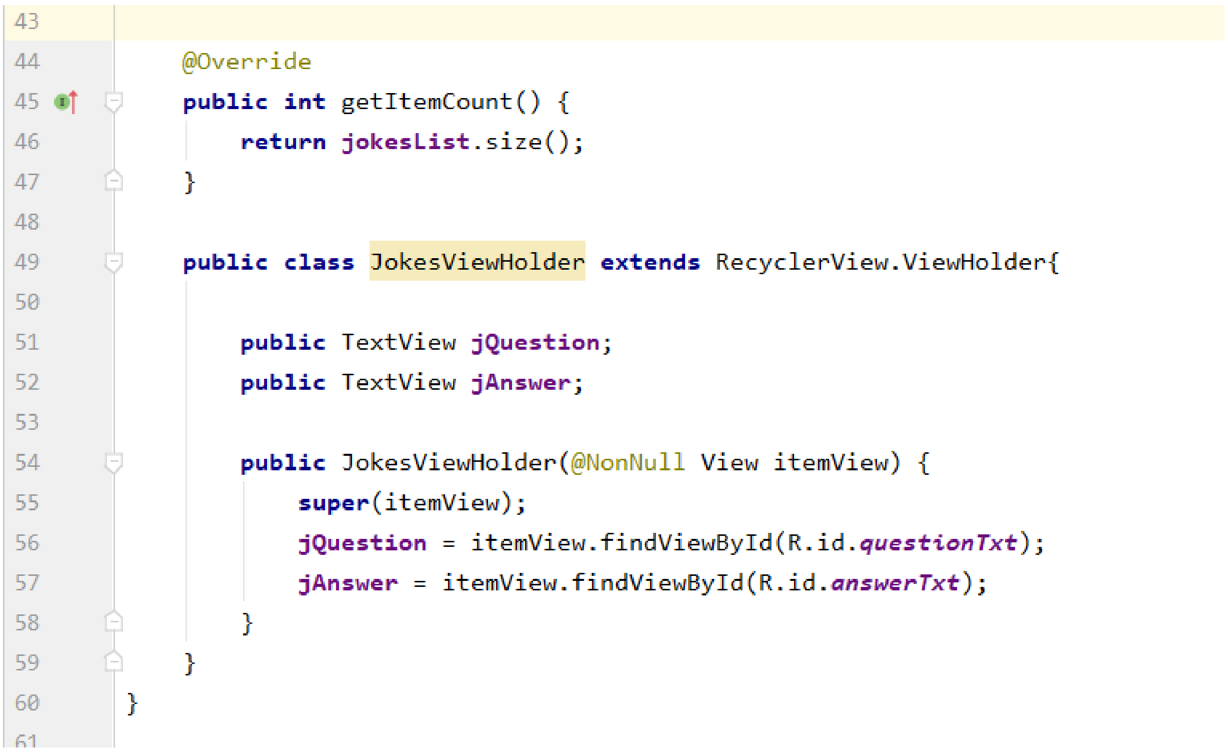


```
    @Override
    public void onBindViewHolder(@NonNull JokesViewHolder holder, int position) {
        Jokes currentJoke = jokesList.get(position);

        String jokesQ = currentJoke.getQuestion();
        String jokesA = currentJoke.getAnswer();

        holder.jQuestion.setText(jokesQ);
        holder.jAnswer.setText(jokesA);
    }
```

Your final JokesAdapter class should look as follows:



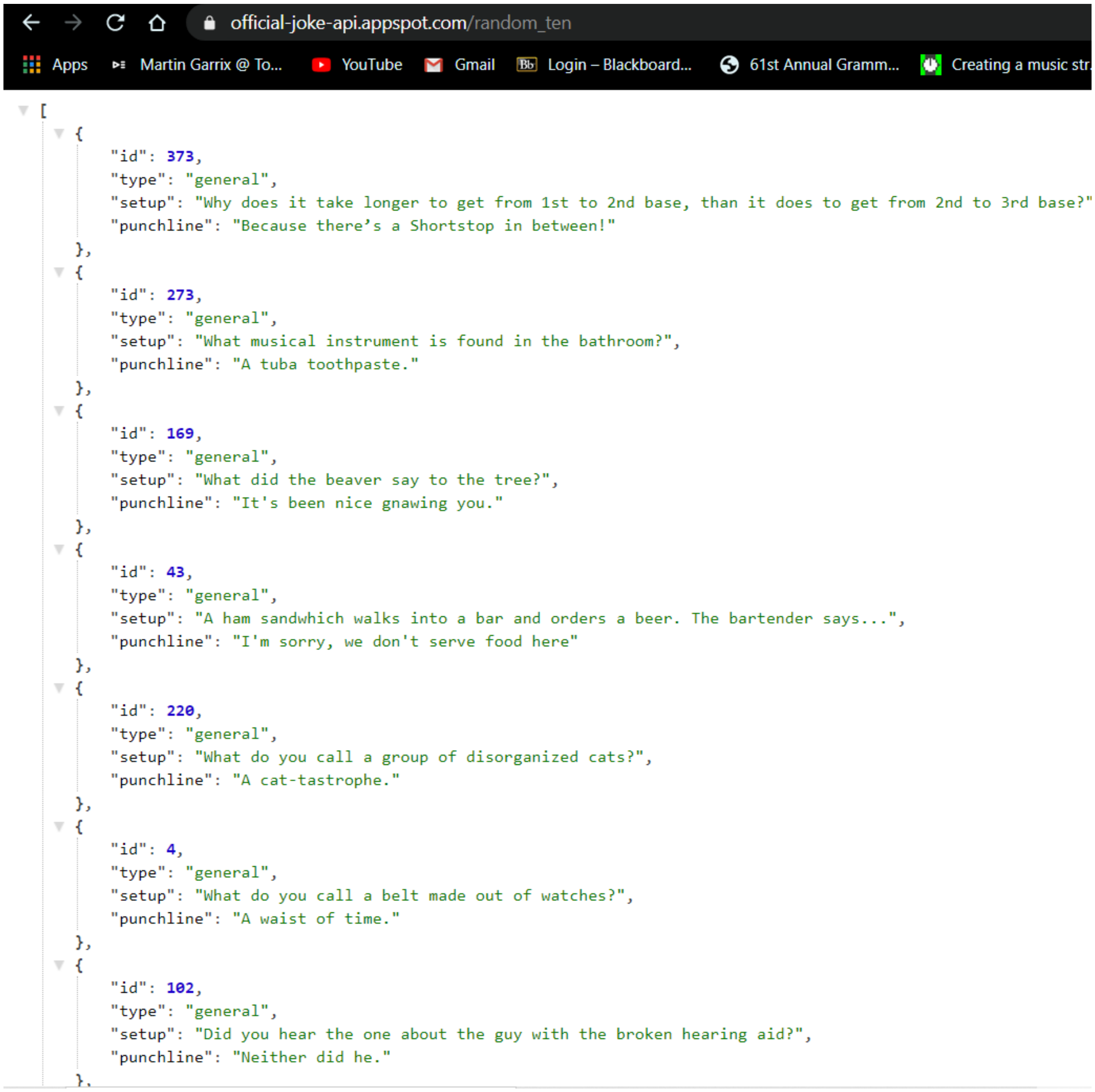
```java
package com.example.shivangi.jokesapi;

import ...

public class JokesAdapter extends RecyclerView.Adapter<JokesAdapter.JokesViewHolder> {

    private Context jContext;
    private ArrayList<Jokes> jokesList;

    public JokesAdapter(Context jContext, ArrayList<Jokes> jokesList) {
        this.jContext = jContext;
        this.jokesList = jokesList;
    }


    @NonNull
    @Override
    public JokesViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

        View view = LayoutInflater.from(jContext).inflate(R.layout.item_layout,parent, attachToRoot: false);
        return new JokesViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull JokesViewHolder holder, int position) {
        Jokes currentJoke = jokesList.get(position);

        String jokesQ = currentJoke.getQuestion();
        String jokesA = currentJoke.getAnswer();

        holder.jQuestion.setText(jokesQ);
        holder.jAnswer.setText(jokesA);
    }
```



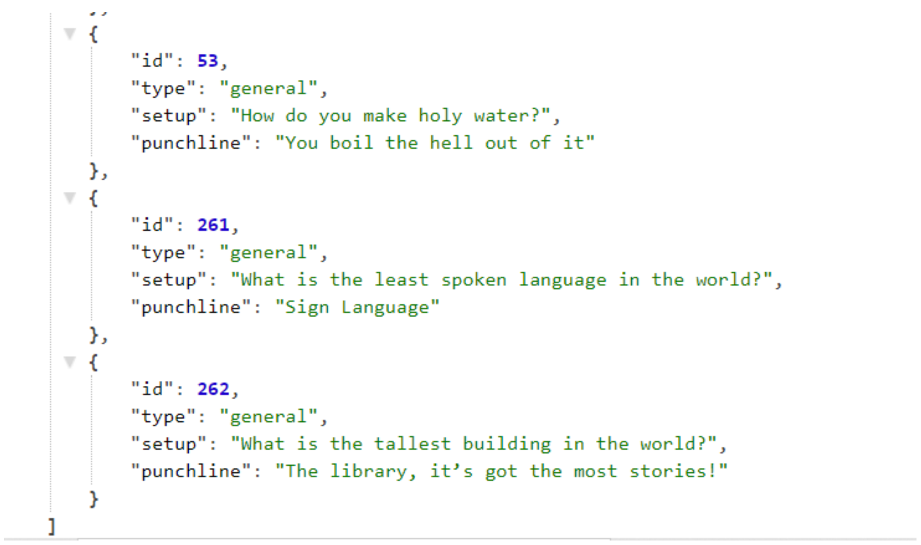
```java

    @Override
    public int getItemCount() {
        return jokesList.size();
    }

    public class JokesViewHolder extends RecyclerView.ViewHolder{

        public TextView jQuestion;
        public TextView jAnswer;

        public JokesViewHolder(@NonNull View itemView) {
            super(itemView);
            jQuestion = itemView.findViewById(R.id.questionTxt);
            jAnswer = itemView.findViewById(R.id.answerTxt);
        }
    }
}
```

Now that we have created the adapter for our ArrayList, let's move to MainActiivty.java.

If you go to the URL I provided at the beginning of this document (https://official-joke-api.appspot.com/random_ten), it will display the following data:



```
[
  {
    "id": 373,
    "type": "general",
    "setup": "Why does it take longer to get from 1st to 2nd base, than it does to get from 2nd to 3rd base?"
    "punchline": "Because there’s a Shortstop in between!"
  },
  {
    "id": 273,
    "type": "general",
    "setup": "What musical instrument is found in the bathroom?",
    "punchline": "A tuba toothpaste."
  },
  {
    "id": 169,
    "type": "general",
    "setup": "What did the beaver say to the tree?",
    "punchline": "It's been nice gnawing you."
  },
  {
    "id": 43,
    "type": "general",
    "setup": "A ham sandwhich walks into a bar and orders a beer. The bartender says...",
    "punchline": "I'm sorry, we don't serve food here"
  },
  {
    "id": 220,
    "type": "general",
    "setup": "What do you call a group of disorganized cats?",
    "punchline": "A cat-tastrophe."
  },
  {
    "id": 4,
    "type": "general",
    "setup": "What do you call a belt made out of watches?",
    "punchline": "A waist of time."
  },
  {
    "id": 102,
    "type": "general",
    "setup": "Did you hear the one about the guy with the broken hearing aid?",
    "punchline": "Neither did he."
  },
```

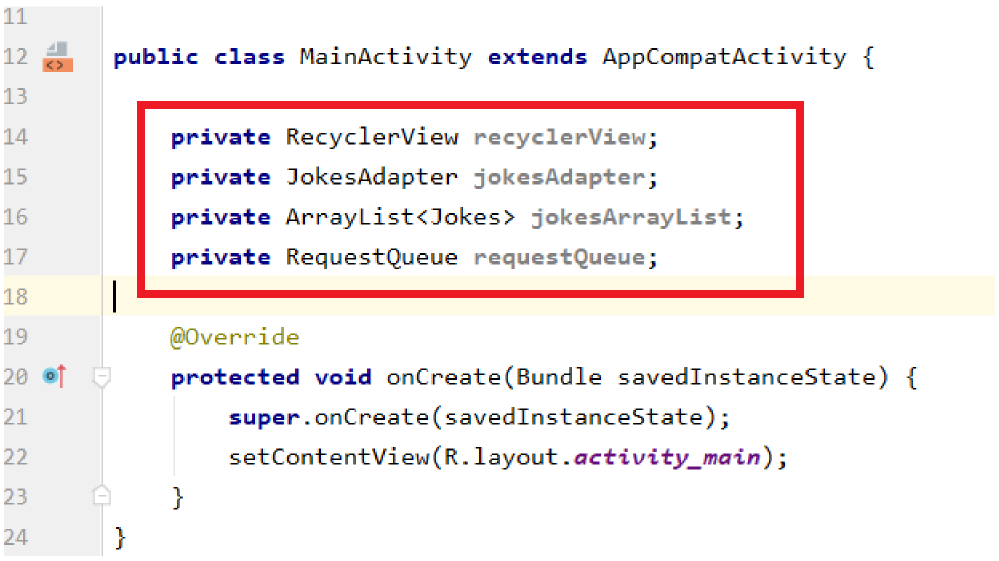


```
  {
      "id": 53,
      "type": "general",
      "setup": "How do you make holy water?",
      "punchline": "You boil the hell out of it"
  },
  {
      "id": 261,
      "type": "general",
      "setup": "What is the least spoken language in the world?",
      "punchline": "Sign Language"
  },
  {
      "id": 262,
      "type": "general",
      "setup": "What is the tallest building in the world?",
      "punchline": "The library, it’s got the most stories!"
  }
]
```

This is the JSON data we will retrieve.

NOTE: the values in the data will vary as it will retrieve random strings of jokes.

Add the following class variables to your MainActivity.java class.



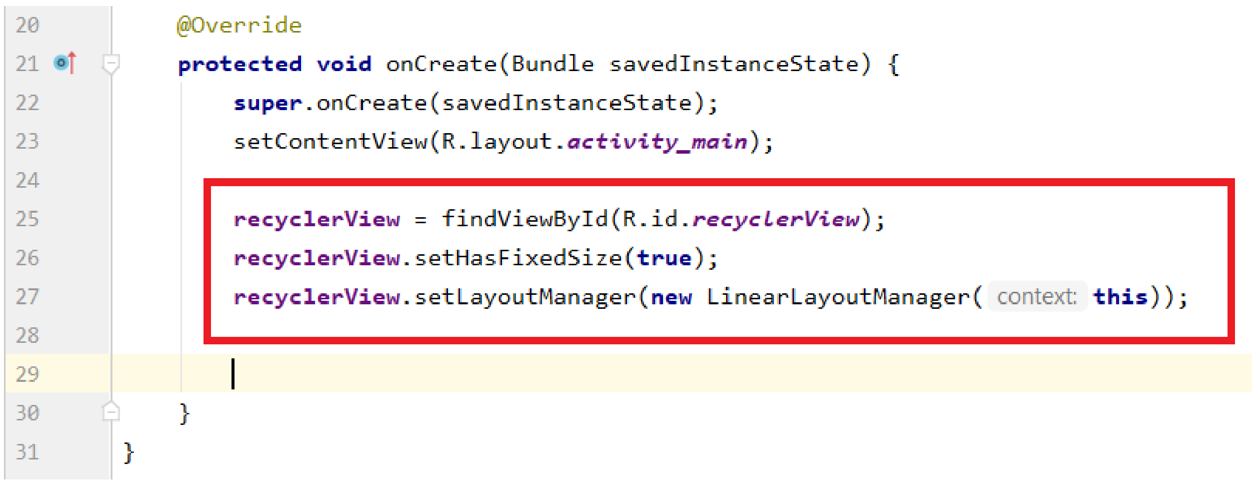
```java

public class MainActivity extends AppCompatActivity {

    private RecyclerView recyclerView;
    private JokesAdapter jokesAdapter;
    private ArrayList<Jokes> jokesArrayList;
    private RequestQueue requestQueue;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

A RequestQueue is used to queue all the requests and handle the responses. It manages the worker threads for running the network operations besides reading and writing to and from the cache.

Volley provides two classes for Json requests:

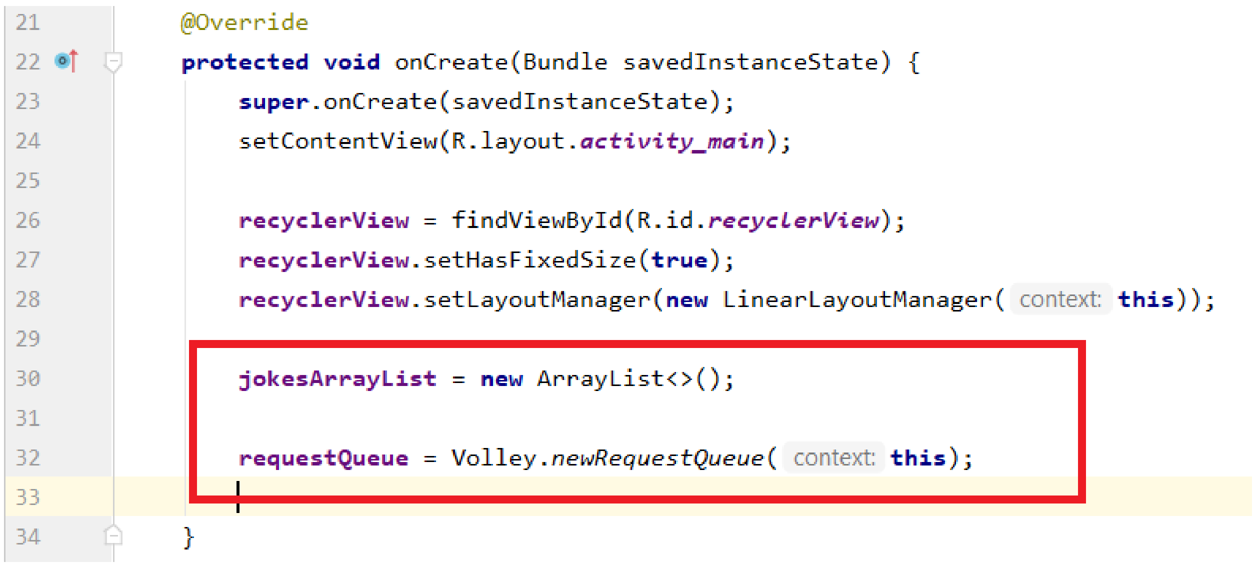
1) **JsonArrayRequest** – it is used to retrieve a JSONArray from a Url.
2) **JsonObjectRequest** – it is used to retrieve JSONObject from a Url.

Now we will access the RecyclerView component in our activity_main.xml file and set the LayoutManager for our RecyclerView. In the code below, we have used the setHasFixedSize() method to ensure that the layout is fixed. This increases the performance of the app.



```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        recyclerView = findViewById(R.id.recyclerView);
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager( context: this));

    }
}
```

We will create a new ArrayList to store the data we will retrieve from the API and request a new volley queue for our MainActivty.java class. Add the following code:



```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        recyclerView = findViewById(R.id.recyclerView);
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager( context: this));

        jokesArrayList = new ArrayList<>();

        requestQueue = Volley.newRequestQueue( context: this);

    }
```

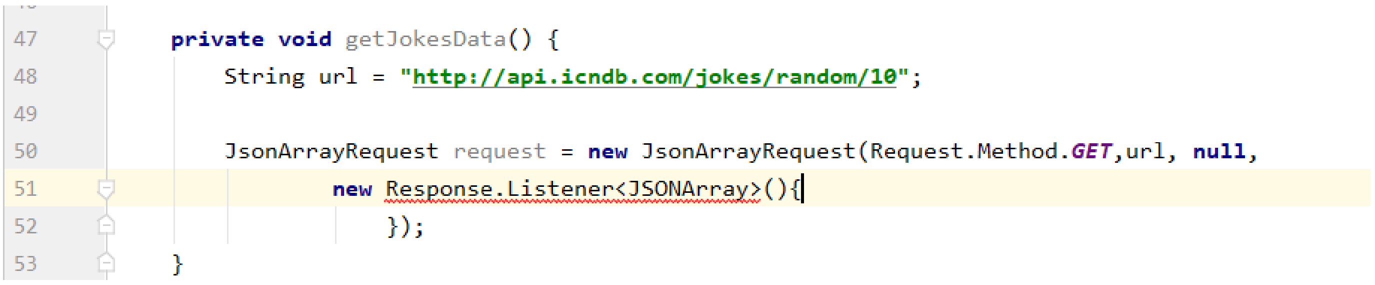
To make our work neat, we will create a method that will parse JSON from the URL. Since our JSON file is an array, we will use JsonArrayRequest to retrieve data.

The request method has the following parameters :

- Request Method – can be GET, POST, PUT etc

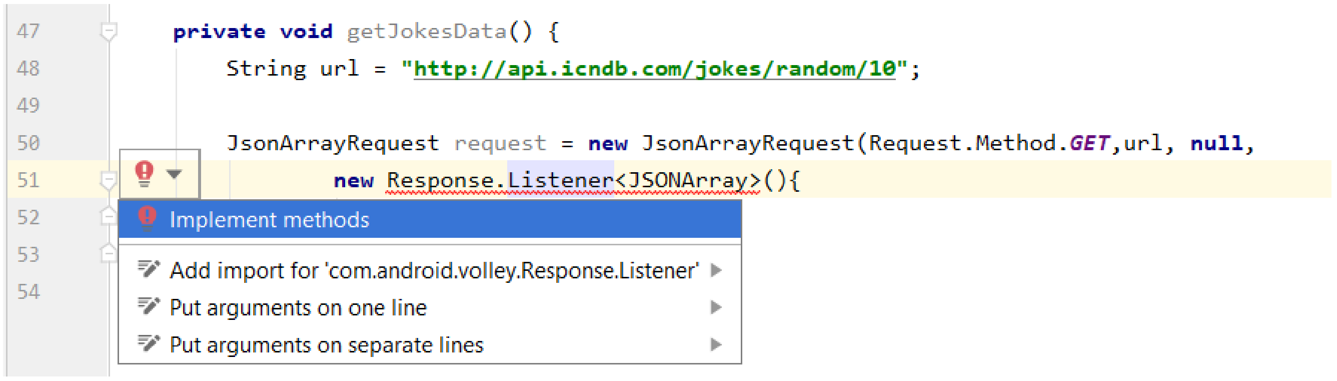
- URL – it is the http url string.
- JSONObject – it is an optional object posted with the request, usually is null.
- ResponseListener – its callback method will contain the response.
- ErrorListener – its callback method contains the details of the problem, if any.

Add the following code below your onCreate() method in the MainActivity.java:



```
    private void getJokesData() {
        String url = "http://api.icndb.com/jokes/random/10";

        JsonArrayRequest request = new JsonArrayRequest(Request.Method.GET,url, null,
                new Response.Listener<JSONArray>(){
                    });
    }
```

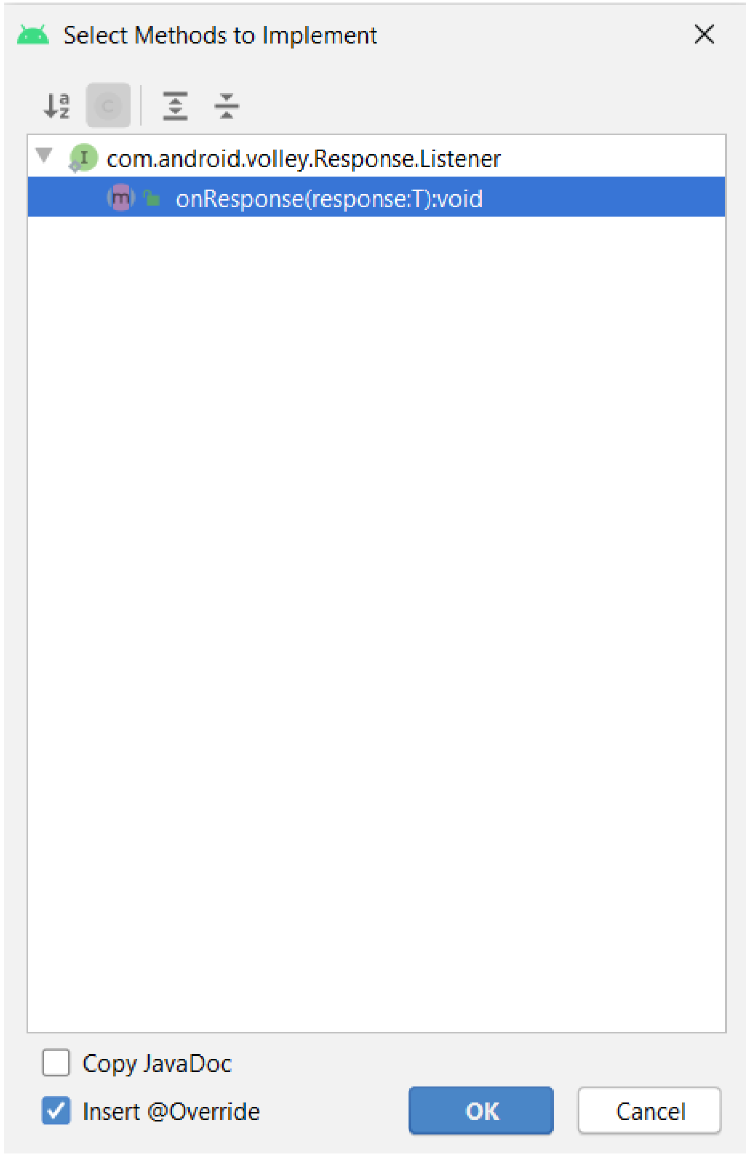
When you click on the code with the red lines, a red bulb will appear on the left. Click on it and select implement methods.



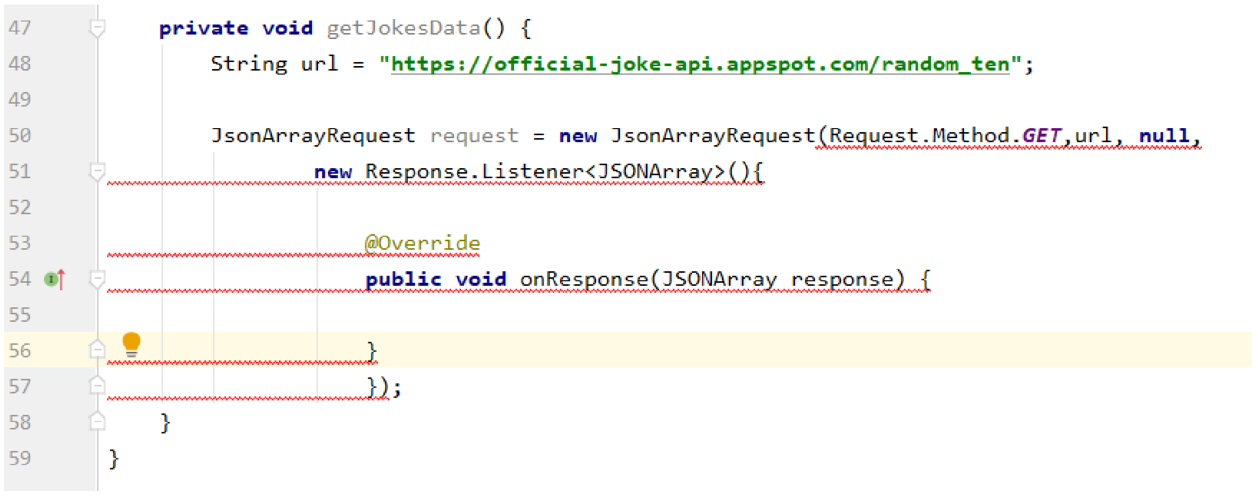
```
    private void getJokesData() {
        String url = "http://api.icndb.com/jokes/random/10";

        JsonArrayRequest request = new JsonArrayRequest(Request.Method.GET,url, null,
                new Response.Listener<JSONArray>(){
```

Implement methods
Add import for 'com.android.volley.Response.Listener'
Put arguments on one line
Put arguments on separate lines

Select ok on the dialog box.

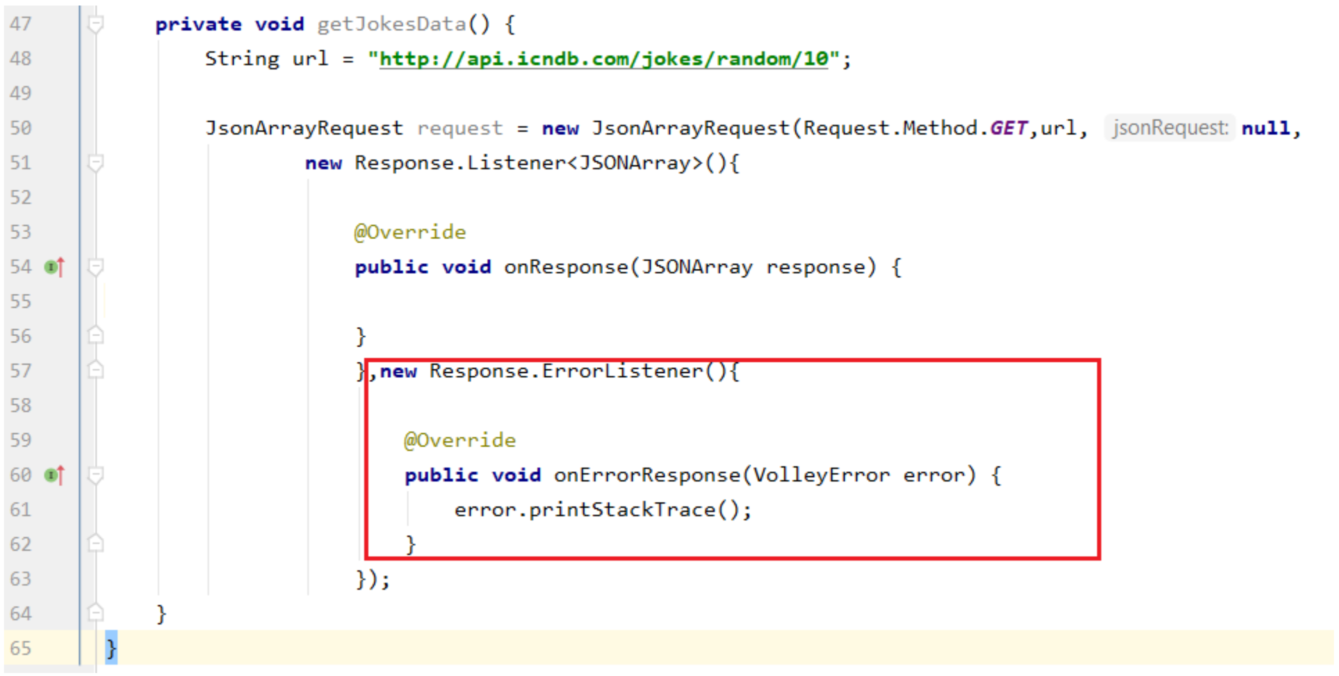

Select Methods to Implement
com.android.volley.Response.Listener
onResponse(response:T):void
Copy JavaDoc
Insert @Override
OK
Cancel

Your code will have many red lines as we have not completed the request. To complete the request, we need to implement the ErrorListener.



```
    private void getJokesData() {
        String url = "https://official-joke-api.appspot.com/random_ten";

        JsonArrayRequest request = new JsonArrayRequest(Request.Method.GET,url, null,
                new Response.Listener<JSONArray>(){

                    @Override
                    public void onResponse(JSONArray response) {

                    }
                    });
    }
}
```

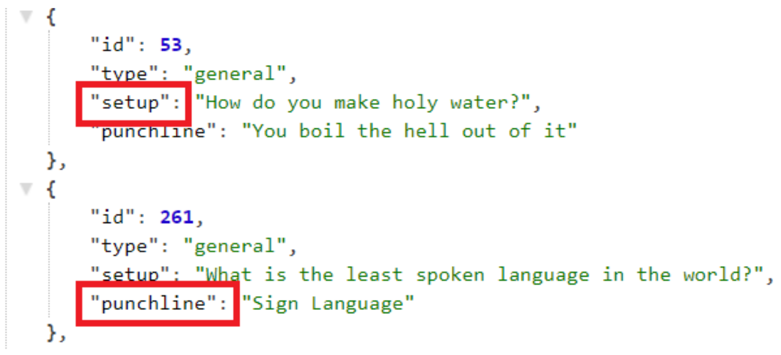
To add the error listener, add the following code after the onResponse(JSONArray response) method.



```
    private void getJokesData() {
        String url = "http://api.icndb.com/jokes/random/10";

        JsonArrayRequest request = new JsonArrayRequest(Request.Method.GET,url,  jsonRequest: null,
                new Response.Listener<JSONArray>(){

                    @Override
                    public void onResponse(JSONArray response) {

                    }
                    },new Response.ErrorListener(){

                        @Override
                        public void onErrorResponse(VolleyError error) {
                            error.printStackTrace();
                        }
                    });
    }
}
```

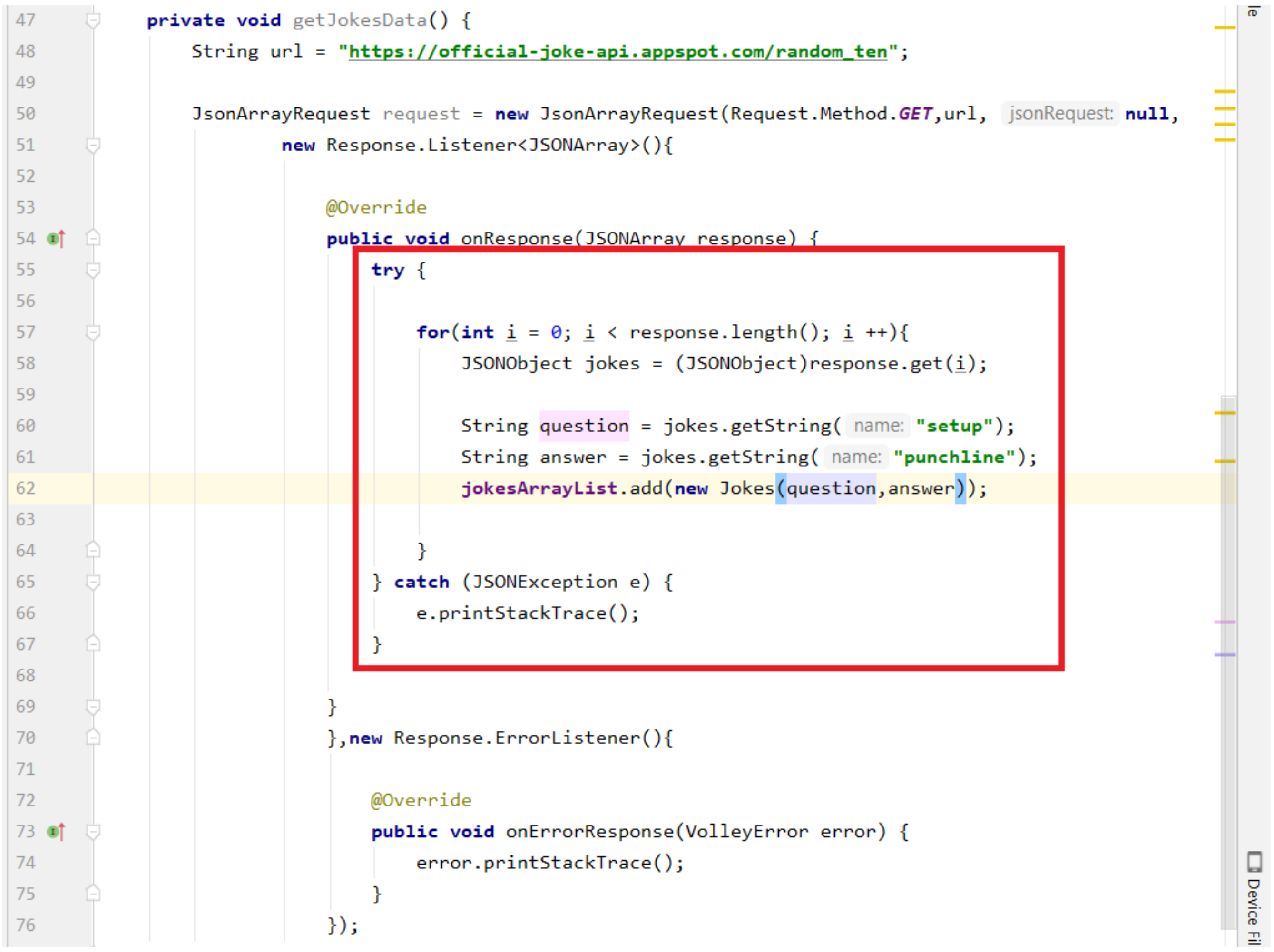
In our onResponse() method, we will declare a new JSONObject variable and use a for loop to get the item at the particular position.

Two variables will hold the data retrieved and with the new array list we created, the data will be passed to that list. In the code below, the question string holds the jokes.getString("setup"); , the setup name is obtained from the array, it refers to the field name in the jsonArray file.



```
{
    "id": 53,
    "type": "general",
    "setup": "How do you make holy water?",
    "punchline": "You boil the hell out of it"
},
{
    "id": 261,
    "type": "general",
    "setup": "What is the least spoken language in the world?",
    "punchline": "Sign Language"
},
```

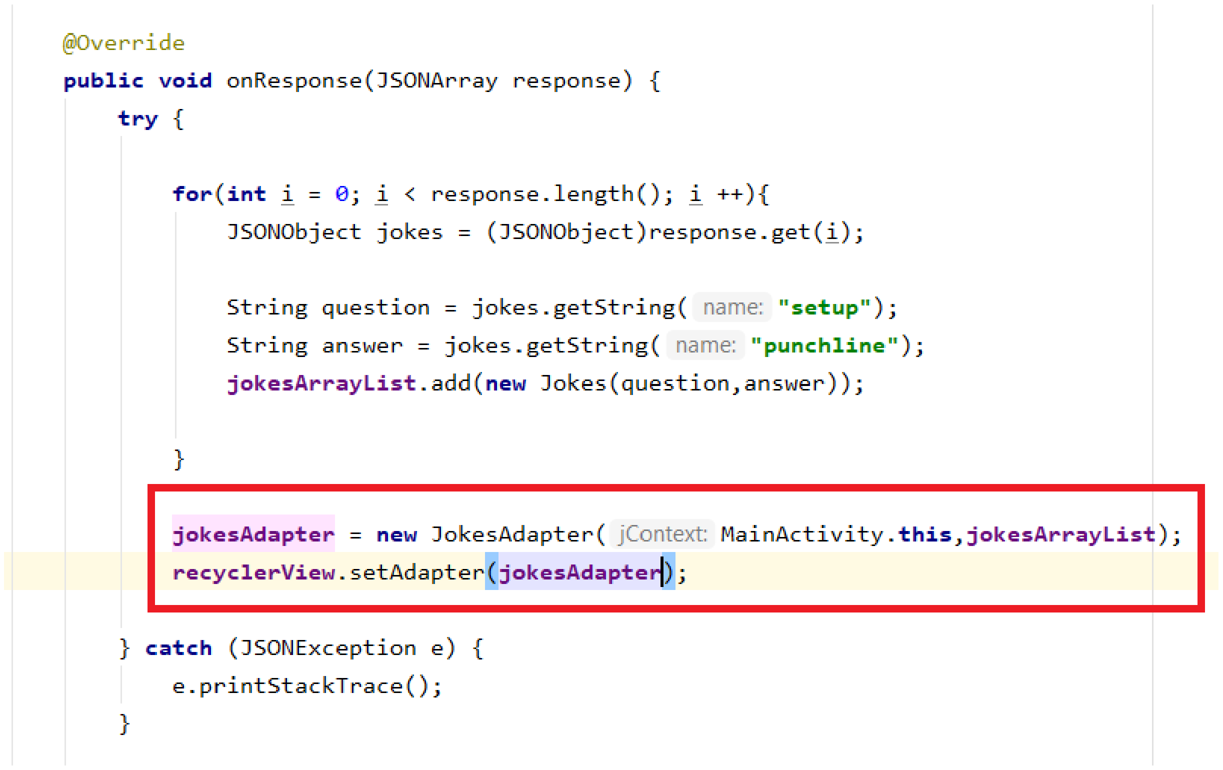
Add the following code:



```
private void getJokesData() {
    String url = "https://official-joke-api.appspot.com/random_ten";

    JsonArrayRequest request = new JsonArrayRequest(Request.Method.GET,url, jsonRequest: null,
            new Response.Listener<JSONArray>(){

                @Override
                public void onResponse(JSONArray response) {
                    try {

                        for(int i = 0; i < response.length(); i ++){
                            JSONObject jokes = (JSONObject)response.get(i);

                            String question = jokes.getString( name: "setup");
                            String answer = jokes.getString( name: "punchline");
                            jokesArrayList.add(new Jokes(question,answer));

                        }
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }

                }
            },new Response.ErrorListener(){

                @Override
                public void onErrorResponse(VolleyError error) {
                    error.printStackTrace();
                }
            });
```

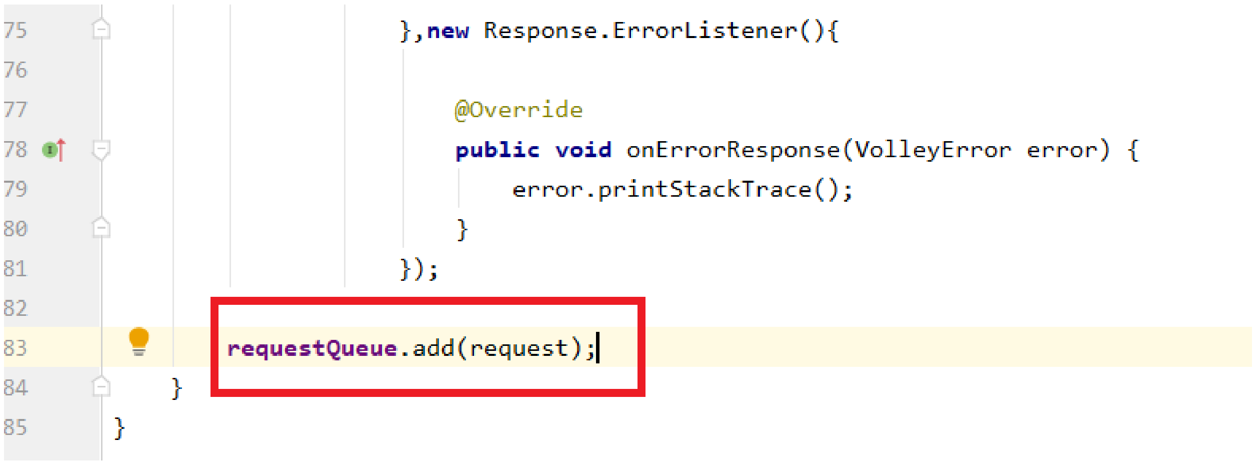
Lastly, we will set up the adapter for our RecyclerView, add the response to our requestQueue and call the getJokesData() method in our onCreate() method.

To set up the adapter, add the following code after the for loop in the try block.



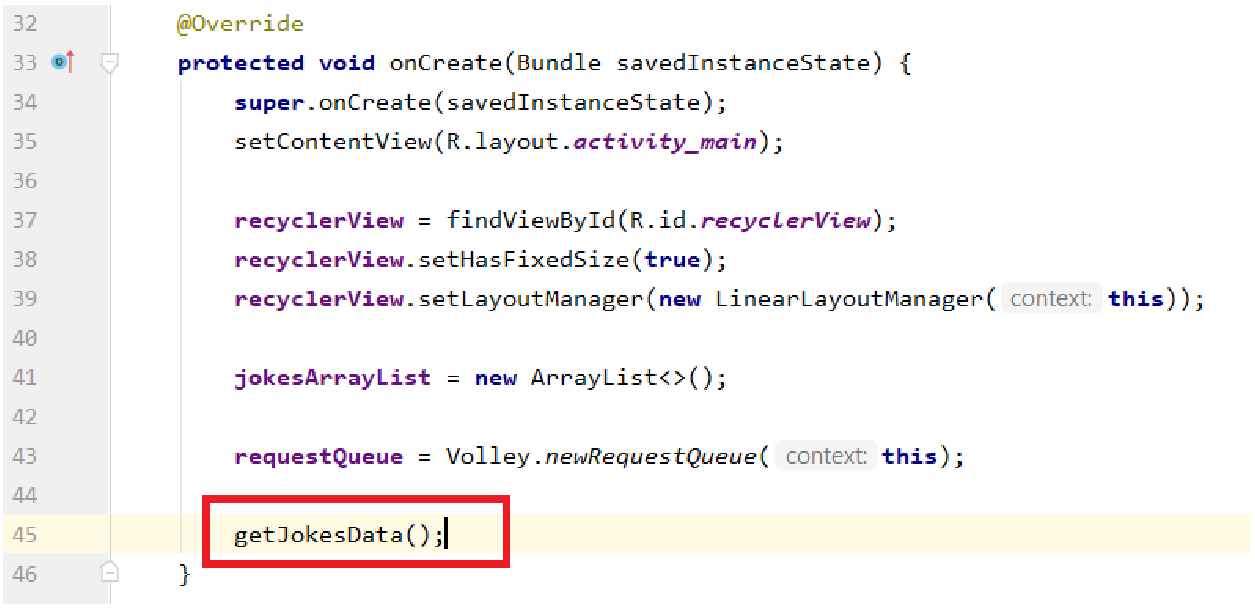
```
@Override
public void onResponse(JSONArray response) {
    try {

        for(int i = 0; i < response.length(); i ++){
            JSONObject jokes = (JSONObject)response.get(i);

            String question = jokes.getString( name: "setup");
            String answer = jokes.getString( name: "punchline");
            jokesArrayList.add(new Jokes(question,answer));

        }

        jokesAdapter = new JokesAdapter( jContext: MainActivity.this,jokesArrayList);
        recyclerView.setAdapter(jokesAdapter);

    } catch (JSONException e) {
        e.printStackTrace();
    }
```

At the end of the getJokesData() method, add the response to the requestQueue.



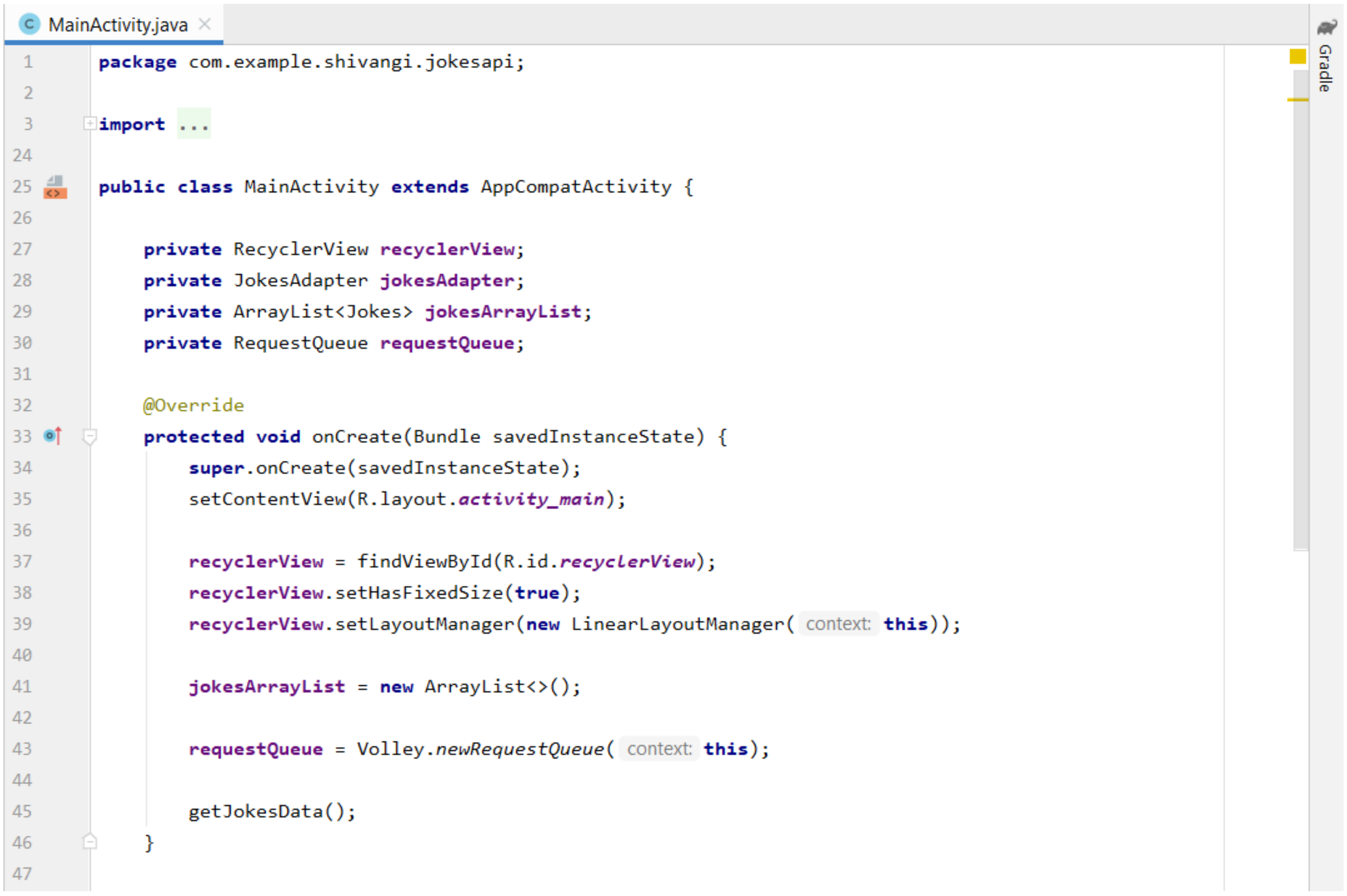
```
                },new Response.ErrorListener(){

                    @Override
                    public void onErrorResponse(VolleyError error) {
                        error.printStackTrace();
                    }
                });

        requestQueue.add(request);
    }
}
```

Finally, call the getJokesData() method in your onCreate() method.

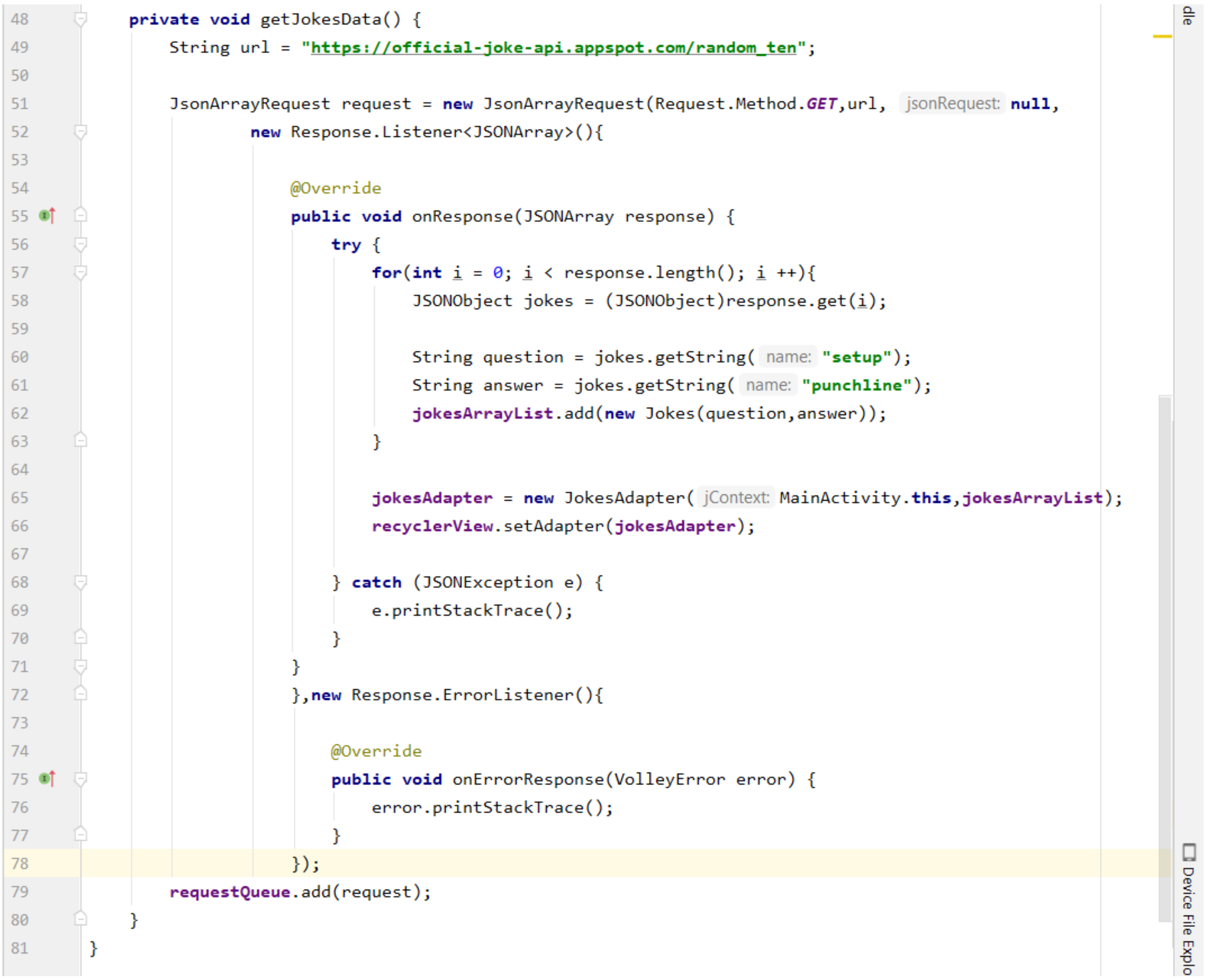


```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        recyclerView = findViewById(R.id.recyclerView);
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager( context: this));

        jokesArrayList = new ArrayList<>();

        requestQueue = Volley.newRequestQueue( context: this);

        getJokesData();
    }
```

Your final MainActivity.java class will look as follows:



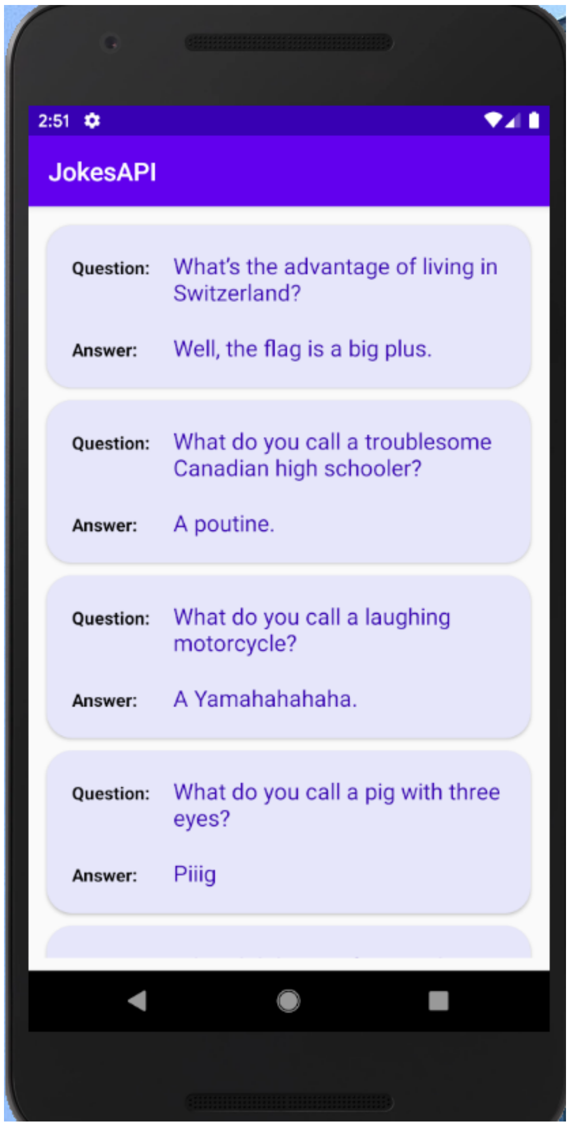
```
package com.example.shivangi.jokesapi;

import ...

public class MainActivity extends AppCompatActivity {

    private RecyclerView recyclerView;
    private JokesAdapter jokesAdapter;
    private ArrayList<Jokes> jokesArrayList;
    private RequestQueue requestQueue;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        recyclerView = findViewById(R.id.recyclerView);
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager( context: this));

        jokesArrayList = new ArrayList<>();

        requestQueue = Volley.newRequestQueue( context: this);

        getJokesData();
    }

```



```
    private void getJokesData() {
        String url = "https://official-joke-api.appspot.com/random_ten";

        JsonArrayRequest request = new JsonArrayRequest(Request.Method.GET,url, jsonRequest: null,
                new Response.Listener<JSONArray>(){

                    @Override
                    public void onResponse(JSONArray response) {
                        try {
                            for(int i = 0; i < response.length(); i ++){
                                JSONObject jokes = (JSONObject)response.get(i);

                                String question = jokes.getString( name: "setup");
                                String answer = jokes.getString( name: "punchline");
                                jokesArrayList.add(new Jokes(question,answer));
                            }

                            jokesAdapter = new JokesAdapter( jContext: MainActivity.this,jokesArrayList);
                            recyclerView.setAdapter(jokesAdapter);

                        } catch (JSONException e) {
                            e.printStackTrace();
                        }
                    }
                },new Response.ErrorListener(){

                    @Override
                    public void onErrorResponse(VolleyError error) {
                        error.printStackTrace();
                    }
                });
        requestQueue.add(request);
    }
}
```

Shivangi Patel

Run the app. The following are screenshots of how the app will be presented.


2:51

JokesAPI

Question: What's the advantage of living in Switzerland?

Answer: Well, the flag is a big plus.

Question: What do you call a troublesome Canadian high schooler?

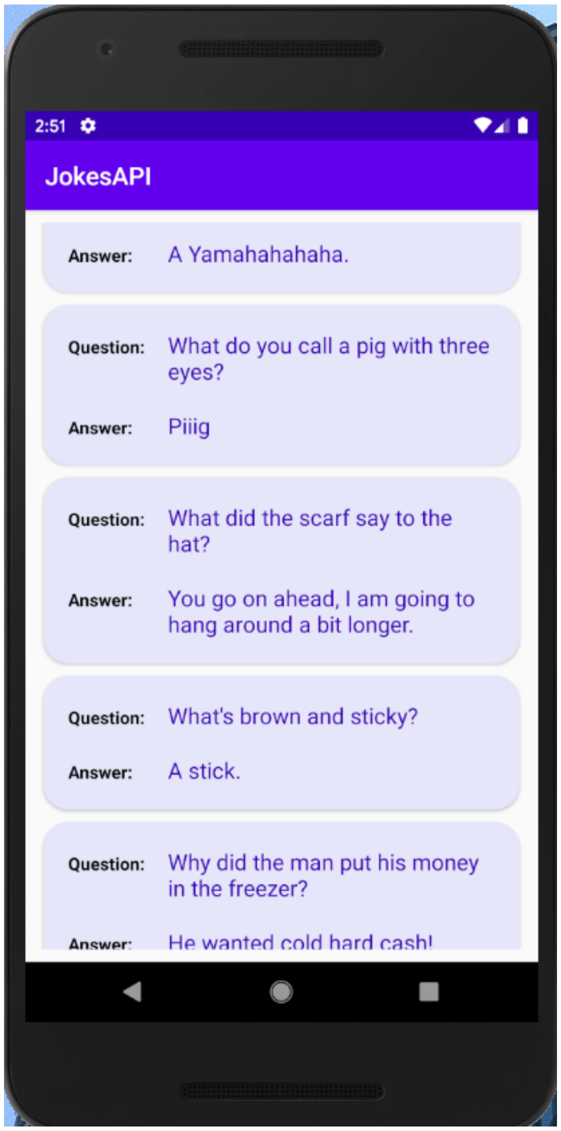
Answer: A poutine.

Question: What do you call a laughing motorcycle?

Answer: A Yamahahahaha.

Question: What do you call a pig with three eyes?

Answer: Piiig


2:51

JokesAPI

Answer: A Yamahahahaha.

Question: What do you call a pig with three eyes?

Answer: Piiig

Question: What did the scarf say to the hat?

Answer: You go on ahead, I am going to hang around a bit longer.

Question: What's brown and sticky?

Answer: A stick.

Question: Why did the man put his money in the freezer?

Answer: He wanted cold hard cash!

RESOURCES

https://developer.android.com/training/volley

https://codelabs.developers.google.com/codelabs/android-training-create-recycler-view/index.html?index=..%2F..android-training#3

https://www.youtube.com/watch?v=y2xtLqP8dSQ

https://developer.android.com/training/volley/request

https://github.com/public-apis/public-apis#art--design