Lesson 06 Demo 02

Implementing HTTP Client in Java

Objective: Implement the use of HTTP Client in Java 11 to efficiently send HTTP requests and handle responses
Tools Required: Eclipse IDE

Prerequisites: None

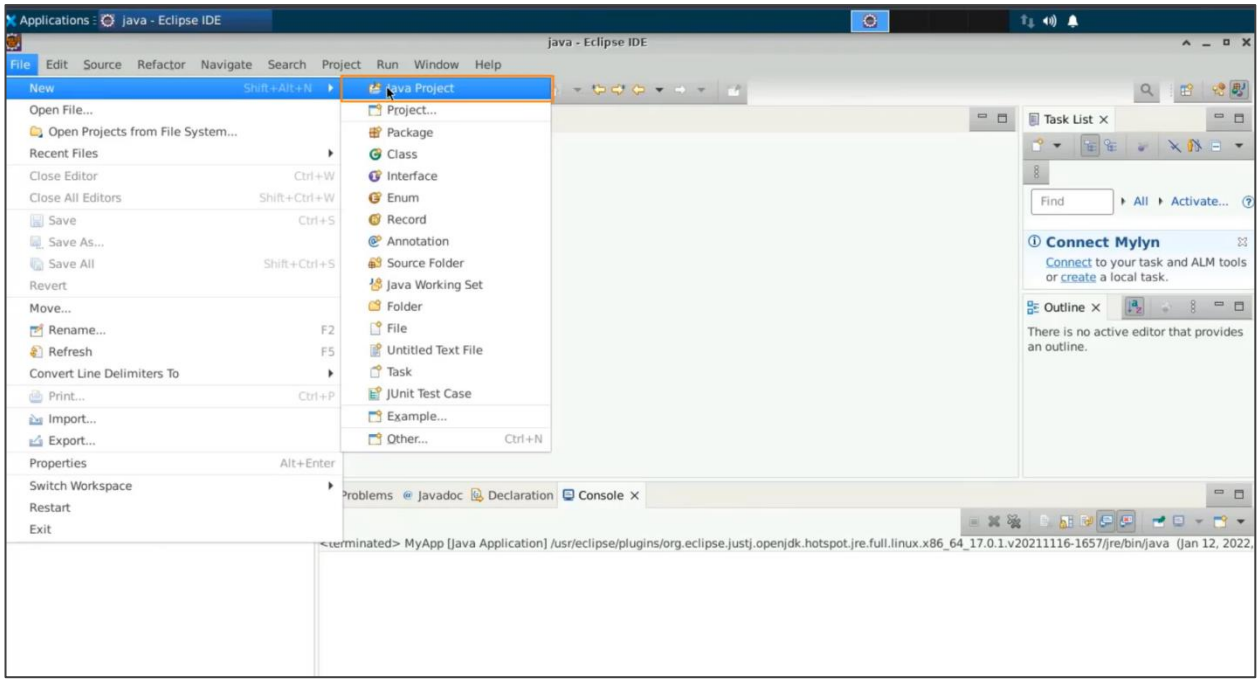
Steps to be followed:

1. Open the Eclipse IDE and create a new Java project
2. Open a web browser and search for newsapi.org
3. Log in to your account to get an API key
4. Hit the URL to get the data in the form of a response
5. Record and concatenate the API key as a separate variable in the Eclipse IDE
6. Create HTTP client object, HTTP request object, and then the response objects
7. Execute the code

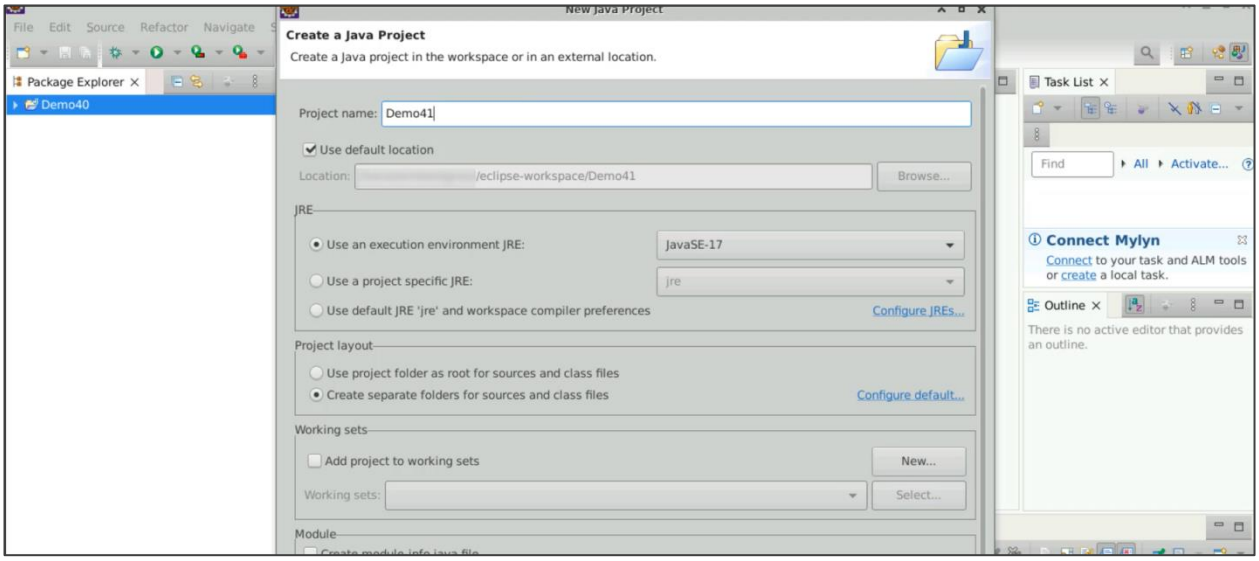
Step 1: Open the Eclipse IDE and create a new Java project

1.1 Open the **Eclipse IDE**

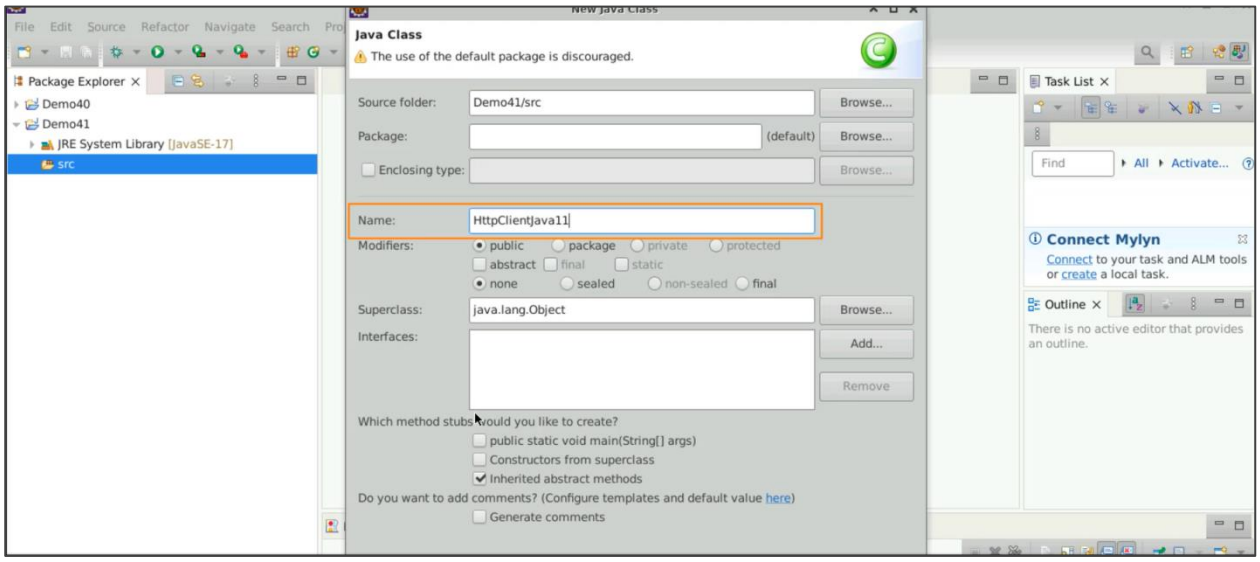
1.2 Select **File**, then **New,** and then **Java project**



1.3 Name the project Demo41, uncheck **Create a module-info.java** file, and press **Finish**

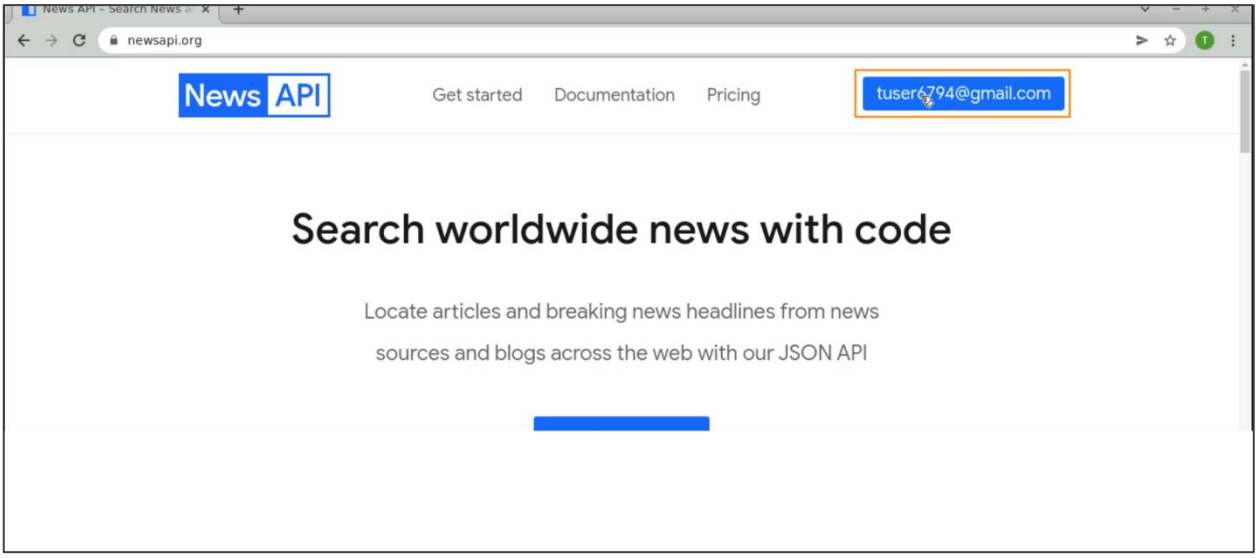


1.4 With **Demo41** selected, right-click on the **src**, and create a new class. Name this class **HttpClientJava11**, select the main method, and then select **Finish**



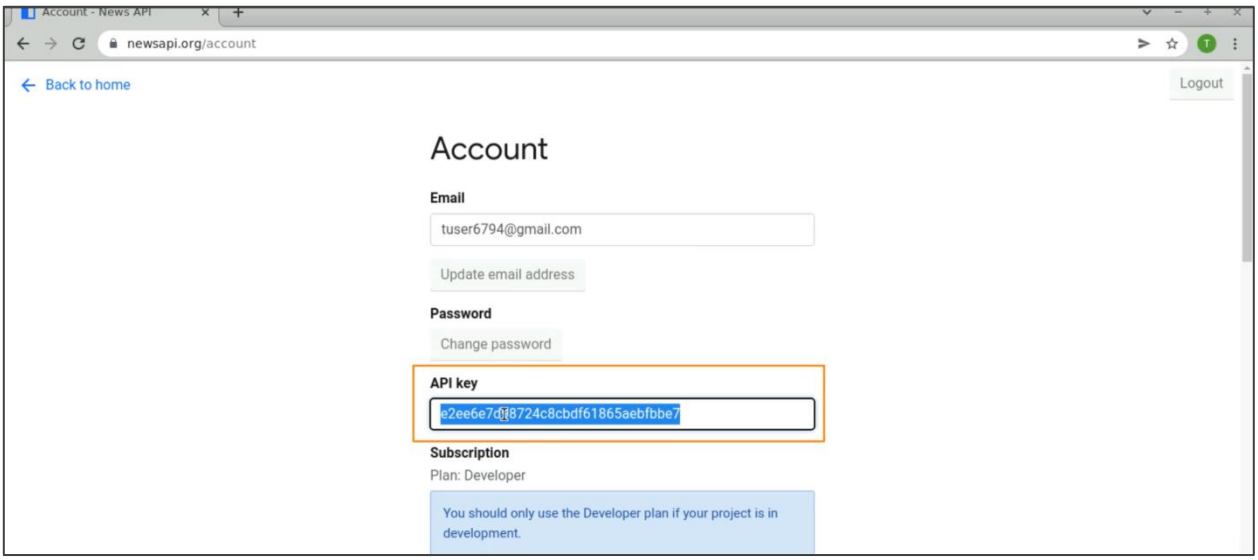
Step 2: Open a web browser and search for newsapi.org

2.1 Open a web browser and search for **newsapi.org**. As you can see, it is already logged in here

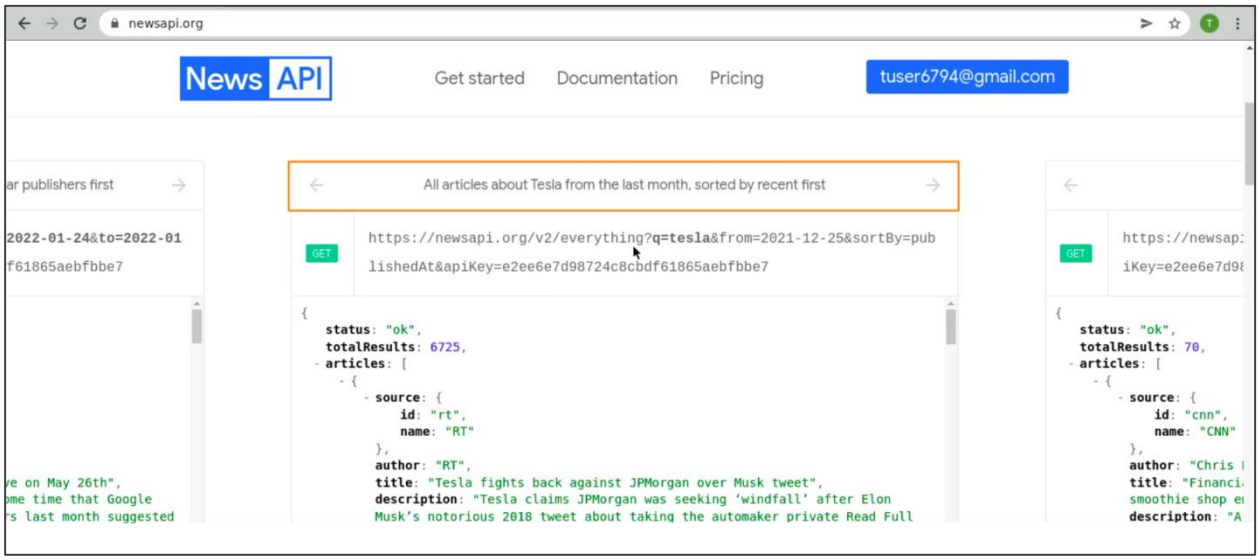


Step 3: Log in to your account to get an API key

3.1 You need to log in to your account to get an **API key**.

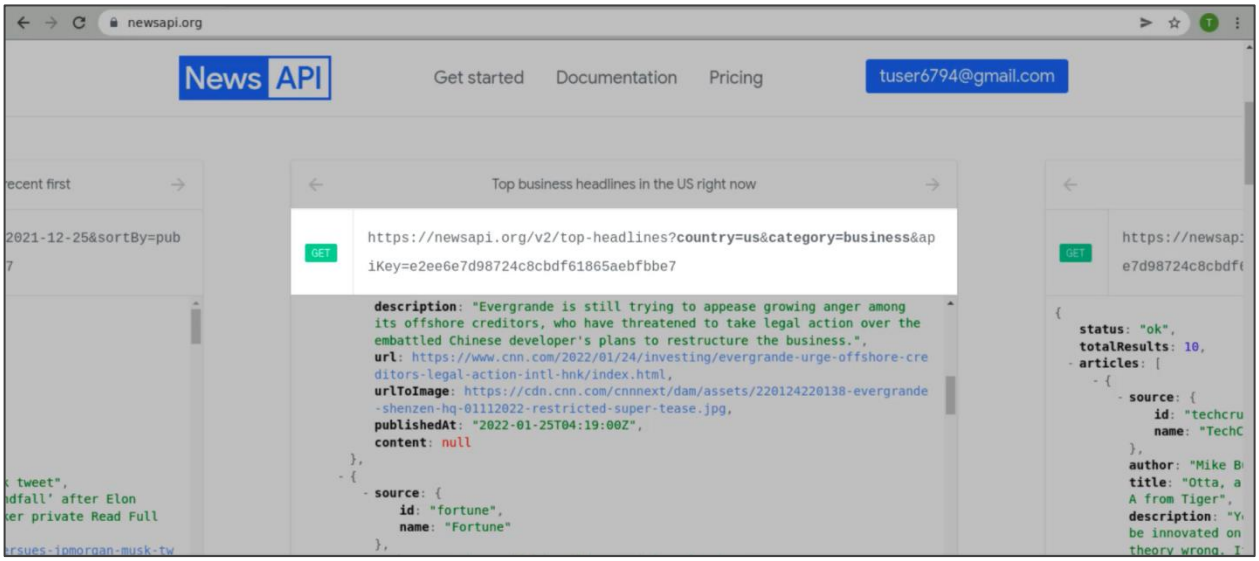


3.2 There are some standard URLs that can provide data, such as articles related to Tesla from the last month



Step 4: Hit the URL to get the data in the form of a response

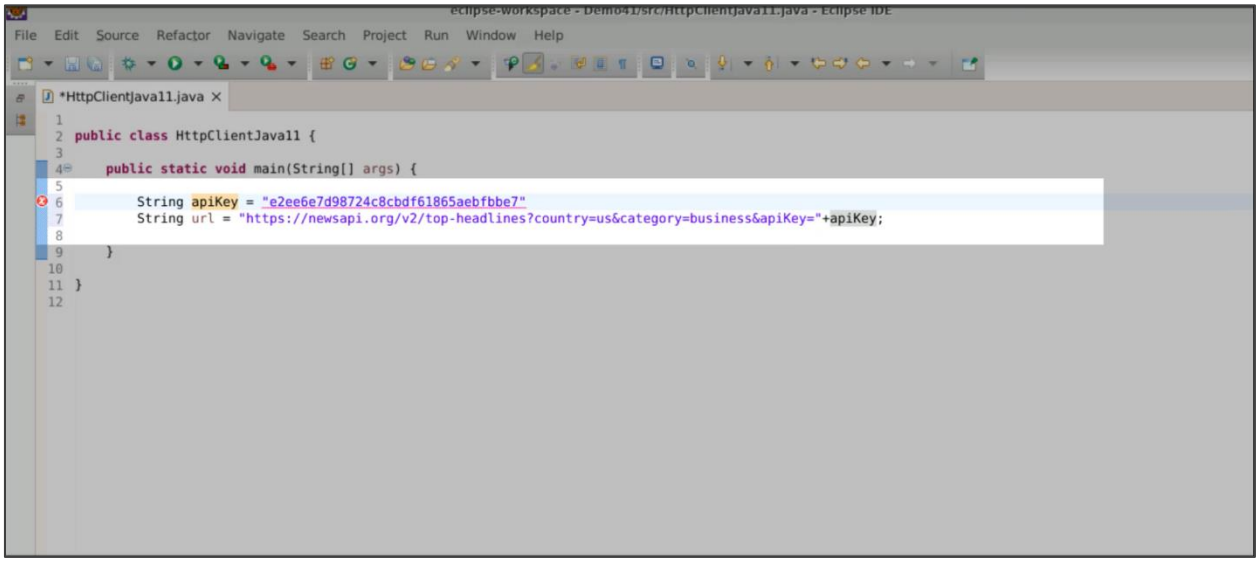
4.1 These URLs will have your API keys associated, and you can hit the URL to get data in the form of a response. First, copy this URL.



4.2 Paste it here. Now, when you hit this URL, you can see the JSON data comes as a response. The status is OK, and these are articles from the New York Times and so on.

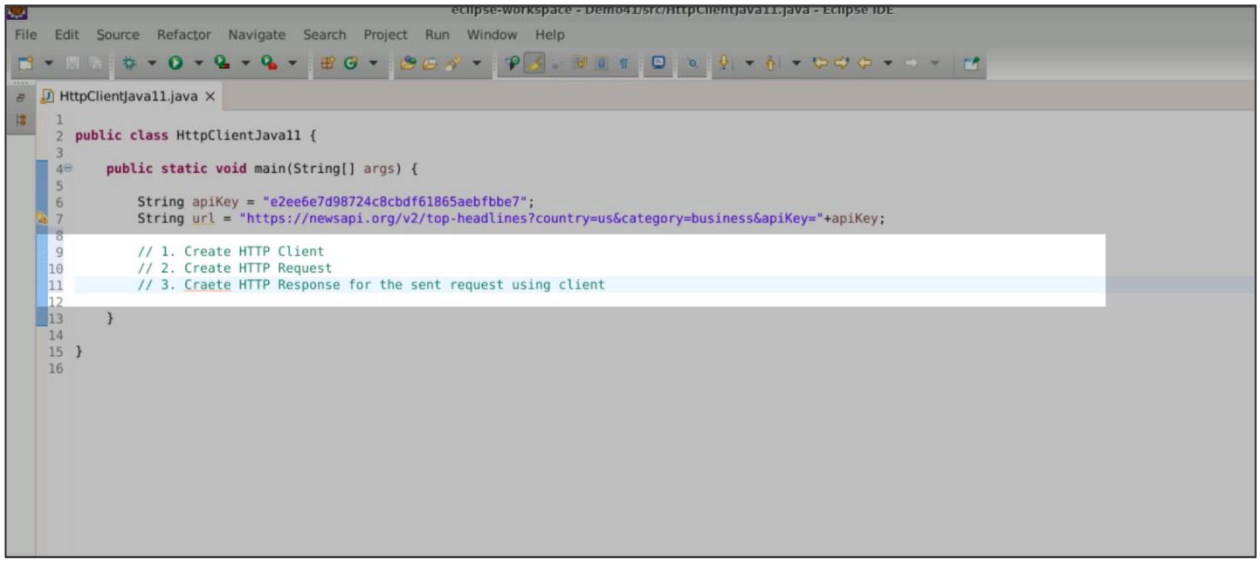
Step 5: Record and concatenate the API key as a separate variable in the Eclipse IDE

5.1 Let us create the URL endpoint. You have the news API URL coming in. As mentioned earlier, for the news API, you will have an API key provided by the website. Let us record this API key as a separate variable, and you can concatenate this API key here

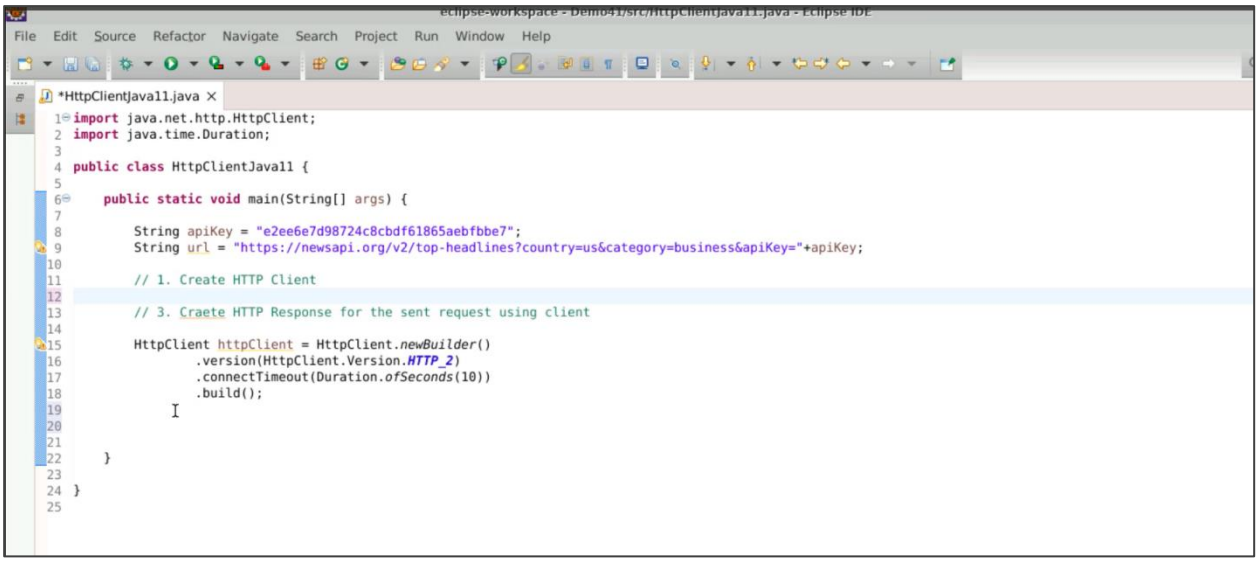


Step 6: Create HTTP client object, HTTP request object, and then the response objects

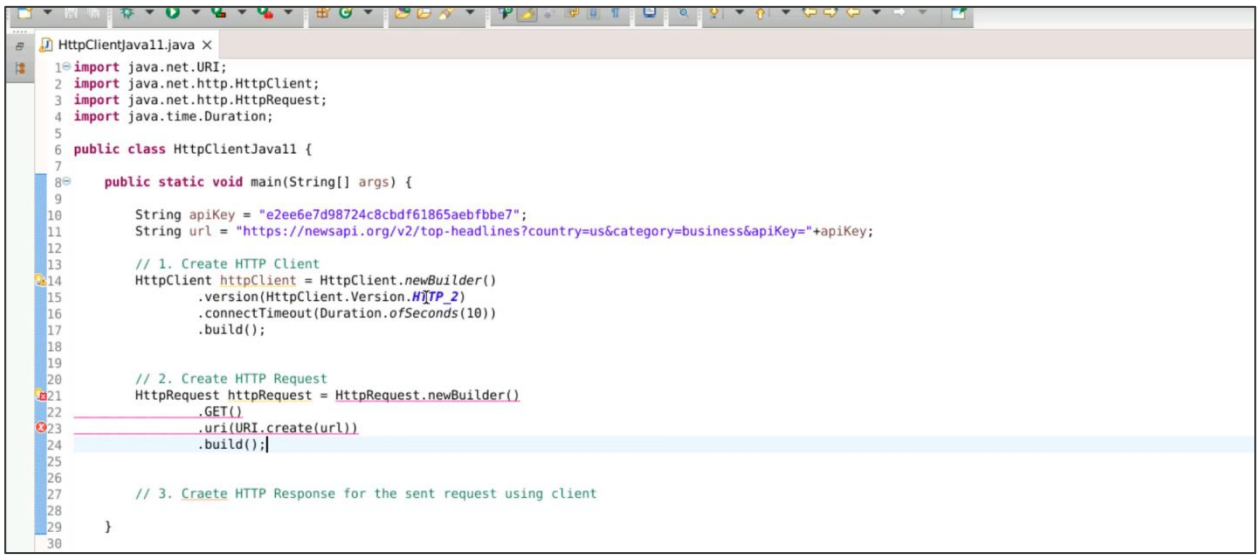
6.1 First, to send the request, you need to create an HTTP client object. Then, you will need to create an HTTP request object and the response objects. These are the three things you need to create in sequence. Let us follow these steps one by one:



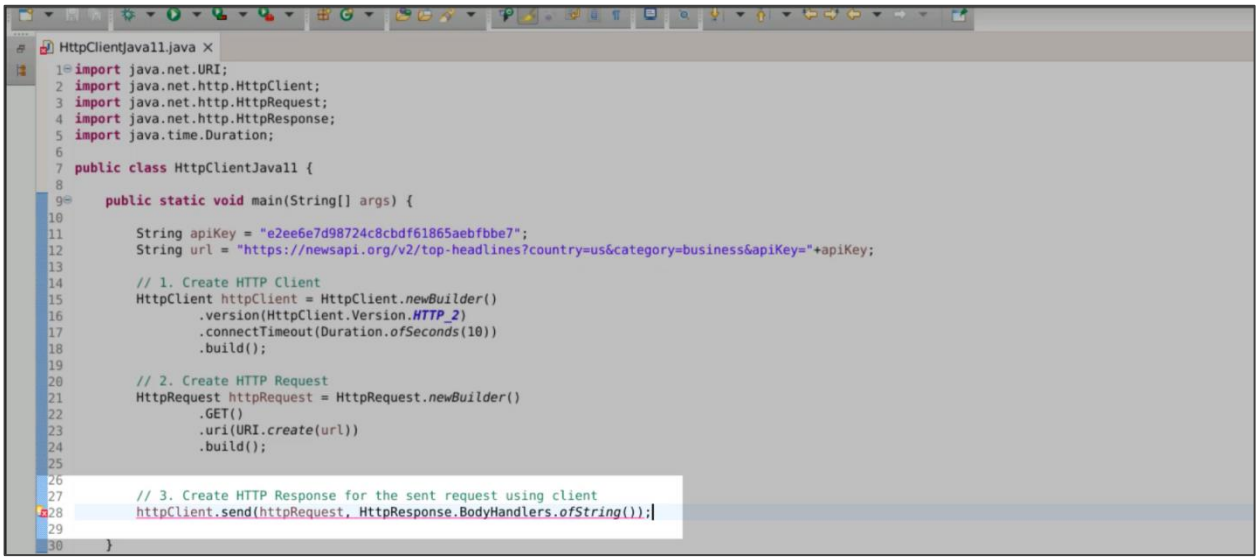
6.2 We can create an HTTP client using the java.net package's HTTP client API. It provides a builder for setting options like the version (e.g., HTTP/2) and timeout (e.g., 10 seconds) using the builder design pattern



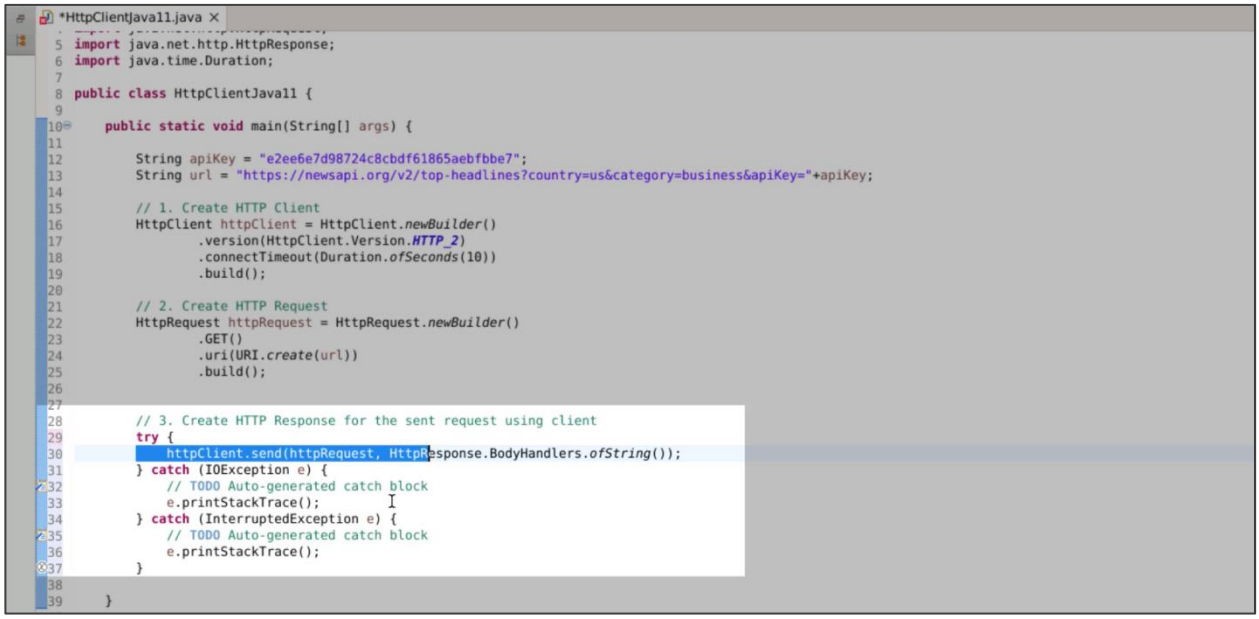
6.3 To send the request, we create an HTTP request object using the Builder API, specify the URI, and execute the **build** method. For a GET request without data, we use the **GET** method



6.4 To handle the response, we use **HTTPClient.send** with inputs for the body handler and the HTTP request. The **HTTPResponse.BodyHandlers** API assists in handling the response, particularly for strings, using the **ofString** method



6.5 Click on the error bulb. It shows that this code can throw an exception, and you should surround your code with a try-catch. Thus, surround with try-catch to handle exceptions



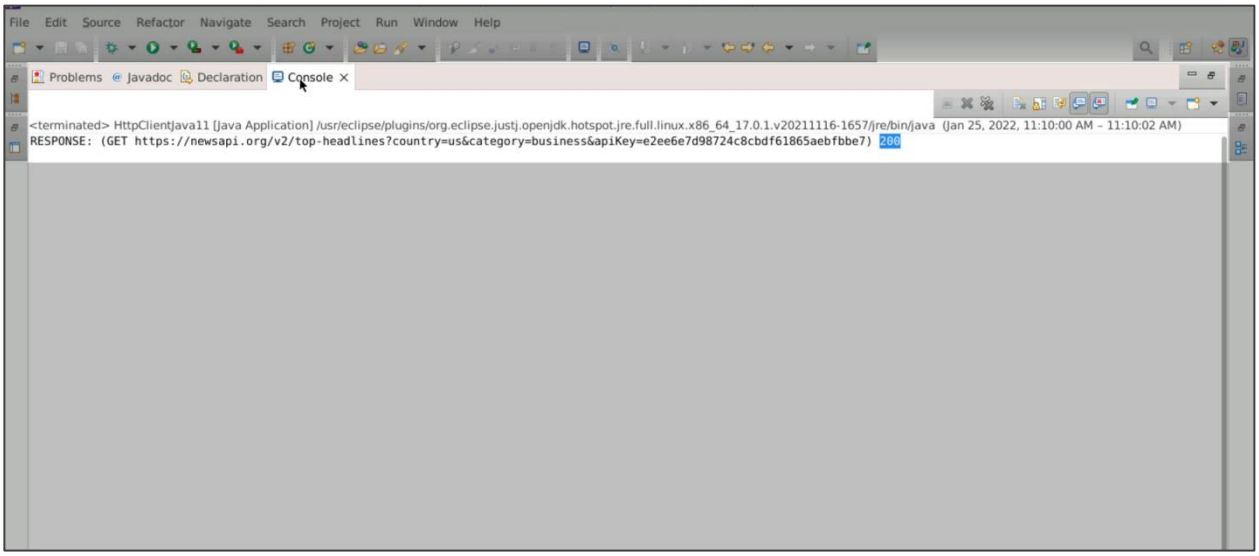
6.6 Create the left-hand side for this expression, which is your HTTP response. The HTTP response will be of type String. This means you are ready with the JSON, but JSON is a string. Thus, you have your HTTP response

```
public class HttpClientJava11 {

    public static void main(String[] args) {

        String apiKey = "e2ee6e7d98724c8cbdf61865aebfbbe7";
        String url = "https://newsapi.org/v2/top-headlines?country=us&category=business&apiKey="+apiKey;

        // 1. Create HTTP Client
        HttpClient httpClient = HttpClient.newBuilder()
                .version(HttpClient.Version.HTTP_2)
                .connectTimeout(Duration.ofSeconds(10))
                .build();

        // 2. Create HTTP Request
        HttpRequest httpRequest = HttpRequest.newBuilder()
                .GET()
                .uri(URI.create(url))
                .build();


        // 3. Create HTTP Response for the sent request using client
        try {
             HttpResponse<String> httpResponse = httpClient.send(httpRequest, HttpResponse.BodyHandlers.ofString());
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
```

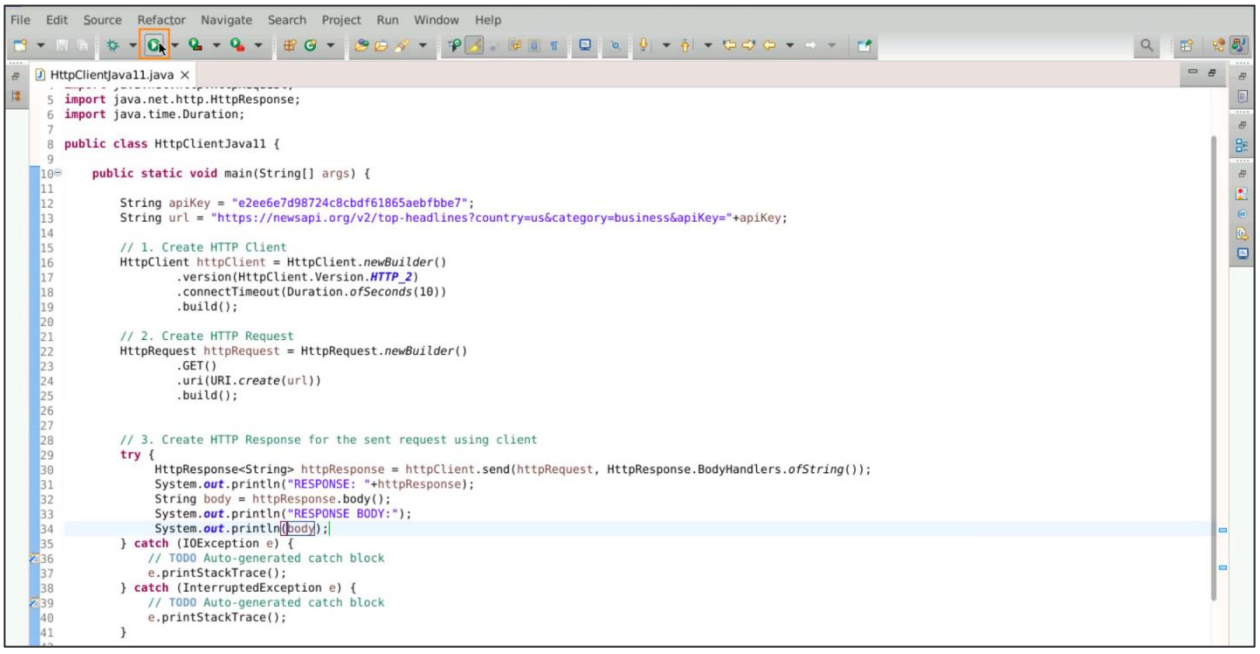
6.7 With the HTTP client, you can send a request and receive a response. We will print the response as it is, without any changes or formatting

```
File  Edit  Source  Refactor  Navigate  Search  Project  Run  Window  Help

HttpClientJava11.java

import java.net.http.HttpResponse;
import java.time.Duration;

public class HttpClientJava11 {

    public static void main(String[] args) {

        String apiKey = "e2ee6e7d98724c8cbdf61865aebfbbe7";
        String url = "https://newsapi.org/v2/top-headlines?country=us&category=business&apiKey="+apiKey;

        // 1. Create HTTP Client
        HttpClient httpClient = HttpClient.newBuilder()
                .version(HttpClient.Version.HTTP_2)
                .connectTimeout(Duration.ofSeconds(10))
                .build();

        // 2. Create HTTP Request
        HttpRequest httpRequest = HttpRequest.newBuilder()
                .GET()
                .uri(URI.create(url))
                .build();


        // 3. Create HTTP Response for the sent request using client
        try {
            HttpResponse<String> httpResponse = httpClient.send(httpRequest, HttpResponse.BodyHandlers.ofString());
            System.out.println("RESPONSE: "+httpResponse);
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (InterruptedException e) {
```

6.8 Let us run this code. As you can see, there is a response for URL get and you can note that the response code is 200, which means OK and that it's a successful request with the successful response

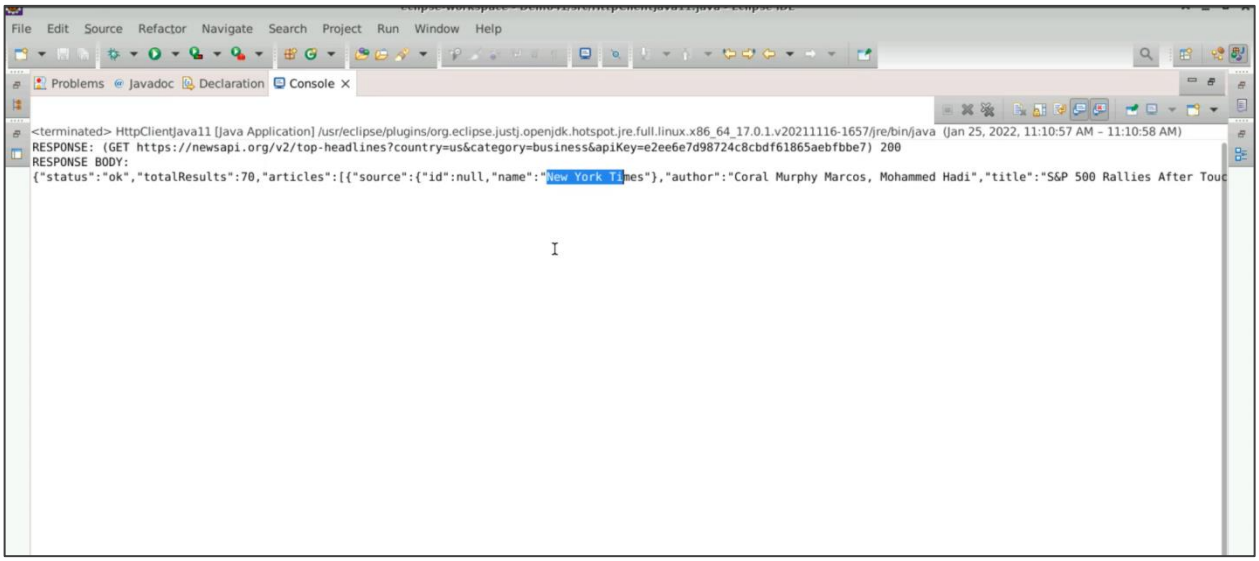


6.9 Get the contents of the response, which is like a body. Write HTTP **response.body** and convert it to a string if you wish. However, when you call **response.body**, it is already a string. Therefore, print the body



Step 7: Execute the code

7.1 Next, run the final code. As you can see, after the response, you have the response body which says the status is OK, and there are 70 articles found. This is the same content you see when you hit this URL in the browser



By following these steps, you have successfully implemented the use of HTTP Client in Java 11 to efficiently send HTTP requests and handle responses.