CASE STUDY - SELENIUM

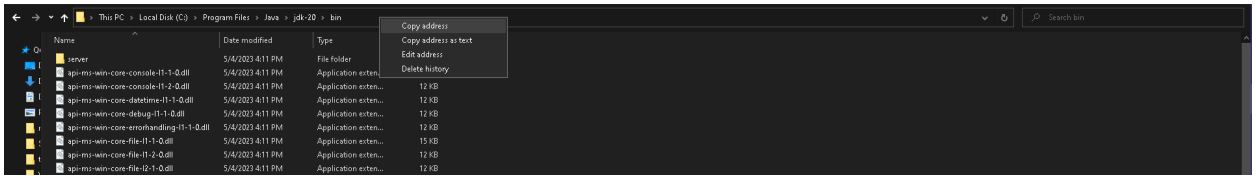
You work as a Devops Engineer in Ventura Soft Pvt Ltd. Recently a new product has been released by the company which demands creating test cases from scratch. This product is a search engine.

You being new in the team have been asked to present a POC for your skills. Assume the search engine is bing.com, search on Bing for the term "intellipaat", and print the title of the search webpage in the console.

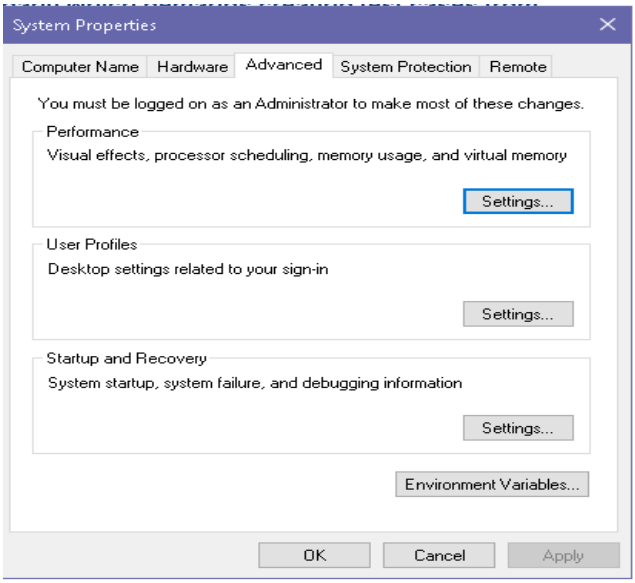
This test case should be packaged, and should run on AWS.

Let us install:
1. Java JDK installation from:
https://www.oracle.com/java/technologies/downloads/#jdk20-windows
For windows.

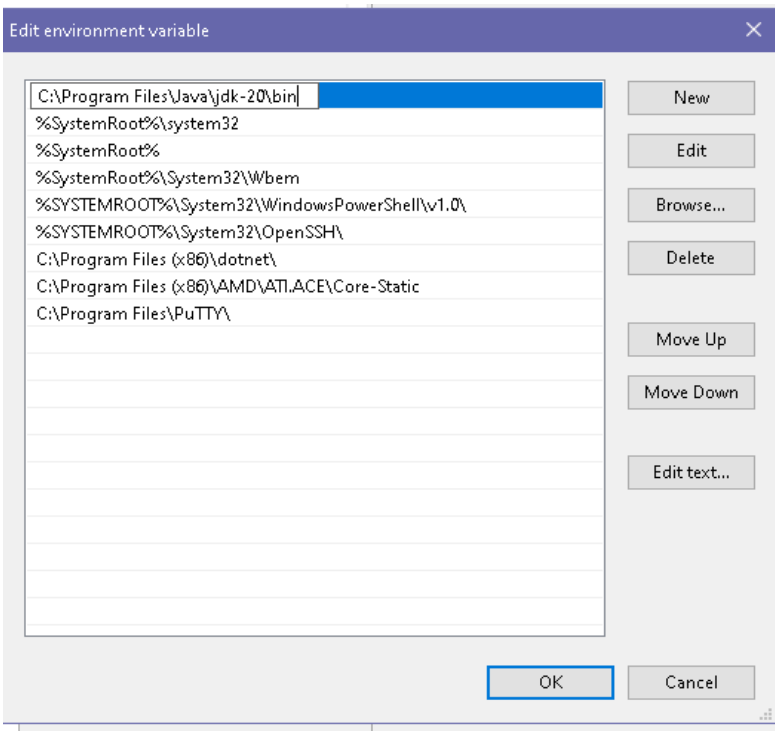
Follow the usual installation procedure.
Now, after installation, we need to add the environment variables to the path as follows.
Goto the installed folder and goto bin and copy the complete folder path.



Now, go to Control Panel > System & Security > System >Advanced System Settings



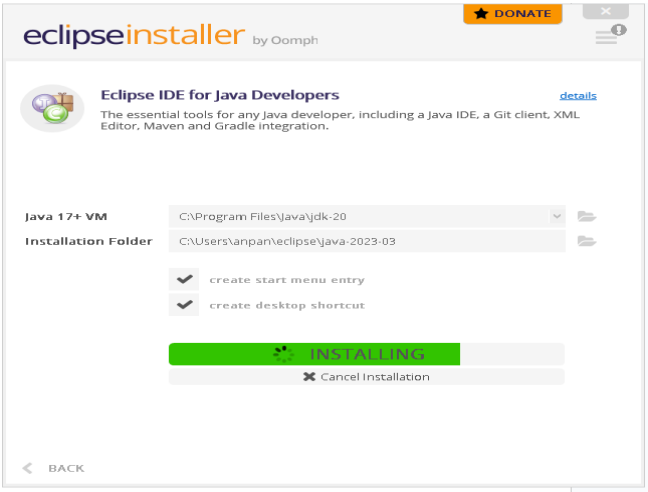
Click on Environment Variables.
Click on Path in the System Variables section, then paste the previously copied path.



Now open the command prompt and run the following command.

2 Eclipse installation.

Choosing IDE is completely a personal choice.
We will be using ECLIPSE IDE for JAVA Developers.

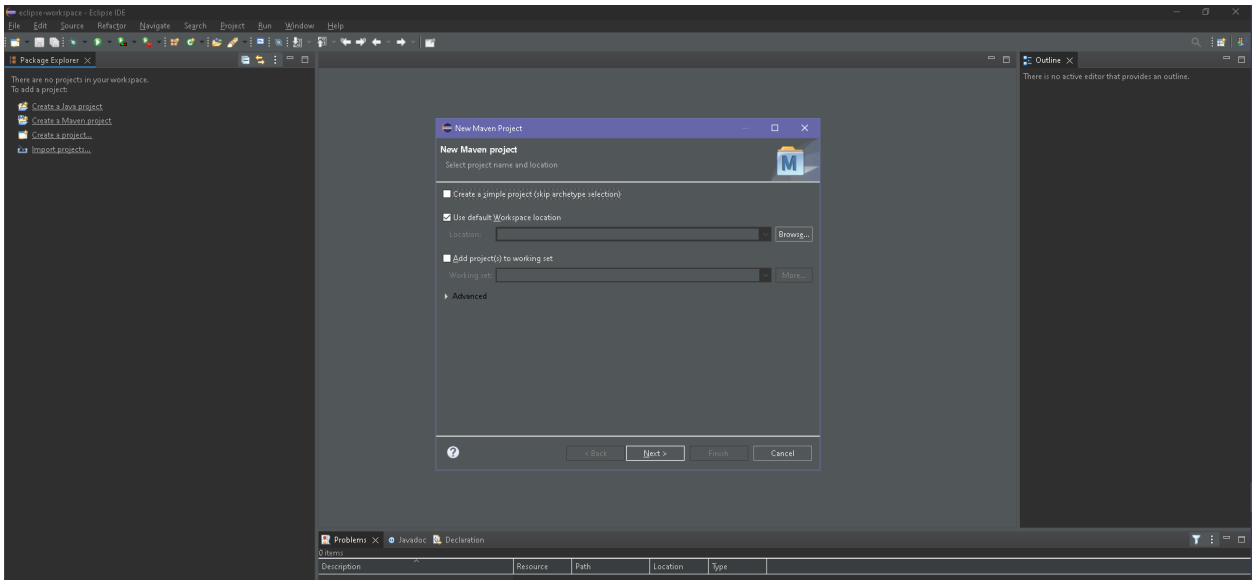


Now that we have successfully set up Java and Eclipse in our environment, we will be Performing a Selenium Test Case using Maven. Before that let us install chrome drivers which we will need later.

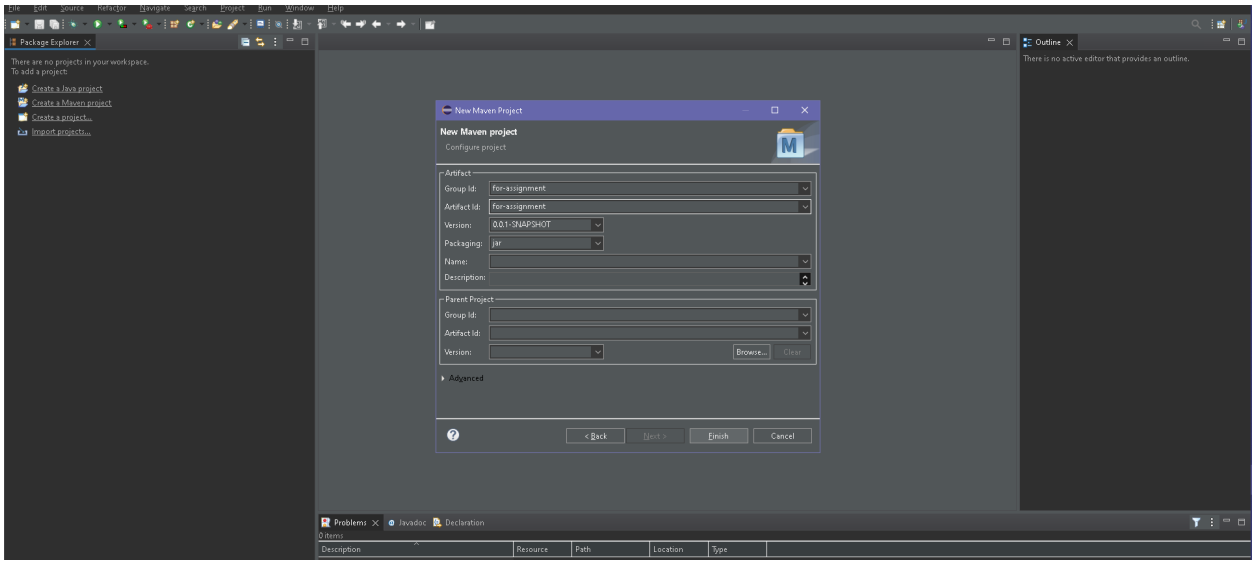
3. Chrome Driver Installation:
https://chromedriver.storage.googleapis.com/112.0.5615.49/chromedriver_win32.zip
Extract it and locate it for now.

4. Selenium Test Case.

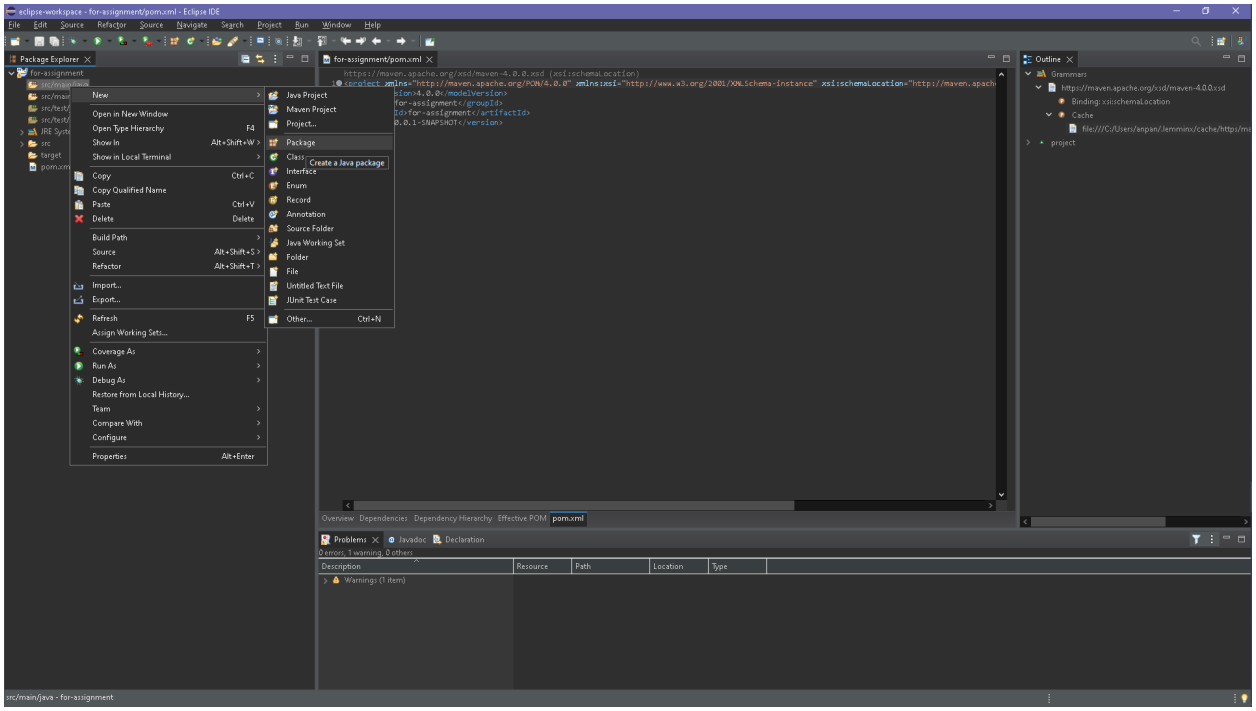
Open eclipse and start a new Maven Project:
Check the Create a Simple Project box. And click Next.

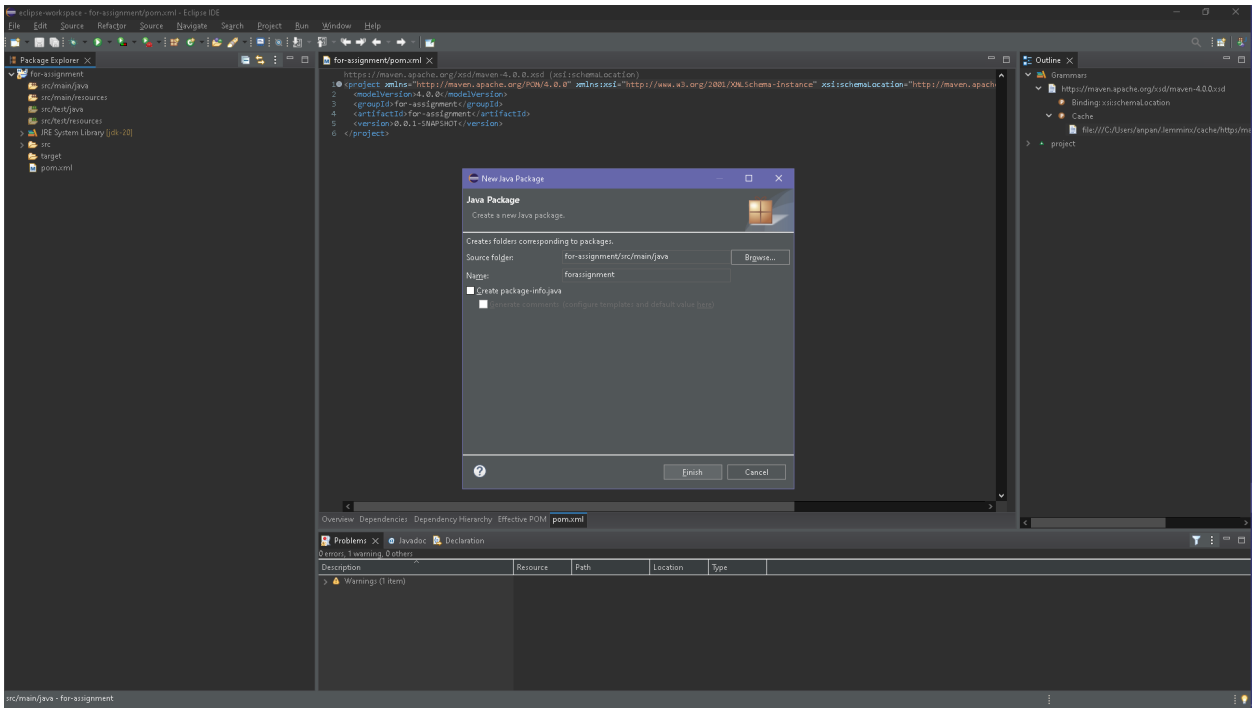


Enter the group id and artifact id of your choice.

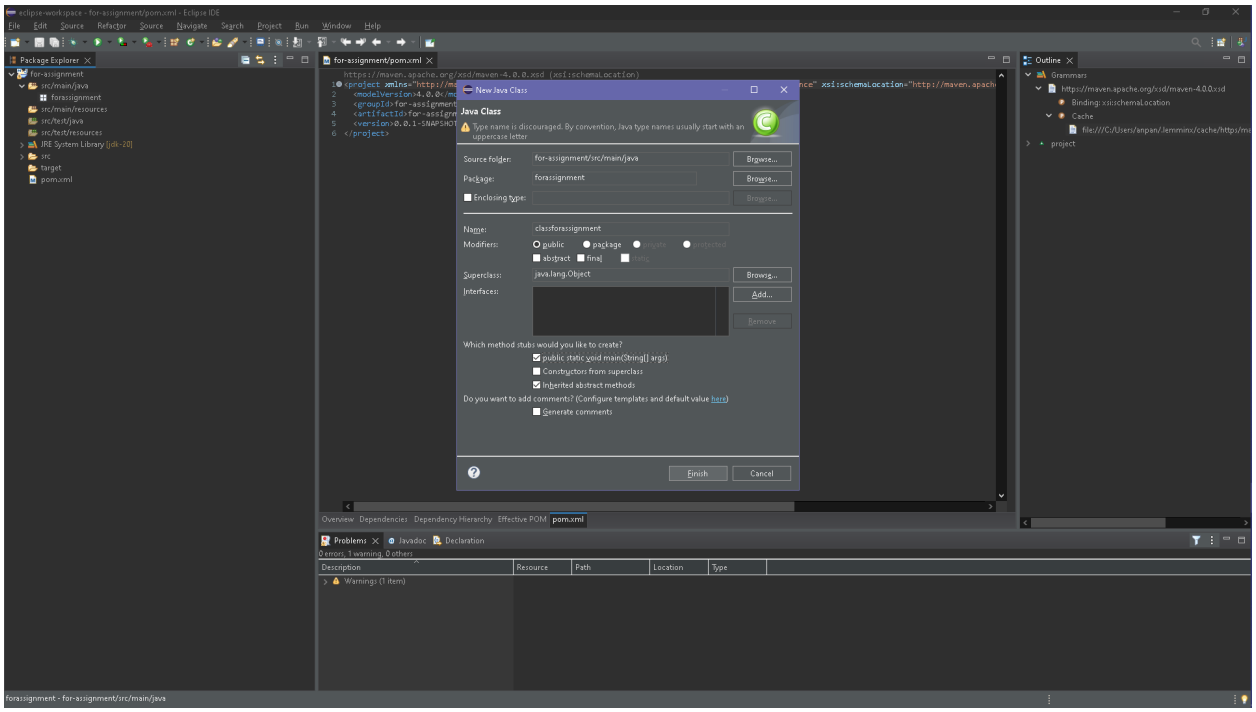


Right click on src/main/java and create a java package. Enter a Package Name.Then click on Finish.





Now create a Java Class by right clicking on package, name the class. Check the Public static void main(String[] args) box. And click on Finish.



What we have now is,

Class.java file:

```
package forassignment;
public class classforassignment {
	public static void main(String[] args) {
		// TODO Auto-generated method stub
	}
}
```

Pom.xml file:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>
 <groupId>for-assignment</groupId>
 <artifactId>for-assignment</artifactId>
 <version>0.0.1-SNAPSHOT</version>
</project>
```

Pom.xml for this project:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>
 <groupId>for-assignment</groupId>
 <artifactId>for-assignment</artifactId>
 <version>0.0.1-SNAPSHOT</version>
  <dependencies>
    <dependency>
      <groupId>org.seleniumhq.selenium</groupId>
      <artifactId>selenium-java</artifactId>
      <version>3.141.59</version>
    </dependency>
    <dependency>
      <groupId>io.github.bonigarcia</groupId>
      <artifactId>webdrivermanager</artifactId>
      <version>5.1.0</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.1</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-jar-plugin</artifactId>
        <version>3.2.0</version>
        <configuration>
          <archive>
            <manifest>
              <addClasspath>true</addClasspath>
              <mainClass>classforassignment.java</mainClass>
            </manifest>
          </archive>
        </configuration>
      </plugin>
    </plugins>
  </build>
```

```
</project>
```

Pom.xml explained:

This pom.xml file is a configuration file for Maven, a build automation tool used in Java projects. It specifies the dependencies of the project, plugins to use for the build process, and other configuration details.

The file starts with the project element, which specifies the project metadata such as the group ID, artifact ID, and version.

The dependencies section specifies the external libraries that the project depends on. In this case, the project depends on the Selenium Java library version 3.141.59 and the WebDriverManager library version 5.1.0 for managing the driver executable.

The build section specifies the build process configuration, such as the plugins to use. In this case, it includes the Maven Compiler Plugin with Java version 11 as the source and target level.

Overall, this pom.xml file is a necessary component for building the Selenium project and specifying all the required dependencies.

Classforassignment.java:

```
package forassignment;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class classforassignment {

	public static void main(String[] args) {
		// Set the path to the ChromeDriver executable
		System.setProperty("webdriver.chrome.driver",
"C:\\Users\\anpan\\Downloads\\chromedriver_win32/chromedriver.exe");

		// Create a new instance of the ChromeDriver
		WebDriver driver = new ChromeDriver();

		// Navigate to the Bing homepage
		driver.get("https://www.bing.com");

		// Enter the search term and submit the form
		driver.findElement(By.name("q")).sendKeys("intellipaat");
		driver.findElement(By.name("q")).submit();

		// Get the title of the search result page and print it to the console
		String title = driver.getTitle();
		System.out.println(title);

		// Close the browser
		driver.quit();

	}

}
```

Classforassignment.java explained:

This is a Java program that uses Selenium WebDriver to navigate to the Bing homepage, enter the search term "intellipaat", submit the search form, and print the title of the search results page to the console.

In the first line of the program, the import statement is used to import necessary classes from the Selenium library.

The program defines a classforassignment class that contains a main method.

Inside the main method, a new instance of the ChromeDriver is created using WebDriver driver = new ChromeDriver();.

The get method is called on the driver object to navigate to the Bing homepage at https://www.bing.com.

Then the sendKeys method is used to enter the search term "intellipaat" in the search bar. driver.findElement(By.name("q")).sendKeys("intellipaat");

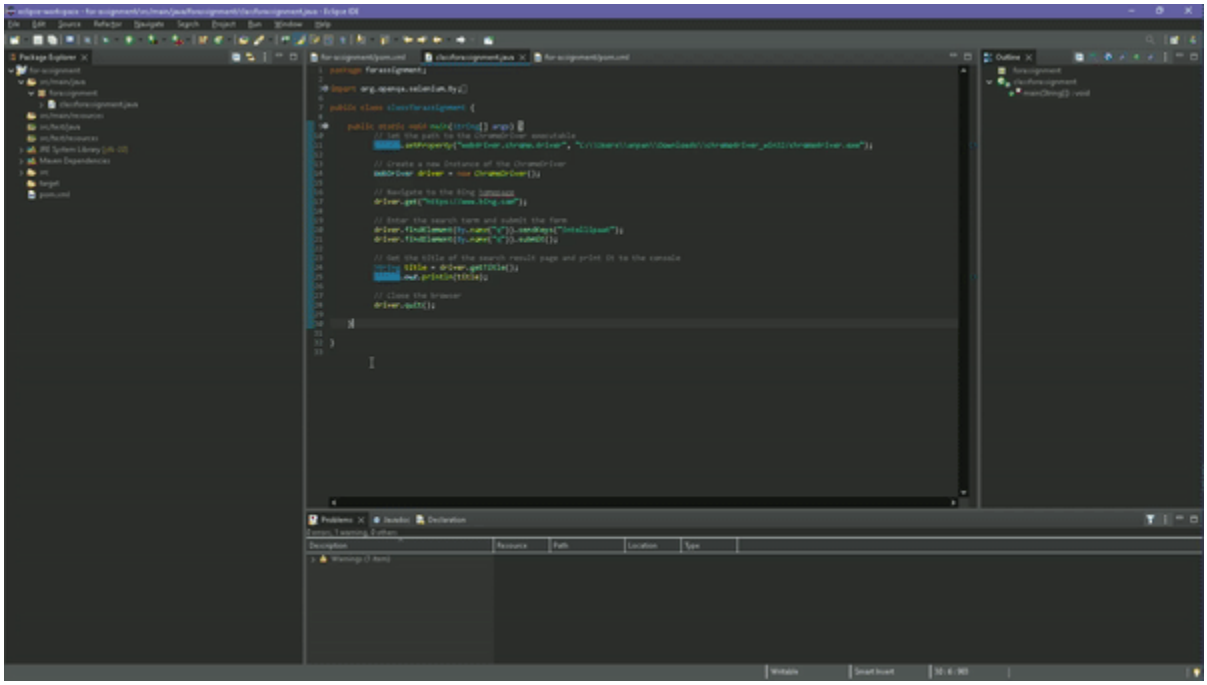
The submit method is called on the search bar to submit the search form. driver.findElement(By.name("q")).submit();

The getTitle method is used to get the title of the search result page and store it in the title variable. String title = driver.getTitle();

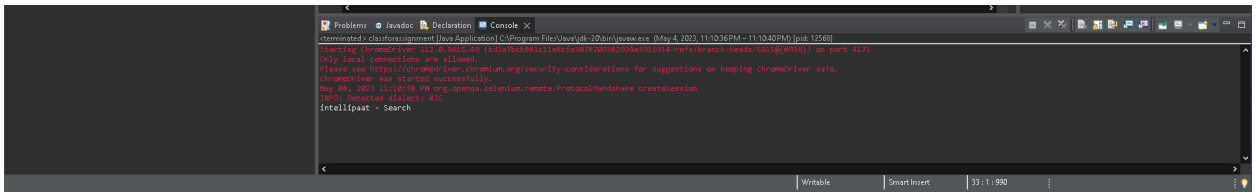
Finally, the println method is used to print the title to the console, and the quit method is called on the driver object to close the browser.

Note that the WebDriver and ChromeDriver classes should be imported at the top of the file using the import statements, and the ChromeDriver executable file path should be set in the system properties.

Output:

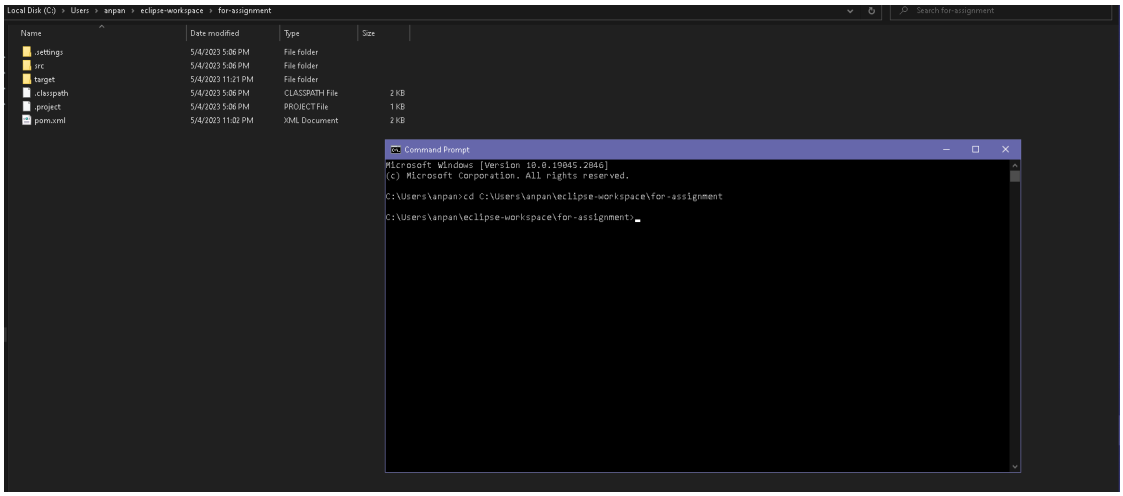


So the output searches on bing.com for the term "intellipaat", and prints the title of the search webpage in the console as required:



Let us package this test so that later it could be ran on AWS.

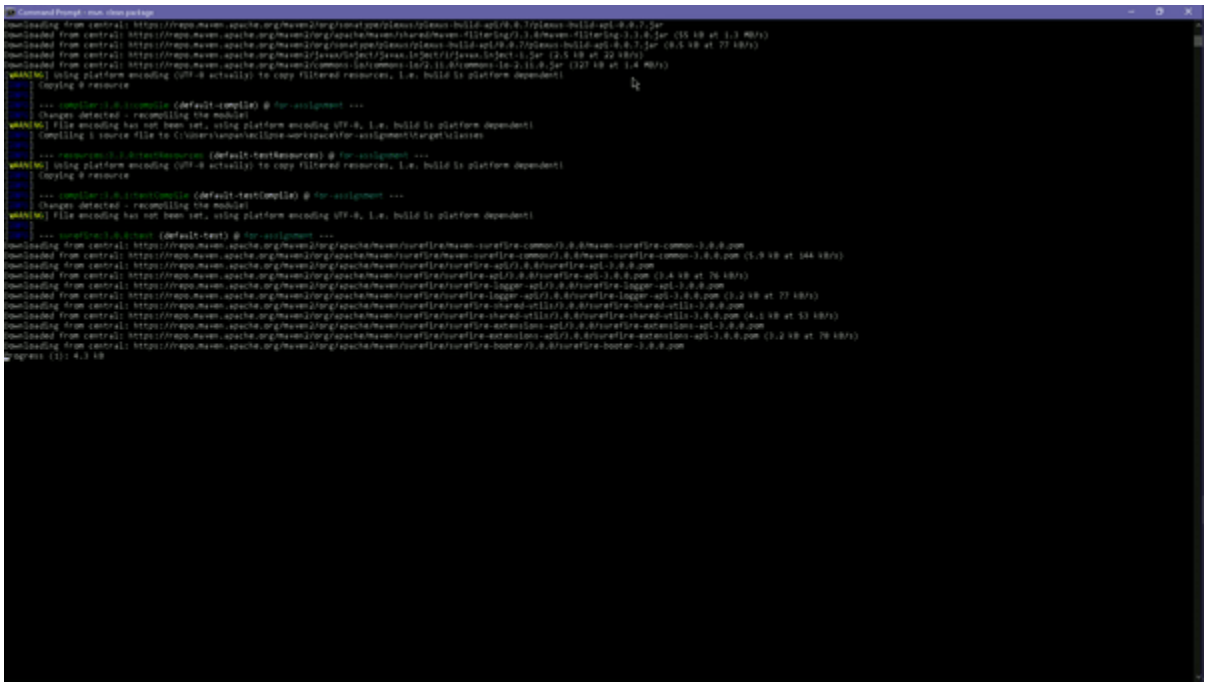
First cd to root directory of your project on command prompt:



Note: Like we did previously for java, install maven and specify bin path of maven to path of environment variables in your system.

Now run: maven clean package

This will create .jar package which we can run on AWS if needed.



To run the package on AWS you can:

Upload the JAR file to an Amazon S3 bucket:
- Open the Amazon S3 console and create a new bucket if you don't have one already. Click on the "Upload" button and select the JAR file you created in the previous step.
- Make sure to set the permission to "public" or "read" so that it can be accessed from the EC2 instance.

Launch an Amazon EC2 instance:
- Open the Amazon EC2 console and launch a new instance using an Amazon Machine Image (AMI) of your choice.
- Make sure to select a security group that allows inbound traffic on port 22 (for SSH access) and any other port your Selenium project may require.

Connect to the EC2 instance:
- Use SSH to connect to the EC2 instance using the public IP address or DNS name. Once you are connected, install Java and any other dependencies your project may require.

Download and run the JAR file:Use the wget command to download the JAR file from the S3 bucket to the EC2 instance.Run the JAR file using the command java -jar <jar_file_name>.jar.
That's it! Your Selenium project should now be running on the AWS EC2 instance.