**Selenium Framework**

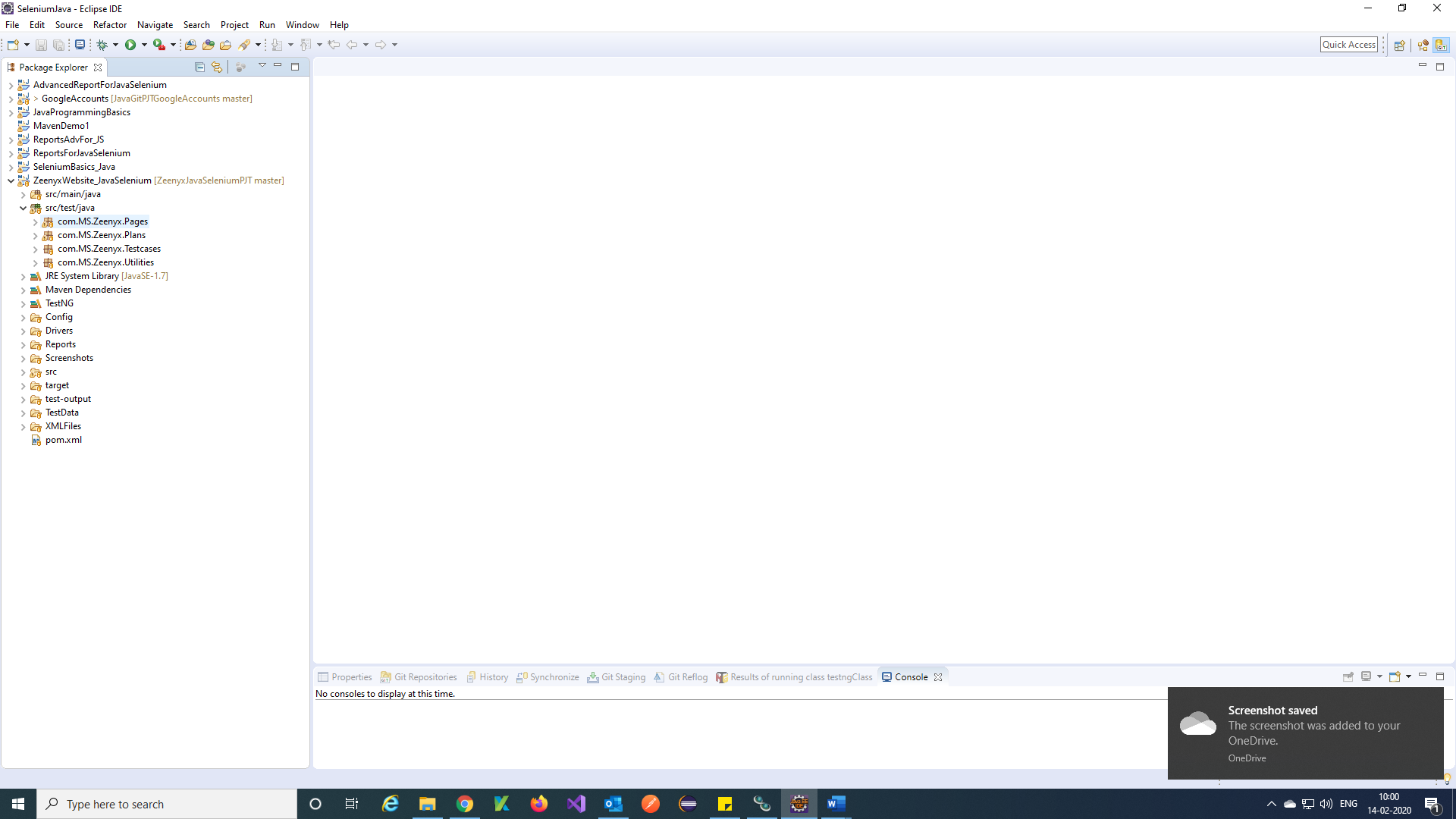
1. **POM (Page Object Model) or Pages**
2. **Utilities**
3. **Base class**
4. **Testcases**
5. **Reports**
6. **TestNg XML Files**

**Prerequisites:**

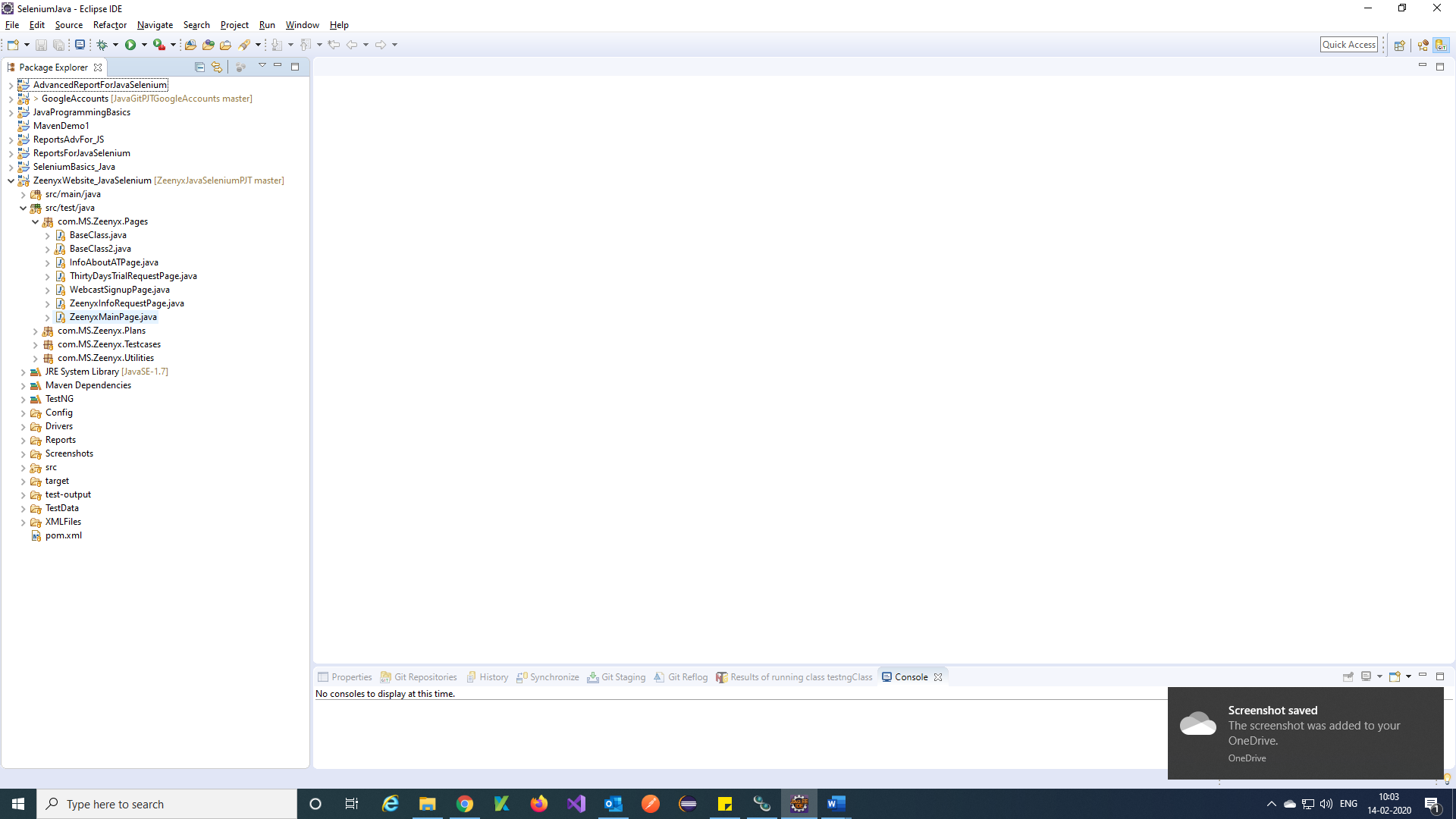
New Maven project

**Page Object Model (POM):**

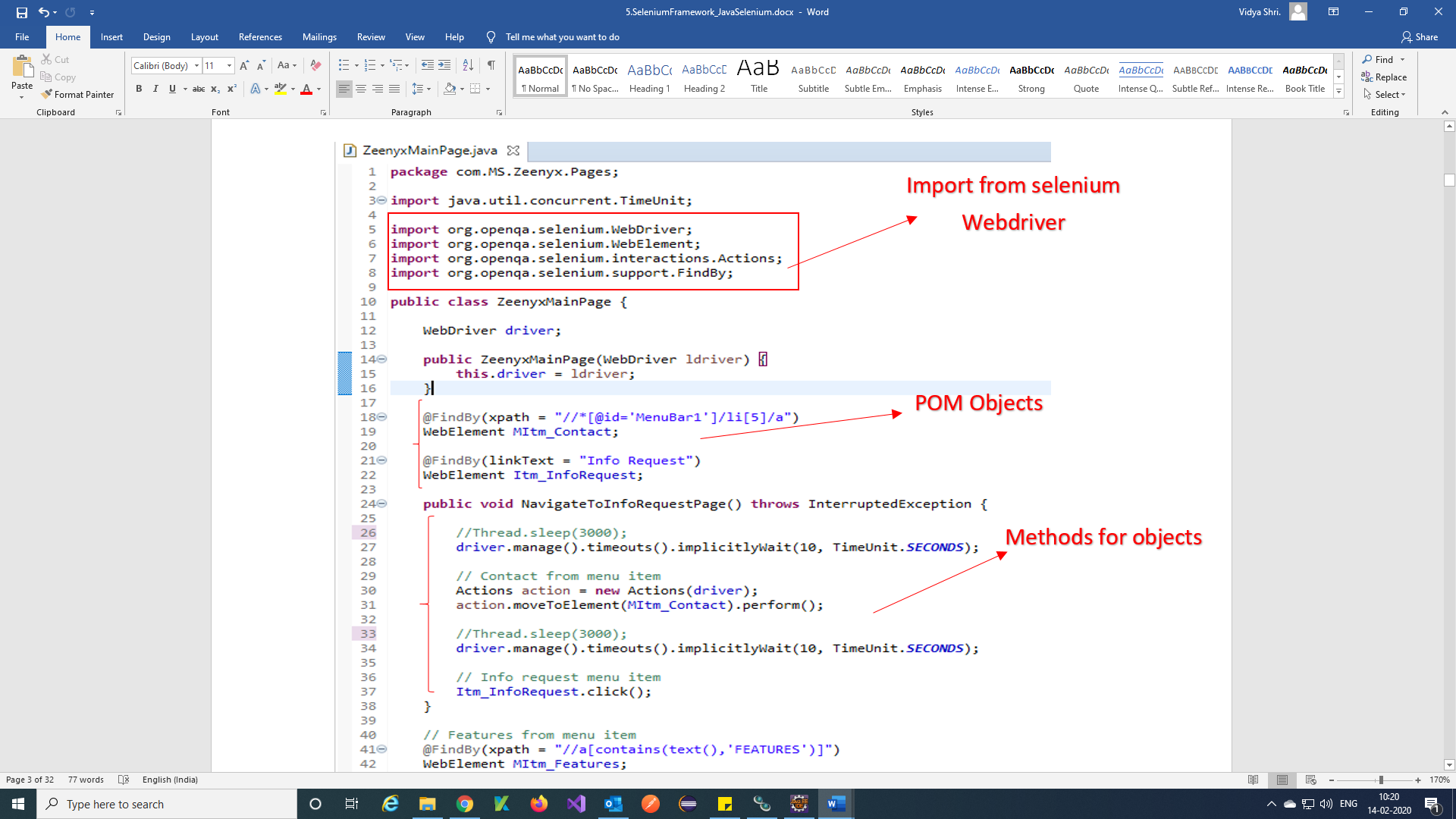
Step 1) Create a Package like Pages for example (com.MS.ZeenyxWebsite.Pages).



Step 2) Create a class for Zeenyx main page and identify the paths



Step 3) In the main page add all the requires objects and methods for the objects.



Note : For all remaining individual pages we are followed POM in this framework.

**Why Page Object Model?**

Starting an UI Automation in Selenium WebDriver is NOT a tough task. You just need to find elements, perform operations on it.

Consider this simple script to login into a website

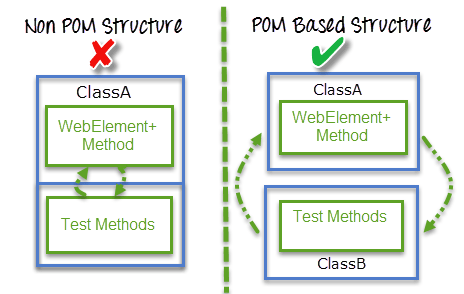
As you can observe, all we are doing is finding elements and filling values for those elements.

This is a small script. Script maintenance looks easy. But with time test suite will grow. As you add more and more lines to your code, things become tough.

The chief problem with script maintenance is that if 10 different scripts are using the same page element, with any change in that element, you need to change all 10 scripts. This is time consuming and error prone.

A better approach to script maintenance is to create a separate class file which would find web elements, fill them or verify them. This class can be reused in all the scripts using that element. In future, if there is a change in the web element, we need to make the change in just 1 class file and not 10 different scripts.

This approach is called **Page Object Model (POM)**. It helps make the code **more readable, maintainable**, and **reusable.**

[](https://www.guru99.com/images/AdvanceSelenium/071514_0722_PageObjectM2.png)

**Advantages of POM**

1. Page Object Pattern says operations and flows in the UI should be separated from verification. This concept makes our code cleaner and easy to understand.
2. The Second benefit is the object repository is independent of test cases, so we can use the same object repository for a different purpose with different tools. For example, we can integrate POM with TestNG/JUnit for functional[Testing](https://www.guru99.com/software-testing.html)and at the same time with JBehave/Cucumber for acceptance testing.
3. Code becomes less and optimized because of the reusable page methods in the POM classes.
4. Methods get more realistic names which can be easily mapped with the operation happening in UI. i.e. if after clicking on the button we land on the home page, the method name will be like ZeenyxMainPage().

**Utilities:**

Step 1) Create a package for Utilities for example com.MS.Zeenyx.Utilities

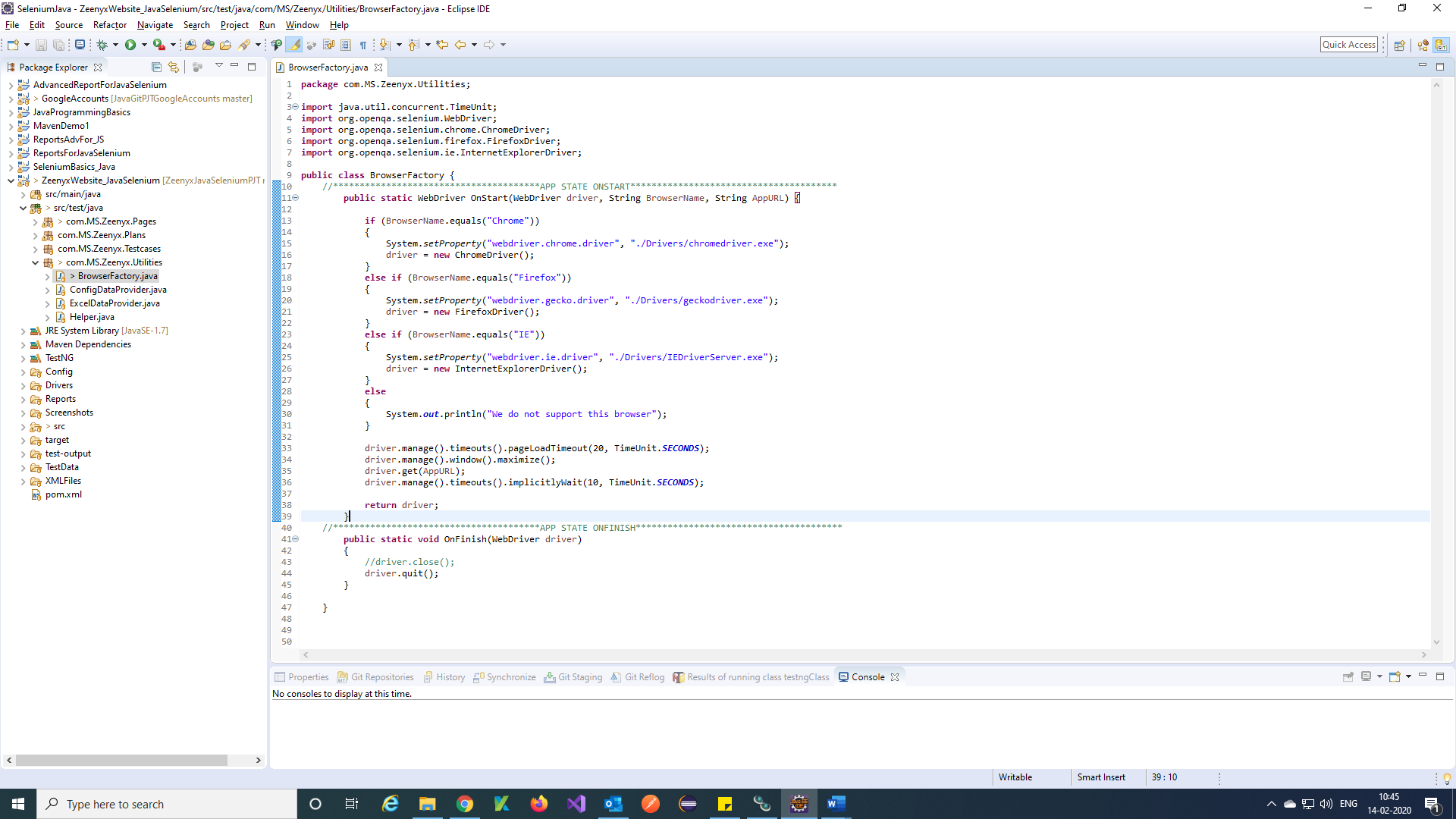
Utilities package contains the Browser Factory class, Helper class, Config data provider class and Excel data provider class are present.



**Browser Factory class:**

Step 1) With frameworks we know that maintaining and passing around a ***WebDriver***object across different tests is a delicate process. Also, the complexity increases when we have to maintain only one instance of a *WebDriver* throughout the test run. To overcome the problem on instantiation of *WebDriver* and maintaining the instance of browser we can use create a small class called browser factory.

A ***browser factory*** class with a basic *WebDriver* initialization logic will look something like this.



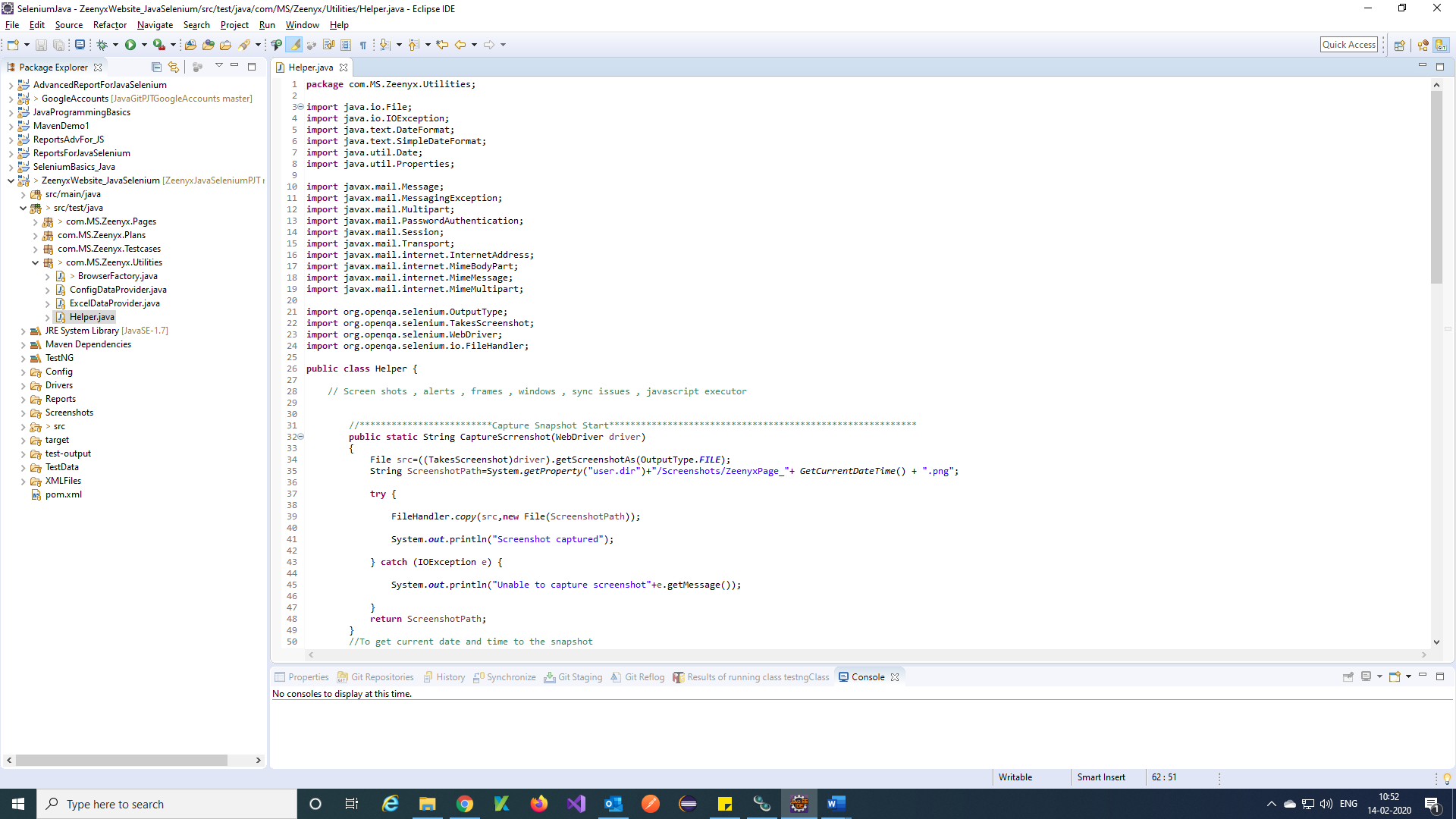
**Helper class:**

Step 1) In Helper class we can use any helping methods for your framework

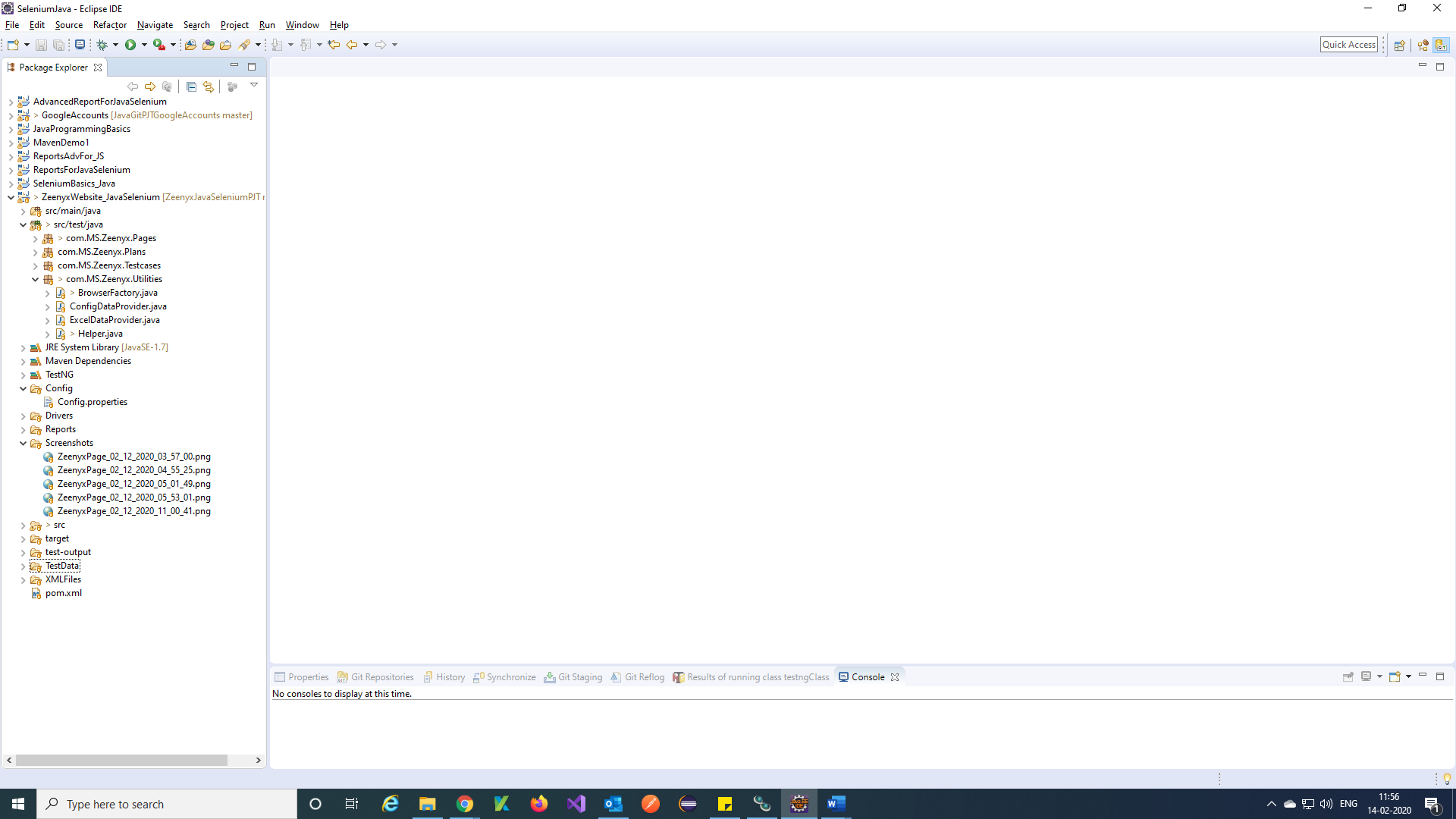
Here we are using **CaptureScreenshot** method, **GetCurrentDateTime** method, **SendMailandAttachReport** method.

**CaptureScreenshot method:**

Step 1) In below picture you can find captureScreenshot method.

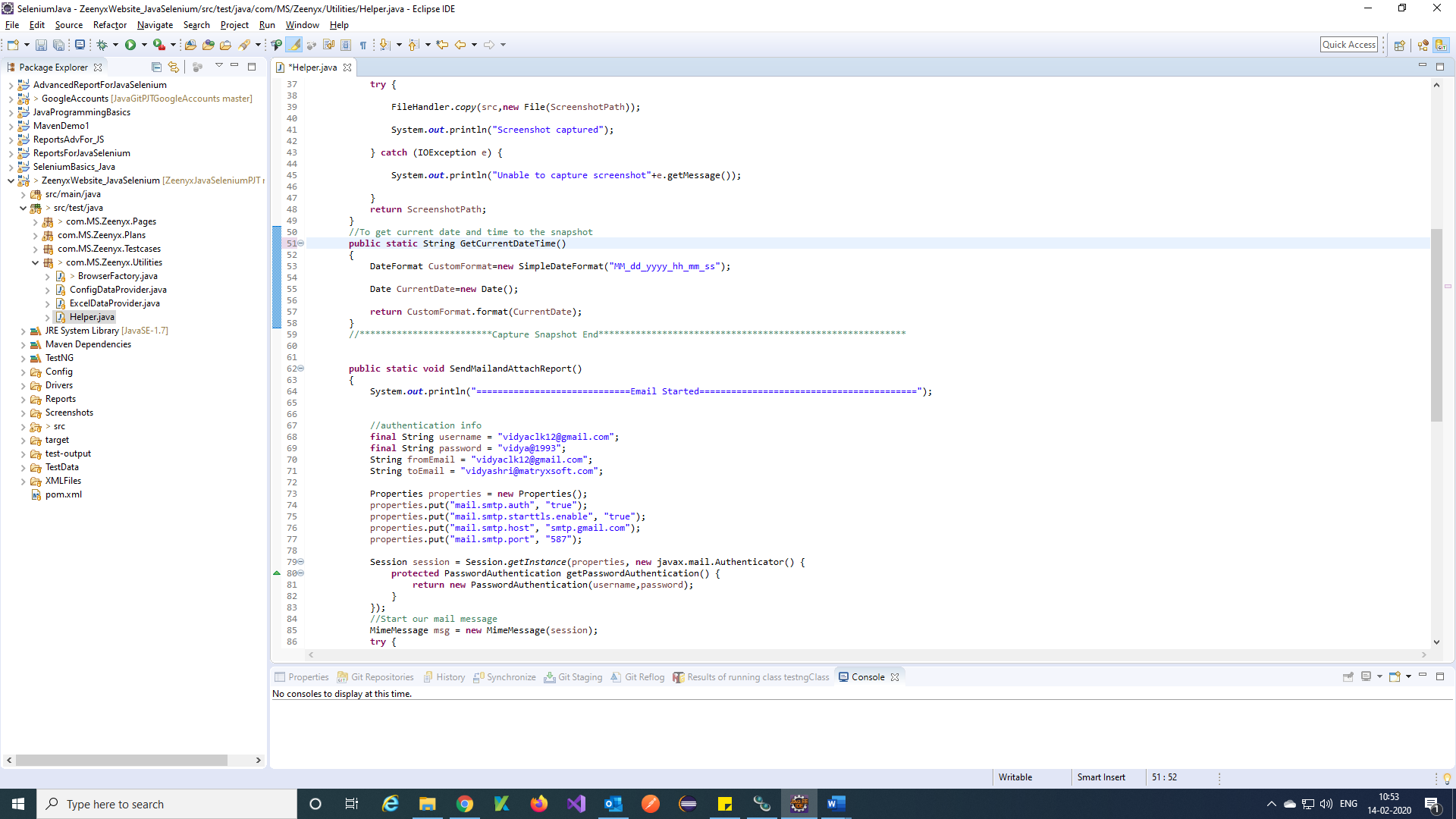


Note: Create a folder called Screenshots in framework and store all Snapshot in that folder.



**GetCurrentDateTime** method:

Step 1) In below picture you can find GetCurrentDateTime method.

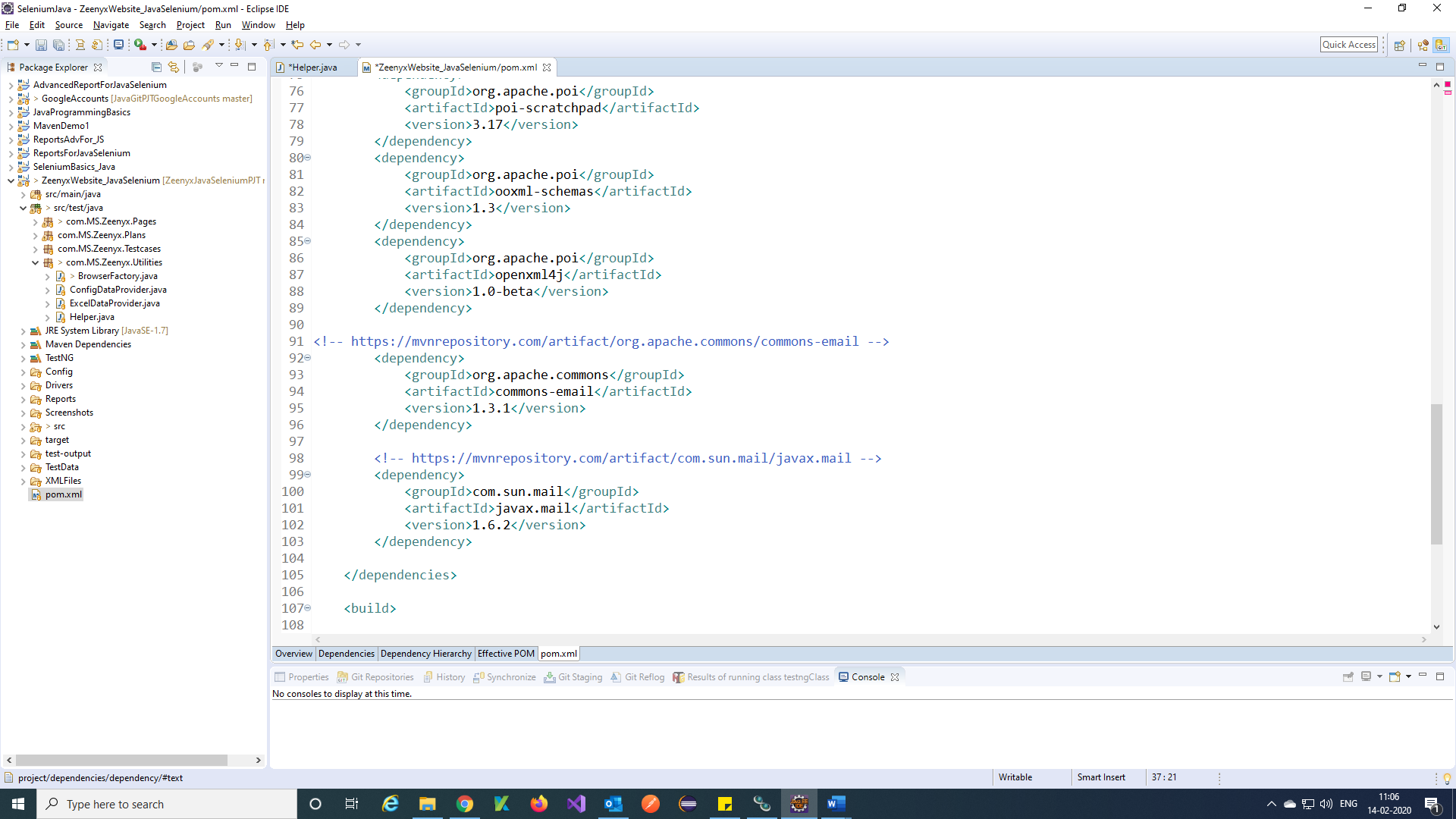


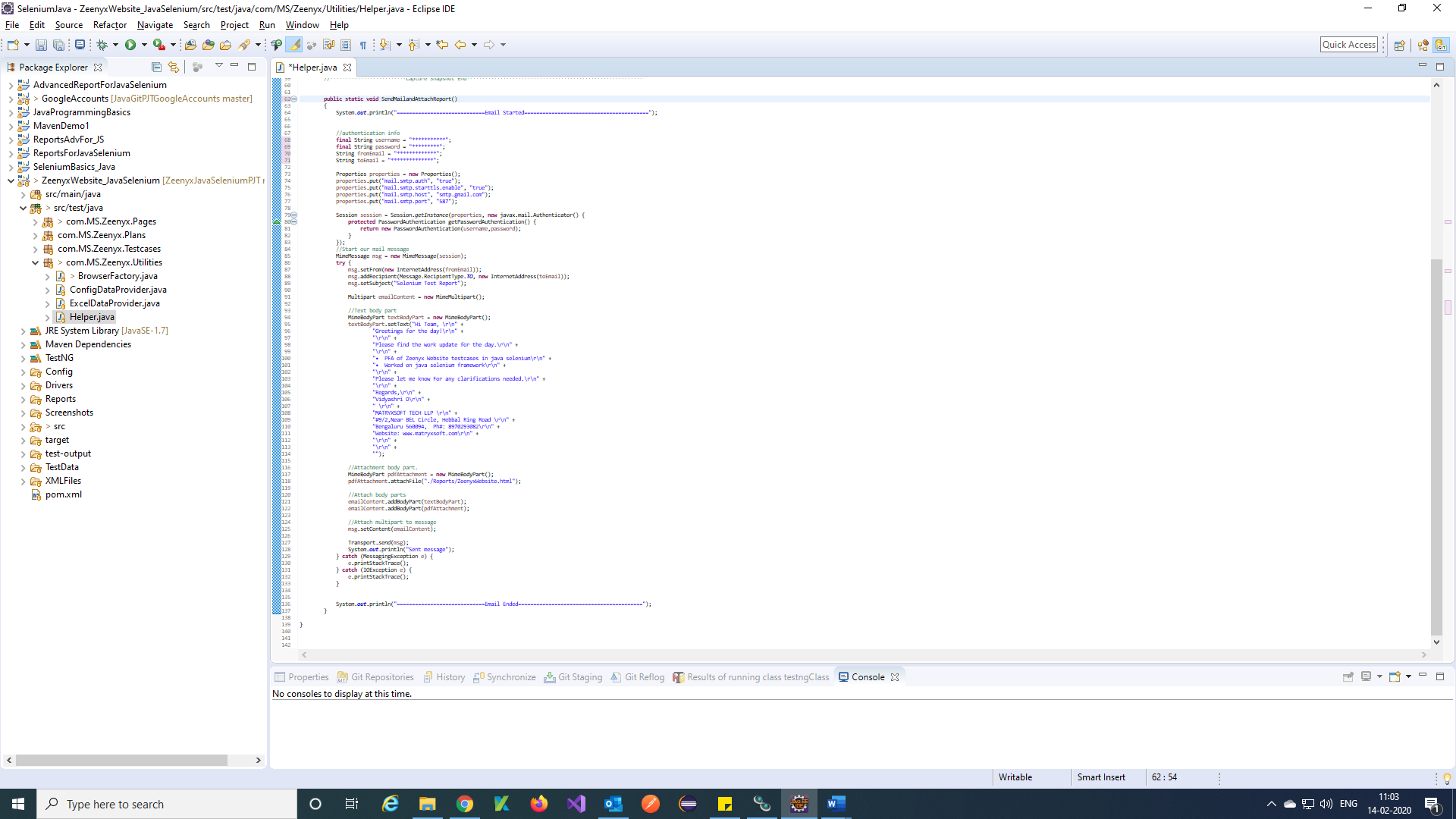
**SendMailandAttachReport** method:

Step 1) In below picture you can find Send mail and Attach Report method.

Note: Before writing Send mail, method add javax mail dependency to pom.xml

URL for dependency is 🡪 <https://mvnrepository.com/artifact/com.sun.mail/javax.mail/1.6.2>



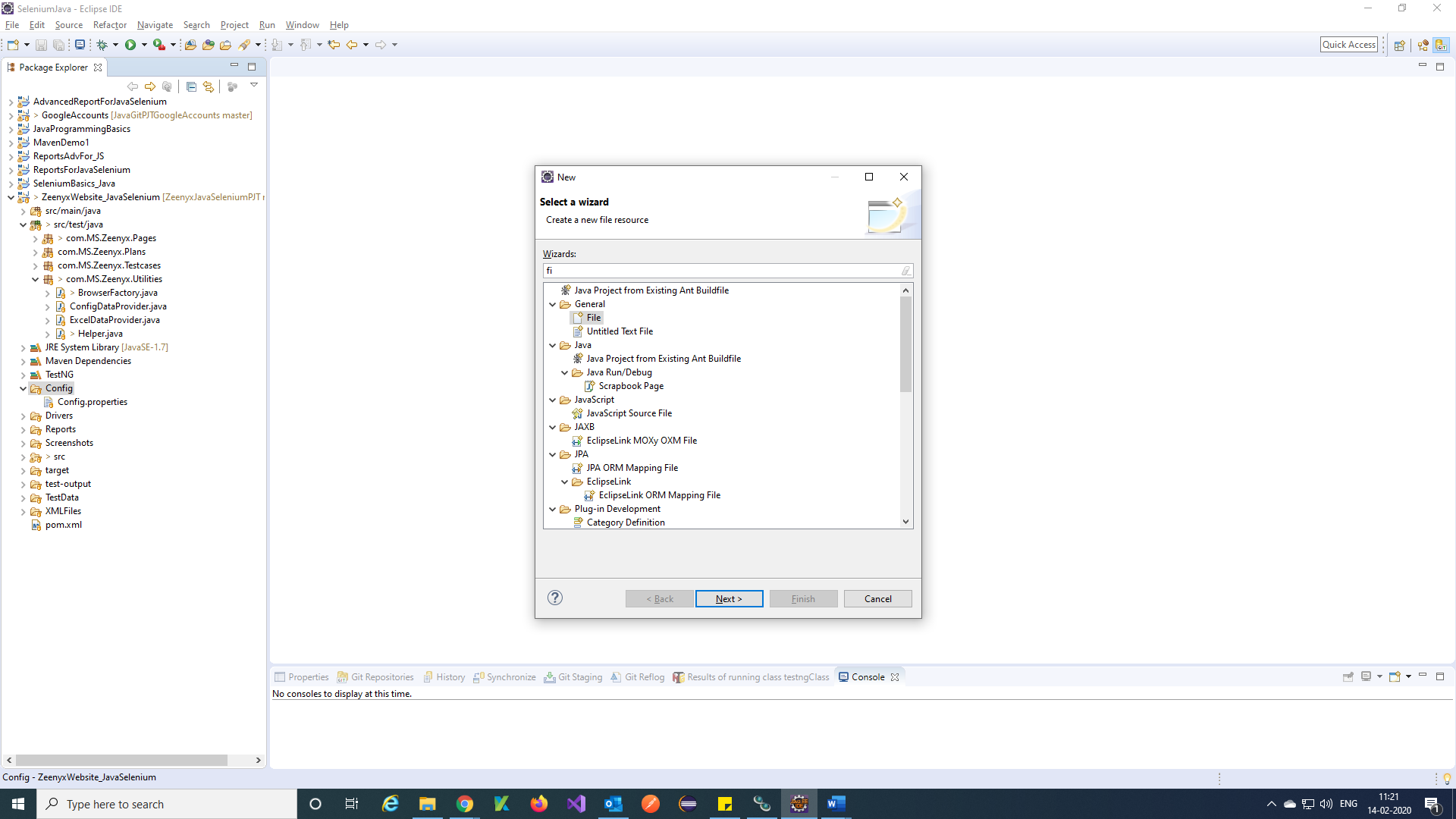


**ConfigDataProvider class:**

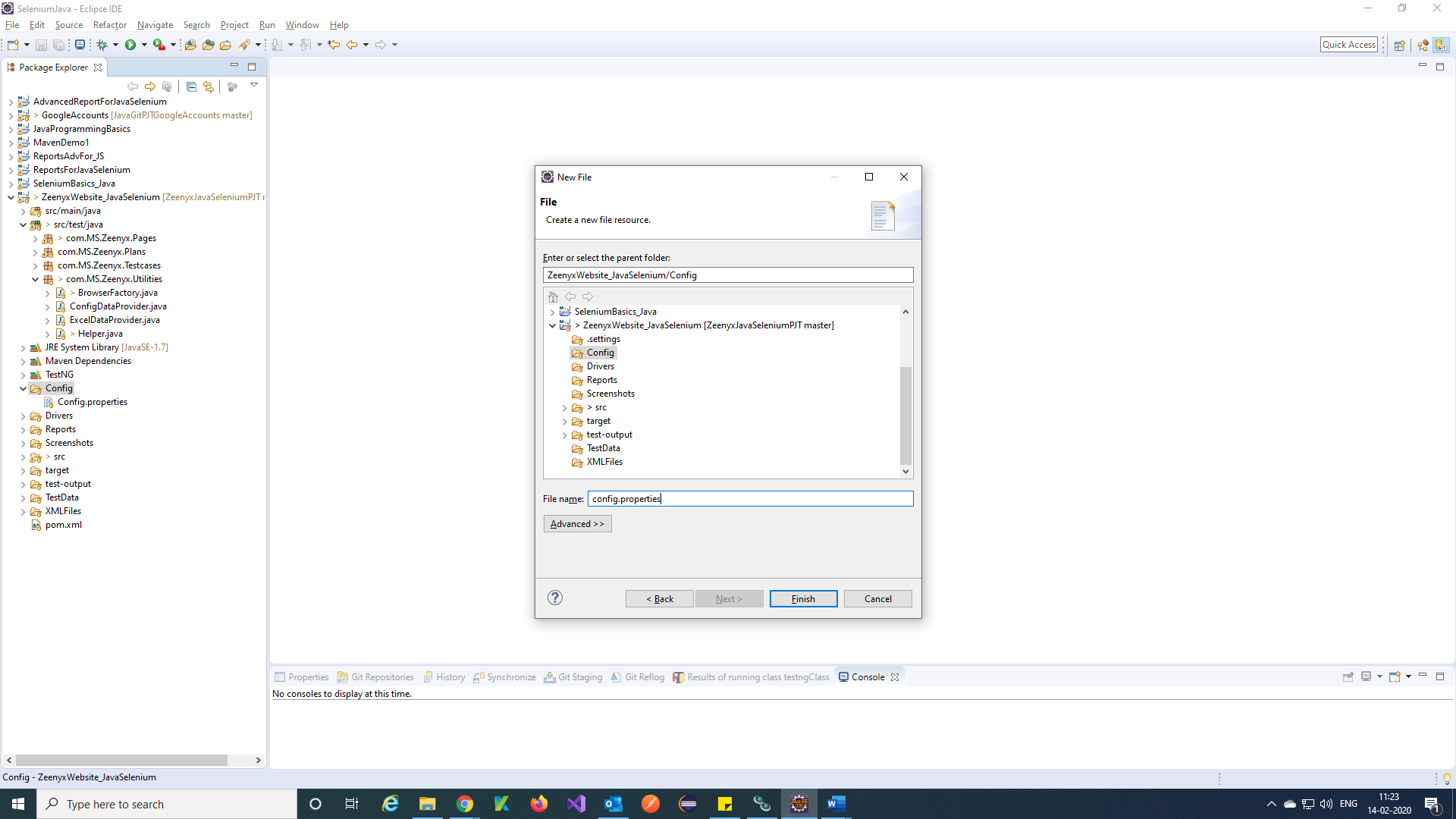
Step 1) Create a folder as Config and in that create a config.properties file to get global data for our framework.

**Create a config property file**:

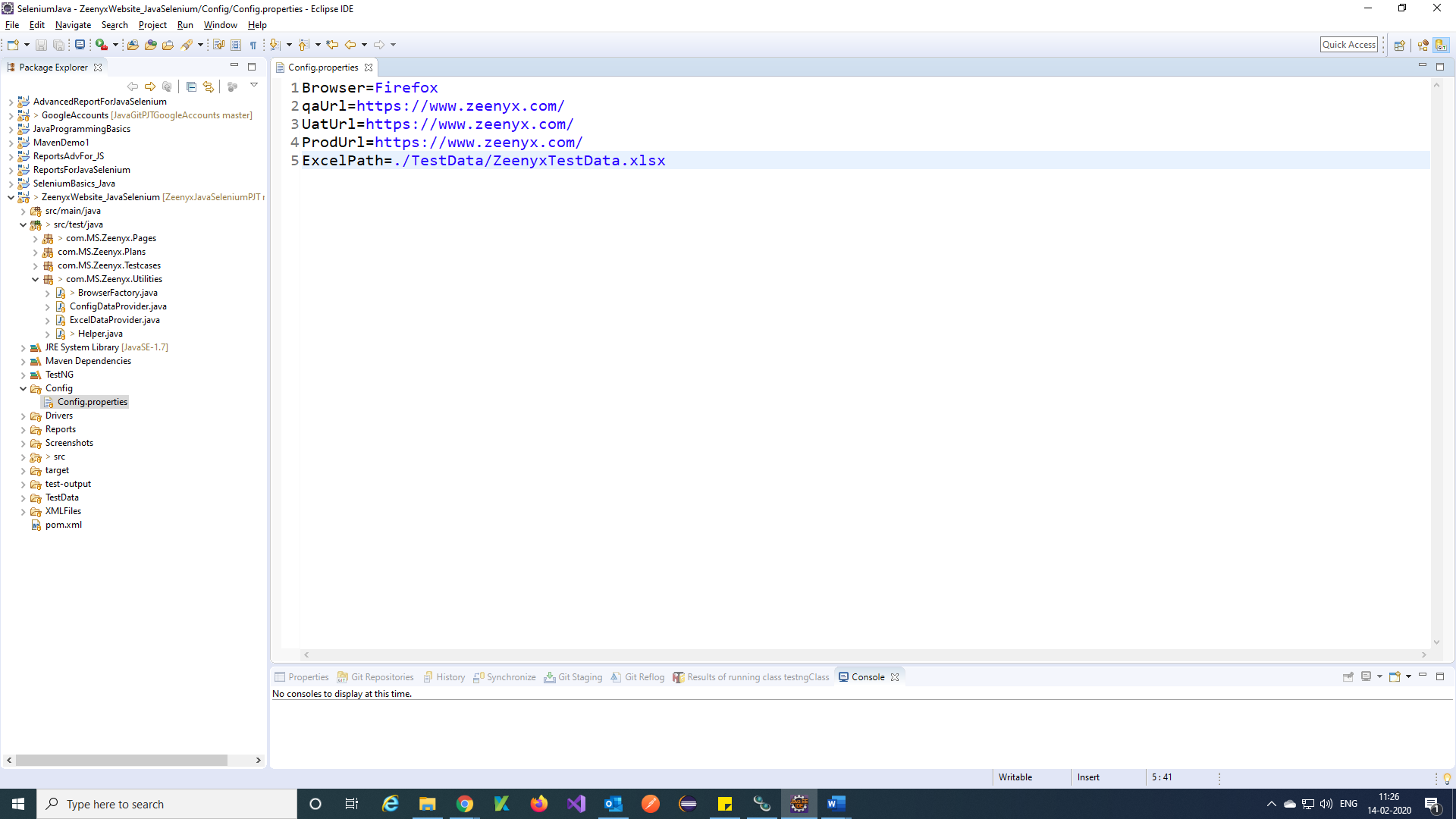
Step 1) Right click on Config folder 🡪 search file🡪 General File 🡪 Next



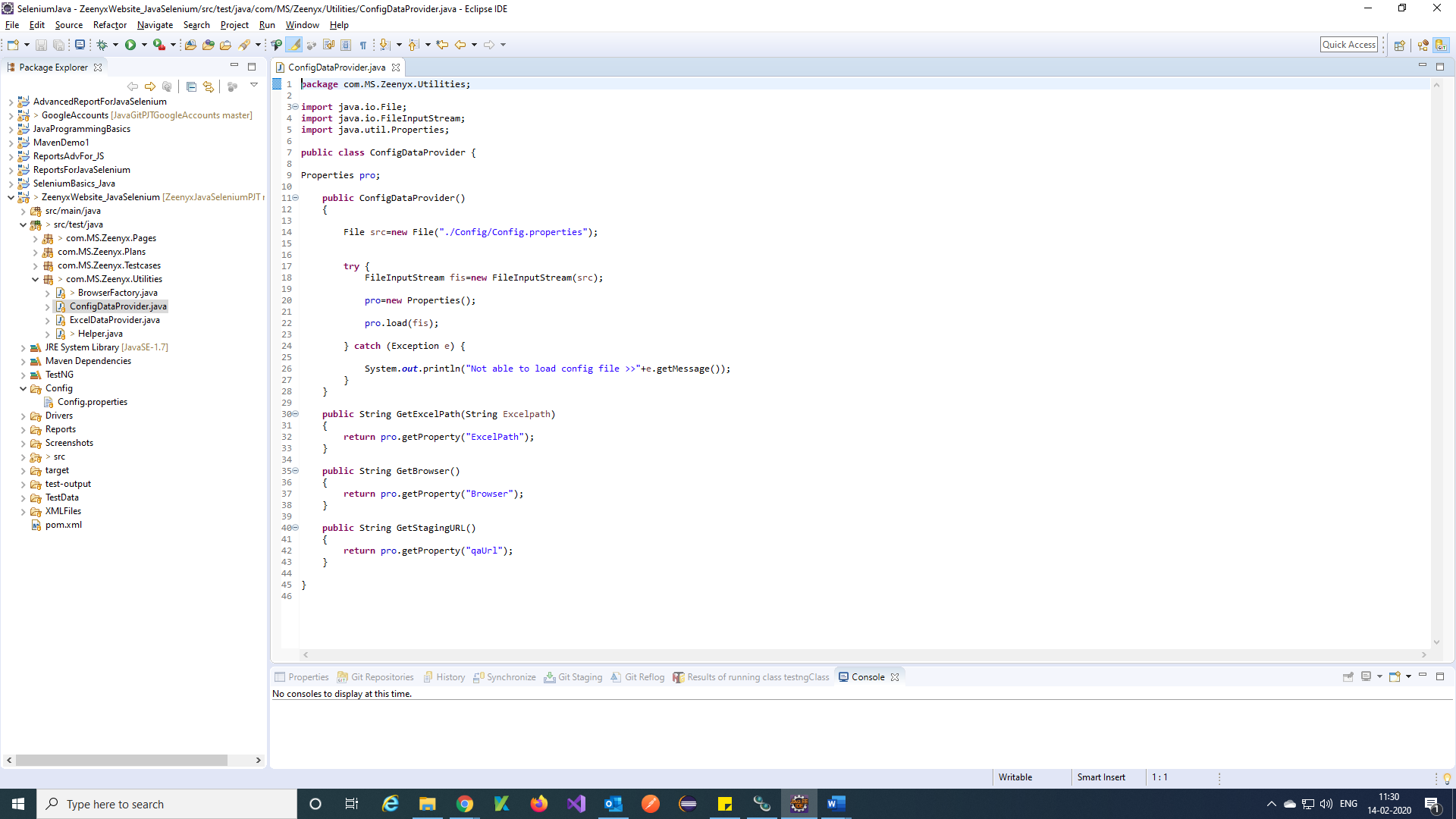
Step 2) Enter file name with extension Config.properties 🡪 Finish



Step 3) Add required global variable data in Config file and save.



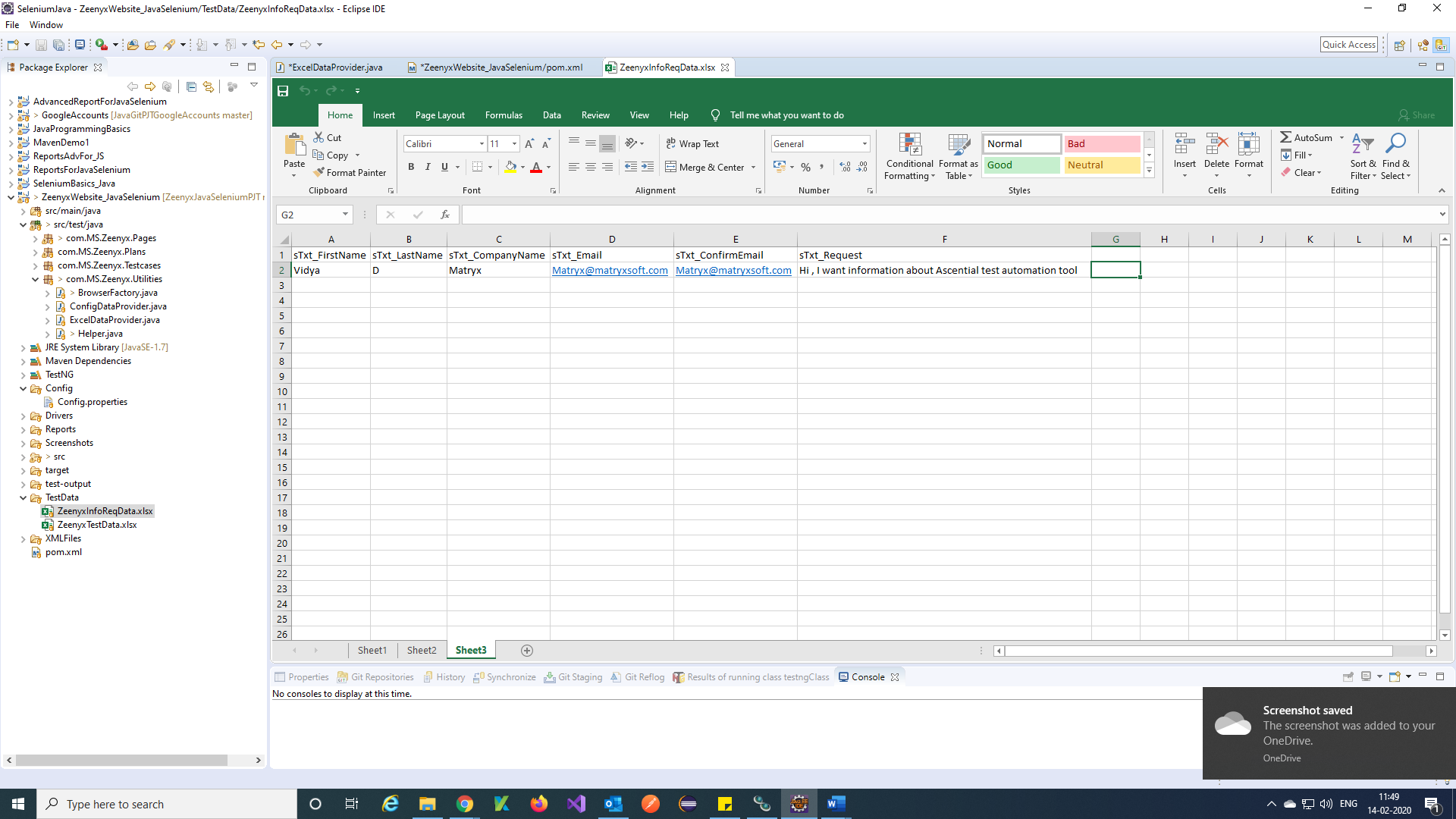
Step 2) Write a function in ConfigDataProvider class to retrieve config data into your framework.



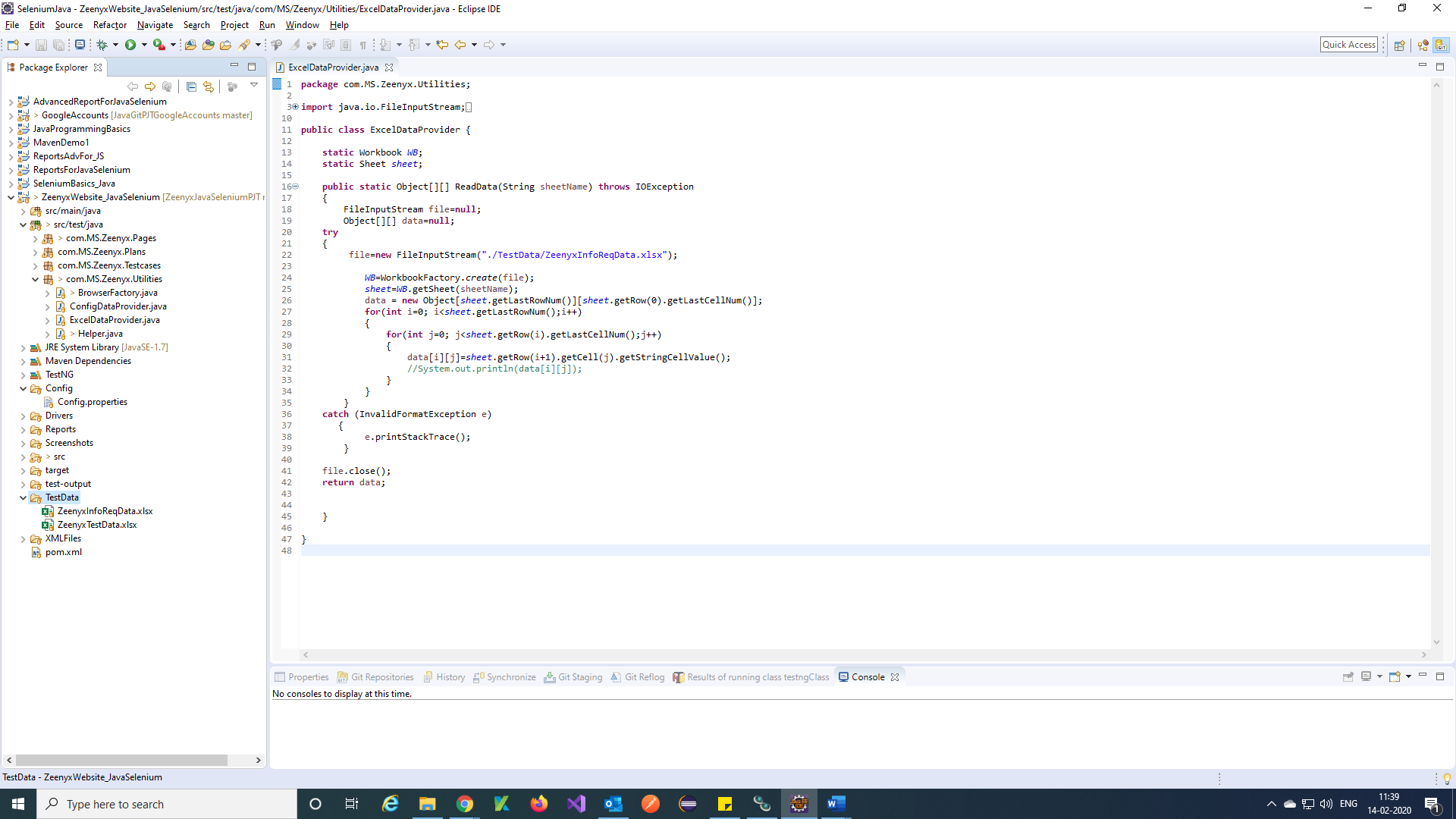
**ExcelDataProvider class:**

Step 1) Create a folder called TestData and

create Testcases data in excel and add that excel file to TestData folder.



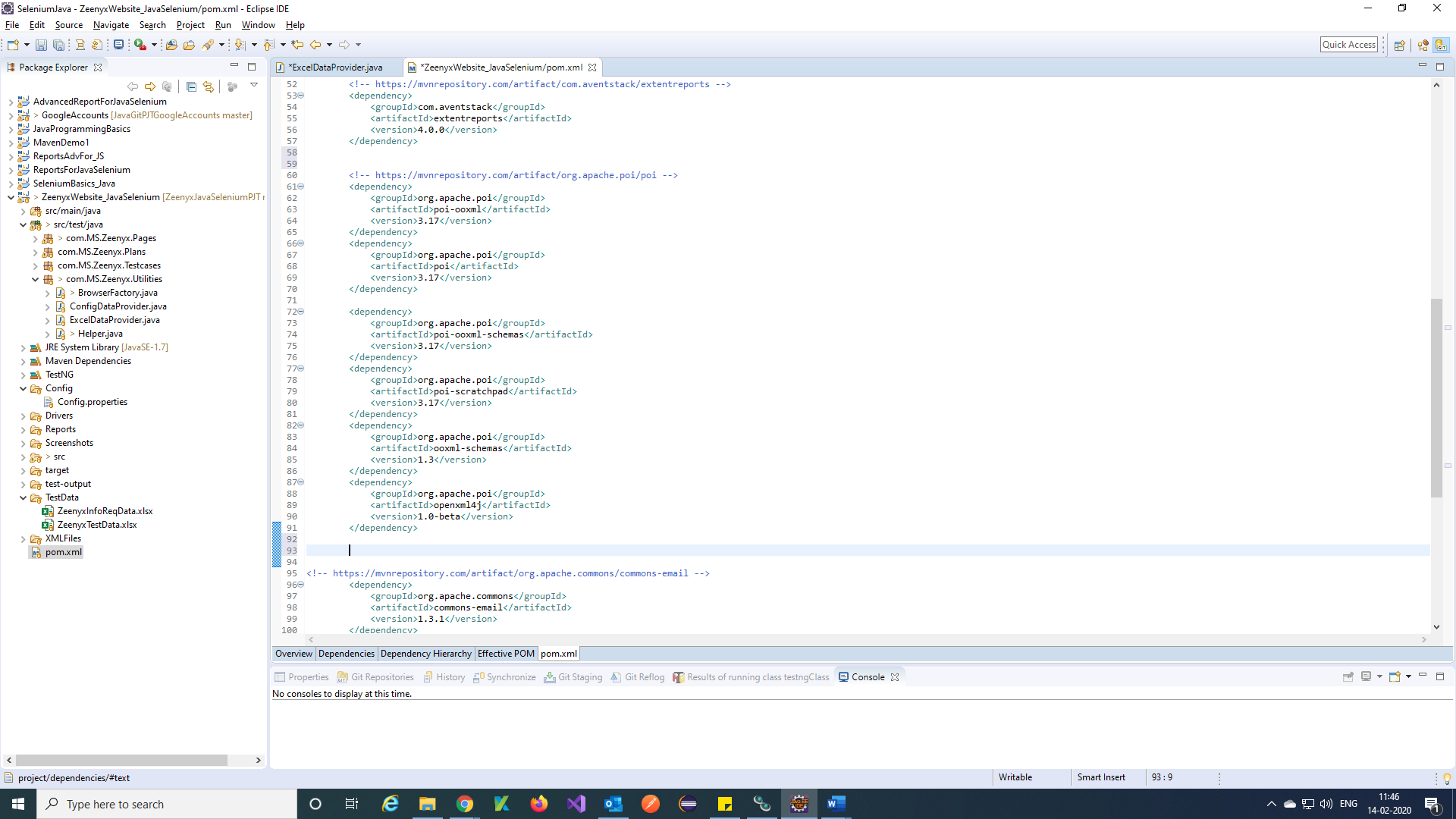
Note: Just copy your Excel file and paste it to TestData folder in Your project.

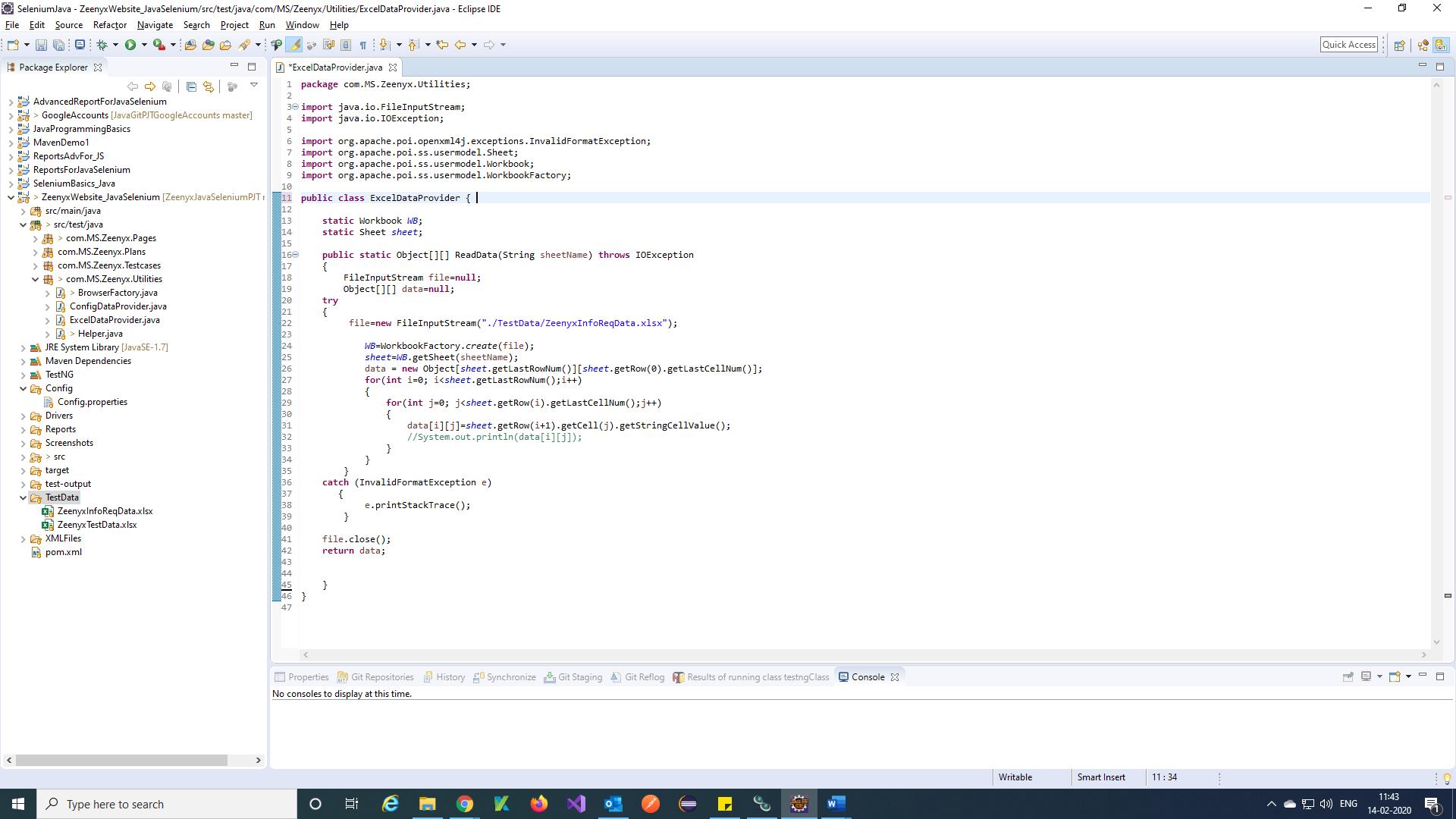


Step 2) Write a function in ExcelDataProvider class to retrieve excel file data into your framework.

Note: Before writing ExcelDataProvider function method add **apache poi** dependency to pom.xml

URL for dependency is 🡪 <https://mvnrepository.com/search?q=apache+poi>





**BaseClass:**

Step 1) Create a class called **BaseClass** in **Pages** package

What is **Base Class** in Selenium:

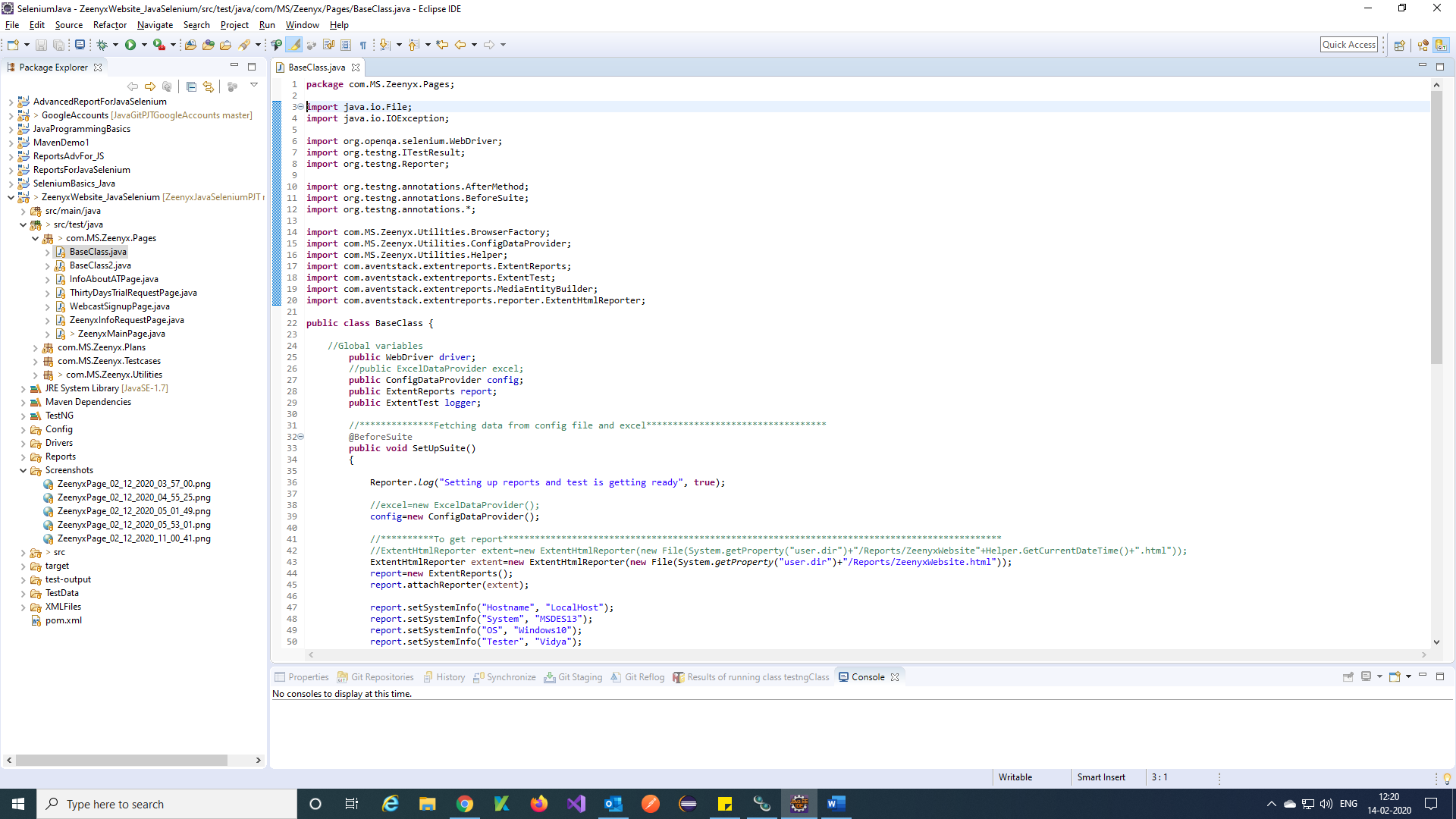
Base class in the main class which will take care of Browser setup, loading [configuration file](http://learn-automation.com/object-repository-in-selenium-webdriver/)and other reusable methods like [screenshot](http://learn-automation.com/how-to-capture-screenshot-for-failed-test-cases-in-selenium-webdriver/), handling [sync issues](http://learn-automation.com/best-way-to-handle-synchronization-in-selenium-webdriver/)and many more. Using Base class, we can avoid code duplication. Reuse code as much we can.

**Note**:

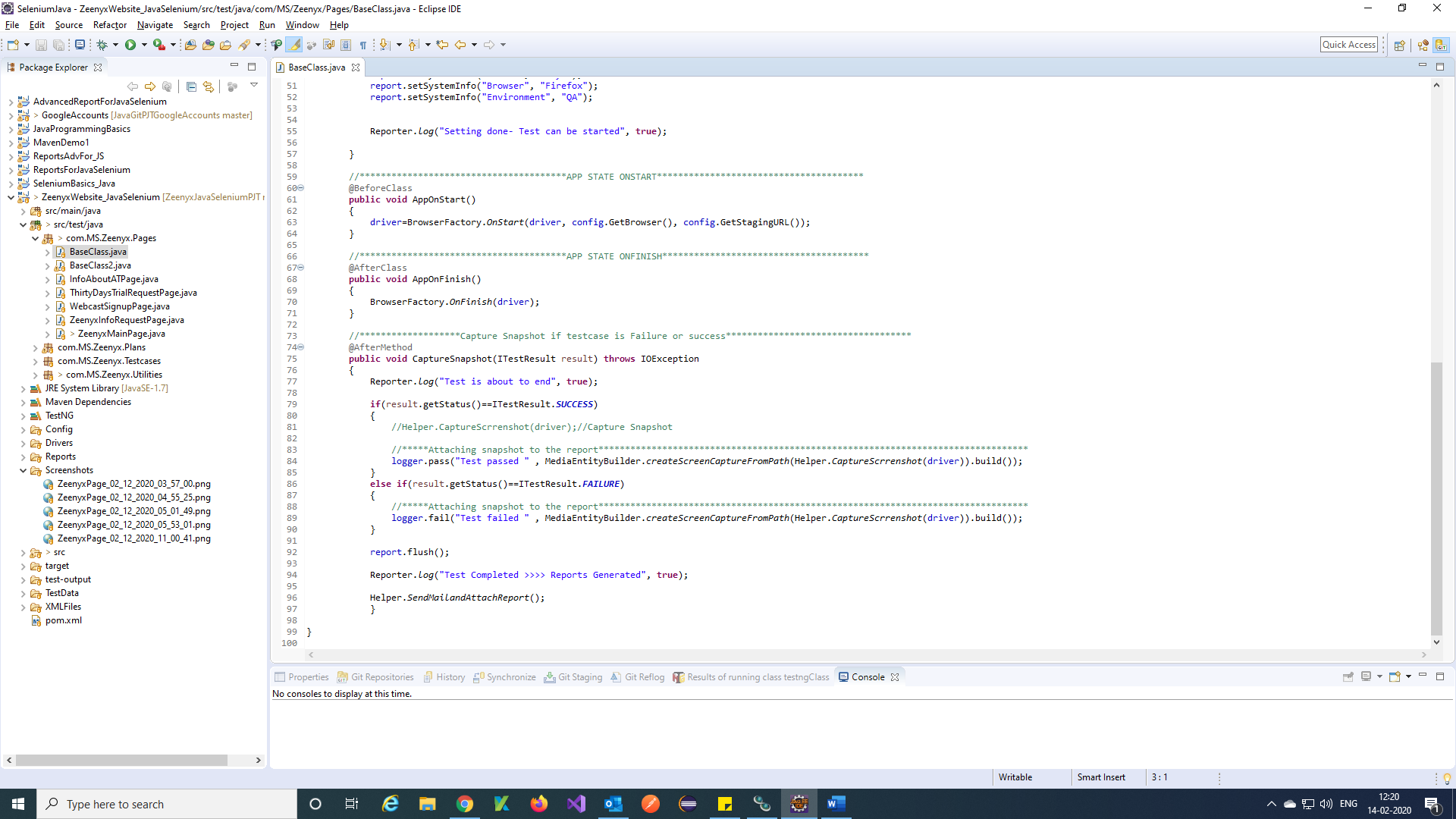
Preconditions before jumping into base class Once go through below concepts:

TestNG should be [installed](http://learn-automation.com/how-to-install-testng-in-eclipse/) and little knowledge on [TestNG Framework](http://learn-automation.com/testng-tutorials-for-beginners/)**.**

Understanding of [Inheritance](https://www.youtube.com/watch?v=iDd-OfxOz0U) in JAVA.



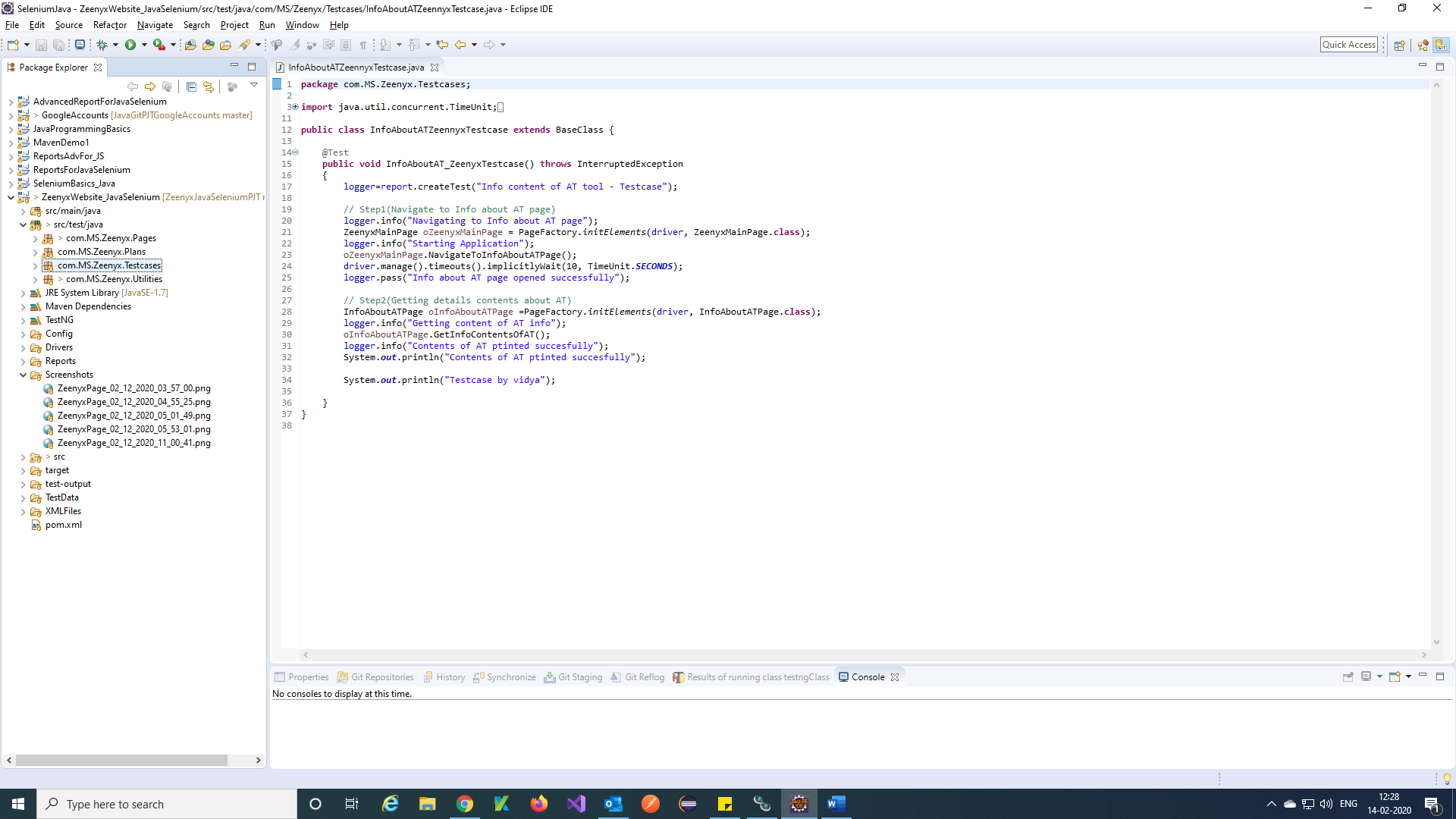
 Continues….



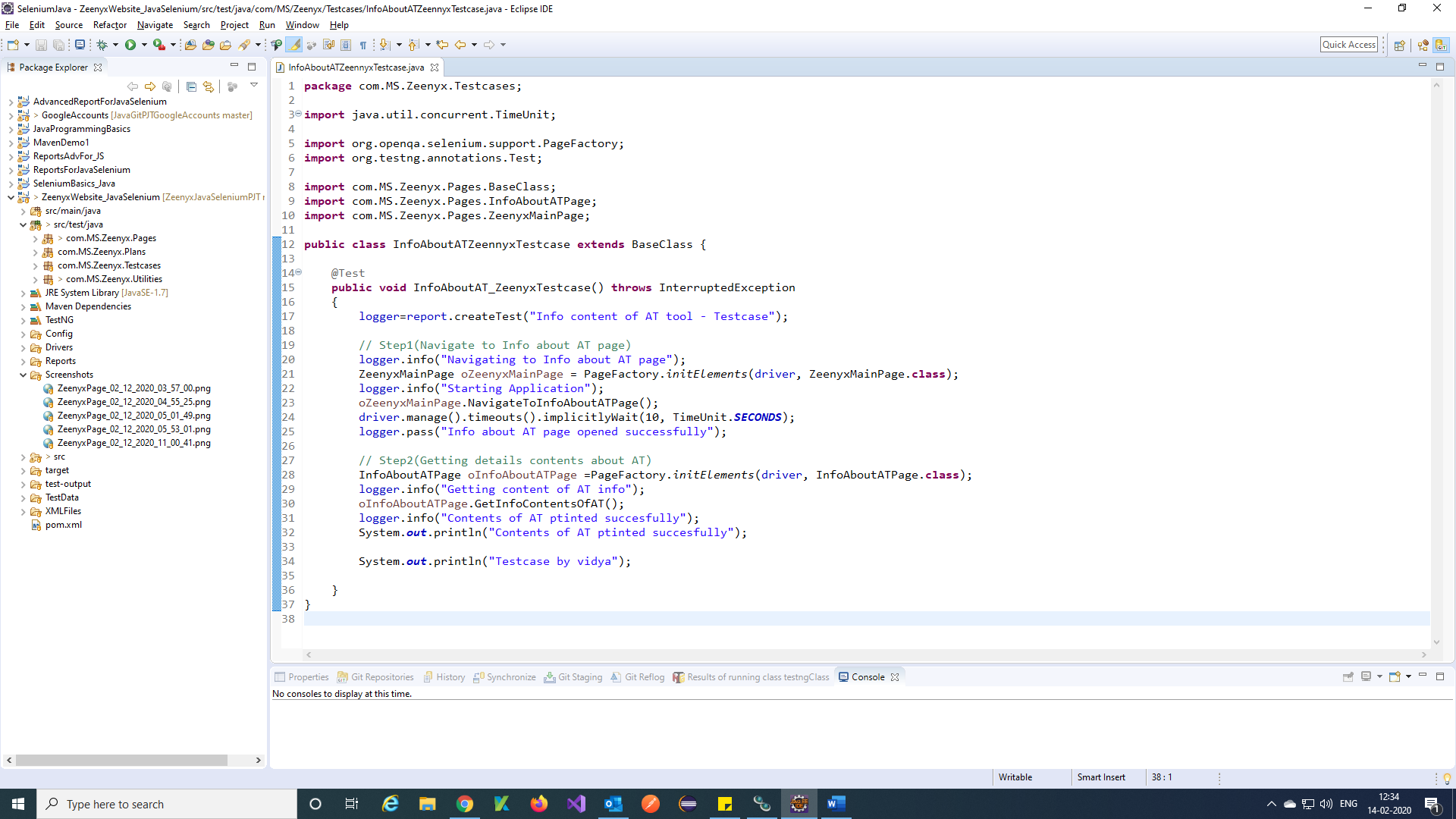
Note: We are calling Browser factory, Config data provider, Helper classes in base class

**Testcases:**

Step 1) Create a package for Testcases for example com.MS.Zeenyx.Testcases



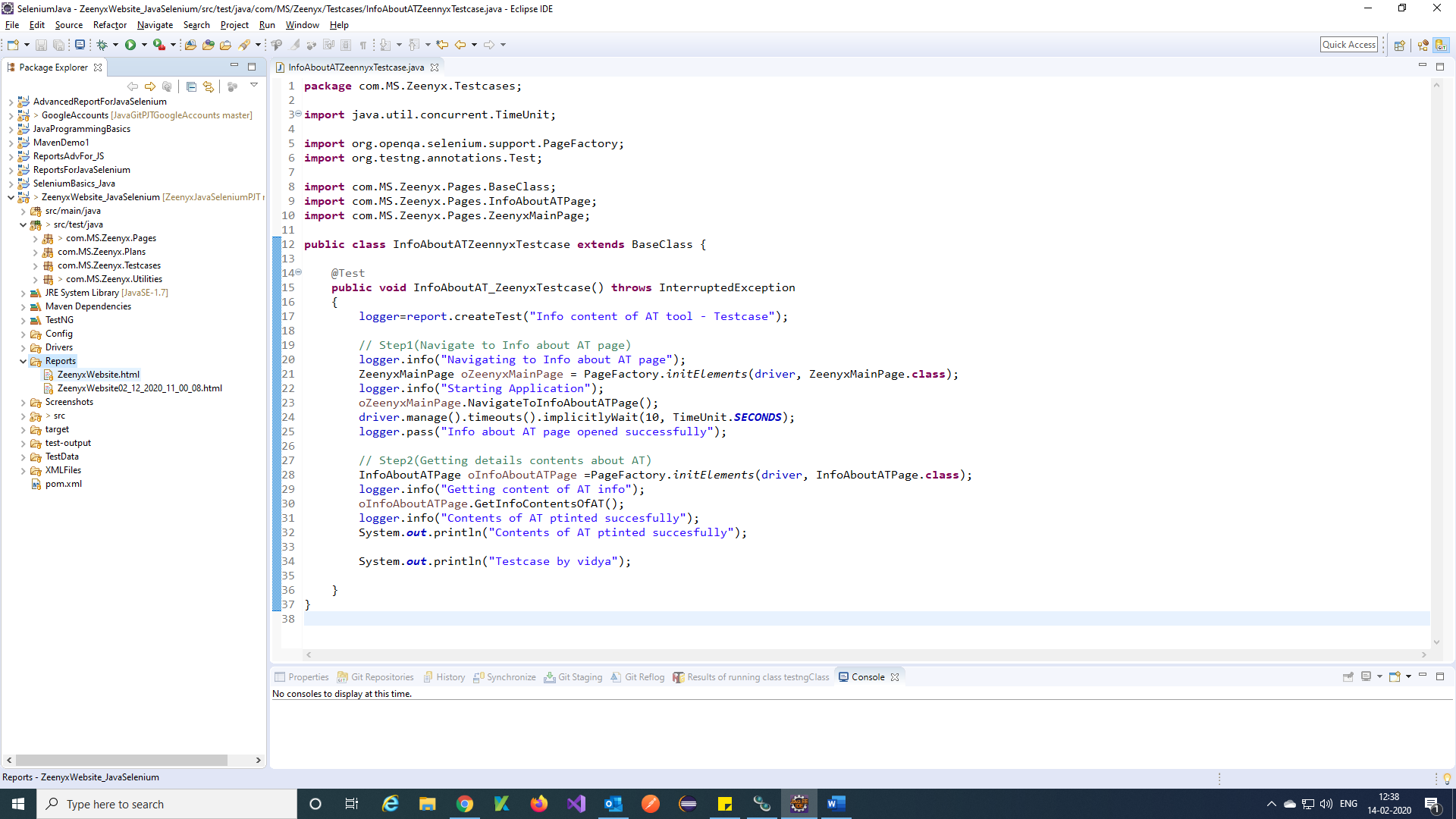
Step 2) Create a class for a testcase and call pages methods in tests.



Note: Here we called BaseClass using Inheritance of java and also used initElements to call page methods.

**Reports:**

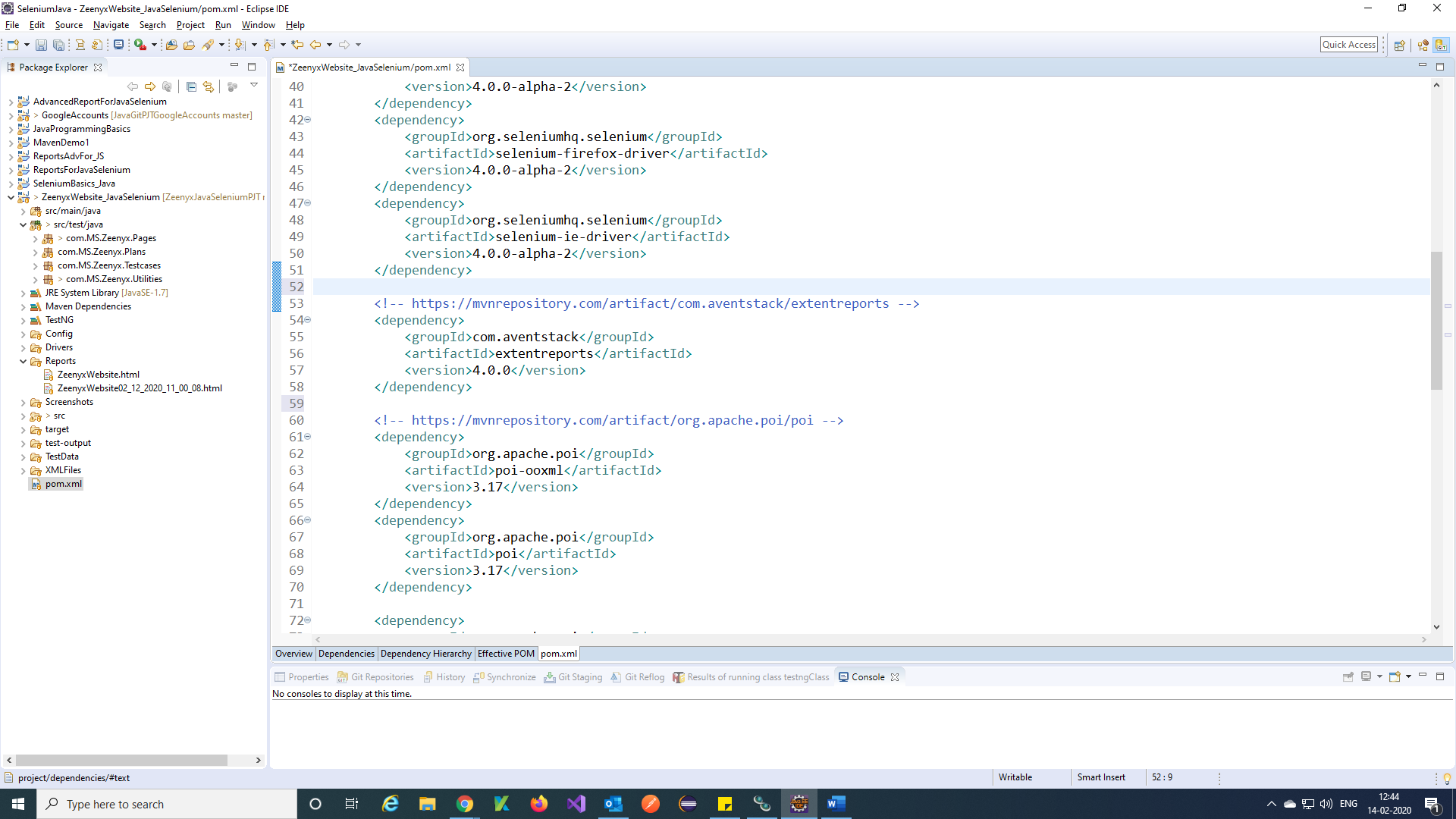
Step 1) Create a folder called reports in your framework and place all the report in the Reports folder.

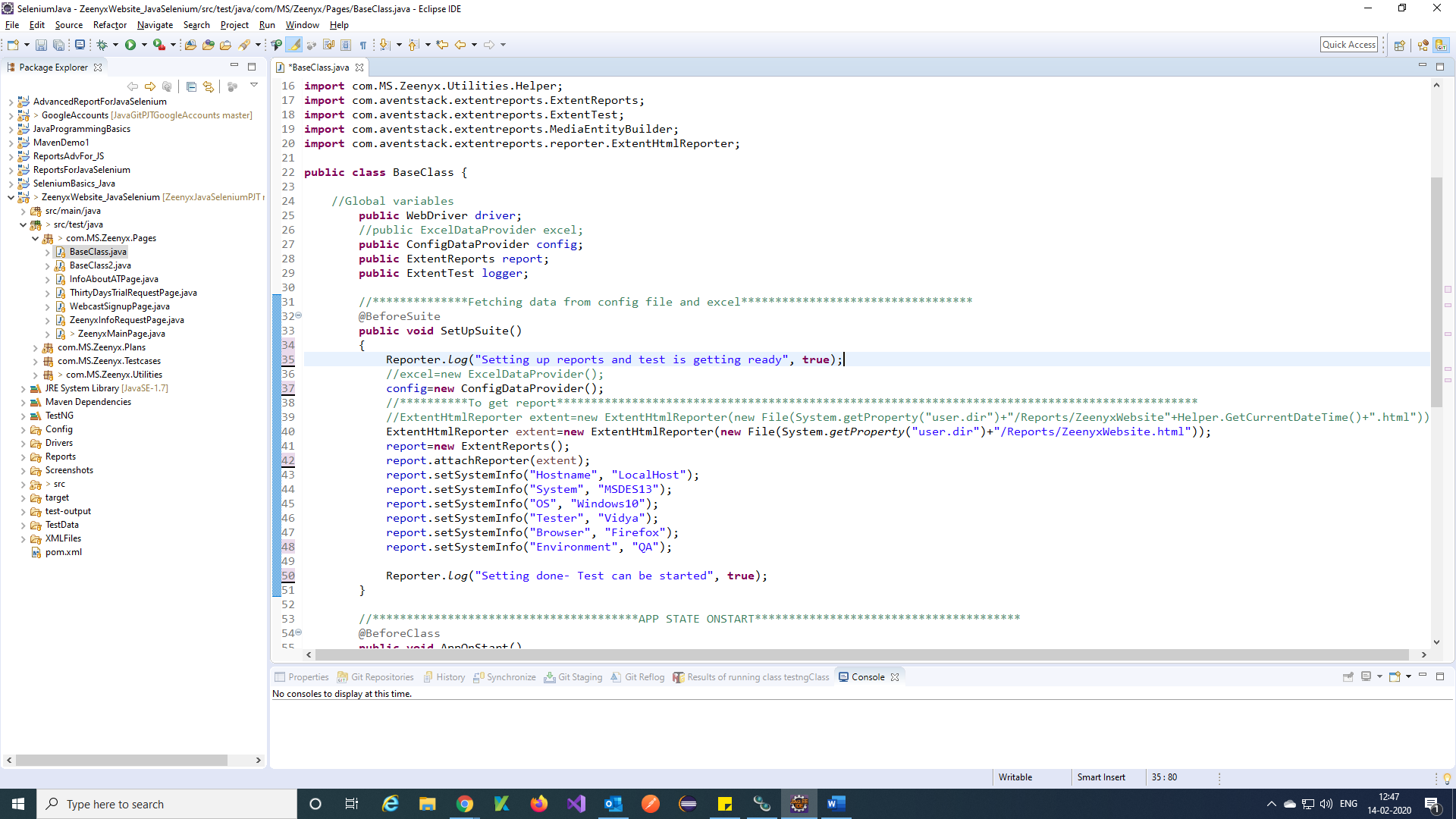


Step 2) Write Report function in base class using @ before Suite TestNG annotation

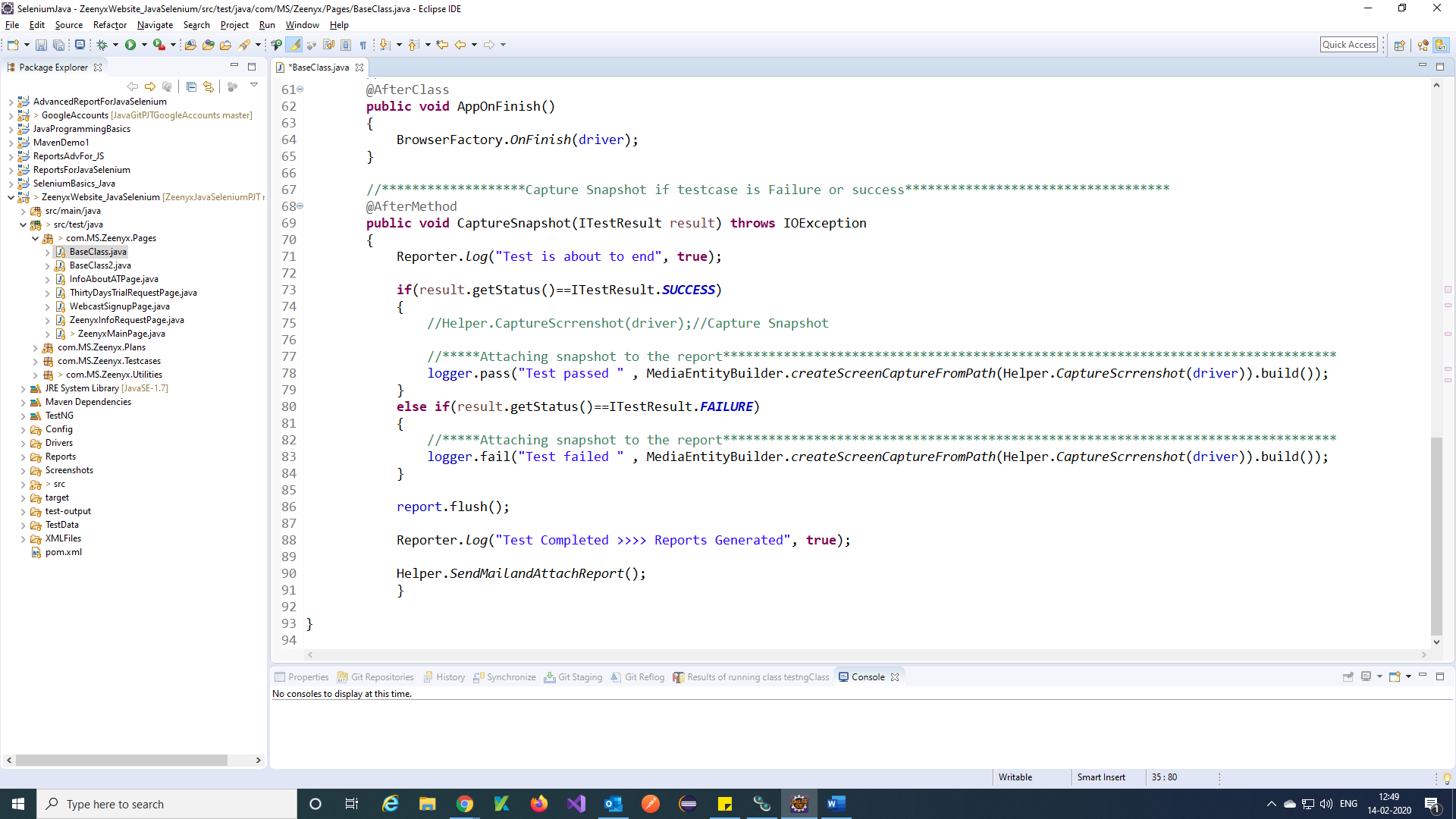
Note: Before writing reports, function add extentreports dependency to pom.xml

URL for dependency is 🡪 <https://mvnrepository.com/artifact/com.aventstack/extentreports/4.0.0>

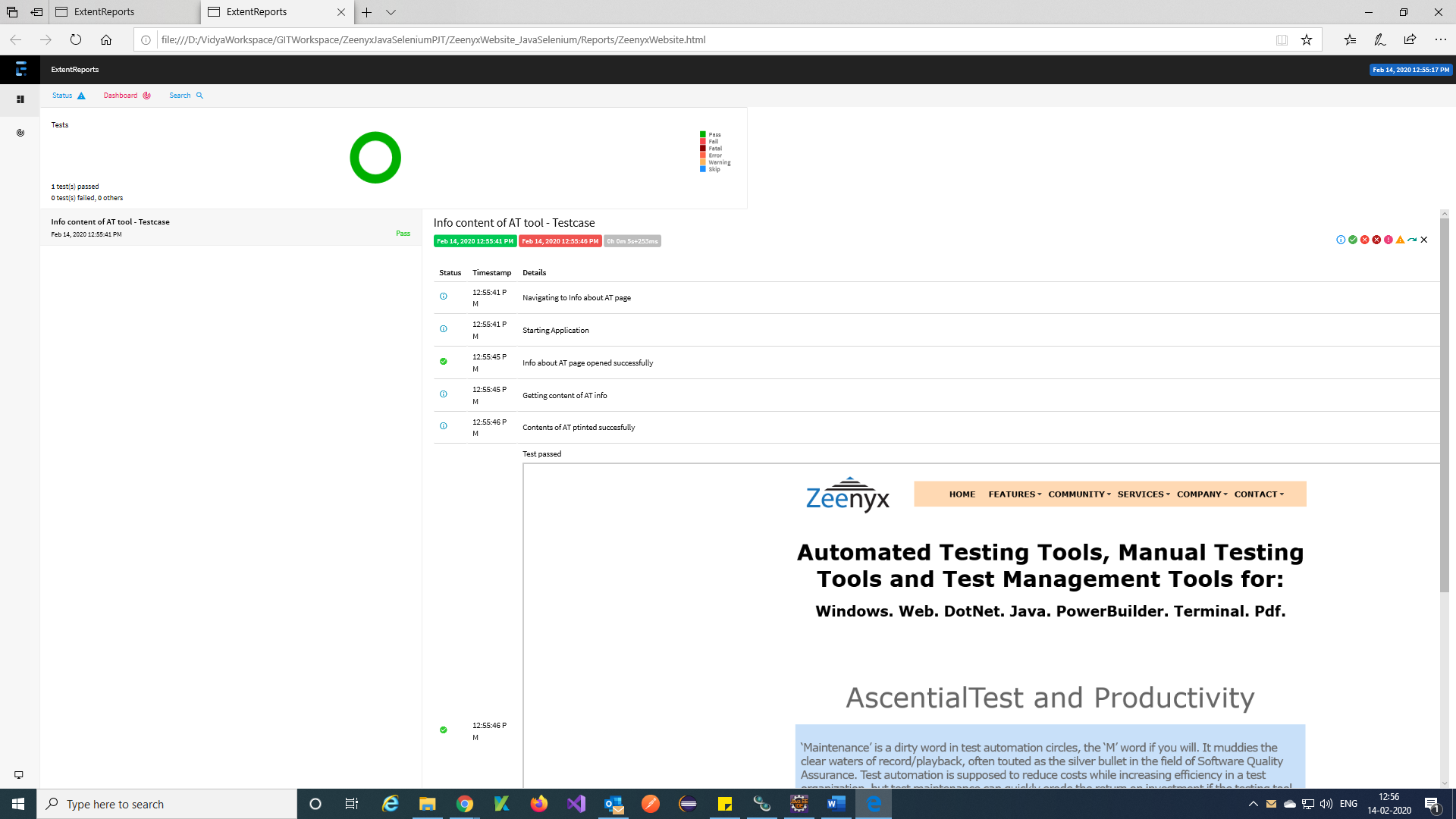




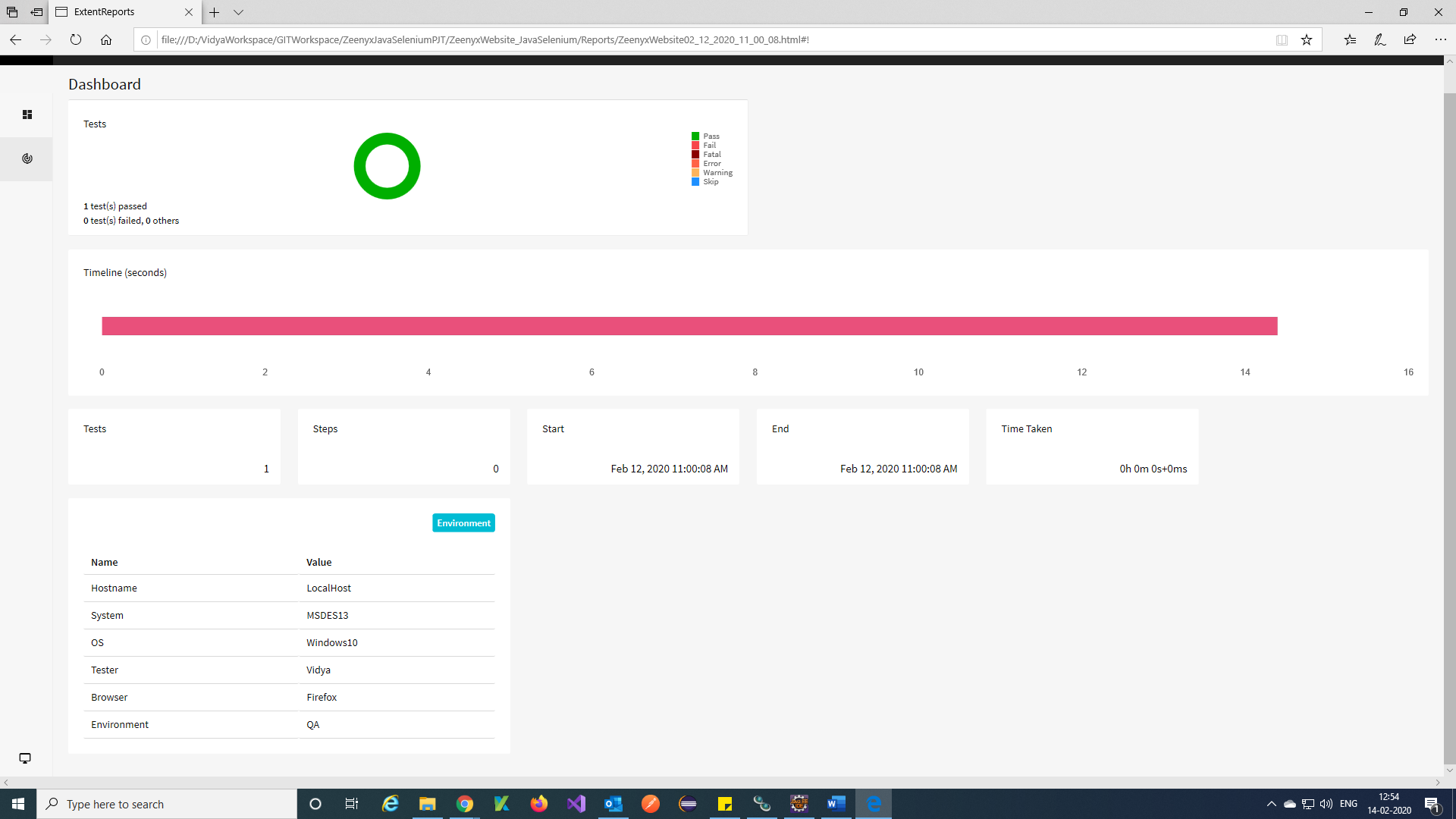
Step 3) Attach snapshot to report and close extent report using flush()



Step 4) We will get report like below



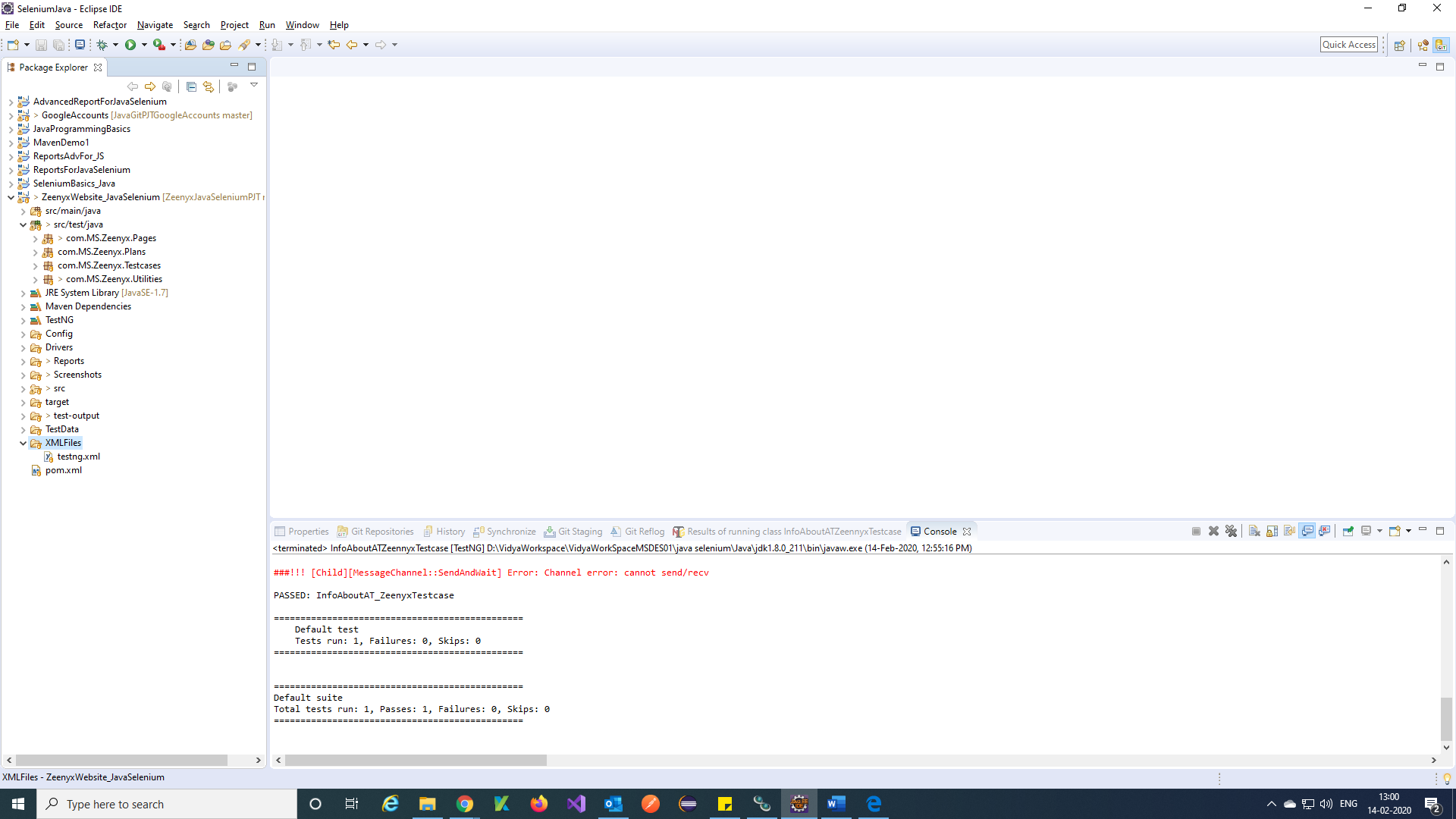
Continues…



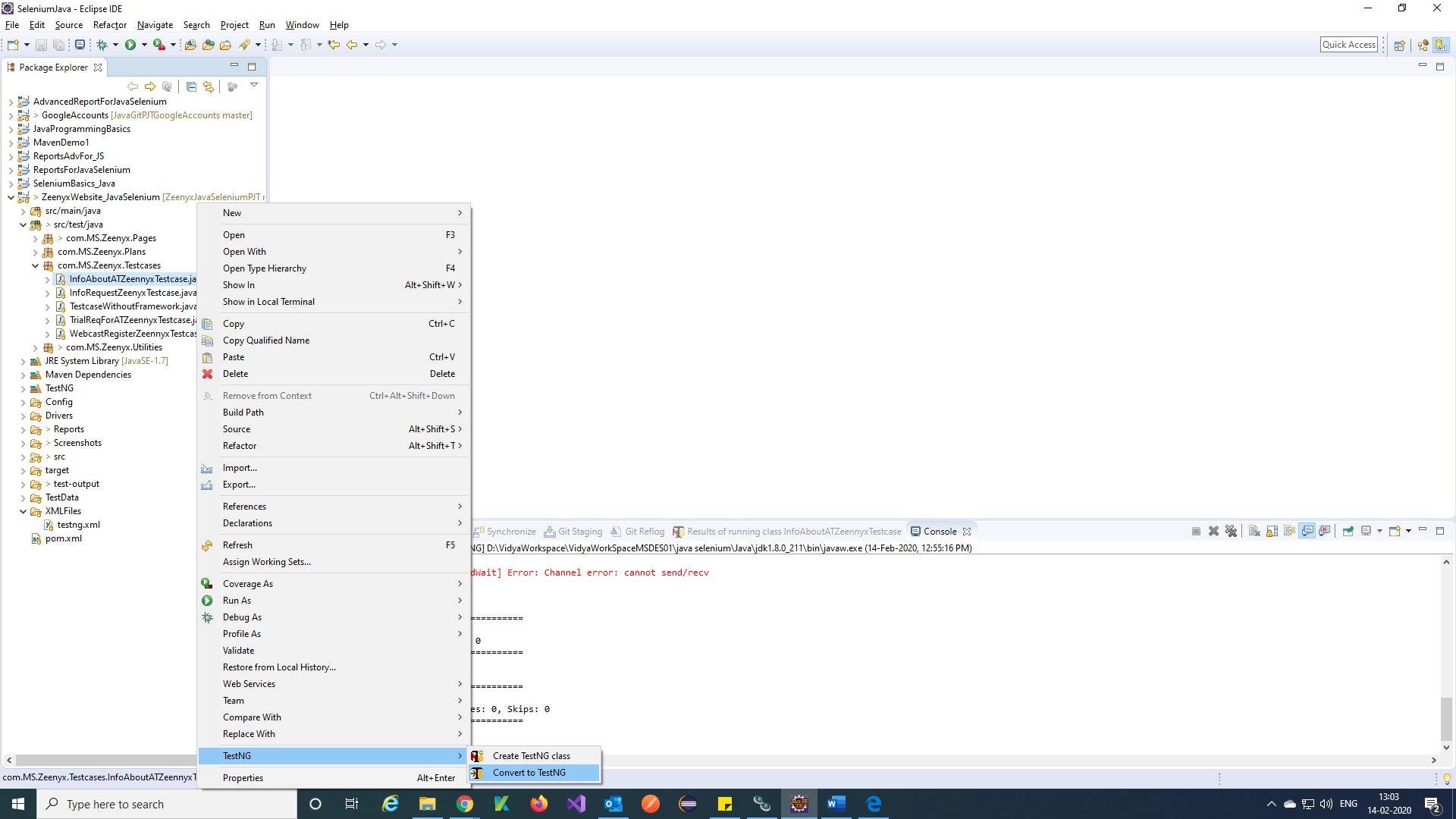
**TestNg XML Files:**

**Create XML files for the classes:**

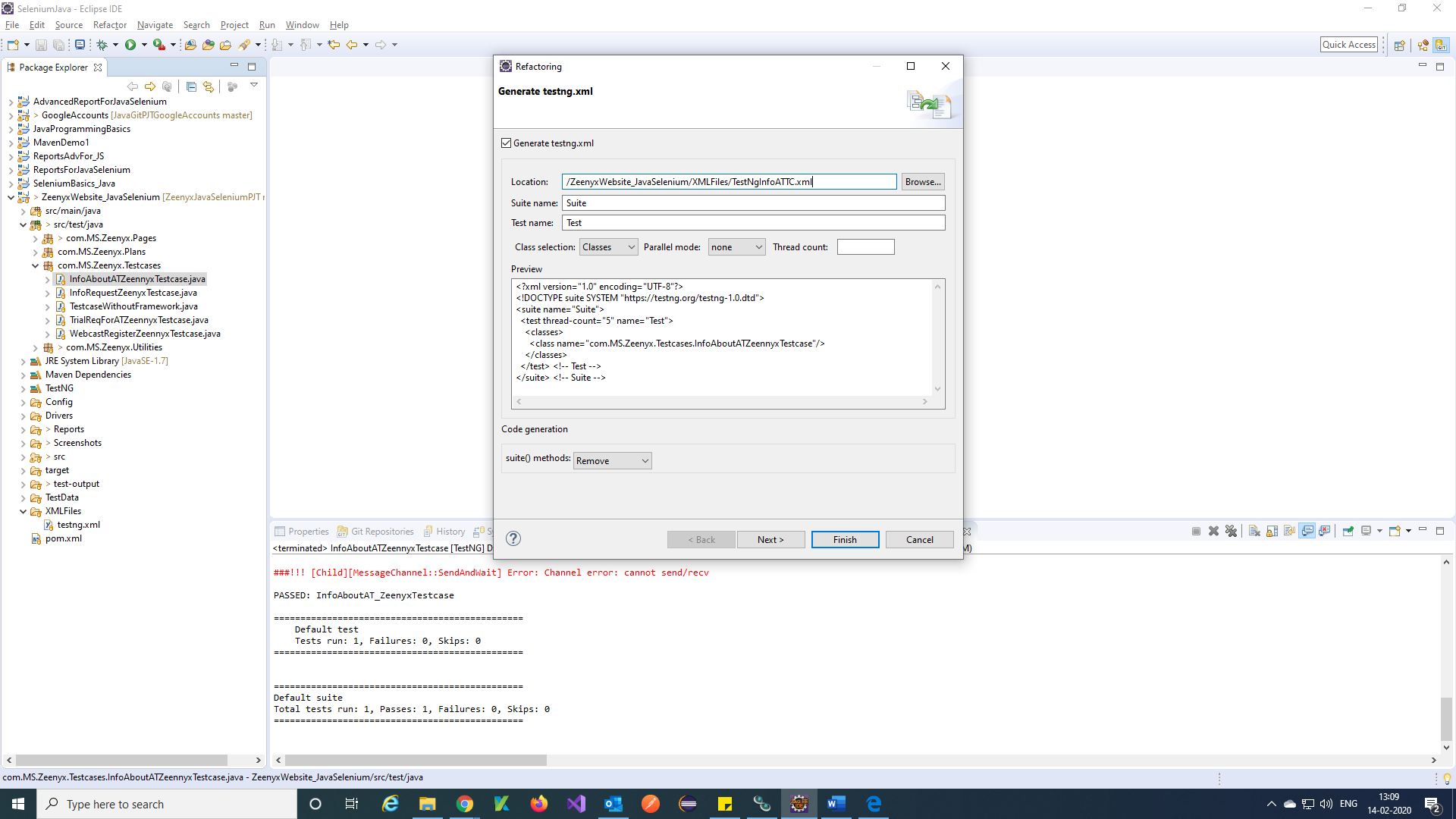
Step 1) Create a folder called XMLFiles in Framework and create all the TestNg file in XMLFiles folders.



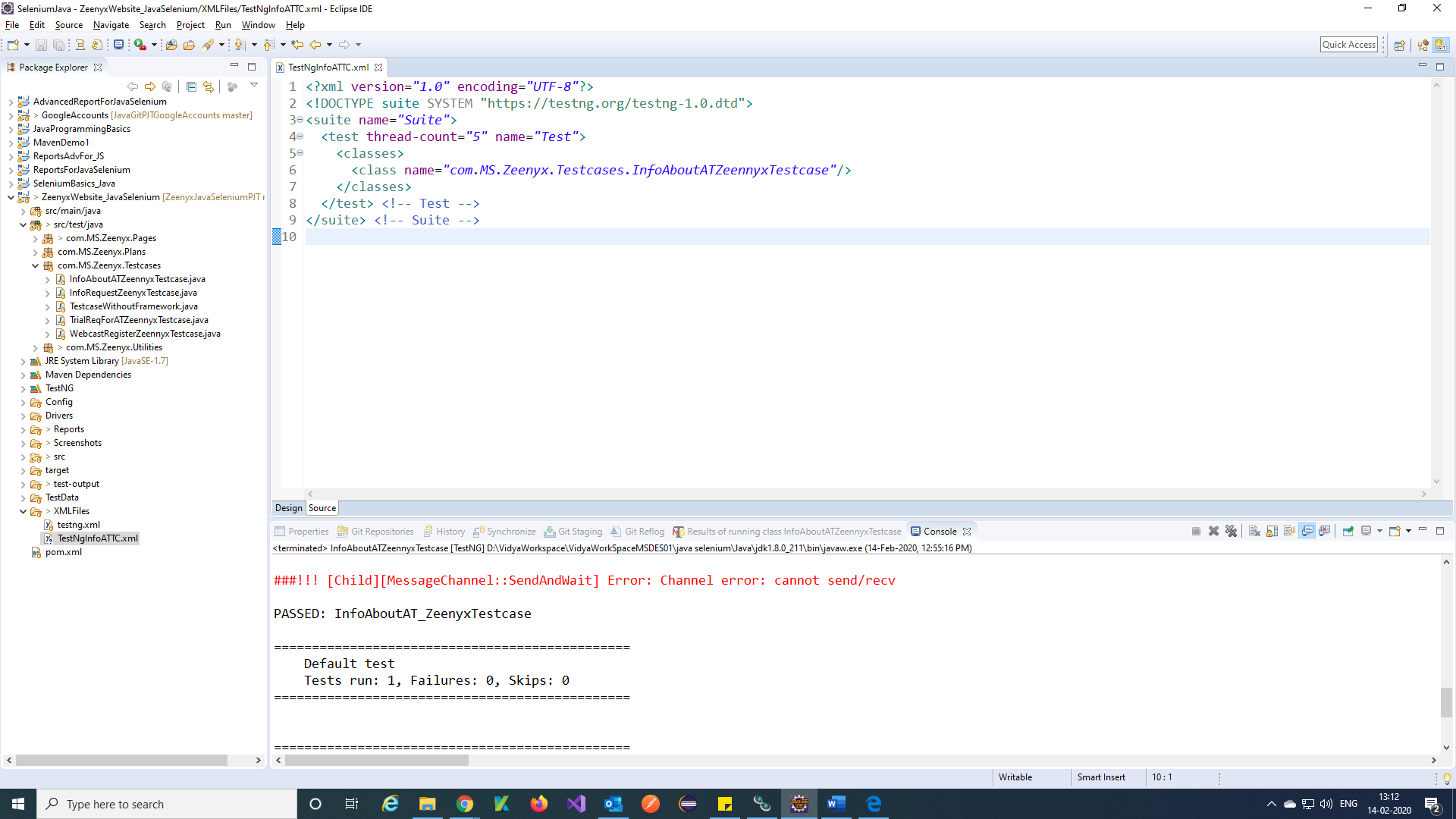
Step 2) Right Click on any testcases 🡪 TestNG 🡪 Convert to TestNG.



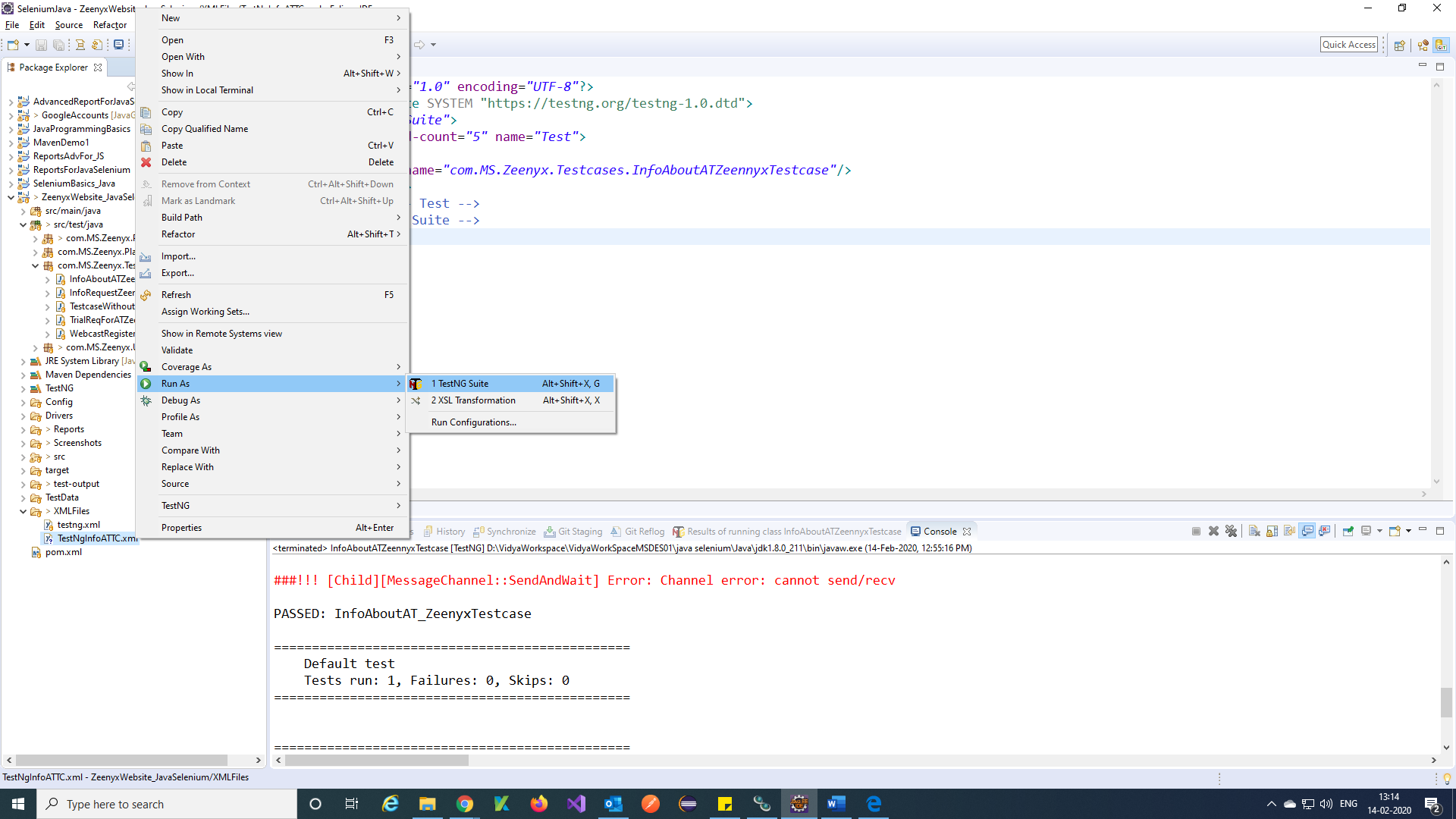
Step 3) Select path to XMLFiles Folder in your project and also Give name for TestNg file 🡪 Finish.



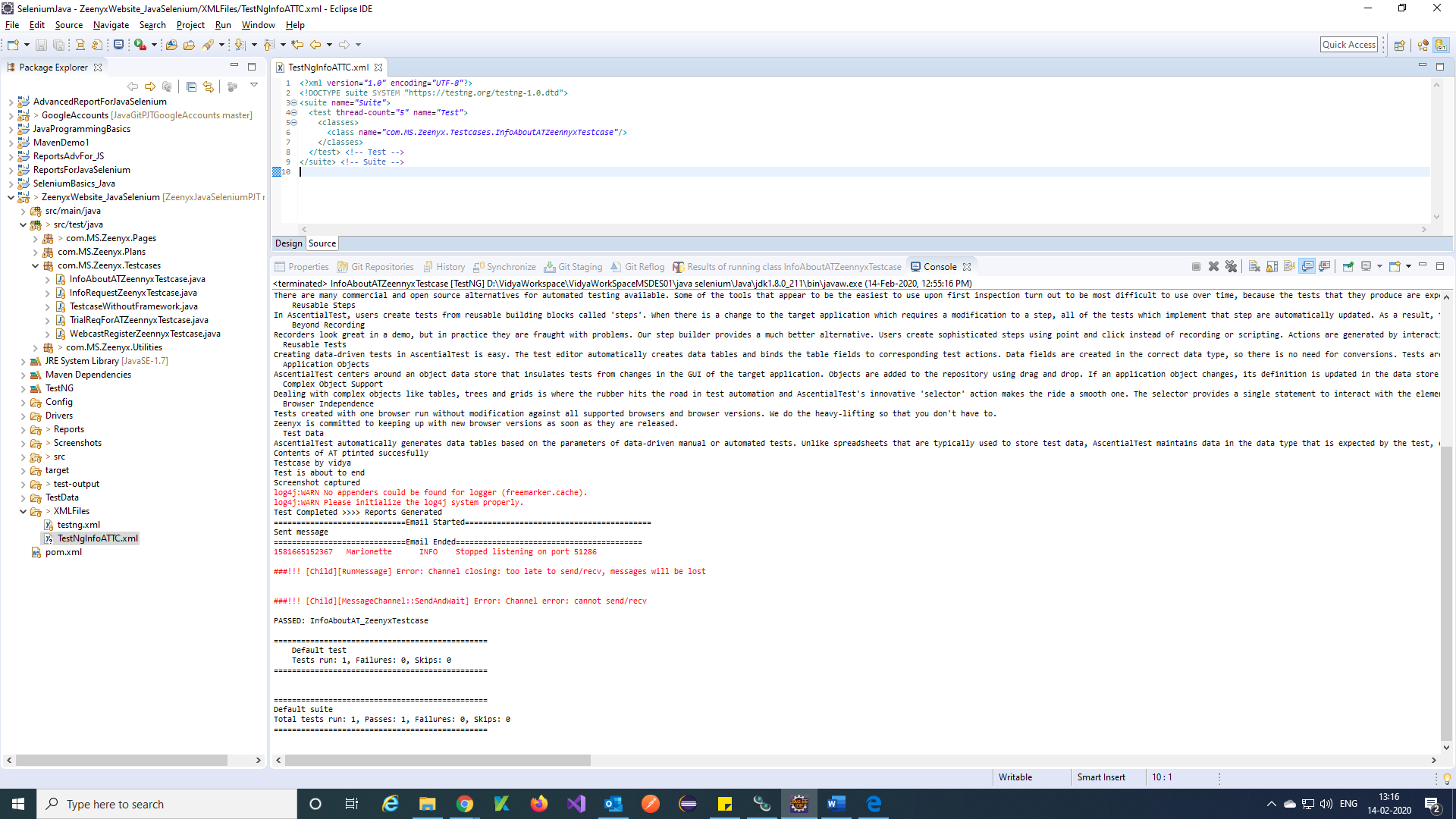
Step 4) It will create TestNg file



Step 5) Right click on created TestNg file 🡪 Run As 🡪 TestNg Suite



Step 6) It will run and gives you a console and report



**Reference:**

<https://www.youtube.com/watch?v=vFXL4nMWvXI&list=PL6flErFppaj0WwNOMFeXPVlNCDuJyPYFi>

Note: Total 10 videos referred for framework.