Hi everyone. In this tutorial, we are going to learn how to check code coverage of your application using Jacoco tool. So, guys, what exactly this core coverage? So, basically you can say core coverage means to measure the number of lines of our code are executed during the automated tests suppose assume I wrote a method with 50 lines and I write a test case for it so from 50 lines of my code, how many number of lines is covered with my test case?

That's what the code coverage, right? So let's understand it with one example and we are going to generate the code coverage report of our application so let's go to Eclipse create a Maven project so I'm creating here one standalone application give name something like we are going to write a program for string palindrome.

**public** **class** App {

**public** **boolean** isPalindrome(String input) {

**if** (input == **null**) {

**throw** **new** IllegalArgumentException("input shouldn't be null");

}

**if** (input.equals(reverse(input))) {

**return** **true**;

} **else** {

**return** **false**;

}

}

**private** String reverse(String input) {

String rev = "";

**for** (**int** i = input.length() - 1; i >= 0; i--) {

rev = rev + input.charAt(i);

}

**return** rev;

}

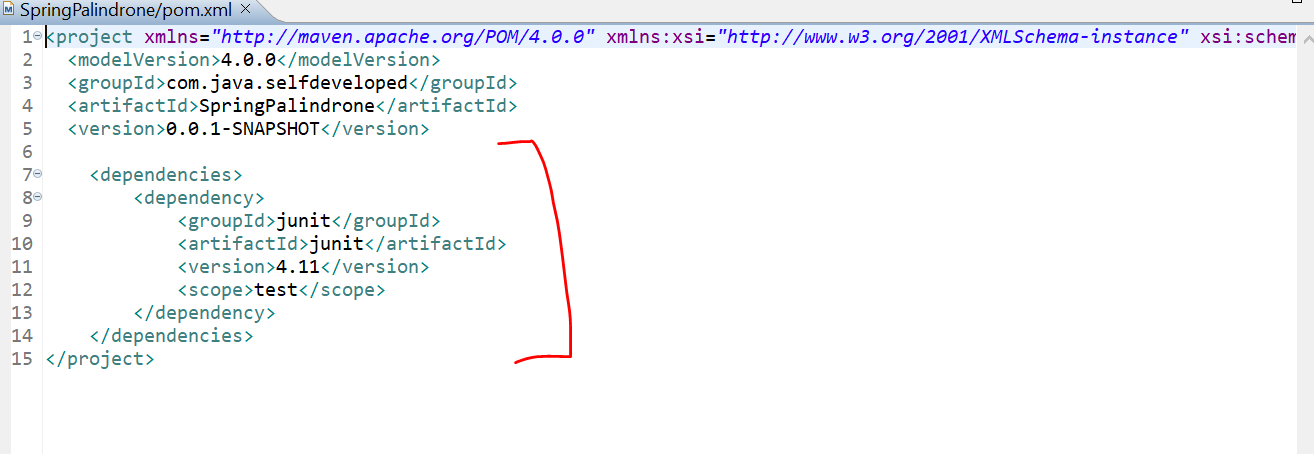
**public** **static** **void** main(String[] args) {

}

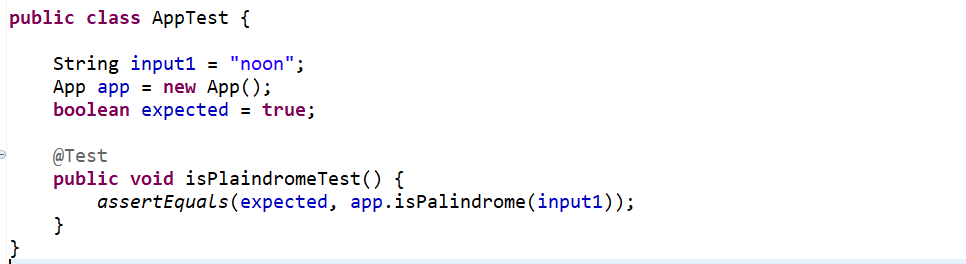
}

now let's write a test case for this each Palindrome method.

Here we need to change the Junit version. Let’s add the Junit version of 4.11.



Later we will add the Jacoco plugin here first let’s add the Test – Case.



In assertEquals() we are expecting true and what actually we are getting, so just call the method with an input parameter.

If it is returning true and we are expecting true, then our test case will be success.

Now we need to add the Jacoco plugin, maven provided plugin for Jacoco through which we can generate the coverage report.

I have that plugin let’s add it in pom.xml..

<build>

<plugins>

<plugin>

<groupId>org.jacoco</groupId>

<artifactId>jacoco-maven-plugin</artifactId>

<version>0.8.3</version>

<executions>

<execution>

<id>prepare-agent</id>

<goals>

<goal>prepare-agent</goal>

</goals>

</execution>

<execution>

<id>report</id>

<phase>prepare-package</phase>

<goals>

<goal>report</goal>

</goals>

</execution>

<execution>

<id>post-unit-test</id>

<phase>test</phase>

<goals>

<goal>report</goal>

</goals>

<configuration>

<!-- Sets the path to the file which contains the execution data. -->

<dataFile>target/jacoco.exec</dataFile>

<!-- Sets the output directory for the code coverage report. -->

<outputDirectory>target/my-reports</outputDirectory>

</configuration>

</execution>

</executions>

<configuration>

<systemPropertyVariables>

<jacoco-agent.destfile>target/jacoco.exec</jacoco-agent.destfile>

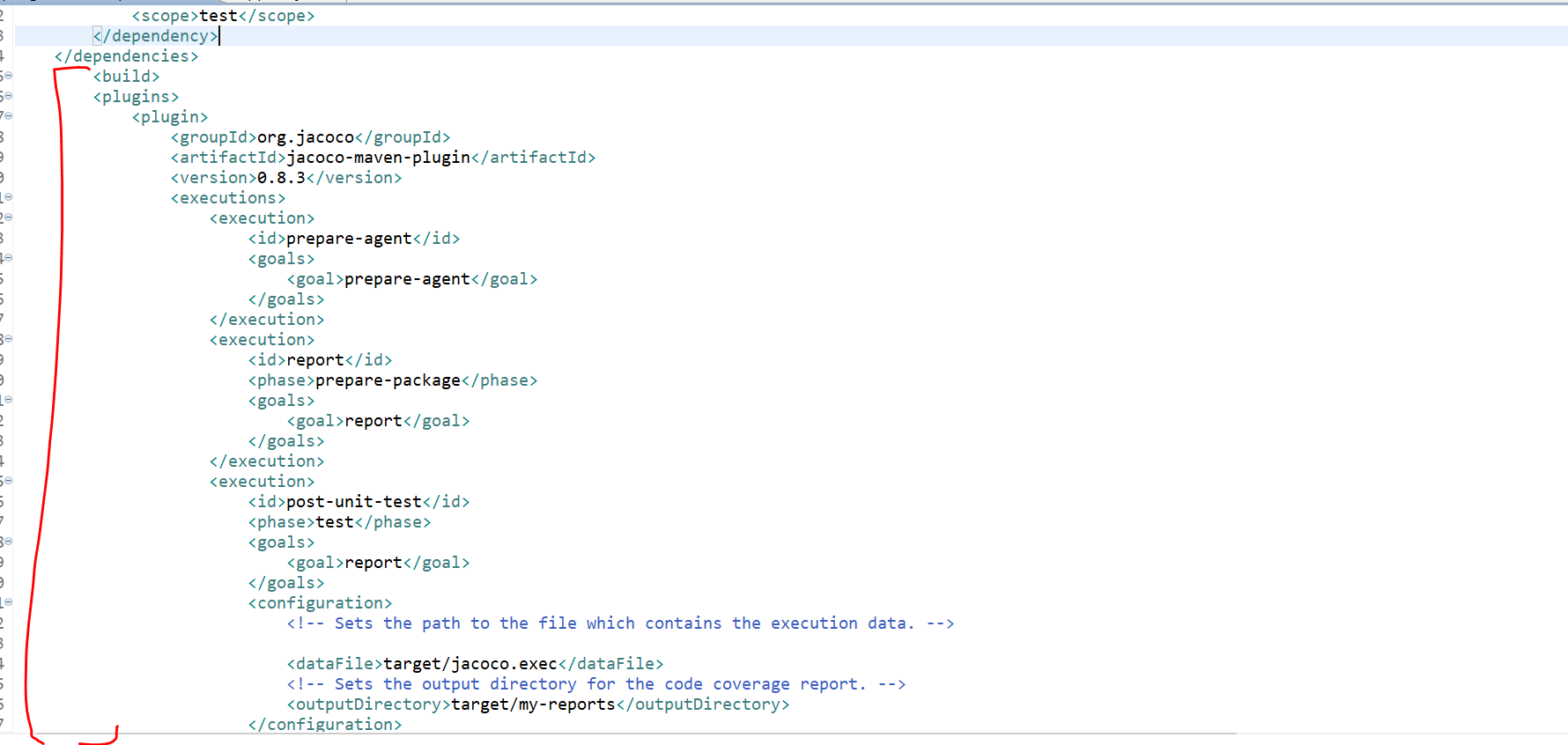
</systemPropertyVariables>

</configuration>

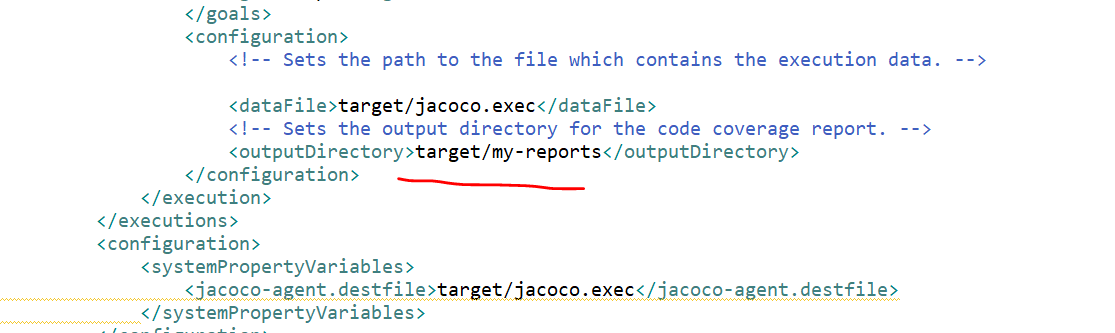
</plugin>

</plugins>

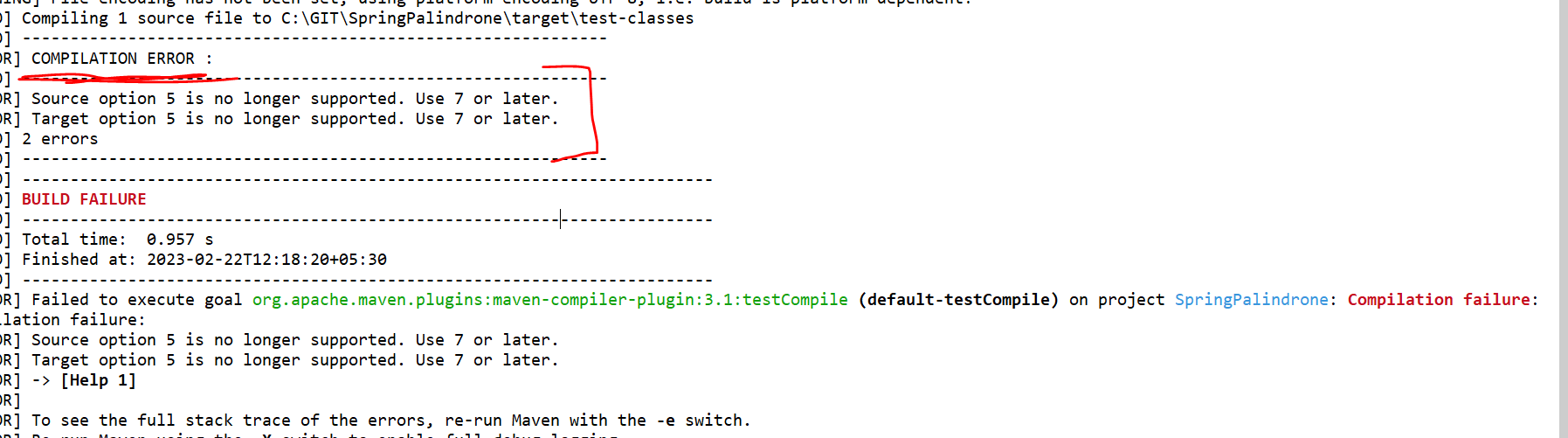
</build>



So, report will be generate in the target directory with my-reports folder.



Now let’s run our test case.



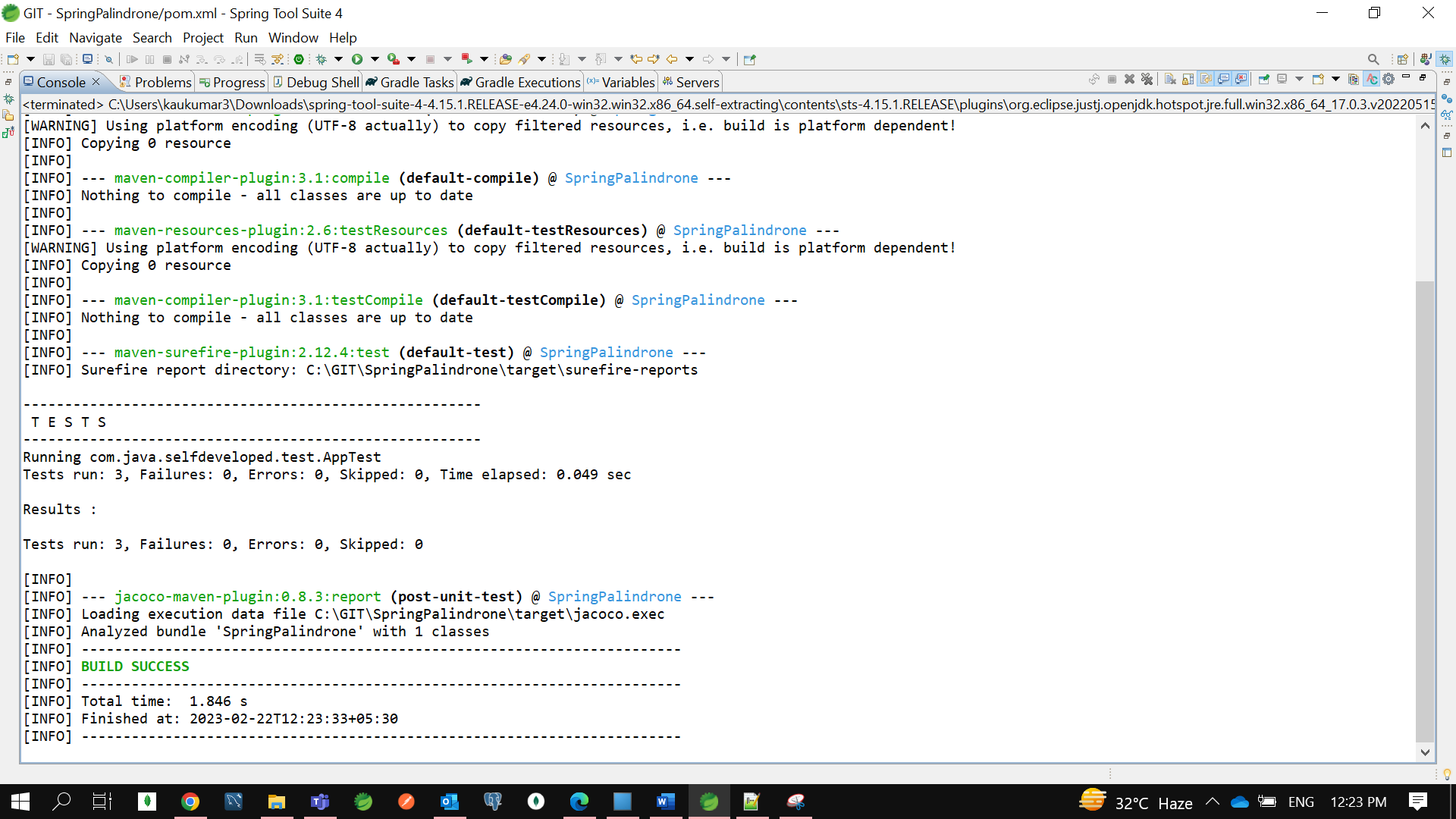
Just follow below steps-🡪

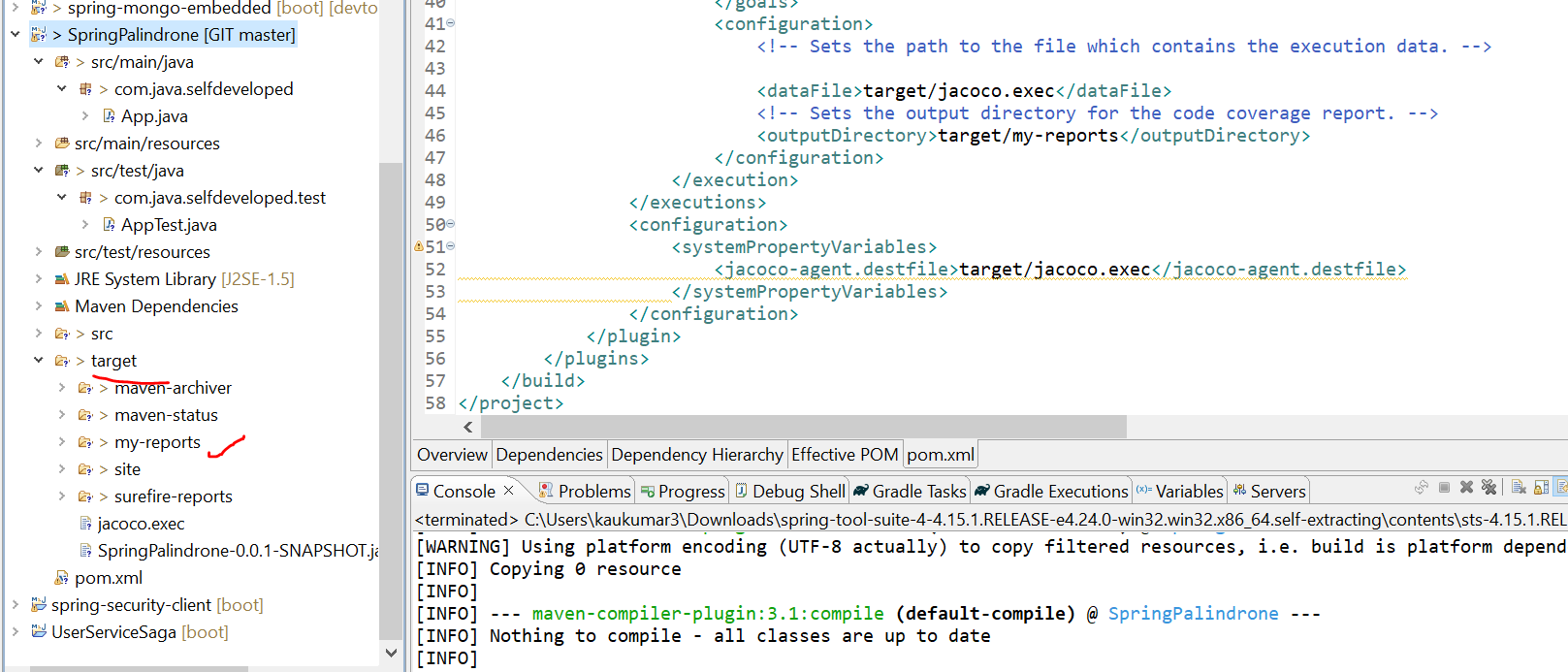
18

In Eclipse This helped me:

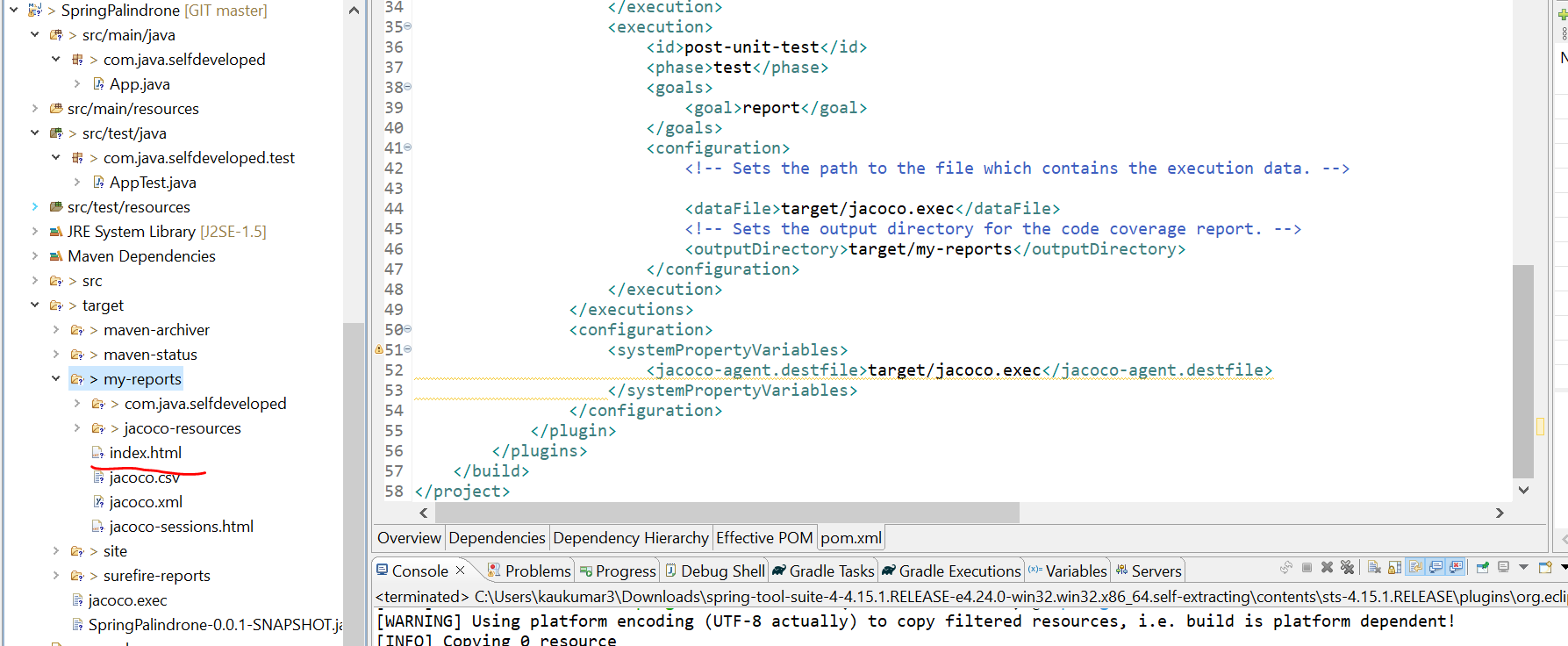
1. Right Click on Project.
2. Click on Build path.
3. Click on Configure Build path.
4. It opens a Java Build path window.
5. Click on Java Compiler in the Left side.
6. It navigates to Java Compiler window in that to set the Compiler compliance level is set as according to your jre version(ex if java version is 1.8 then choose 1.8) as select.
7. Click on [Apply] button.
8. Click on [OK] button.
9. Right click on Project > Maven > Update the project.
10. Right click on Project > Run As > Maven installs -- The pom.xml file is running and java jars are download and installed to project.
11. Right click on Project > Run As > Maven Test -- The pom.xml file is running, and java jars are download and installed to project.

Then you got the Build Success message, and your maven project is created successfully.

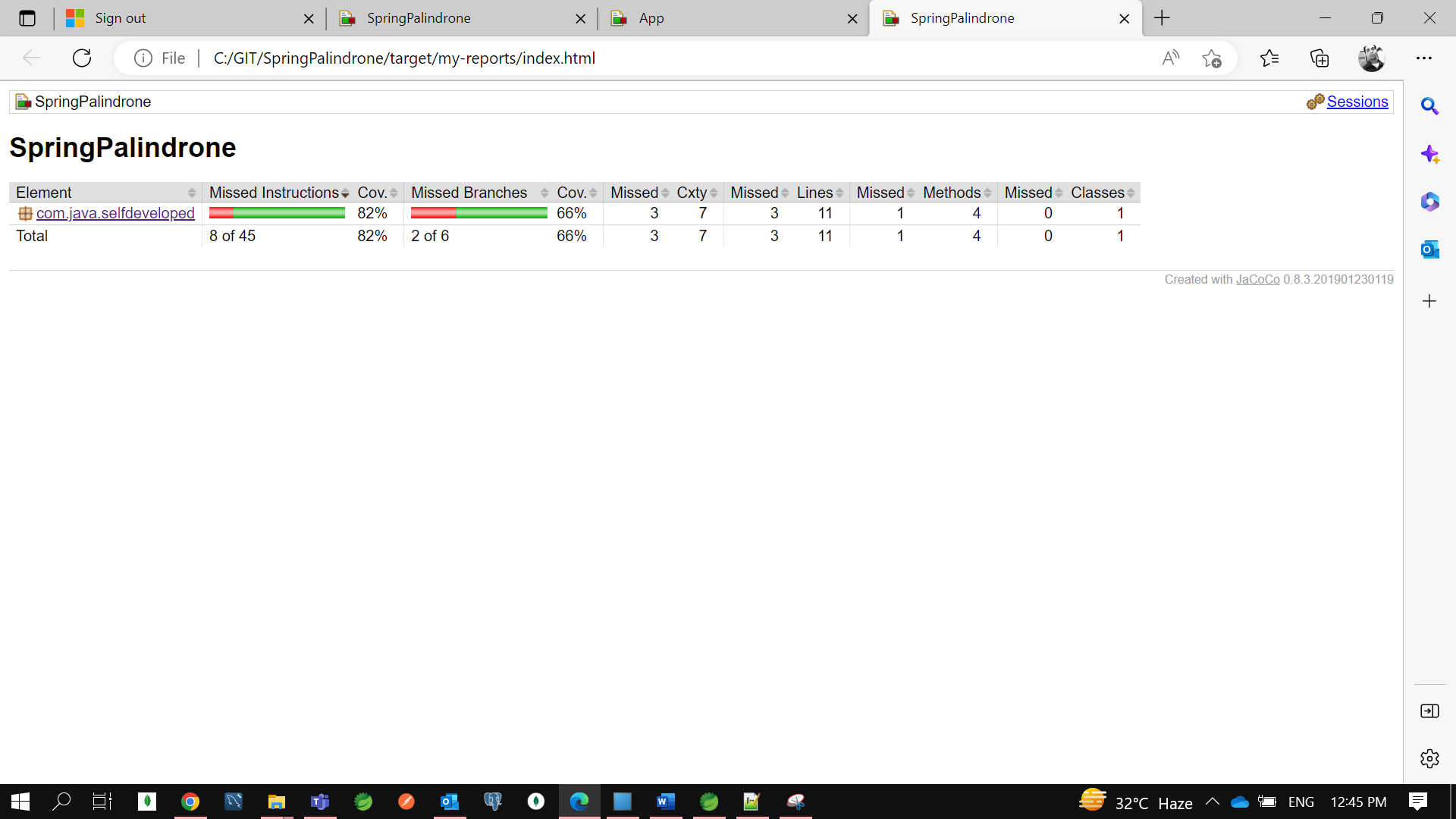


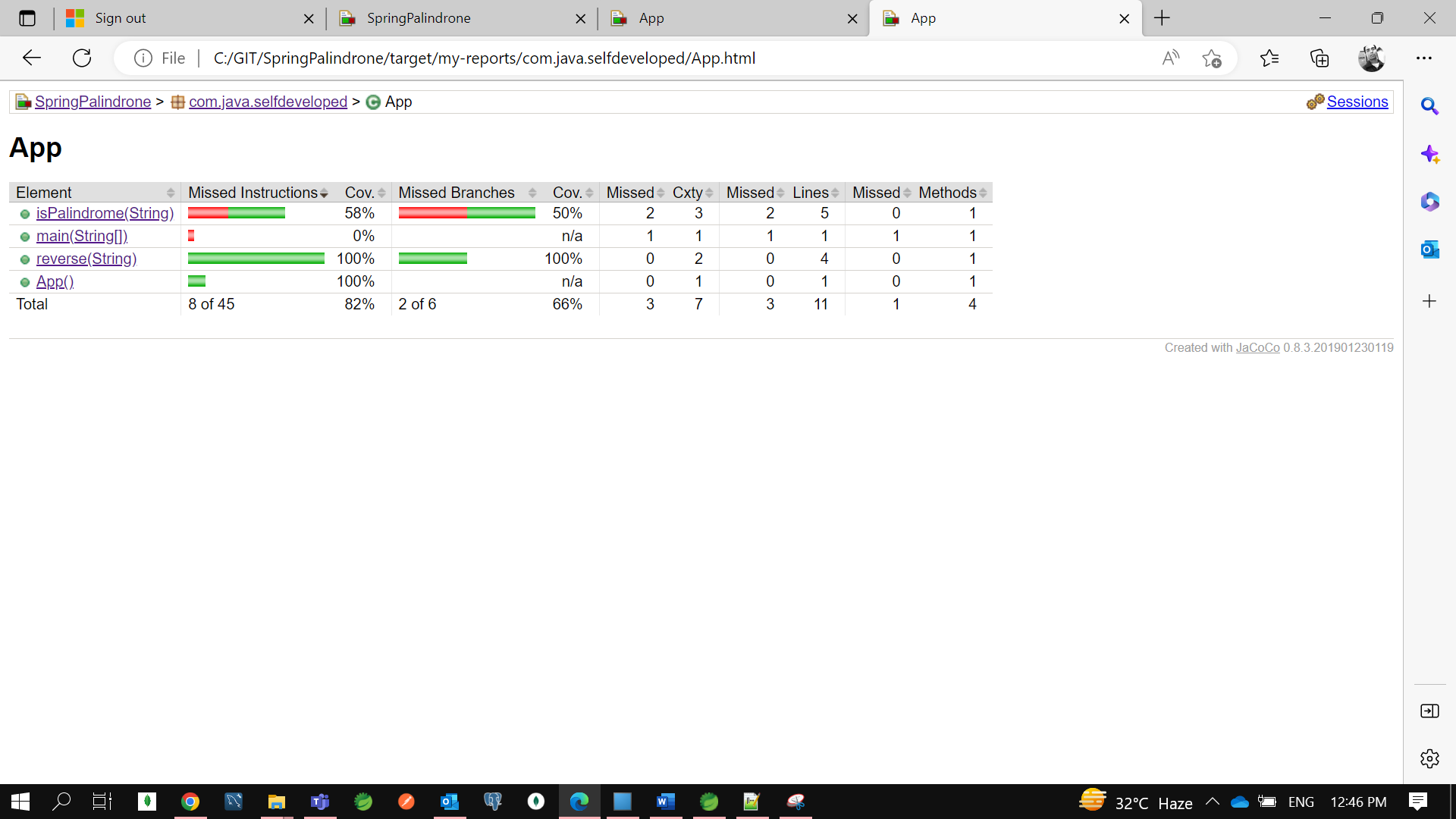


Now in target folder we can see my-reports folder is created which containing few of the documents and data.



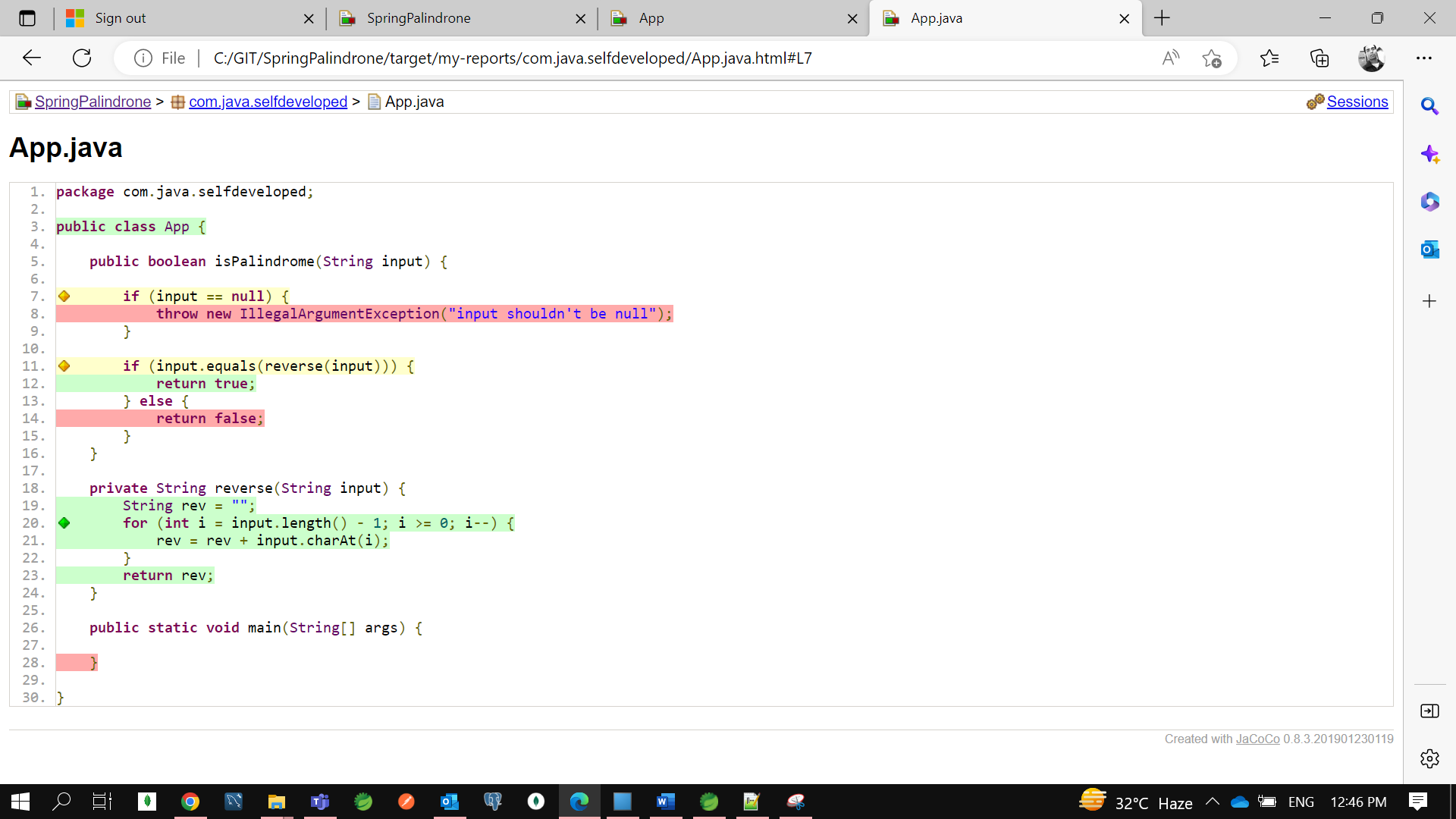
Click on index.html and open with web browser.





We can see 77% coverage is there but still 23% not covered.

So, let’s check which line is not covered…



So, we have a method called **is Palindrome**()

So, we forgot to remove this method psvm() which we are not using…

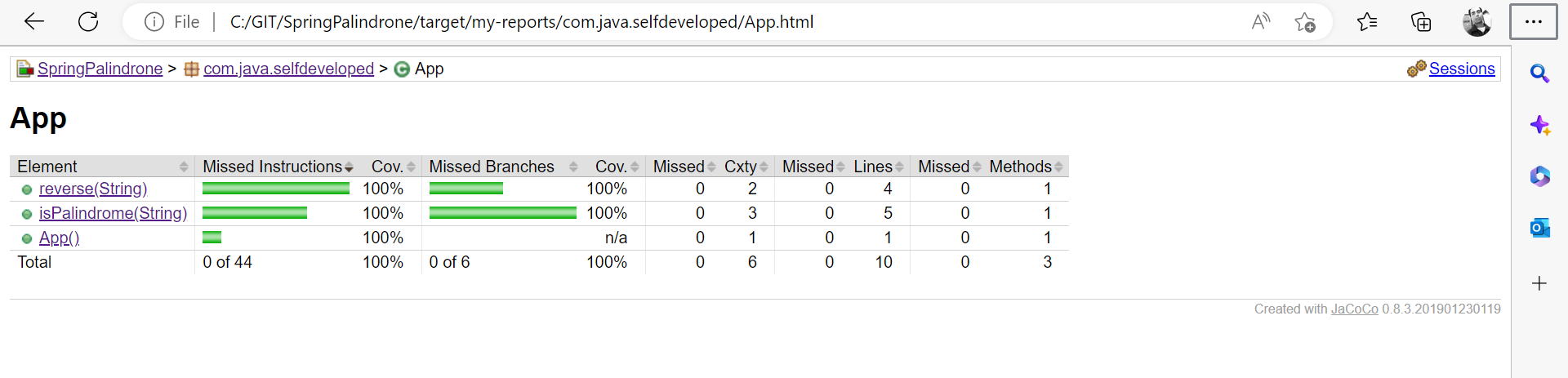
So, you can see null check we have not covered which is in red color, but you can see yellow color mark me partially we have covered.

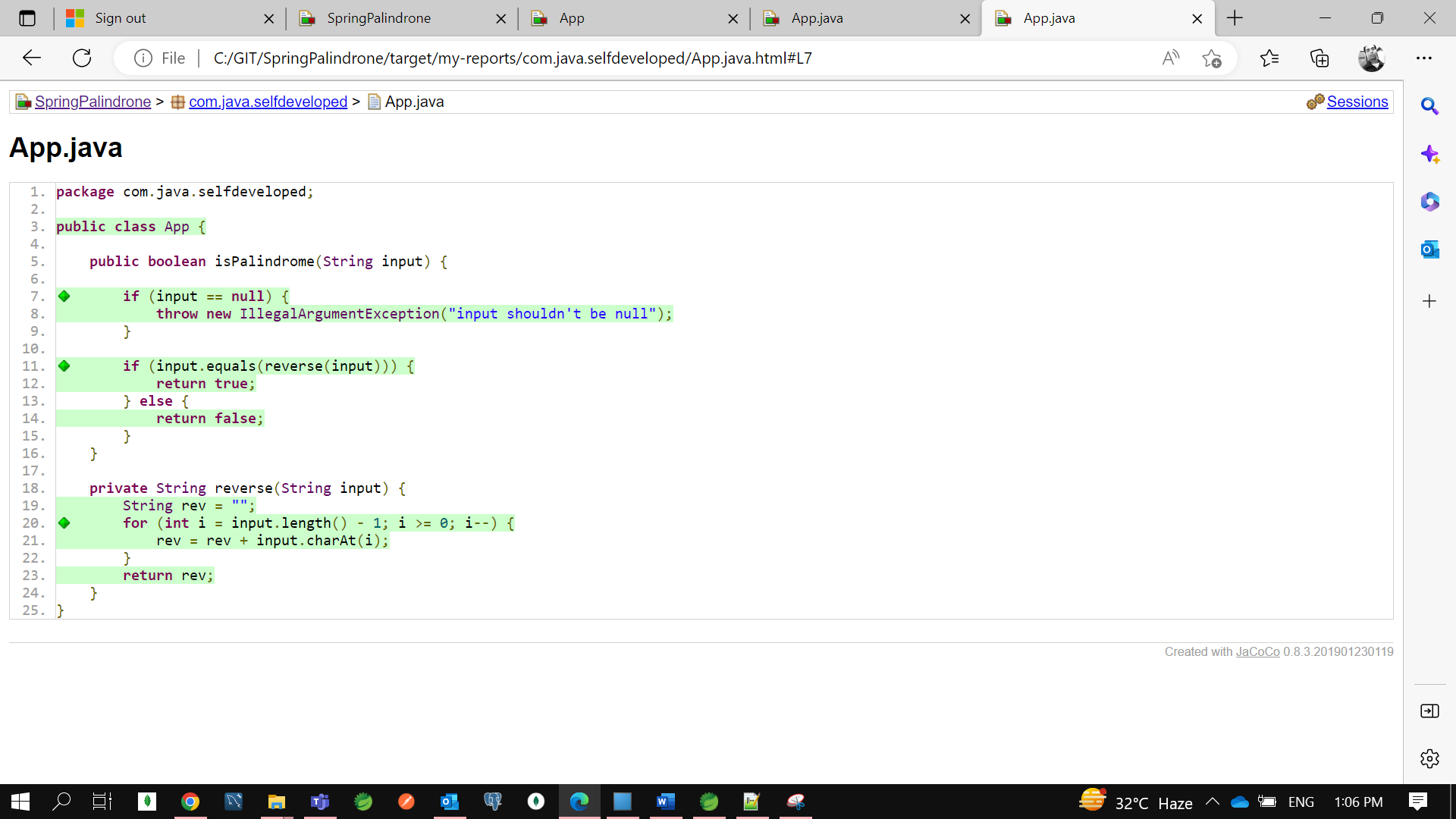
So, for each and every case we need to write a Test case . we need to write a test case for returning false and for Illegal Argument check as well.

So how to handle the exception in Testcase, we are not going to handle so, based on exception we want to execute the Test.

So, now let’s RunAS -> Maven Test







Now the coverage is 100%.now the actual mean for coverage is for each statement where your application will be break for that you need to write the testcase. So, green diamond means fully covered yellow diamond means partially covered and red diamond means not covered.