Lesson 01 Demo 02

Writing Test Cases with Various eCommerce Scenarios

Objective: To write unit test cases with various scenarios for e-commerce applications. Also, configure JUnit5 in the project

Tool required: Eclipse IDE

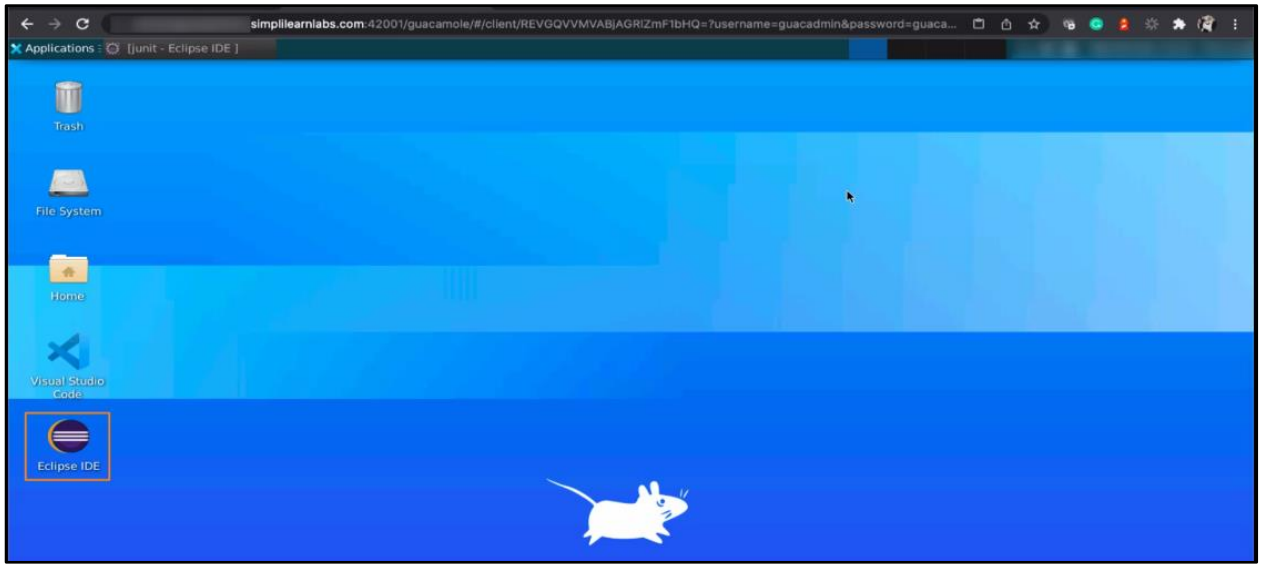
Prerequisites: None

Steps to be followed:

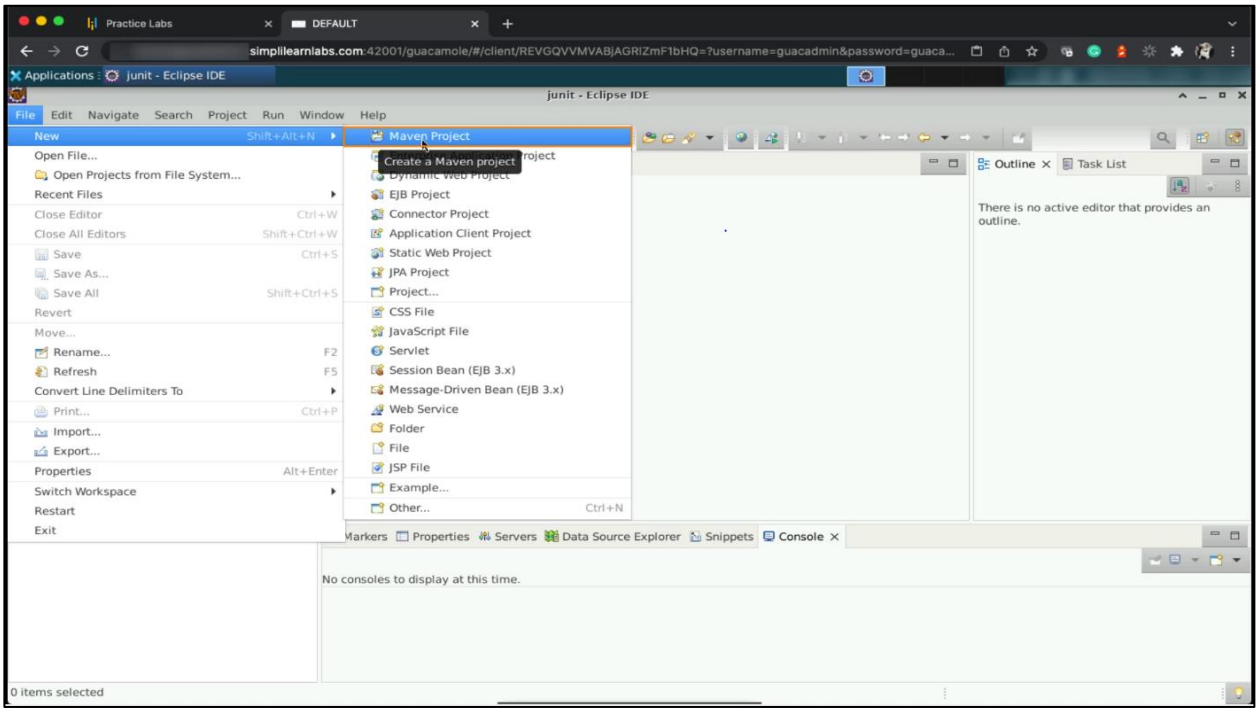
1. Creating a new Maven project
2. Creating an algorithmic method
3. Writing a test case to create an algorithmic method
4. Implementing JUnit5 and configuring the project with a newer version
5. Creating more test cases with output values
6. Packaging application

Step 1: Creating a new Maven project

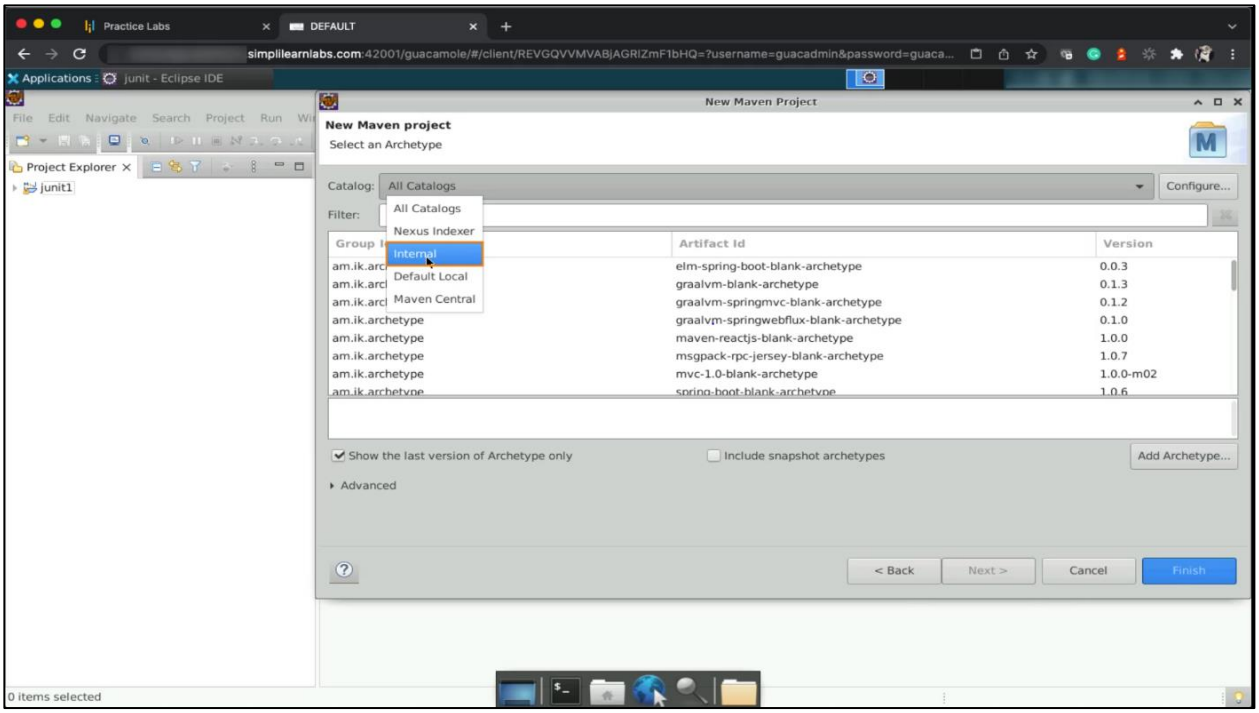
1.1 Open **Eclipse IDE**



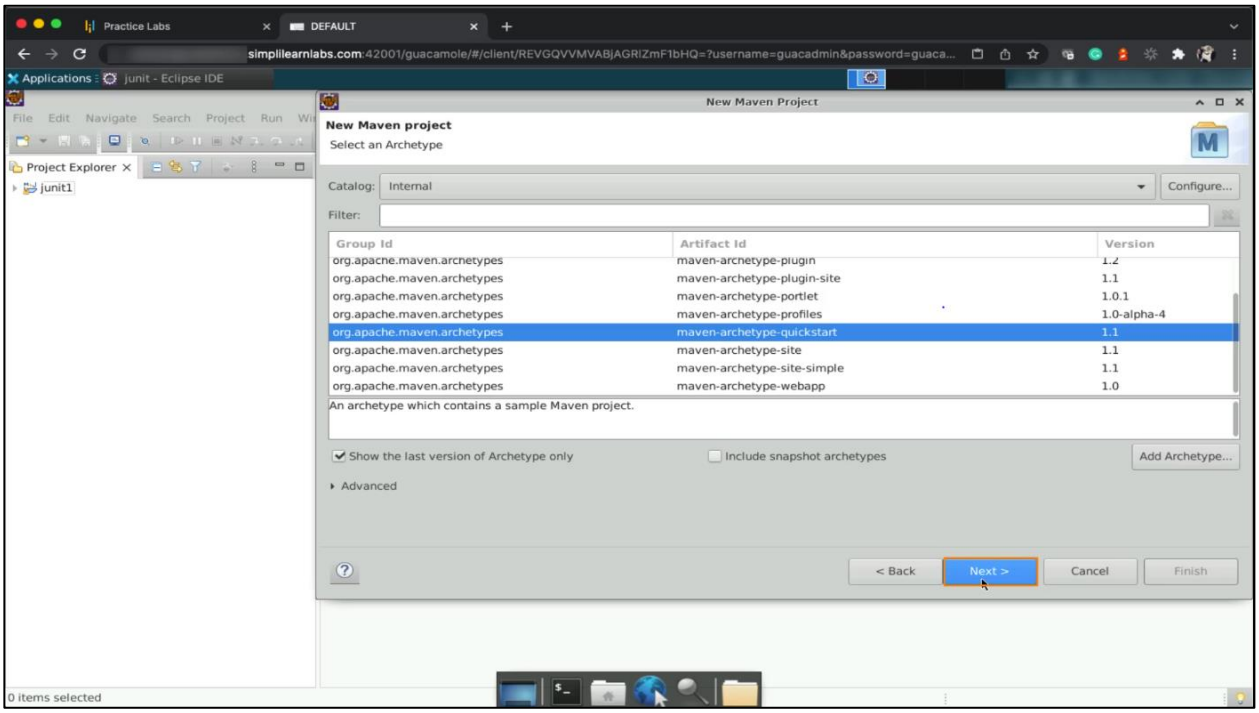
1.2 Go to **Files > New** and click **Maven Project** to create a new file



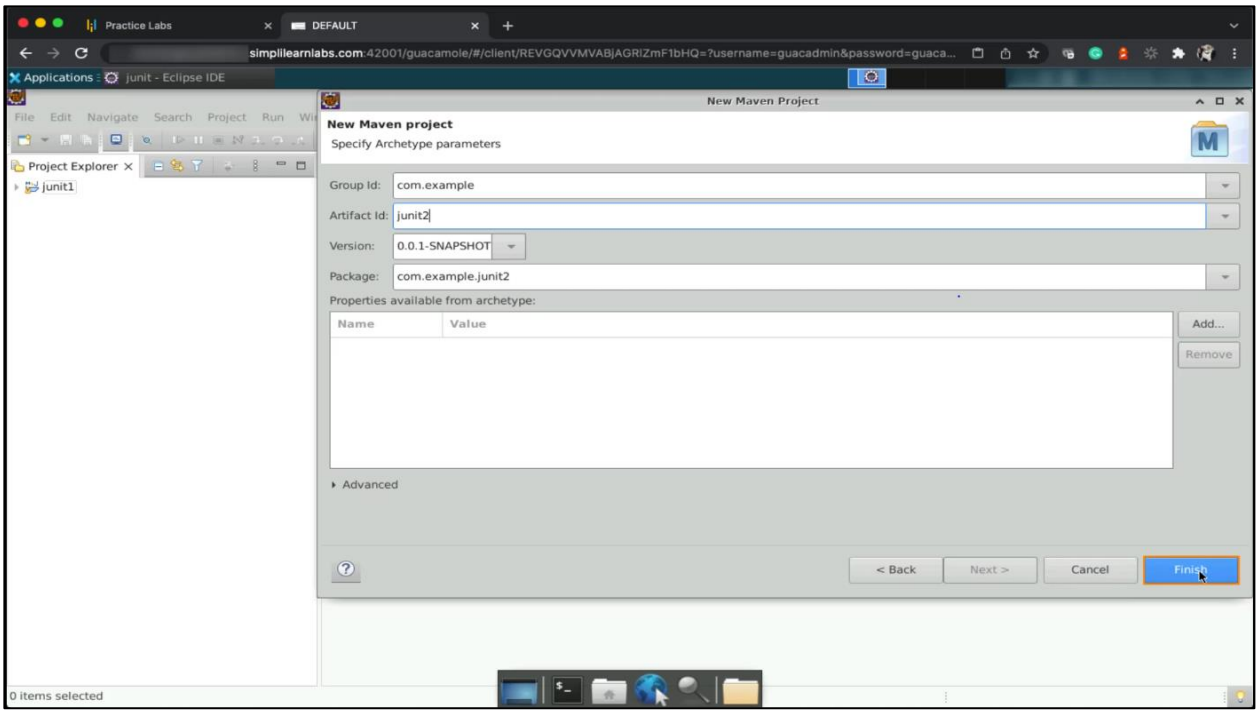
1.3 Select **Next** and then choose the catalog as **Internal**



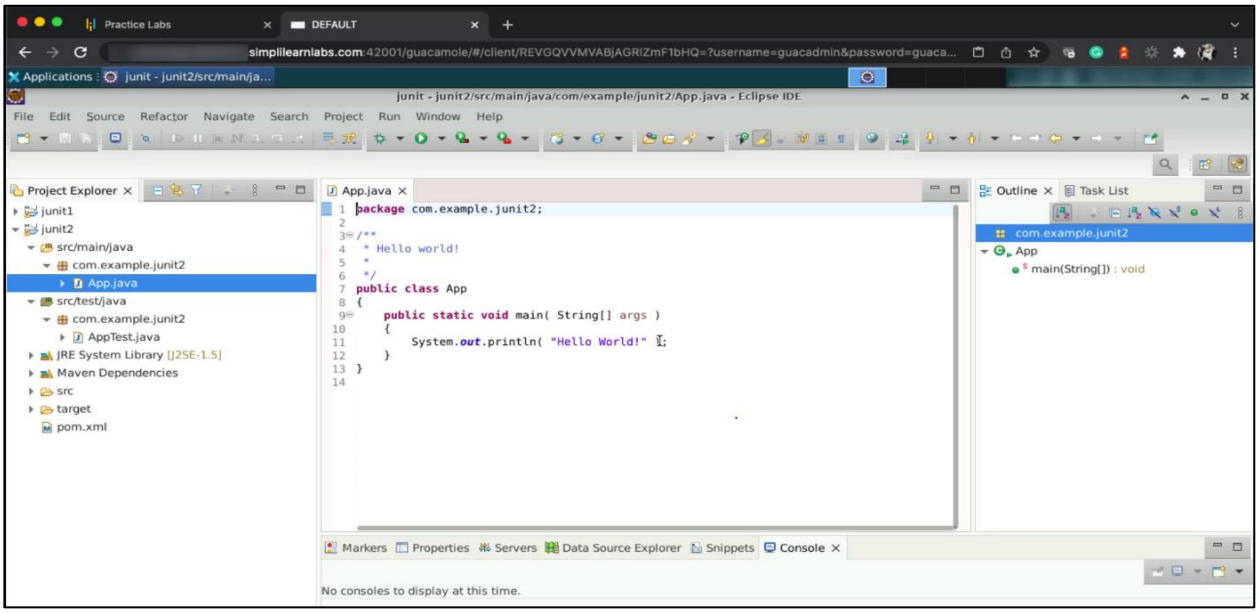
1.4 Select the **archetype** as **quickstart** and then click on **Next**



1.5 Name the **Group id** as **com.example** and **Artifact id** as **junit2.** Click **Finish**

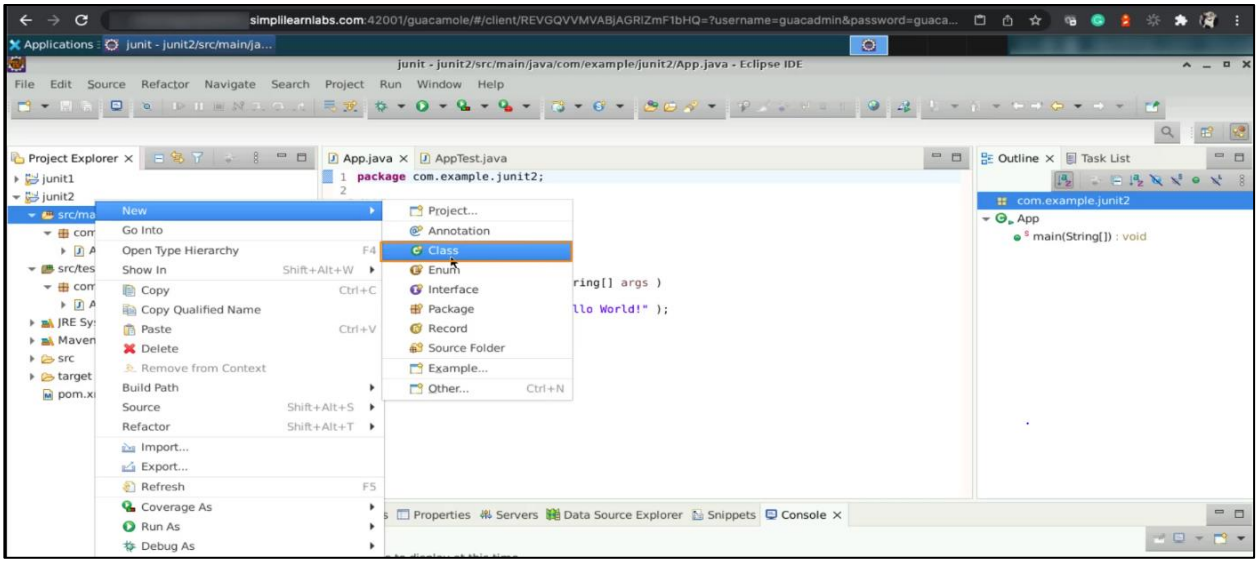


The junit2 maven project is created with all the basic files and configurations.

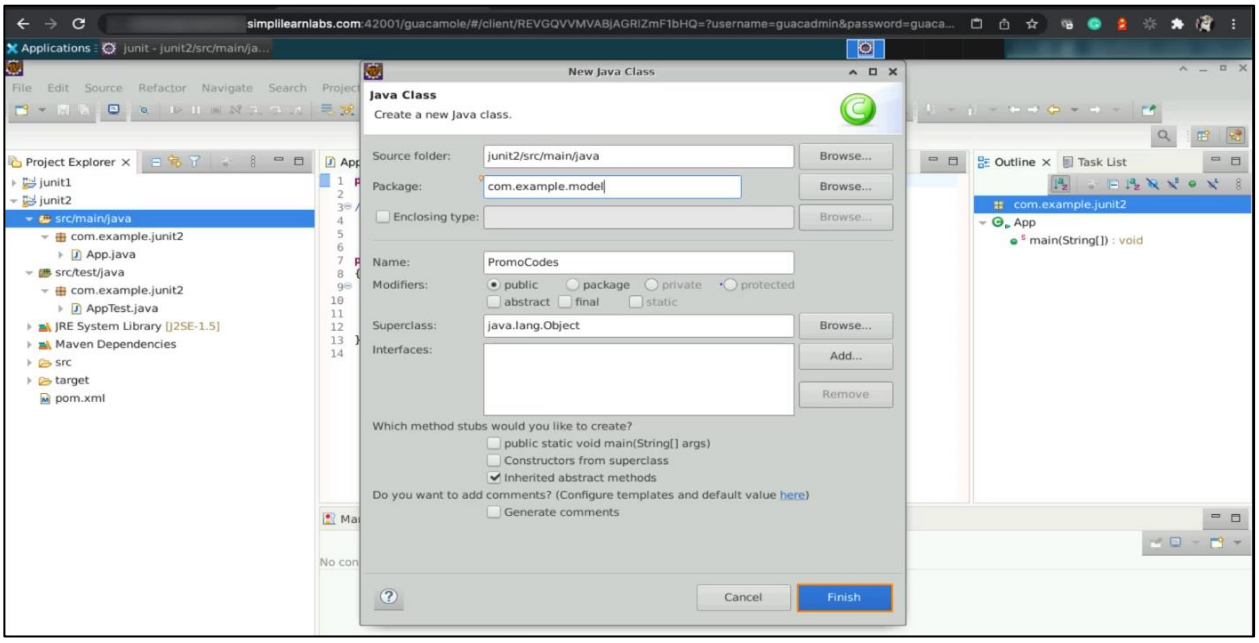


Step 2: Creating an algorithmic method

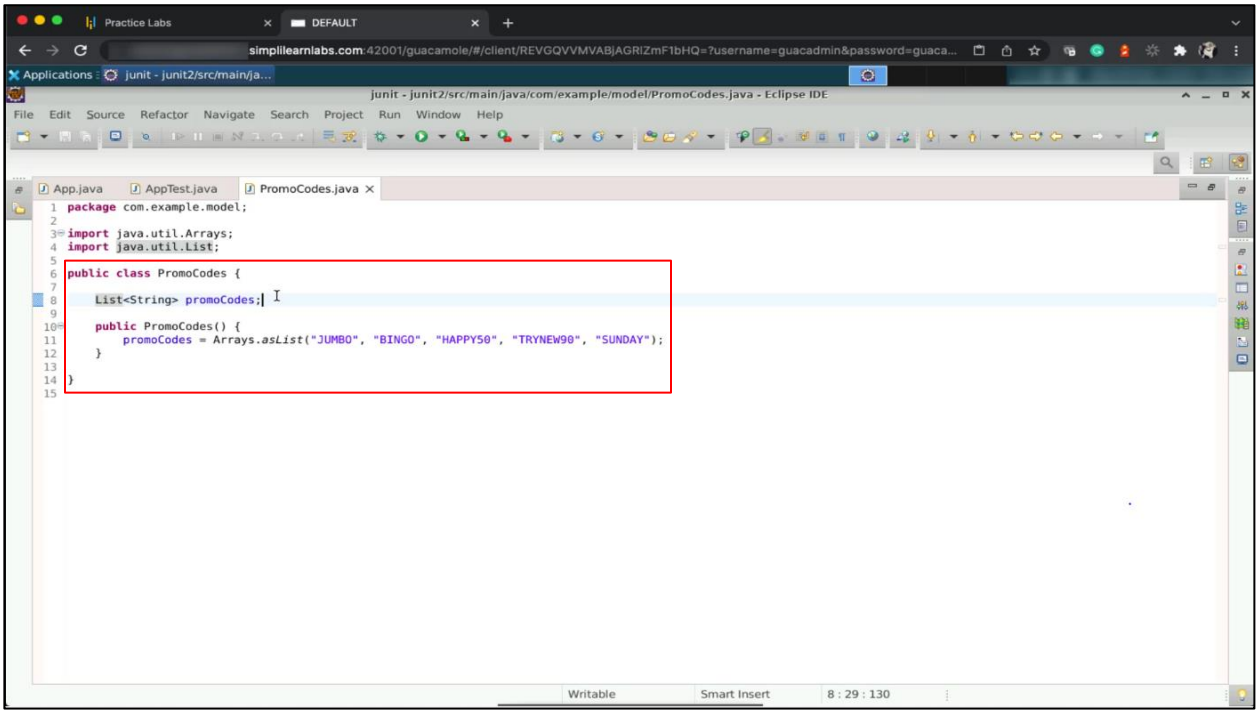
2.1 Go to **Project Explorer,** right-click on **src/main/java > New** and create a new **Class**



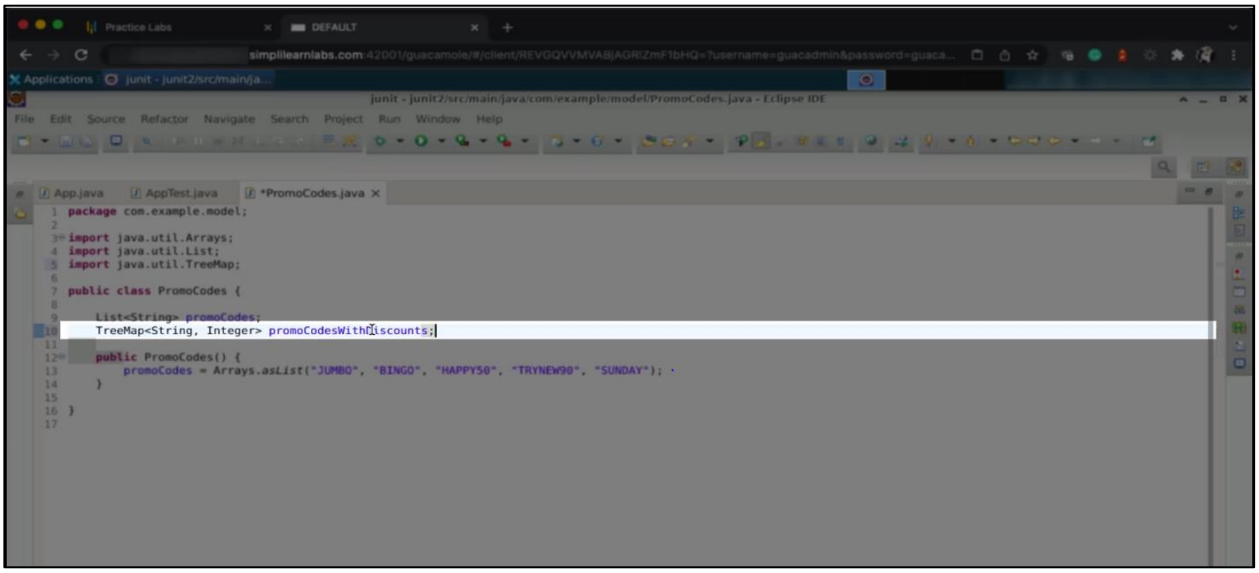
2.2 Name the class **PromoCodes.** Change package name to **com.example.model** and click **Finish**



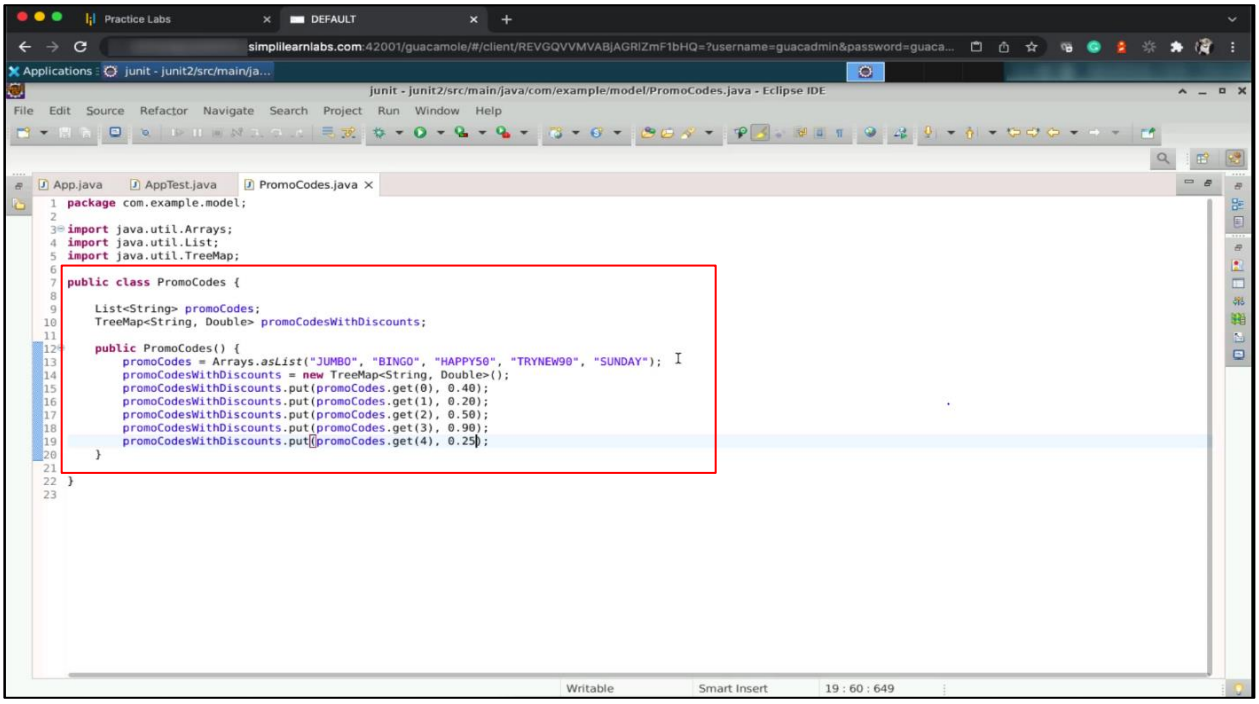
2.3 Now, go to the **PromoCodes.java** file. First, declare all the promo codes by creating an array list of type strings.



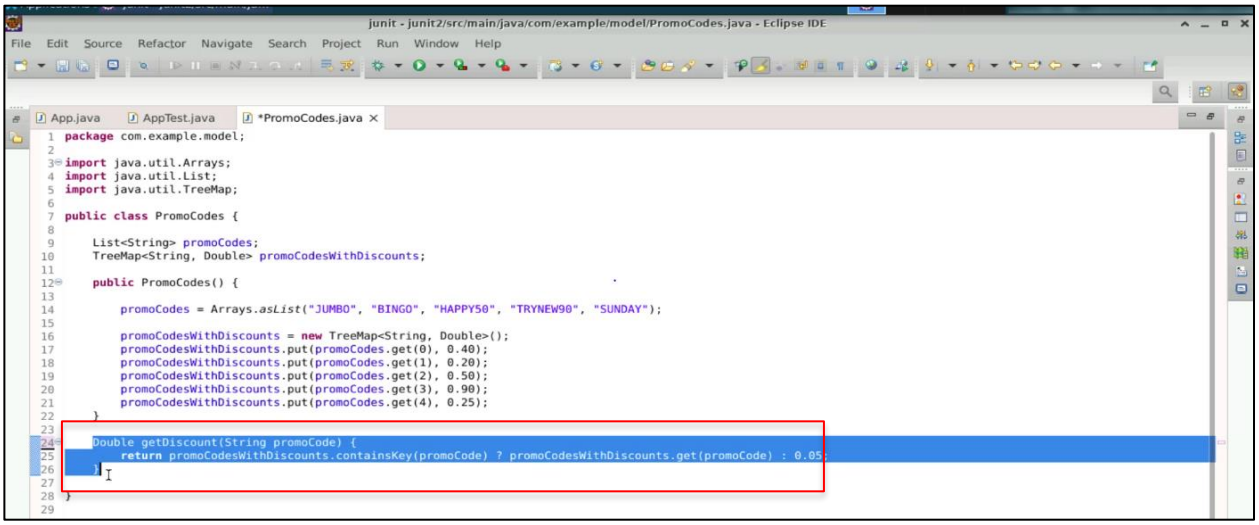
2.4 Create a **TreeMap** to determine which promo code is going to offer how much discount. You need to pass the **key-value** pair. Here, the key is **String,** and the value is **Integer**



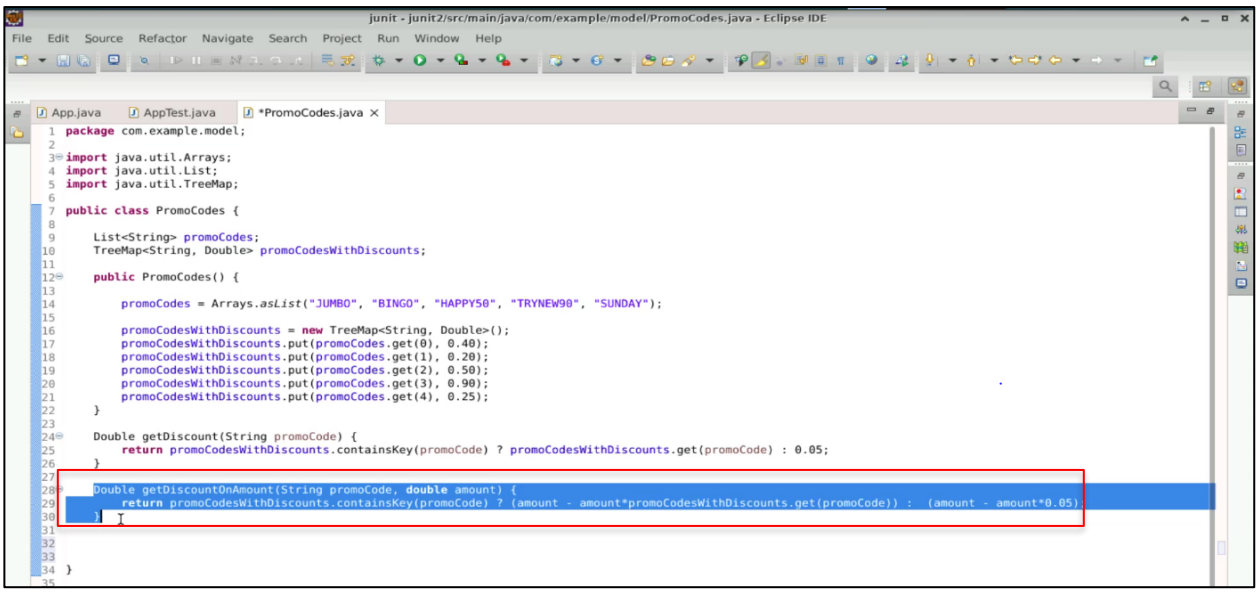
2.5 Create two different **data structures** to make it easier to work on and create all the **key-value** pairs of promo codes and discounts. Also, change **Integer** to **Double**



2.6 Create a method **getDiscount()** to return the discount value if the promo code exists, else return **5%** of discount



2.7 Create one more method **getDiscountOnAmount()** to get the final amount after the discount has been subtracted from the total amount, depending on the promo code. If the promo code does not exist, get the final amount after a 5% discount on the original amount



2.8 Create one more method **getDiscountOnAmountWithTaxes()** to get the final amount after the discount has been subtracted and **18% tax** added to the total amount

```java
package com.example.model;

import java.util.Arrays;
import java.util.List;
import java.util.TreeMap;

public class PromoCodes {

    List<String> promoCodes;
    TreeMap<String, Double> promoCodesWithDiscounts;

    public PromoCodes() {

        promoCodes = Arrays.asList("JUMBO", "BINGO", "HAPPY50", "TRYNEW90", "SUNDAY");

        promoCodesWithDiscounts = new TreeMap<String, Double>();
        promoCodesWithDiscounts.put(promoCodes.get(0), 0.40);
        promoCodesWithDiscounts.put(promoCodes.get(1), 0.20);
        promoCodesWithDiscounts.put(promoCodes.get(2), 0.50);
        promoCodesWithDiscounts.put(promoCodes.get(3), 0.90);
        promoCodesWithDiscounts.put(promoCodes.get(4), 0.25);
    }

    Double getDiscount(String promoCode) {
        return promoCodesWithDiscounts.containsKey(promoCode) ? promoCodesWithDiscounts.get(promoCode) : 0.05;
    }

    Double getDiscountOnAmount(String promoCode, double amount) {
        return promoCodesWithDiscounts.containsKey(promoCode) ? (amount - amount*promoCodesWithDiscounts.get(promoCode)) :  (amount - amount*0.05);
    }

    Double getDiscountOnAmountWithTaxes(String promoCode, double amount) {
        double amountAfterDiscount = getDiscountOnAmount(promoCode, amount);
        return (amountAfterDiscount + (0.18*amountAfterDiscount));
    }
```

2.9 For better understanding, change the method name **getDiscountOnAmount()** to **getAmountAfterDiscount()** and **getDiscountOnAmountWithTaxes()** to **getAmountAfterDiscountWithTaxes()**. Also, make necessary name changes in the code

```java
import java.util.Arrays;
import java.util.List;
import java.util.TreeMap;

public class PromoCodes {

    List<String> promoCodes;
    TreeMap<String, Double> promoCodesWithDiscounts;

    public PromoCodes() {

        promoCodes = Arrays.asList("JUMBO", "BINGO", "HAPPY50", "TRYNEW90", "SUNDAY");

        promoCodesWithDiscounts = new TreeMap<String, Double>();
        promoCodesWithDiscounts.put(promoCodes.get(0), 0.40);
        promoCodesWithDiscounts.put(promoCodes.get(1), 0.20);
        promoCodesWithDiscounts.put(promoCodes.get(2), 0.50);
        promoCodesWithDiscounts.put(promoCodes.get(3), 0.90);
        promoCodesWithDiscounts.put(promoCodes.get(4), 0.25);
    }

    Double getDiscount(String promoCode) {
        return promoCodesWithDiscounts.containsKey(promoCode) ? promoCodesWithDiscounts.get(promoCode) : 0.05;
    }

    Double getAmountAfterDiscount(String promoCode, double amount) {
        return promoCodesWithDiscounts.containsKey(promoCode) ? (amount - amount*promoCodesWithDiscounts.get(promoCode)) :  (amount - amount*0.05);
    }

    Double getAmountAfterDiscountWithTaxes(String promoCode, double amount) {
        double amountAfterDiscount = getAmountAfterDiscount(promoCode, amount);
        return (amountAfterDiscount + (0.18*amountAfterDiscount));
    }

}
```

2.10 Now, create a method **checkPromoCode()** which will return a **Boolean value**. If a promo code exists, this method returns **true**, if not, it returns **false**

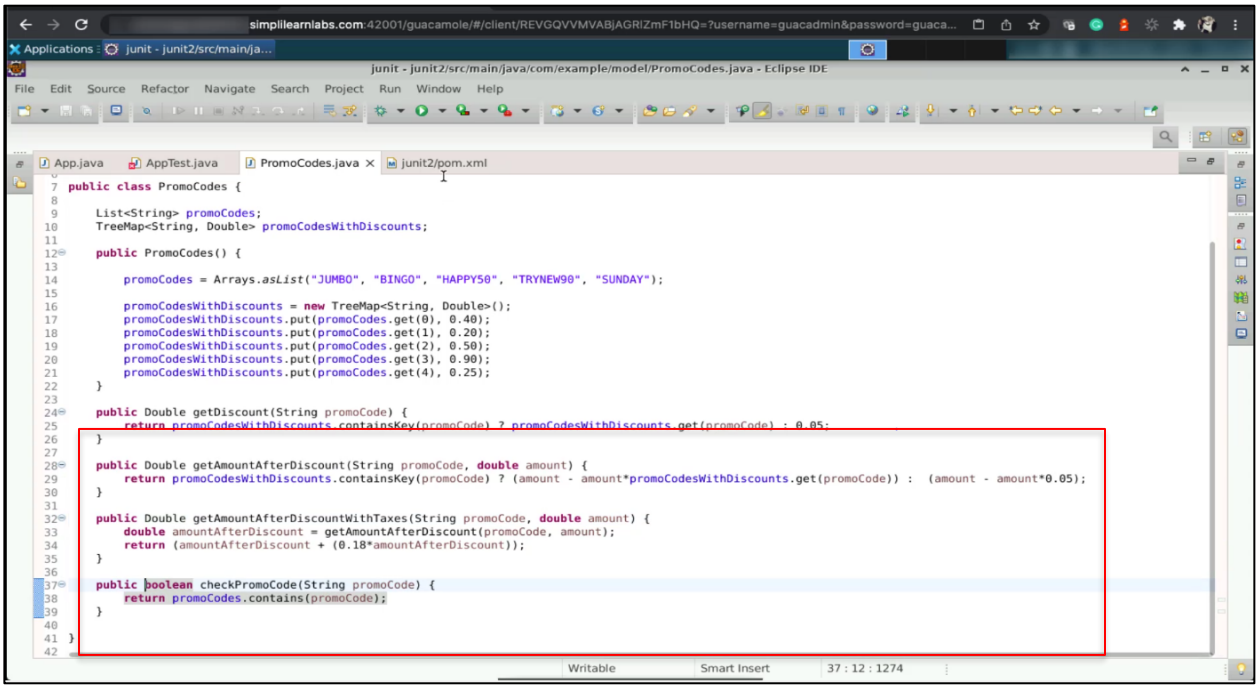
```java
public class PromoCodes {

    List<String> promoCodes;
    TreeMap<String, Double> promoCodesWithDiscounts;

    public PromoCodes() {

        promoCodes = Arrays.asList("JUMBO", "BINGO", "HAPPY50", "TRYNEW90", "SUNDAY");

        promoCodesWithDiscounts = new TreeMap<String, Double>();
        promoCodesWithDiscounts.put(promoCodes.get(0), 0.40);
        promoCodesWithDiscounts.put(promoCodes.get(1), 0.20);
        promoCodesWithDiscounts.put(promoCodes.get(2), 0.50);
        promoCodesWithDiscounts.put(promoCodes.get(3), 0.90);
        promoCodesWithDiscounts.put(promoCodes.get(4), 0.25);
    }

    Double getDiscount(String promoCode) {
        return promoCodesWithDiscounts.containsKey(promoCode) ? promoCodesWithDiscounts.get(promoCode) : 0.05;
    }

    Double getAmountAfterDiscount(String promoCode, double amount) {
        return promoCodesWithDiscounts.containsKey(promoCode) ? (amount - amount*promoCodesWithDiscounts.get(promoCode)) :  (amount - amount*0.05);
    }

    Double getAmountAfterDiscountWithTaxes(String promoCode, double amount) {
        double amountAfterDiscount = getAmountAfterDiscount(promoCode, amount);
        return (amountAfterDiscount + (0.18*amountAfterDiscount));
    }

    boolean checkPromoCode(String promoCode) {
        return promoCodes.contains(promoCode);
    }

}
```

Step 3: Writing a test case to create an algorithmic method

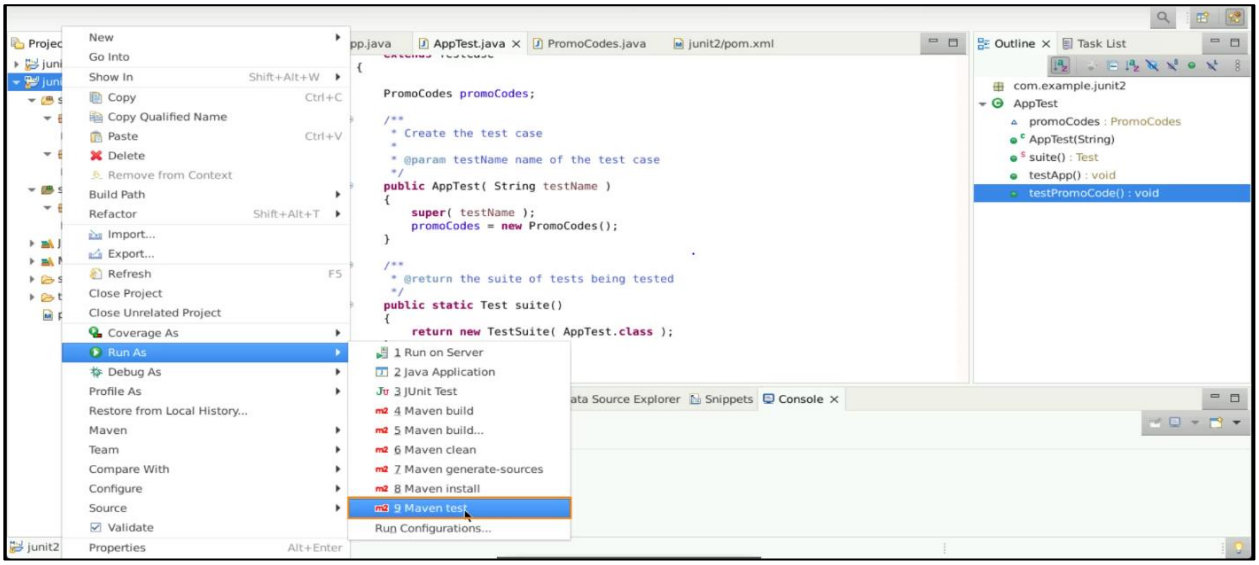
3.1 Go to the **AppTest.java** file. Create a new test case **testPromoCode().** Inside this test case, create the **assertEquals()** assertion method to compare the expected and actual output of methods. To get the actual value, create a **promoCodes** object

```java
{

    PromoCodes promoCodes;

    /**
     * Create the test case
     *
     * @param testName name of the test case
     */
    public AppTest( String testName )
    {
        super( testName );
        promoCodes = new PromoCodes();
    }

    /**
     * @return the suite of tests being tested
     */
    public static Test suite()
    {
        return new TestSuite( AppTest.class );
    }

    /**
     * Rigourous Test :-)
     */
    public void testApp()
    {
        assertTrue( true );
    }

    public void testPromoCode() {
        assertEquals(true, promoCodes.checkPromoCode("bingo"));
    }
}
```

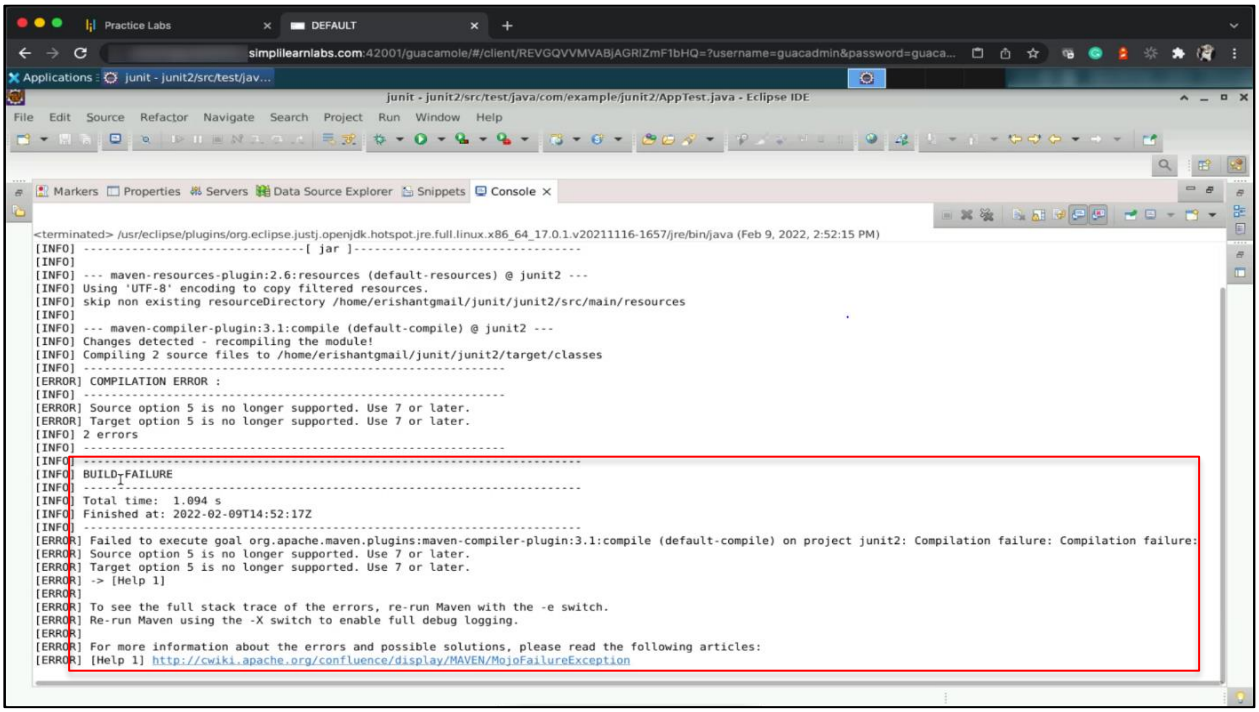
3.2 Go to the **PromoCodes.java** file and declare all methods **public** to access them outside the package



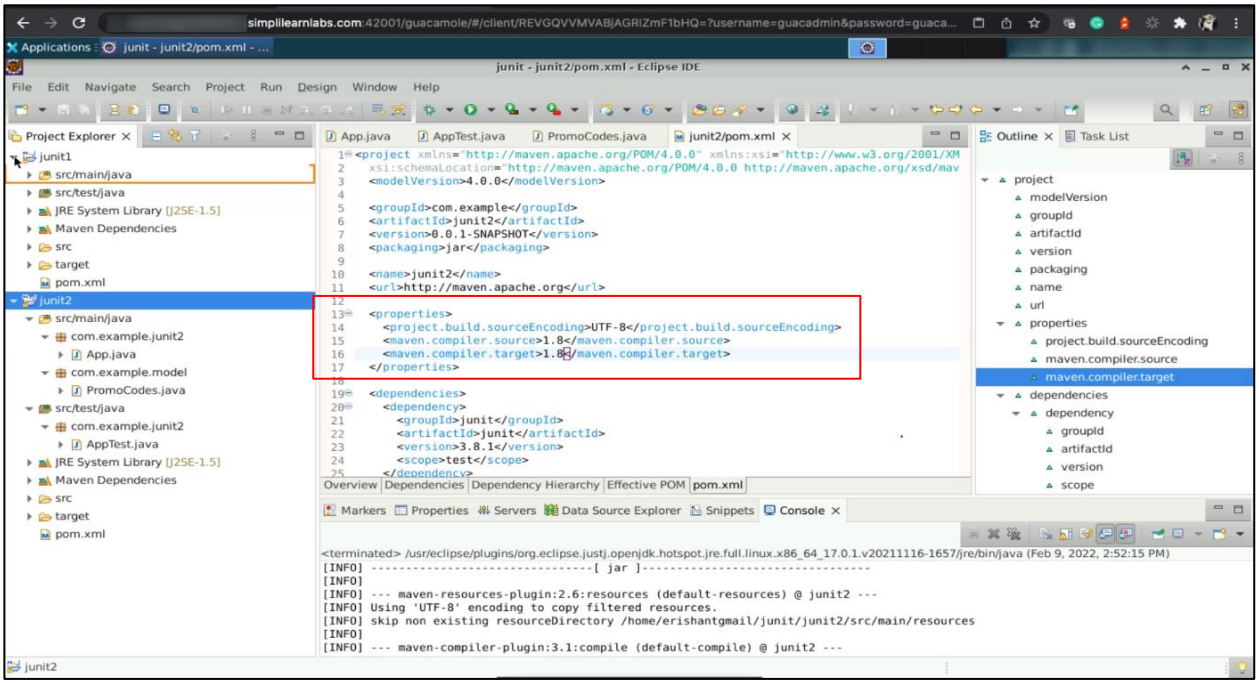
3.3 Now, right-click on the project name **junit2**. Select **Run As** and run the project **Maven test**



In the console, there is a **COMPILATION ERROR** because of the source and target version.



3.4 Go to the **pom.xml** file and set the Maven compiler properties for the source and target



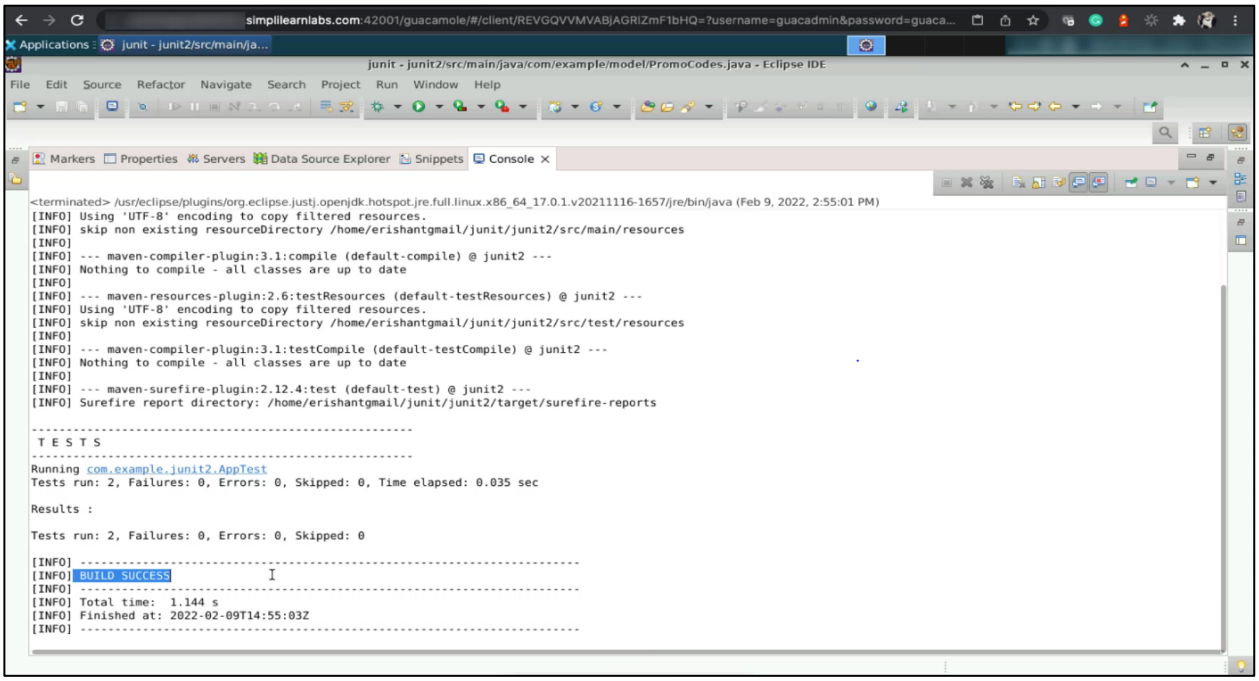
3.5 Rerun the project as a Maven test. In the console output, two test cases are running. One test case will fail, that is **testPromoCode()**, where the output is false but the expected output is true.

```
<terminated> /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211116-1657/jre/bin/java (Feb 9, 2022, 2:53:01 PM)
Running com.example.junit2.AppTest
Tests run: 2, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 0.036 sec <<< FAILURE!
testPromoCode(com.example.junit2.AppTest)  Time elapsed: 0.006 sec  <<< FAILURE!
junit.framework.AssertionFailedError: expected:<true> but was:<false>
        at junit.framework.Assert.fail(Assert.java:47)
        at junit.framework.Assert.failNotEquals(Assert.java:282)
        at junit.framework.Assert.assertEquals(Assert.java:64)
        at junit.framework.Assert.assertEquals(Assert.java:149)
        at junit.framework.Assert.assertEquals(Assert.java:155)
        at com.example.junit2.AppTest.testPromoCode(AppTest.java:46)


Results :

Failed tests:   testPromoCode(com.example.junit2.AppTest): expected:<true> but was:<false>

Tests run: 2, Failures: 1, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.729 s
[INFO] Finished at: 2022-02-09T14:53:04Z
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:2.12.4:test (default-test) on project junit2: There are test failures.
[ERROR]
[ERROR] Please refer to /home/erishantgmail/junit/junit2/target/surefire-reports for the individual test results.
[ERROR] -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException
```

3.6 Go back to the **PromoCodes.java** file and update the return statement with **promoCodes.toUpperCase()** to convert any other input to uppercase because all the promo codes are in uppercase

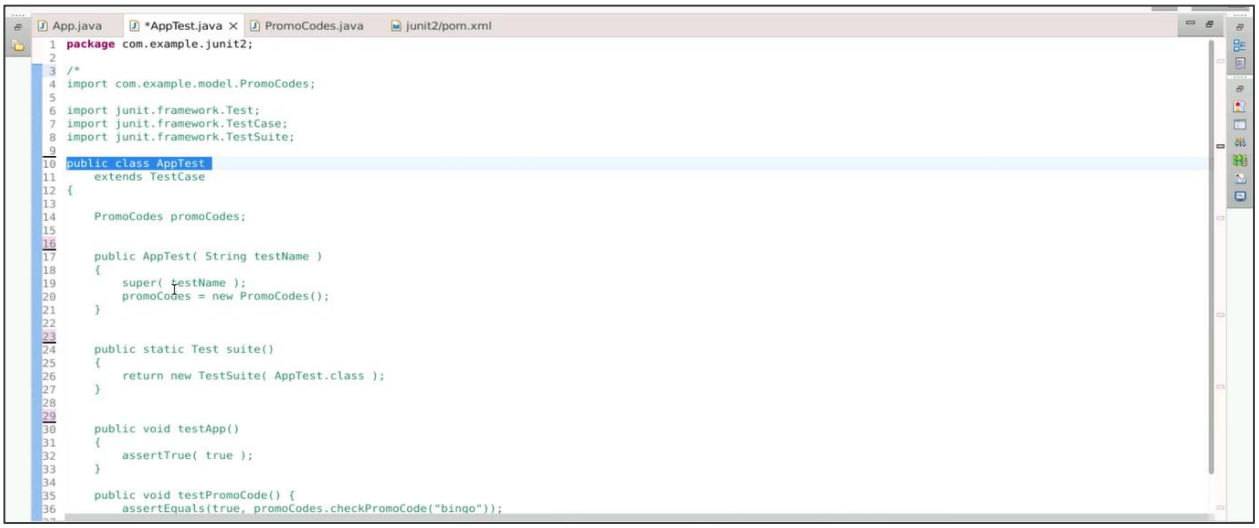
```
public class PromoCodes {

    List<String> promoCodes;
    TreeMap<String, Double> promoCodesWithDiscounts;

    public PromoCodes() {

        promoCodes = Arrays.asList("JUMBO", "BINGO", "HAPPY50", "TRYNEW90", "SUNDAY");

        promoCodesWithDiscounts = new TreeMap<String, Double>();
        promoCodesWithDiscounts.put(promoCodes.get(0), 0.40);
        promoCodesWithDiscounts.put(promoCodes.get(1), 0.20);
        promoCodesWithDiscounts.put(promoCodes.get(2), 0.50);
        promoCodesWithDiscounts.put(promoCodes.get(3), 0.90);
        promoCodesWithDiscounts.put(promoCodes.get(4), 0.25);
    }

    public Double getDiscount(String promoCode) {
        return promoCodesWithDiscounts.containsKey(promoCode) ? promoCodesWithDiscounts.get(promoCode) : 0.05;
    }

    public Double getAmountAfterDiscount(String promoCode, double amount) {
        return promoCodesWithDiscounts.containsKey(promoCode) ? (amount - amount*promoCodesWithDiscounts.get(promoCode)) :  (amount - amount*0.05);
    }

    public Double getAmountAfterDiscountWithTaxes(String promoCode, double amount) {
        double amountAfterDiscount = getAmountAfterDiscount(promoCode, amount);
        return (amountAfterDiscount + (0.18*amountAfterDiscount));
    }

    public boolean checkPromoCode(String promoCode) {
        return promoCodes.contains(promoCode.toUpperCase());
    }
```

3.7 Rerun the project as a **Maven test.** You will see both test cases are running, and the test cases are **BUILD SUCCESS**

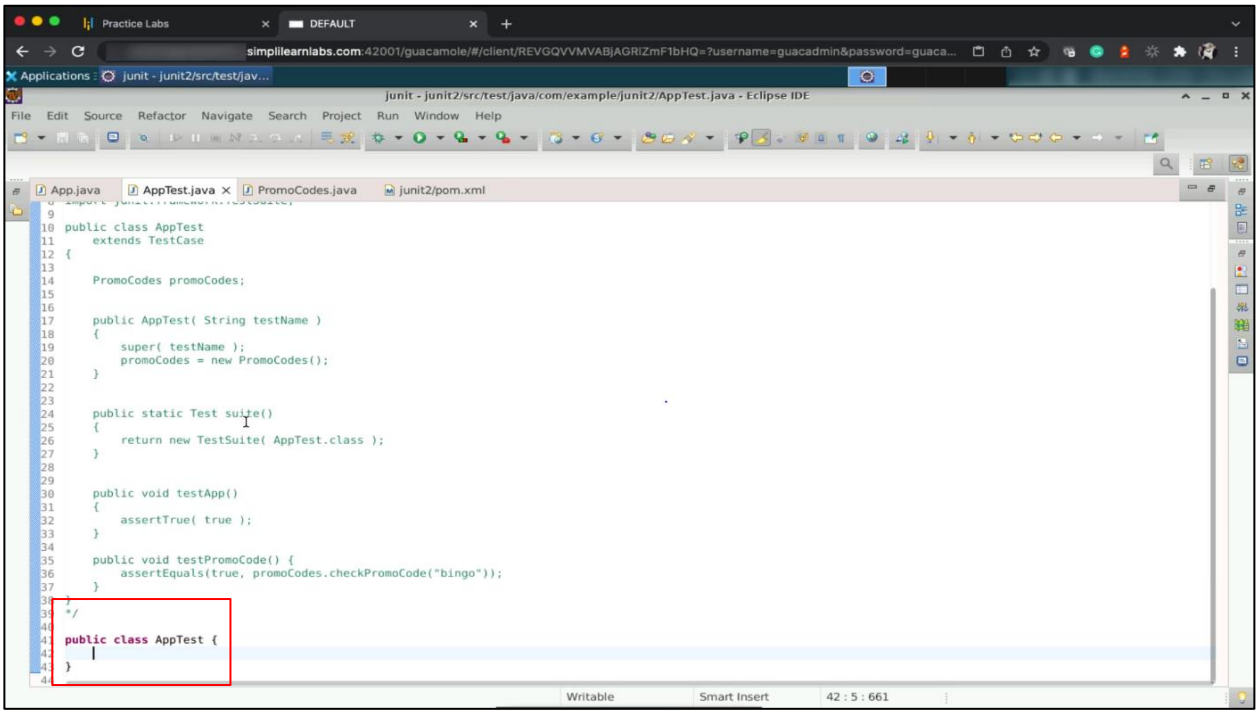


Step 4: Implementing JUnit5 and configuring the project with a newer version

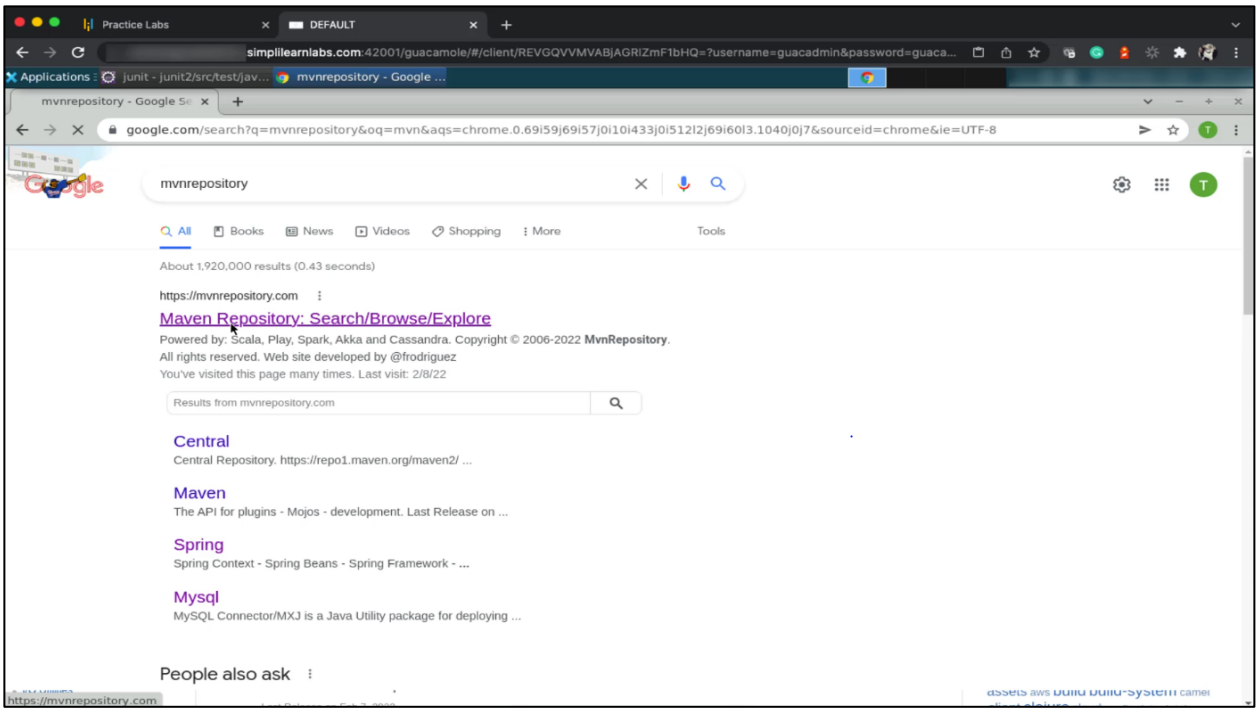
4.1 Go to the **AppTest.java** file and comment on the entire code



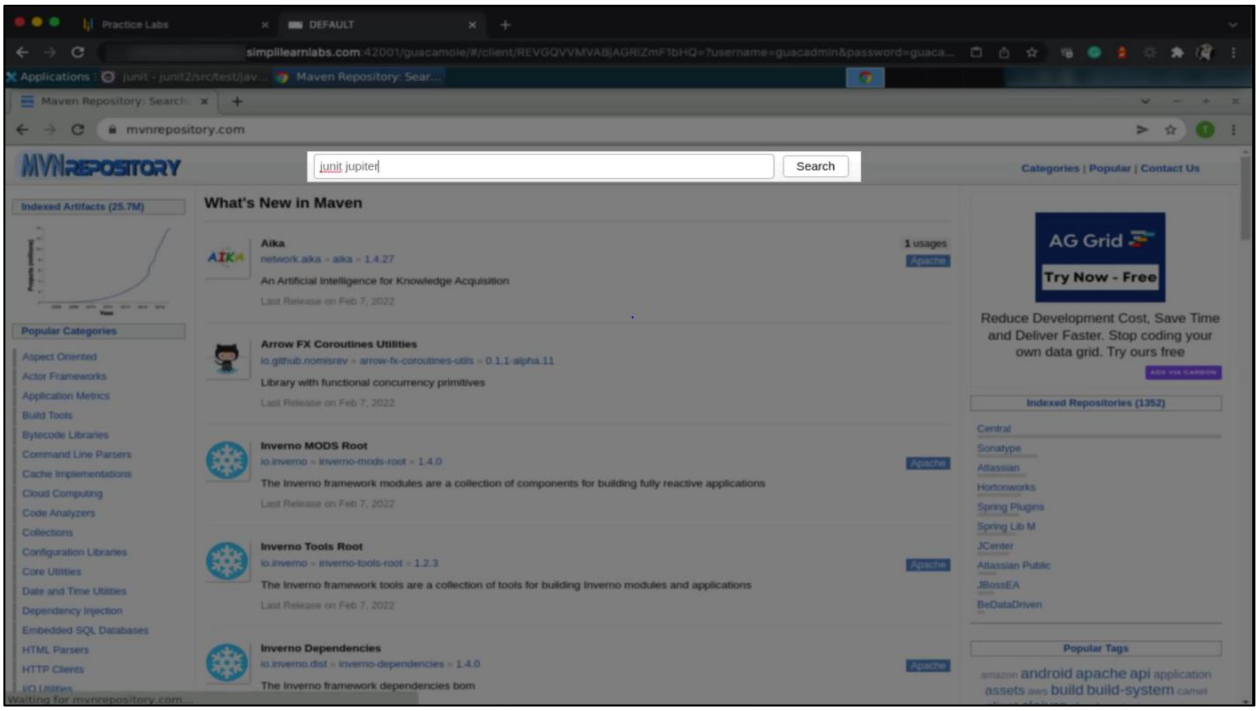
4.2 You need a class for test cases, so add the **public class AppTest**



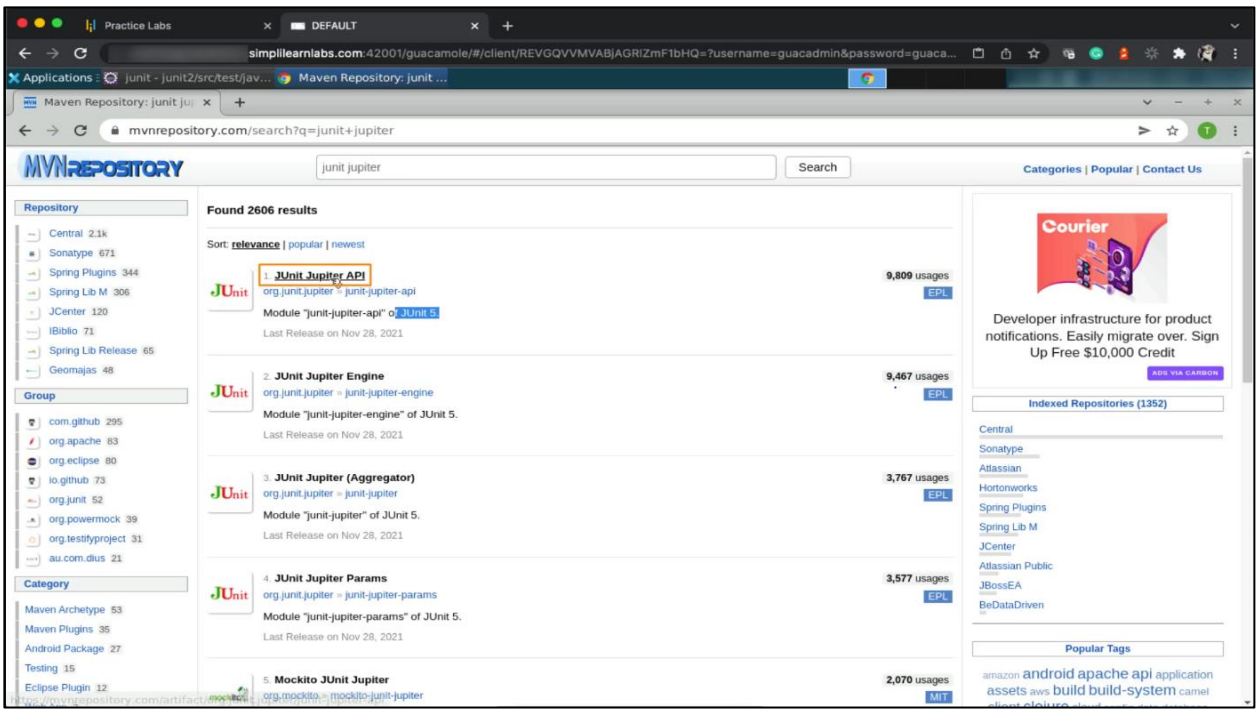
4.3 Go to the **web browser** and search for the **Maven repository**



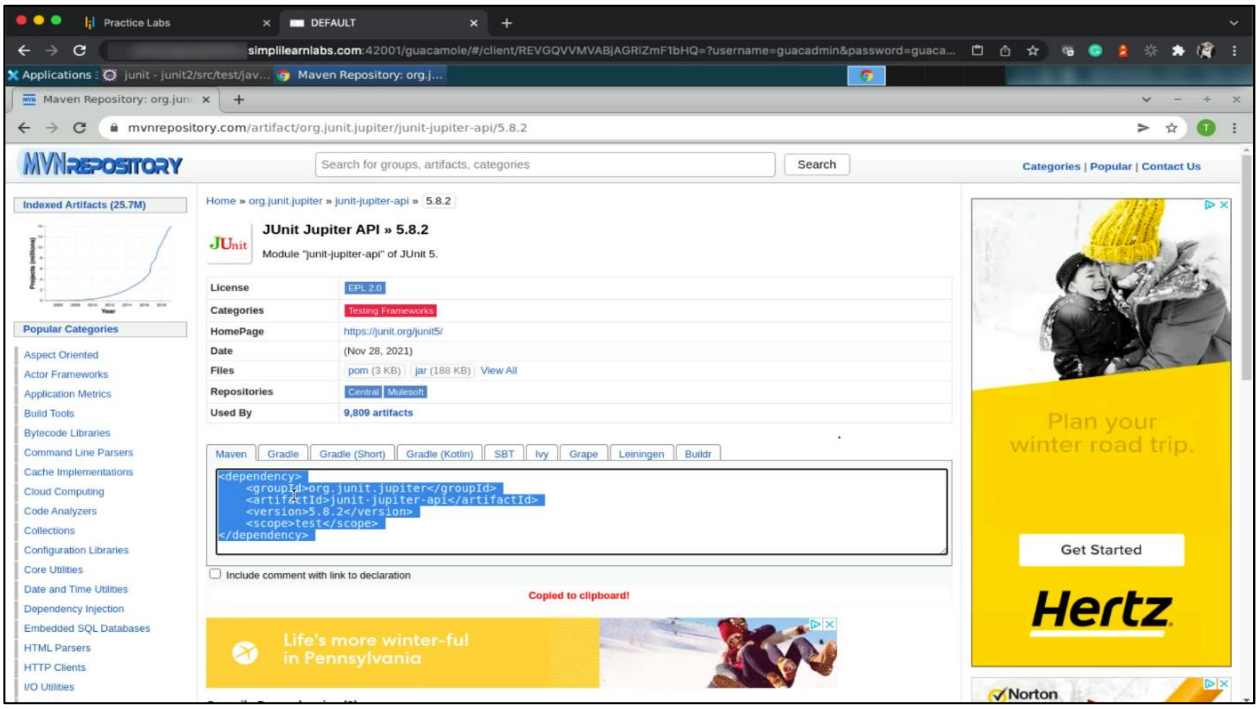
4.4 Write **JUnit Jupiter** in the search box



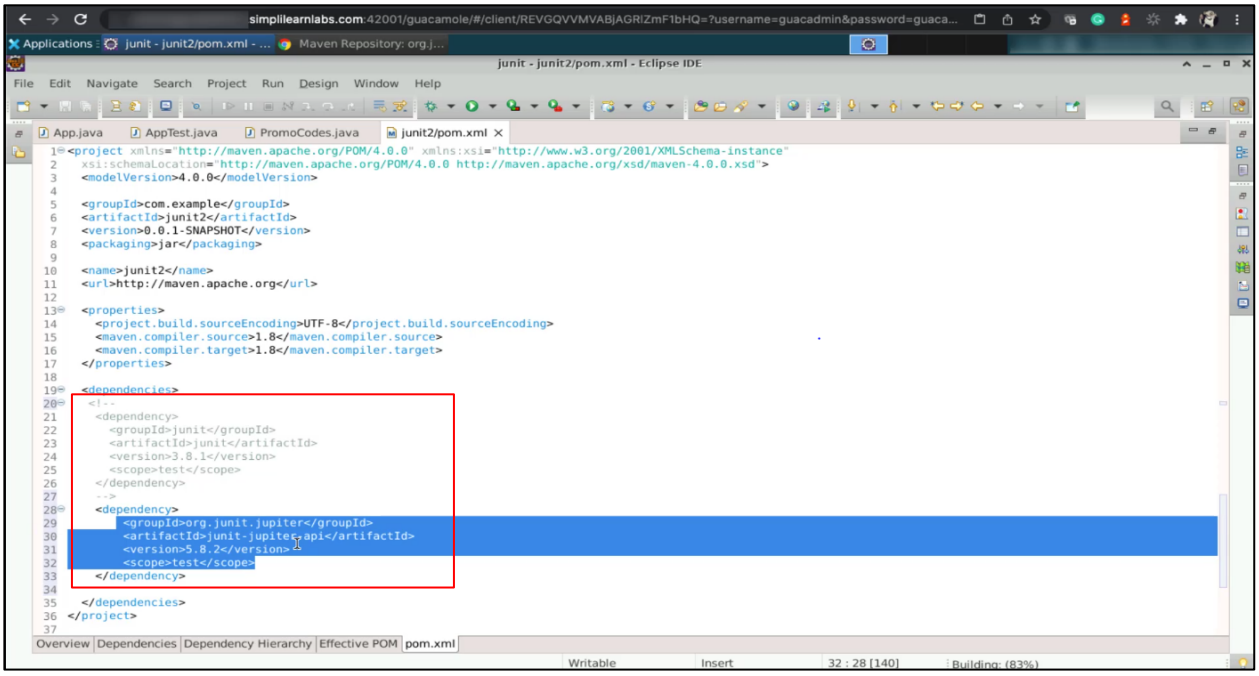
4.5 Click on **JUnit Jupiter API**, which is coming as **JUnit5**



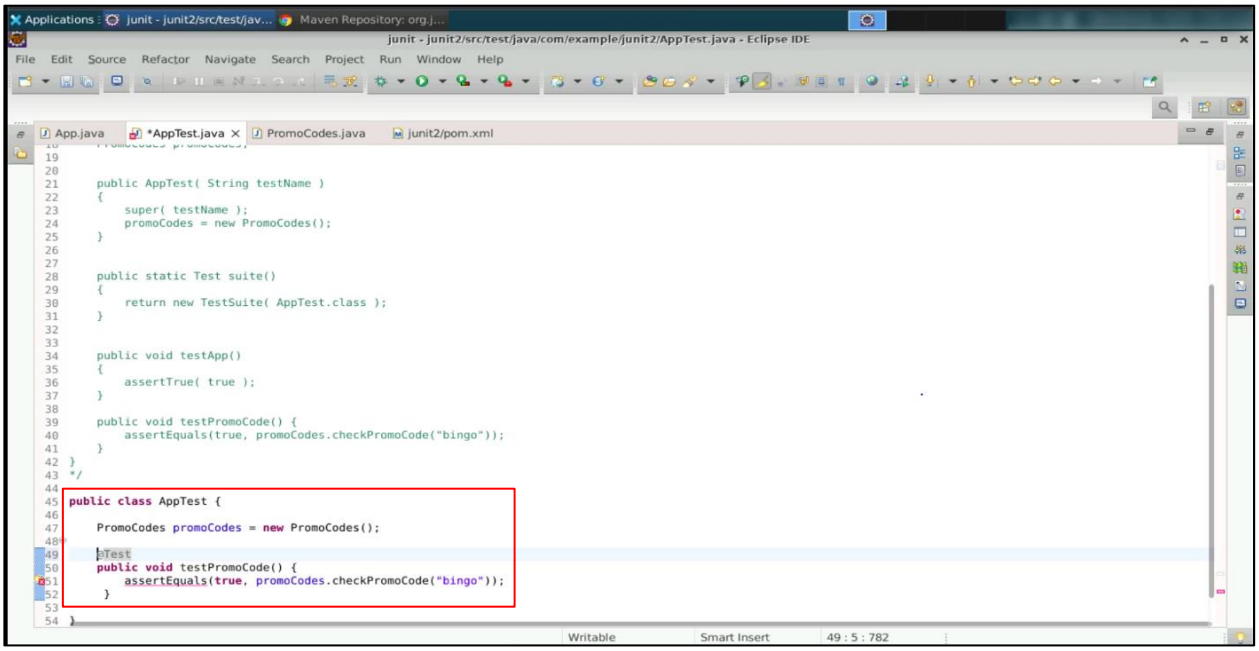
4.6 Open the **5.8.2** latest version and copy the **dependency code**



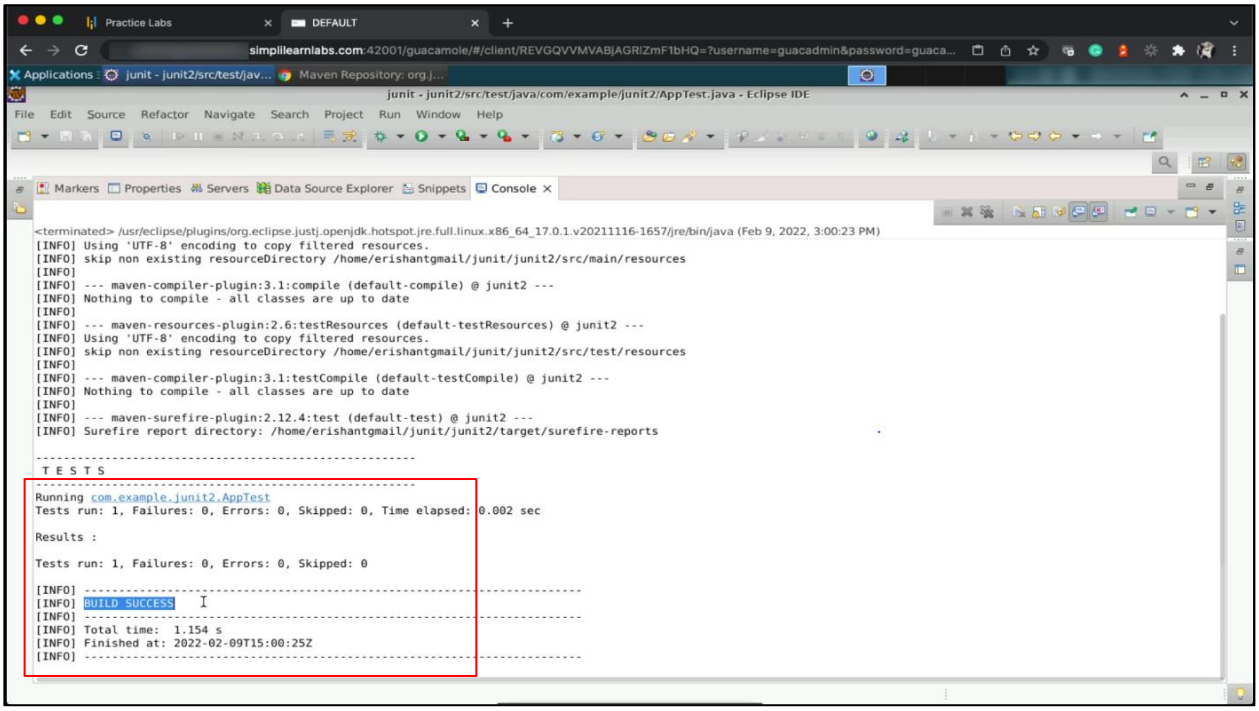
4.7 Come back to the project and go to the **pom.xml** file. Comment old dependency of **JUnit** and the paste new dependency



4.8 Now, go to the **AppTest.java** file and create test case **testPromoCode()** and **promoCodes** object same as previous and annotate it as **@Test**

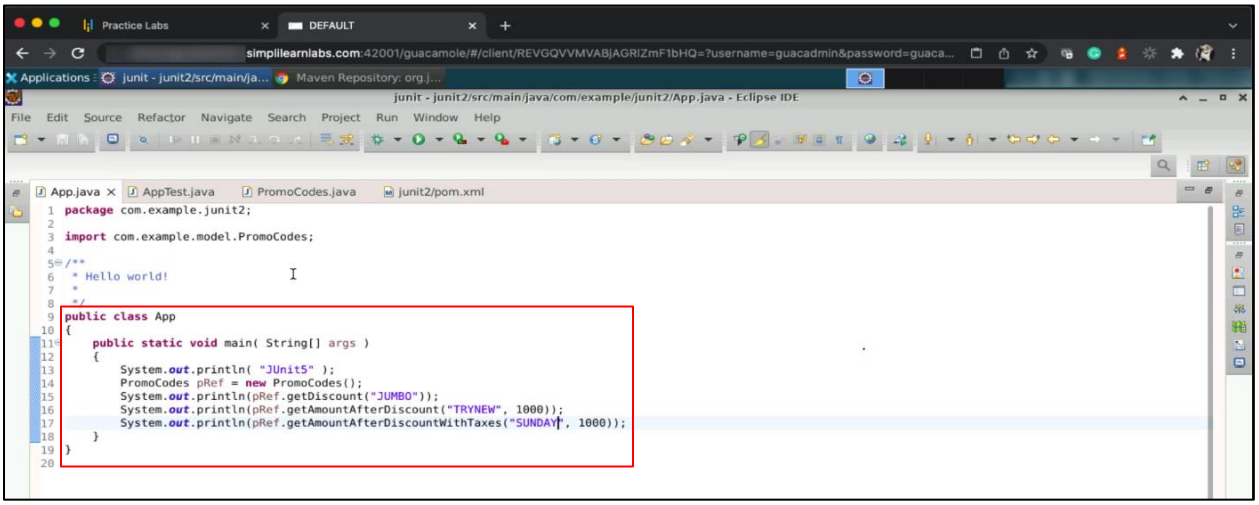


4.9 Rerun the project as a **Maven test.** In the output, you can see 1 test running and **BUILD SUCCESS** which shows that our dependency configuration was done successfully

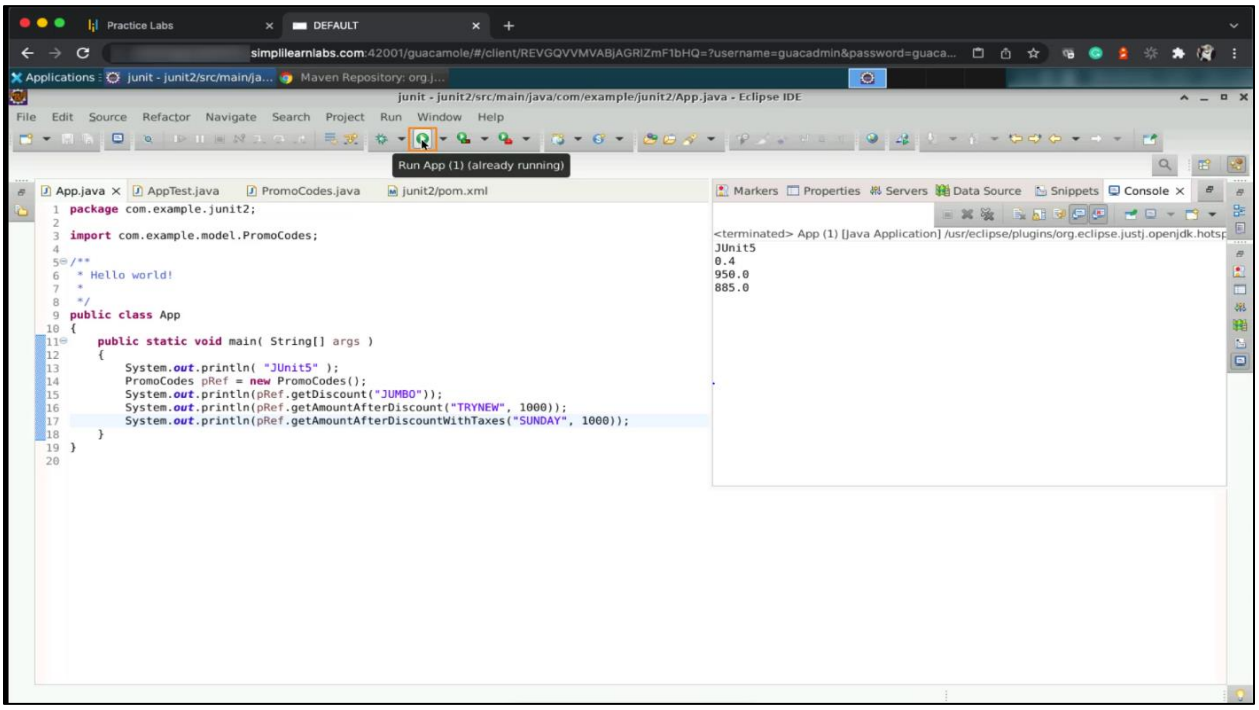


Step 5: Creating more test cases with output values

5.1 Go to the **App.java** file. Change the print statement to **JUnit5**. Create the **pRef** object of the **PromoCodes**. Using this **pRef** object, execute all the methods created and print the outputs



5.2 Run the code and record all outputs to use in test cases



5.3 Go back to the **AppTest.java** file. Create a few more test cases **testDiscount()**, **testAmountDiscount(), testAmountDiscountTaxes(),** and use **assertEquals()** assertion to compare the outputs recorded from the last steps from test cases

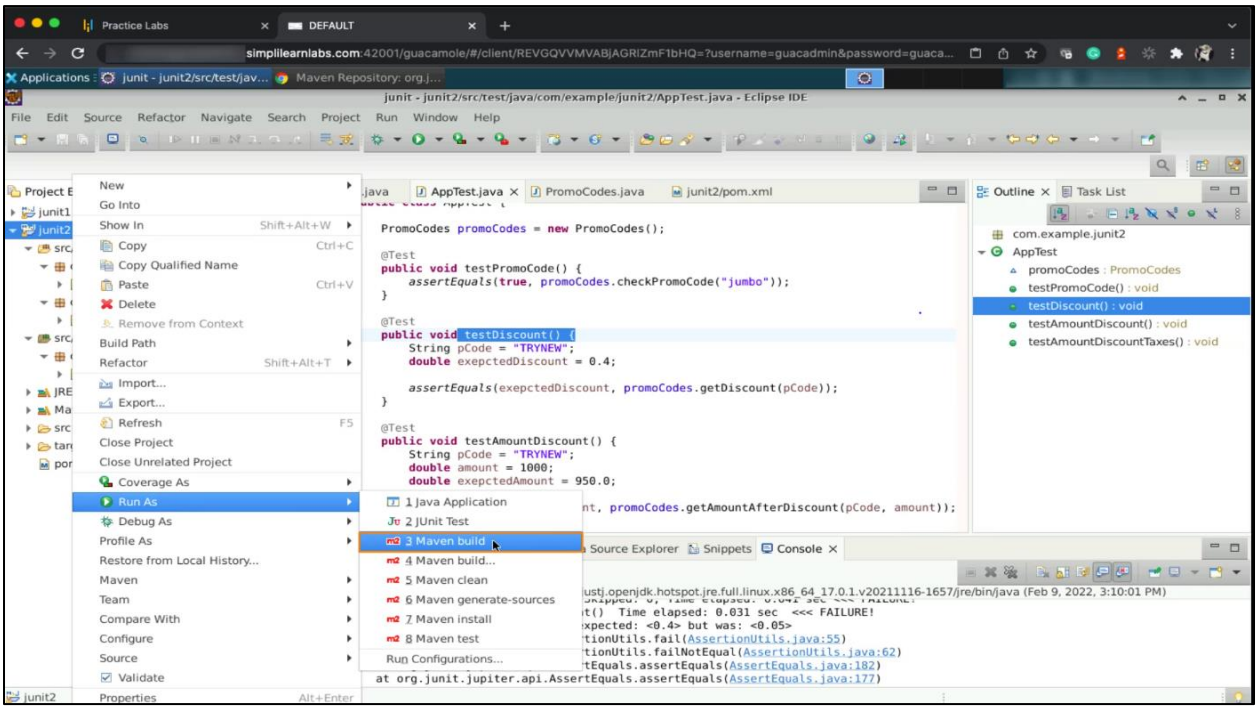
```
    PromoCodes promoCodes = new PromoCodes();

    @Test
    public void testPromoCode() {
        assertEquals(true, promoCodes.checkPromoCode("HEELO"));
    }

    @Test
    public void testDiscount() {
        String pCode = "JUMBO";
        double exepctedDiscount = 0.4;

        assertEquals(exepctedDiscount, promoCodes.getDiscount(pCode));
    }

    @Test
    public void testAmountDiscount() {
        String pCode = "TRYNEW";
        double amount = 1000;
        double exepctedAmount = 950.0;

        assertEquals(exepctedAmount, promoCodes.getAmountAfterDiscount(pCode, amount));
    }

    @Test
    public void testAmountDiscountTaxes() {
        String pCode = "SUNDAY";
        double amount = 1000;
        double exepctedAmount = 885.0;

        assertEquals(exepctedAmount, promoCodes.getAmountAfterDiscountWithTaxes(pCode, amount));
    }
}
```

5.4 Rerun the project as a **Maven test.** In the output, you can see all the test cases passed

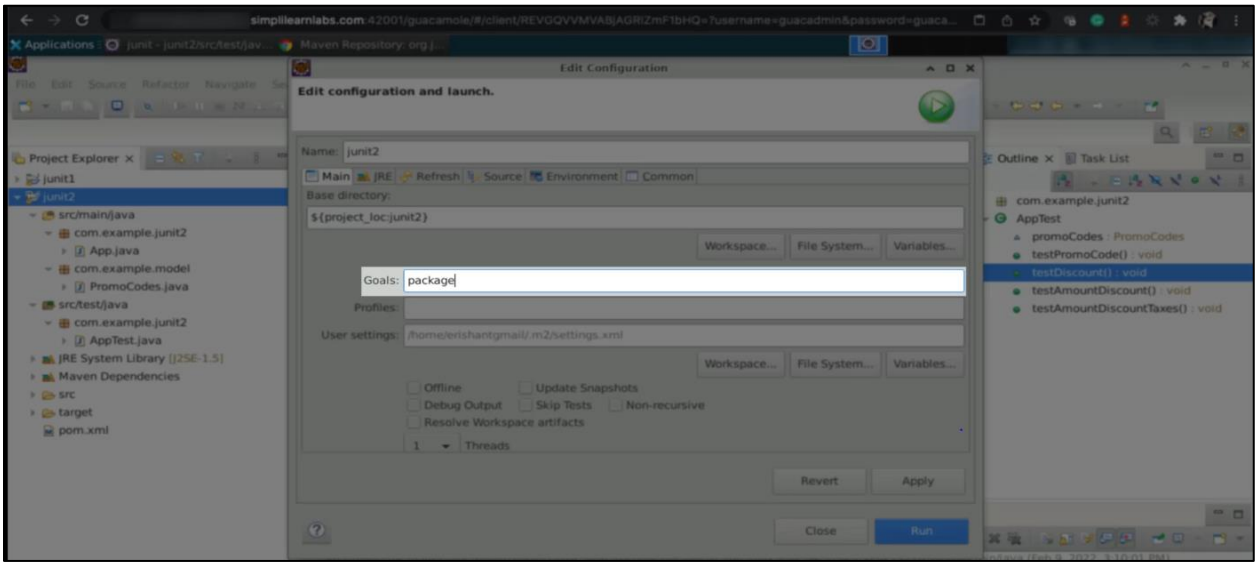
```
<terminated> /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211116-1657/jre/bin/java (Feb 9, 2022, 3:09:10 PM)
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/erishantgmail/junit/junit2/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ junit2 ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ junit2 ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/erishantgmail/junit/junit2/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ junit2 ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ junit2 ---
[INFO] Surefire report directory: /home/erishantgmail/junit/junit2/target/surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.example.junit2.AppTest
Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.001 sec

Results :

Tests run: 4, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.255 s
[INFO] Finished at: 2022-02-09T15:09:12Z
[INFO] ------------------------------------------------------------------------
```

Step 6: Packaging application

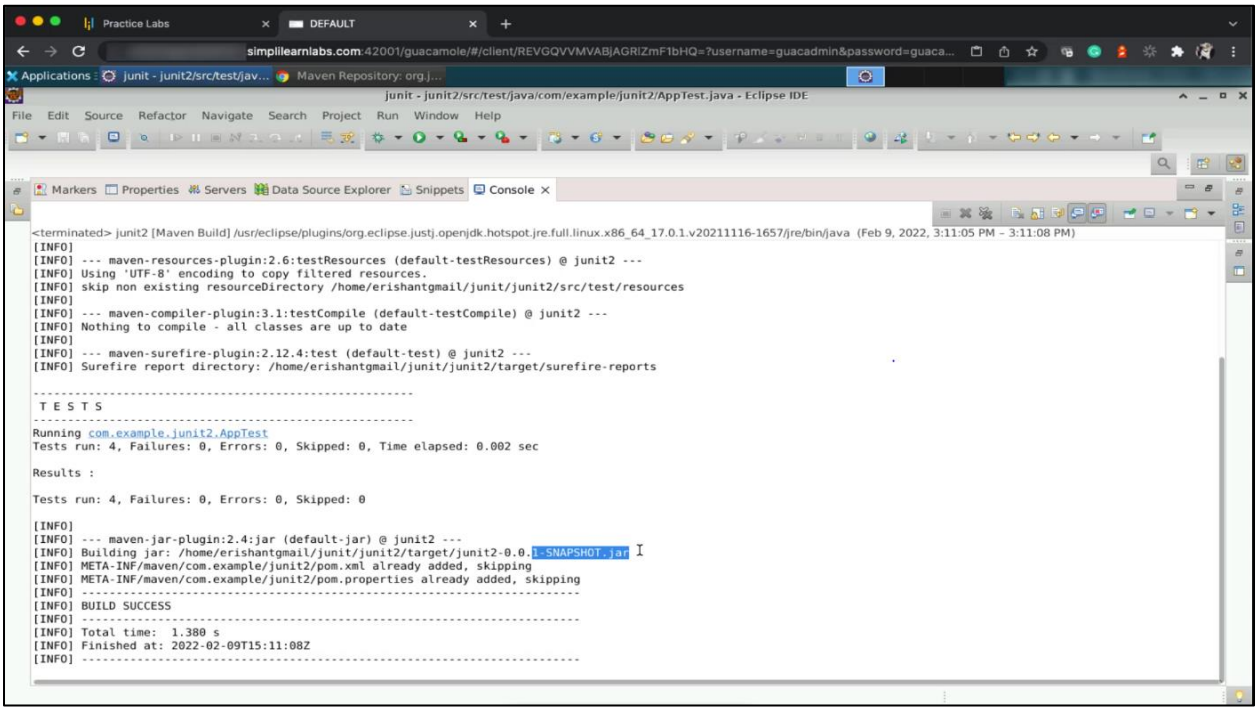
6.1 Go back to the project, and **right-click** on the project name. Go to **Run As,** and this time run the project as **Maven build**



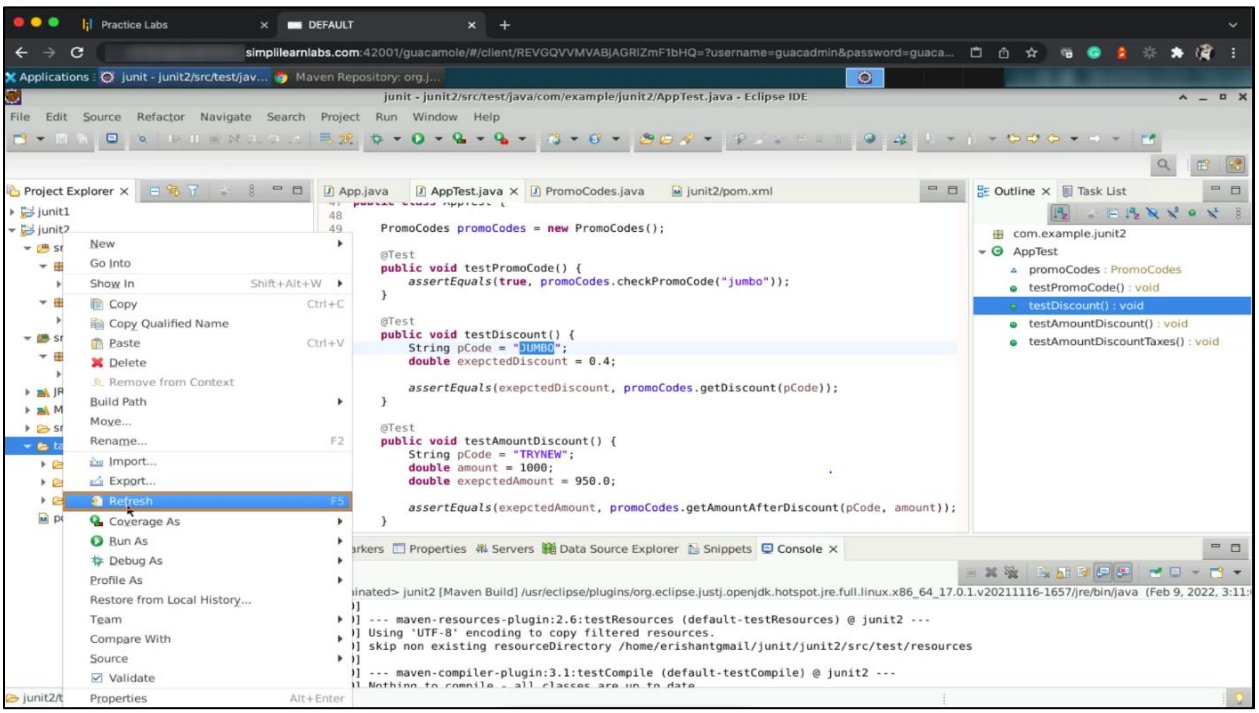
6.2 Name the Goals as a **package**. Click on **Apply** and then click on **Run**



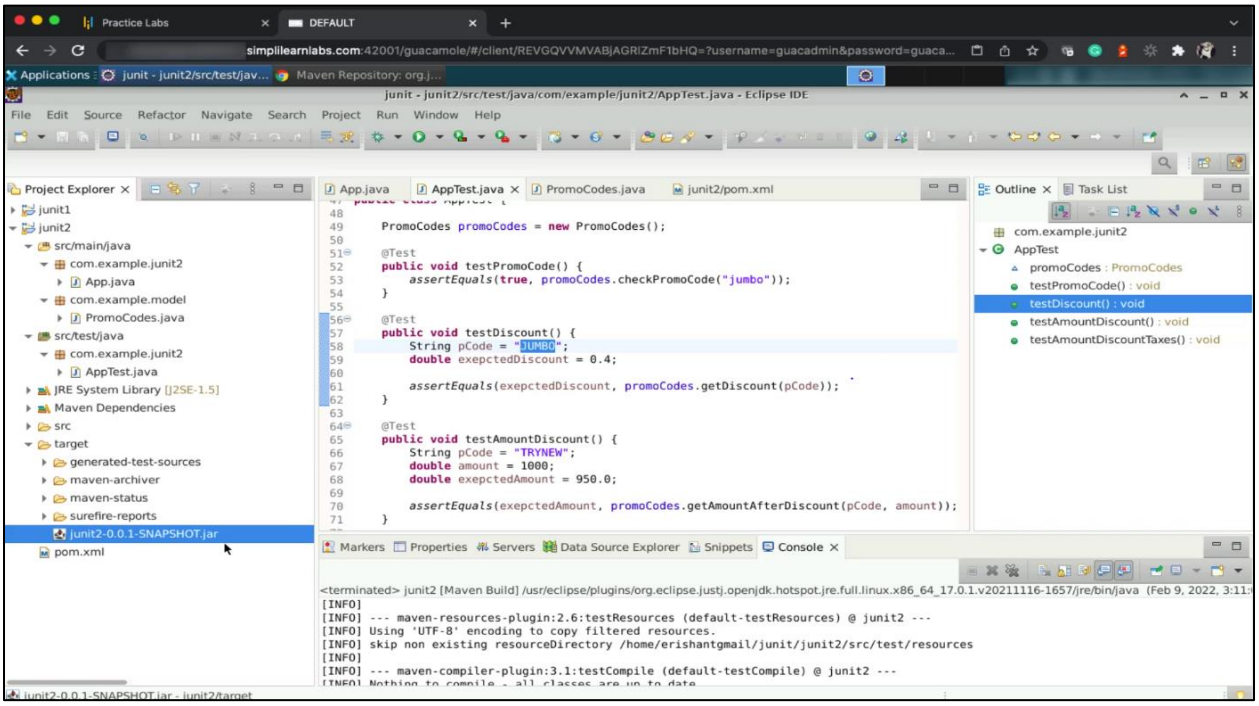
In the output, you can see Maven has created a **.jar** file.



6.3 Go back to **Project Explorer**. Right-click on the **target folder** and click on **Refresh**



You can see the jar file created in the target folder.



This is how we can work on test cases inside JUnit and work on e-store promo codes as a scenario. Here, we also learned to configure the latest JUnit version.