

## Lesson 01 Demo 05

### Maven Surefire Plugin

**Objective:** To explore Maven Surefire plugin by defining test cases and adding different configurations for test cases

**Tools Required:** Visual Studio Code and Maven

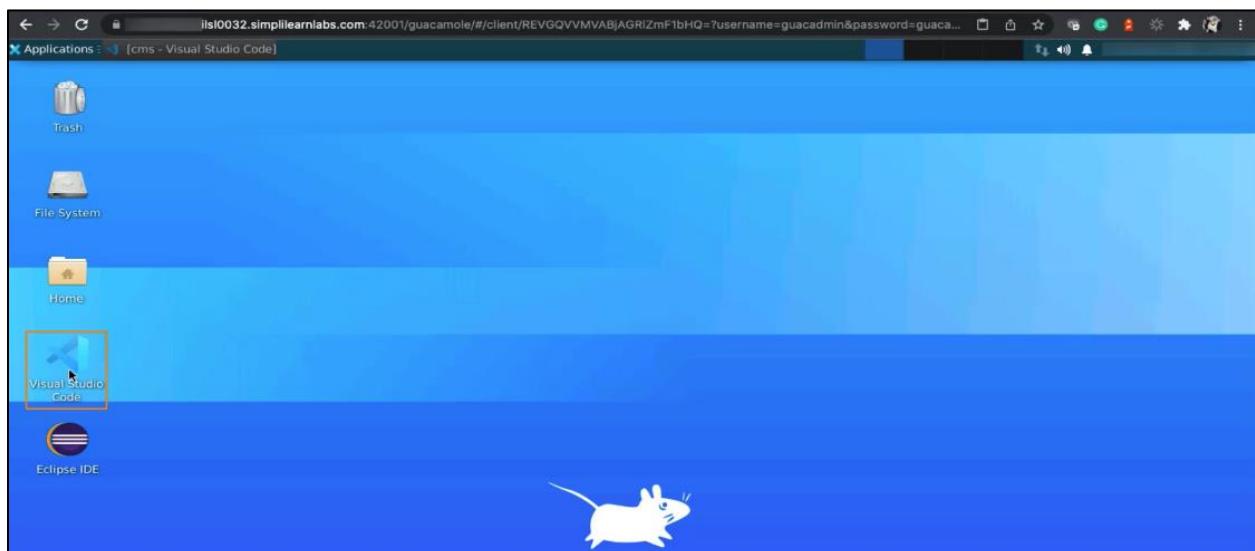
**Prerequisites:** None

#### Steps to be followed:

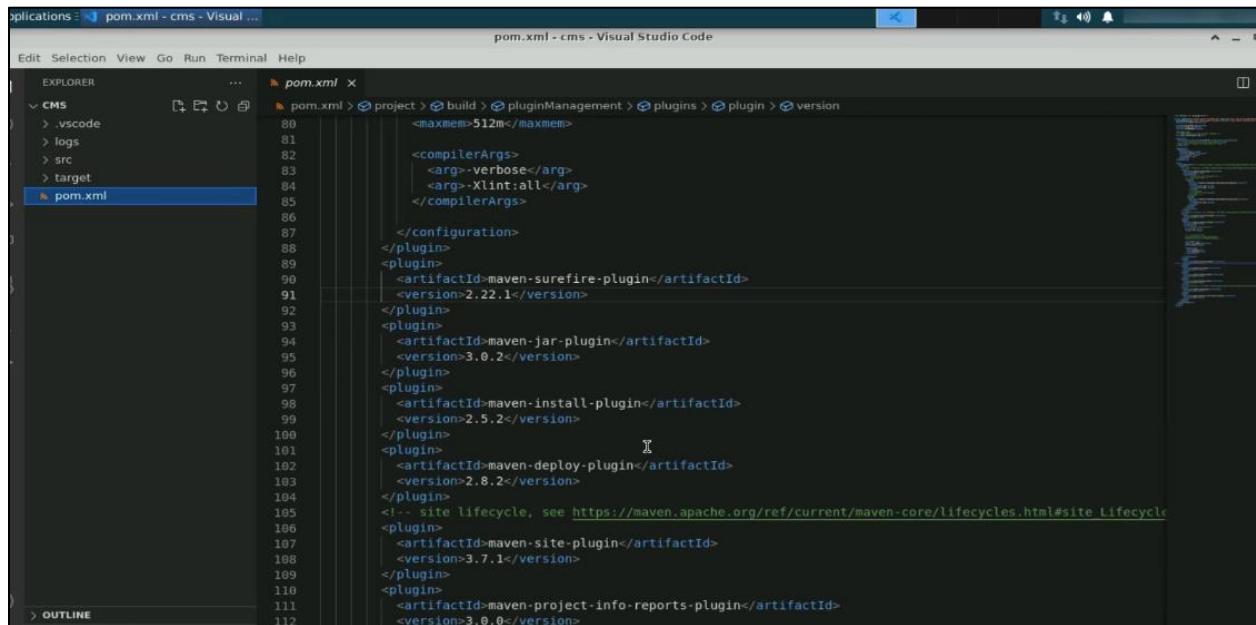
1. Check Maven Surefire plugin
2. Define test cases
3. Test defined test cases
4. Add different configurations for test cases
5. Work with the security manager

#### Step 1: Check Maven Surefire plugin

##### 1.1 Open Visual Studio Code



## 1.2 Under the CMS project, open the **pom.xml** file



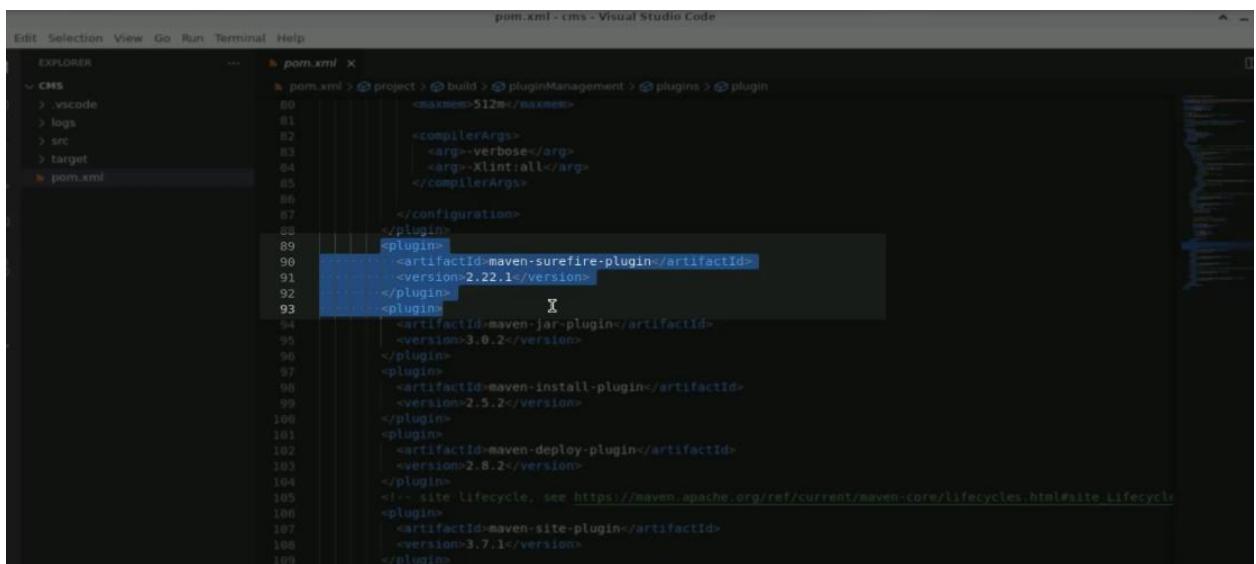
```

<project>
  <build>
    <pluginManagement>
      <plugins>
        <plugin>
          <artifactId>maven-surefire-plugin</artifactId>
          <version>2.22.1</version>
        </plugin>
        <plugin>
          <artifactId>maven-jar-plugin</artifactId>
          <version>3.0.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-install-plugin</artifactId>
          <version>2.5.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-deploy-plugin</artifactId>
          <version>2.8.2</version>
        </plugin>
        <!-- site lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#site_Lifecycle
        <plugin>
          <artifactId>maven-site-plugin</artifactId>
          <version>3.7.1</version>
        </plugin>
        <plugin>
          <artifactId>maven-project-info-reports-plugin</artifactId>
          <version>3.0.0</version>
        </plugin>
      </plugins>
    </pluginManagement>
  </build>
</project>

```

Note: Please refer to the previous demo on how to create the CMS project

You can see the plugin as **maven-surefire-plugin**:

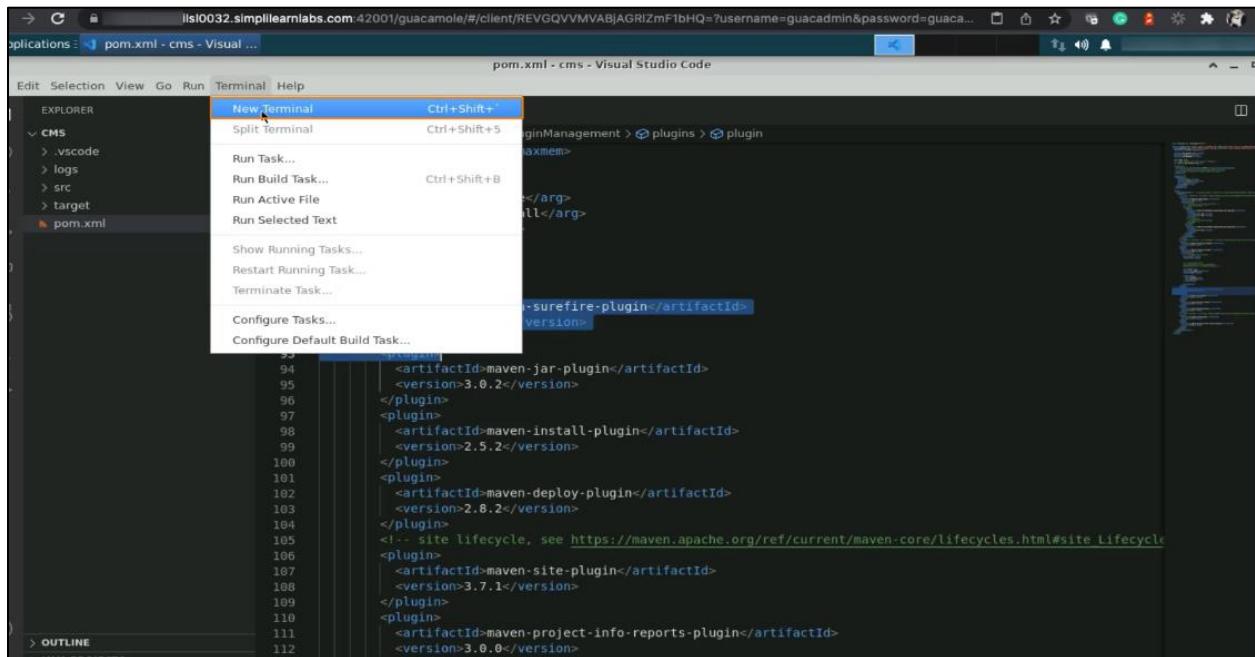


```

<project>
  <build>
    <pluginManagement>
      <plugins>
        <plugin>
          <artifactId>maven-surefire-plugin</artifactId>
          <version>2.22.1</version>
        </plugin>
        <plugin>
          <artifactId>maven-jar-plugin</artifactId>
          <version>3.0.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-install-plugin</artifactId>
          <version>2.5.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-deploy-plugin</artifactId>
          <version>2.8.2</version>
        </plugin>
        <!-- site Lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#site_Lifecycle
        <plugin>
          <artifactId>maven-site-plugin</artifactId>
          <version>3.7.1</version>
        </plugin>
        <plugin>
          <artifactId>maven-project-info-reports-plugin</artifactId>
          <version>3.0.0</version>
        </plugin>
      </plugins>
    </pluginManagement>
  </build>
</project>

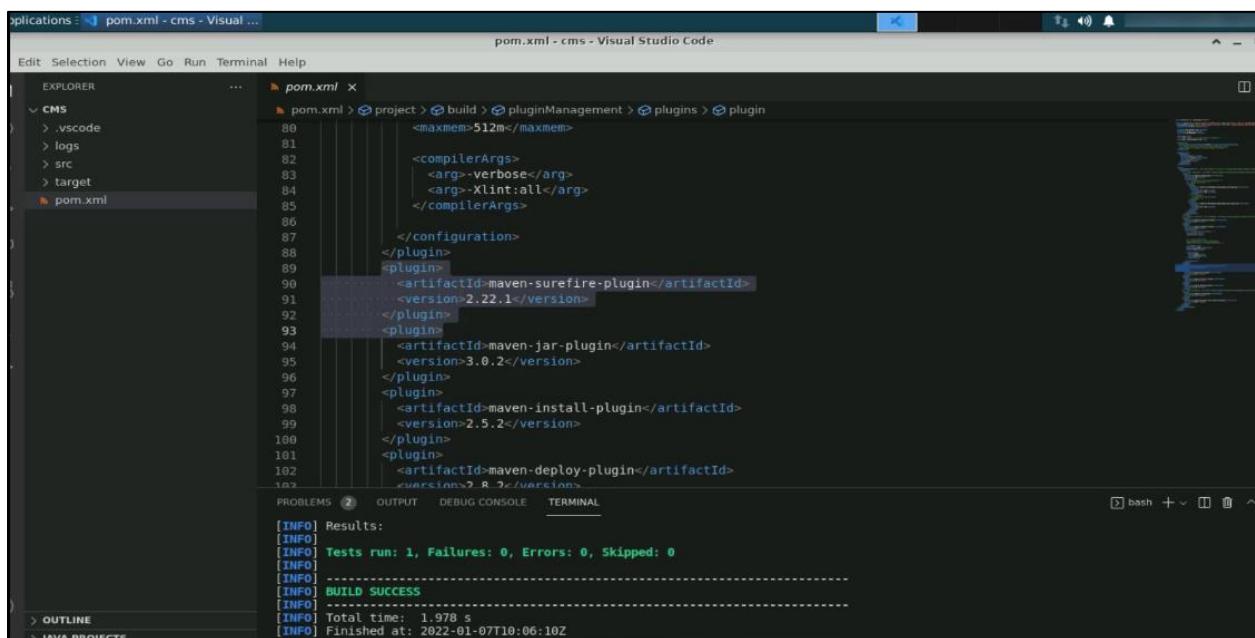
```

### 1.3 Go to the Terminal tab and click on New Terminal



### 1.4 Run the below command to check Surefire plugin usages and run the test cases:

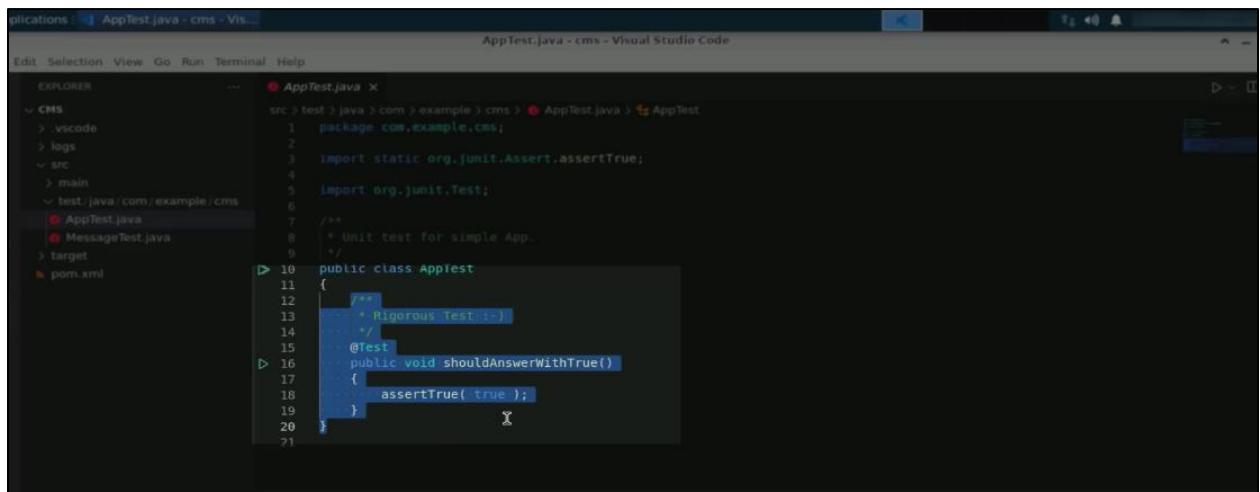
**mvn test**



The command **mvn test** is used in Maven to execute the unit tests for a Java project. It compiles the source code and runs all the test cases in the project.

## Step 2: Define test cases

2.1 Under src, open the **AppTest.java** from the **test** directory



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure:
  - src
  - test
  - java
  - com
  - example
  - cms
  - AppTest.java (selected)
  - MessageTest.java
- Code Editor:** The file **AppTest.java** is open, displaying the following code:

```
package com.example.cms;
import static org.junit.Assert.assertTrue;
import org.junit.Test;
/**
 * Unit test for simple App.
 */
public class AppTest
{
    /**
     * Rigorous Test :-)
     */
    @Test
    public void shouldAnswerWithTrue()
    {
        assertTrue( true );
    }
}
```

2.2 Go to **Message.java** and create a few methods in it

```
public static String getMessage(int code){  
}
```

```

src > main > java > com > example > cms > Message.java > getMessage(int)
1 package com.example.cms;
2
3 public class Message {
4     public static String getMessage(int code){
5         return "Customer Registered Successfully";
6     }
7 }
8

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

[INFO] Results:  
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0  
[INFO] -----  
[INFO] BUILD SUCCESS

The code snippet declares a public static method named **getMessage** which takes an integer parameter named **code**. The method is expected to return a string value.

## 2.3 Add the if condition and the return statement

```

if(code == 101){
    return "Customer Registered Successfully";
}

```

```

src > main > java > com > example > cms > Message.java > getMessage(int)
1 package com.example.cms;
2
3 public class Message {
4     public static String getMessage(int code){
5         if(code == 101){
6             return "Customer Registered Successfully";
7         }
8     }
9 }
10

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

[INFO] Results:  
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

This code snippet checks if the value of the variable **code** is **101**, and then the function returns the string **Customer Registered Successfully**.

## 2.4 Add the else if case and return statement

```
else if(code== 201){
    return "Customer Not Registered Successfully";
}
```

```
Message.java 1
src > main > java > com > example > cms > Message.java > Message > gteMessage(int)
1 package com.example.cms;
2
3 public class Message {
4     public static String gteMessage(int code){
5         if(code == 101){
6             return "Customer Registered Successfully";
7         }else if(code == 201){
8             return "Customer Not Registered Successfully";
9         }
10    }
11 }
12
```

These two lines of code use the conditional statement **else if** to check if the variable **code** is **201**, and then the function returns a string indicating that the customer was not registered successfully.

## 2.5 Add else case and the return statement

```
else{
    return "Please Try Again Later";
}
```

```
Message.java 1
src > main > java > com > example > cms > Message.java > Message > gteMessage(int)
1 package com.example.cms;
2
3 public class Message {
4     public static String gteMessage(int code){
5         if(code == 101){
6             return "Customer Registered Successfully";
7         }else if(code == 201){
8             return "Customer Not Registered Successfully";
9         }else{
10            return "Please Try Again Later";
11        }
12    }
13 }
14
15
```

This code snippet uses the **else** keyword to define what should happen if a certain condition is not met in an if statement. In this case, if the condition is not met, the function will return the string **Please Try Again Later**.

## Step 3: Test defined test cases

### 3.1 Create a **test** method in the **MessageTest.java** file to test the cases

```
@Test
public void testMessage(){
}
```

The screenshot shows the Visual Studio Code interface with the title bar "MessageTest.java - cms - Visual Studio Code". The left sidebar shows a project structure under "EXPLORER" with files like Message.java, App.java, and AppTest.java. The main editor area contains the following Java code:

```
src > test > java > com > example > cms > MessageTest.java > MessageTest > testMessage()

import org.junit.Test;
public class MessageTest {
    @Test
    public void testMessage(){
    }
}
```

### 3.2 Define a scenario for the test and add the expected and the actual output for it

```
assertEquals("Customer Registered Successfully", Message.getMessage(101));
```

The screenshot shows the Visual Studio Code interface with the title bar "MessageTest.java - cms - Visual Studio Code". The left sidebar shows a project structure under "EXPLORER" with files like Message.java, App.java, and AppTest.java. The main editor area contains the following Java code:

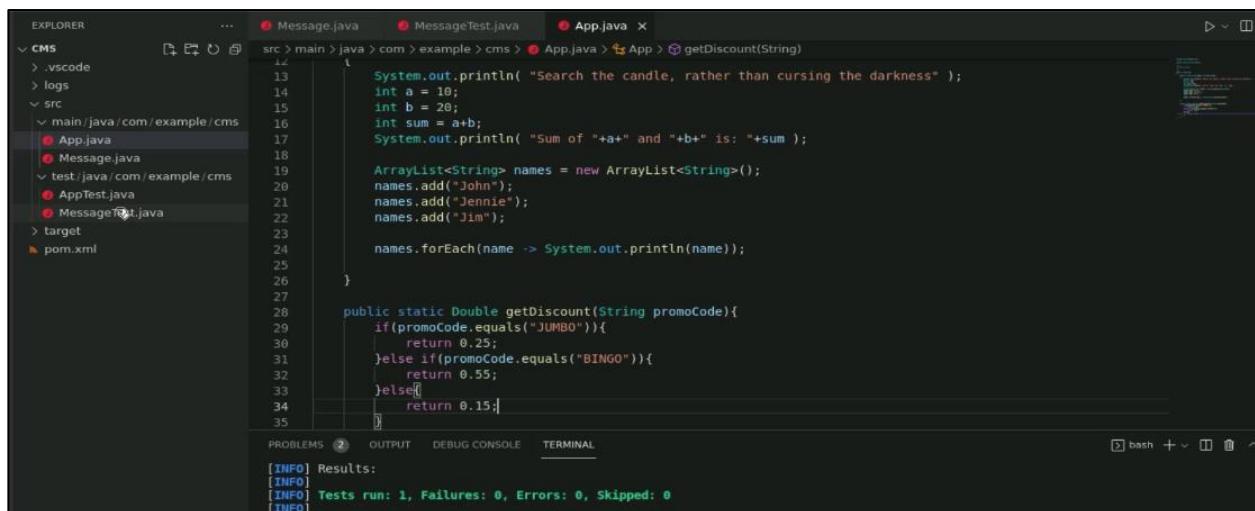
```
src > test > java > com > example > cms > MessageTest.java > MessageTest > testMessage()

import org.junit.Test;
public class MessageTest {
    @Test
    public void testMessage(){
        assertEquals("Customer Registered Successfully", Message.getMessage(101));
    }
}
```

The given code is a test method that verifies if the message returned by the method **getMessage()** with the argument **101** is equal to the expected string **Customer Registered Successfully**. The **assertEquals()** method is used to compare the actual and expected values.

### 3.3 Add a method in **App.java** to define the discount

```
public static Double getDiscount (String promoCode) {
    if(promoCode.equals("JUMBO")){
        return 0.25;
    }else if (promoCode.equals("BINGO")){
        return 0.55;
    }else{
        return 0.15;
    }
}
```



This code defines a static method named **getDiscount** that takes a String argument named **promoCode**. If the **promoCode** is equal to **JUMBO**, the method returns a discount of **25%**; if it is equal to **BINGO**, the method returns a discount of **55%**; and if it is anything else, the method returns a discount of **15%**.

### 3.4 Write test promo code in the **AppTest.java** file and add the **assertEquals** statement, including the actual and expected output

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows the project structure under CMS, including .vscode, logs, src, main/java/com/example/cms (App.java, Message.java), test/java/com/example/cms (AppTest.java, MessageTest.java), target, and pom.xml.
- EDITOR:** Displays the content of AppTest.java:
 

```

1 package com.example.cms;
2 import static org.junit.Assert.assertEquals;
3 import org.junit.Test;
4 /**
5  * Unit test for simple App.
6  */
7 public class AppTest {
8
9     /**
10      * Rigorous Test :-)
11     */
12     @Test
13     public void testPromoCode() {
14         assertEquals(0.55, App.getDiscount("BINGO"));
15     }
16 }
      
```
- TERMINAL:** Shows the build results:
 

```

[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
      
```

### 3.5 Change the expected output value to a double data type value

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows the project structure under CMS, including .vscode, logs, src, main/java/com/example/cms (App.java, Message.java), test/java/com/example/cms (AppTest.java, MessageTest.java), target, and pom.xml.
- EDITOR:** Displays the content of AppTest.java, with the line `assertEquals(Double.valueOf(0.55), App.getDiscount("BINGO"));` highlighted in yellow:
 

```

1 package com.example.cms;
2 import static org.junit.Assert.assertEquals;
3 import org.junit.Test;
4 /**
5  * Unit test for simple App.
6  */
7 public class AppTest {
8
9     /**
10      * Rigorous Test :-)
11     */
12     @Test
13     public void testPromoCode() {
14         assertEquals(Double.valueOf(0.55), App.getDiscount("BINGO"));
15     }
16 }
      
```
- TERMINAL:** Shows the build results:
 

```

[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 1.978 s
[INFO] Finished at: 2022-01-07T10:06:10Z
[INFO]
      
```
- OUTLINE:** Shows sections like OUTLINE, JAVA PROJECTS, and MAVEN.

### 3.6 Use the **mvn test** again to run the test cases

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows the project structure under the CMS root. Files listed include .vscode, logs, src, main/java/com/example/cms (App.java, Message.java), test/java/com/example/cms (AppTest.java, MessageTest.java), target, and pom.xml.
- MessageTest.java** is the active file, showing the following code:

```
src > test > java > com > example > cms > MessageTest.java > ...
1 package com.example.cms;
2
3 import static org.junit.Assert.assertEquals;
4
5 import org.junit.Test;
6
7 public class MessageTest {
8
9     @Test
10    public void testMessage(){
11        assertEquals("Customer Registered Successfully", Message.getMessage(101));
12    }
13
14 }
15
```

- TERMINAL**: Displays the output of a Maven test command:

```
[INFO] Finished at: 2022-01-07T10:06:10Z
[INFO] -----
erishant@gmail@ip-172-31-17-157:~/Downloads/cms$ mvn test
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.google.inject.internal.cglib.core.$ReflectUtils$1 (file:/usr/share/maven/lib/guice.jar) to method java.lang.ClassLoader.defineClass-java.lang.String,byte[],int,int,java.security.ProtectionDomain)
WARNING: Please consider reporting this to the maintainers of com.google.inject.internal.cglib.core.$ReflectUtils$1
[INFO] --- tika-maven-plugin:1.2:analyze (default-cli) @ cms ---
[INFO] Building cms 1.0-SNAPSHOT
[INFO] [ jar ] -----
```

3.7 Change the promo code in the `AppTest.java` file to `Hello55`

The screenshot shows a Java project structure in the left sidebar with files like Message.java, MessageTest.java, App.java, AppTest.java, and pom.xml. The AppTest.java file is open in the editor, showing a test method testPromoCode(). The terminal at the bottom shows the execution of the tests, with output indicating 2 tests run, 0 failures, 0 errors, and 0 skipped.

```
src > test > java > com > example > cms > AppTest.java > AppTest > testPromoCode()
  4 import org.junit.Test;
  5
  6 /**
  7  * Unit test for simple App.
  8  */
  9 public class AppTest
 10 {
 11     /**
 12      * Rigorous Test :-)
 13      */
 14     @Test
 15     public void testPromoCode()
 16     {
 17         assertEquals(Double.valueOf(0.55), App.getDiscount("HELLO55"));
 18     }
 19 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
[INFO] -----  
[INFO] T E S T S  
-----  
[INFO] Running com.example.cms.MessageTest  
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.032 s - in com.example.cms.MessageTest  
[INFO] Running com.example.cms.AppTest  
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.001 s - in com.example.cms.AppTest  
[INFO] -----  
[INFO] Results:  
-----  
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0  
[INFO]
```

3.8 Run the **mvn test** again to run the test cases

The screenshot shows the Visual Studio Code interface with the following details:

- Explorer:** Shows the project structure with files: Message.java, MessageTest.java, App.java, AppTest.java, pom.xml.
- Terminal:** Shows the command `mvn test` being run, resulting in:
 

```
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.001 s - in com.example.cms.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 2.179 s
[INFO] Finished at: 2022-01-07T10:20:56Z
[INFO] -----
[INFO] erikhamza@elip-172-21-17-157:~/Downloads/cms$ mvn test
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.google.inject.internal.cglib.core.$ReflectUtils$1 (file:/usr/share/maven/lib/guice.jar) to method java.lang.ClassLoader.defineClass([Ljava.lang.String;B)V
WARNING: Please consider reporting this to the maintainers of com.google.inject.internal.cglib.core.$ReflectUtils$1
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
```
- PROBLEMS:** Shows 2 errors.
- OUTPUT:** Shows the build results.
- DEBUG CONSOLE:** Not visible.
- TOTAL:** Shows the total build time and completion time.

## Step 4: Add different configurations for test cases

### 4.1 Add configuration in the pom.xml file

```
<configuration></ configuration >
```

The screenshot shows the Visual Studio Code interface with the following details:

- Explorer:** Shows the project structure with files: pom.xml, pom.xml.
- Terminal:** Shows the command `mvn test` being run, resulting in:
 

```
[INFO] T E S T S
[INFO] -----
[INFO] Running com.example.cms.MessageTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.038 s - in com.example.cms.MessageTest
[INFO] Running com.example.cms.AppTest
[ERROR] Tests run: 1, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 0.007 s <<< FAILURE! - in com.example.cms.AppTest
[ERROR] testPromoCode(com.example.cms.AppTest) Time elapsed: 0 s <<< FAILURE!
[java.lang.AssertionError: expected:<0.55> but was:<0.15>
at com.example.cms.AppTest.testPromoCode(AppTest.java:17)
[INFO]
[INFO] Results:
[INFO]
```

### 4.2 Under configurations, add SkipTest and pass the value as true

```
<skipTests>true</skipTests>
```

```

<skipTests>true</skipTests>

```

INFO] Tests run: 1, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 0.038 s - in com.example.cms.MessageTest  
[INFO] Tests run: 1, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 0.007 s <<< FAILURE! - in com.example.cms.AppTest  
[ERROR] testPromoCode(com.example.cms.AppTest) Time elapsed: 0 s <<< FAILURE!  
java.lang.AssertionError: expected:<0.55> but was:<0.15>  
at com.example.cms.AppTest.testPromoCode(AppTest.java:17)

#### 4.3 Run the **mvn clean** command to remove the **target** directory

```

erishant@gmail@ip-172-31-17-157:~/Downloads/cms$ mvn clean

```

#### 4.4 Type the **mvn test** command and run the test cases again to check if it will work or not

```

<configuration>
    <skipTests>true</skipTests>
</configuration>

```

The screenshot shows the VS Code interface with the pom.xml file open. The code editor displays the Maven configuration section, specifically the part where the skipTests element is set to true. The terminal below shows the Maven build output, which includes a warning about illegal reflective access operations and a message indicating a build success.

4.5 To directly use the skip test without writing in the **pom.xml** file, use the below command:

**mvn test -DskipTest=true**

The screenshot shows the VS Code interface with the terminal tab active. The user has run the command `mvn test -DskipTest=true`. The terminal output shows the Maven build process, including the loading of modules and classes, and a message stating that tests are skipped due to the skipTest configuration.

4.6 Add excludes in the configuration

**<excludes></excludes>**

```

<configuration>
    <compilerArgs>
        <arg>-verbose</arg>
        <arg>-Xlint:all</arg>
    </compilerArgs>
</configuration>
</plugin>
<plugins>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>2.22.1</version>
<configuration>
    <!-- <skipTests>true</skipTests> -->
    <excludes>
        <exclude>AppTest.java</exclude>
    </excludes>
</configuration>
</plugin>
</plugins>

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

[INFO] [loading /modules/java_base/java/lang/Double.class]
[INFO] [loading /modules/java_base/java/lang/String.class]
[INFO] [loading /modules/java_base/java/lang/Comparable.class]
[INFO] [loading /modules/java_base/java/lang/Number.class]
[INFO] [wrote /home/erishant@gmail/Downloads/cms/target/classes/com/example/cms/App.class]
[INFO] [wrote /home/erishant@gmail/Downloads/cms/target/test-classes/com/example/cms/AppTest.class]
[INFO] [checking com.example.cms.MessageTest]
[INFO] [loading /home/erishant@gmail/Downloads/cms/target/classes/com/example/cms/Message.class]
[INFO] [wrote /home/erishant@gmail/Downloads/cms/target/test-classes/com/example/cms/MessageTest.class]
[INFO] [total 268ms]
[INFO] --- maven-surefire-plugin:2.22.1:test (default-test) @ cms ---
[INFO] Tests are skipped.
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.739 s
[INFO] Finished at: 2022-01-07T18:22:46Z
[INFO] 

```

OUTLINE JAVA PROJECTS MAVEN

#### 4.7 Under the excludes tag, add the Java file in which you want to exclude

```
<exclude>AppTest.java</exclude>
```

```

<configuration>
    <compilerArgs>
        <arg>-verbose</arg>
        <arg>-Xlint:all</arg>
    </compilerArgs>
</configuration>
</plugin>
<plugins>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>2.22.1</version>
<configuration>
    <!-- <skipTests>true</skipTests> -->
    <excludes>
        <exclude>AppTest.java</exclude>
    </excludes>
    <includes>
        <include>MessageTest.java</include>
    </includes>
</configuration>
</plugin>
</plugins>

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

[INFO] [loading /modules/java_base/java/lang/Double.class]
[INFO] [loading /modules/java_base/java/lang/String.class]
[INFO] [loading /modules/java_base/java/lang/Comparable.class]
[INFO] [loading /modules/java_base/java/lang/Number.class]
[INFO] [loading /home/erishant@gmail/Downloads/cms/target/classes/com/example/cms/App.class]
[INFO] [wrote /home/erishant@gmail/Downloads/cms/target/test-classes/com/example/cms/AppTest.class]
[INFO] [checking com.example.cms.MessageTest]
[INFO] [loading /home/erishant@gmail/Downloads/cms/target/classes/com/example/cms/Message.class]
[INFO] [wrote /home/erishant@gmail/Downloads/cms/target/test-classes/com/example/cms/MessageTest.class]
[INFO] [total 268ms]
[INFO] --- maven-surefire-plugin:2.22.1:test (default-test) @ cms ---
[INFO] Tests are skipped.

```

#### 4.8 Create includes and add a file in it

```
<includes>
    <include>MessageTest.java</include>
</includes>
```

```

<configuration>
    <excludes>
        </excludes>
    </configuration>
</plugin>
<plugin>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>2.22.1</version>
    <configuration>
        <!-- <skipTests>true</skipTests> -->
        <excludes>
            <exclude>AppTest.java</exclude>
        </excludes>
        <includes>
            <include>MessageTest.java</include>
        </includes>
    </configuration>
</plugin>

```

[INFO] [loading /modules/java.base/java/lang/Double.class]  
[INFO] [loading /modules/java.base/java/lang/String.class]  
[INFO] [loading /modules/java.base/java/lang/Comparable.class]  
[INFO] [loading /modules/java.base/java/lang/Number.class]  
[INFO] [loading /home/erishant@gmail/Downloads/cms/target/classes/com/example/cms/App.class]  
[INFO] [wrote /home/erishant@gmail/Downloads/cms/target/test-classes/com/example/cms/AppTest.class]  
[INFO] [checking com.example.cms.MessageTest]  
[INFO] [loading /home/erishant@gmail/Downloads/cms/target/classes/com/example/cms/Message.class]  
[INFO] [wrote /home/erishant@gmail/Downloads/cms/target/test-classes/com/example/cms/MessageTest.class]  
[INFO] [total 269ms]  
[INFO] --- maven-surefire-plugin:2.22.1:test (default-test) @ cms ---  
[INFO] Tests are skipped.  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 2.739 s  
[INFO] Finished at: 2022-01-07T18:22:46Z  
[INFO] -----

#### 4.9 Write **mvn clean** in terminal

```

[INFO] [loading /modules/java.base/java/lang/String.class]
[INFO] [loading /modules/java.base/java/lang/Comparable.class]
[INFO] [loading /modules/java.base/java/lang/Number.class]
[INFO] [loading /home/erishant@gmail/Downloads/cms/target/classes/com/example/cms/App.class]
[INFO] [wrote /home/erishant@gmail/Downloads/cms/target/test-classes/com/example/cms/AppTest.class]
[INFO] [checking com.example.cms.MessageTest]
[INFO] [loading /home/erishant@gmail/Downloads/cms/target/classes/com/example/cms/Message.class]
[INFO] [wrote /home/erishant@gmail/Downloads/cms/target/test-classes/com/example/cms/MessageTest.class]
[INFO] [total 269ms]
[INFO] --- maven-surefire-plugin:2.22.1:test (default-test) @ cms ---
[INFO] Tests are skipped.
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.739 s
[INFO] Finished at: 2022-01-07T18:22:46Z
[INFO] -----

```

#### 4.10 Run the test using the **mvn test** command

```

<configuration>
    <!-- <skipTests>true</skipTests> -->
    <excludes>
        | <exclude>AppTest.java</exclude>
    </excludes>
    <includes>
        | <include>MessageTest.java</include>
    </includes>
</configuration>

```

INFO: -----[ jar ]-----  
INFO: --- maven-clean-plugin:3.1.0:clean (default-clean) @ cms ---  
INFO: Deleting /home/erishant@gmail/Downloads/cms/target  
INFO: Deleting /home/erishant@gmail/Downloads/cms/logs/dev (includes = [\*\*/\*.log, \*\*/\*.tmp], excludes = [])  
INFO: Deleting /home/erishant@gmail/Downloads/cms/logs/prod (includes = [\*\*/\*.log], excludes = [])  
INFO: BUILD SUCCESS  
INFO: Total time: 0.523 s  
INFO: Finished at: 2022-01-07T10:24:54Z  
INFO: erishant@gmail@ip-172-31-17-157:~/Downloads/cms\$ mvn test  
WARNING: An illegal reflective access operation has occurred  
WARNING: Illegal reflective access by com.google.inject.internal.cglib.core.\$ReflectUtils\$1 (file:/usr/share/maven/lib/guice.jar) to  
od java.lang.ClassLoader.defineClass(java.lang.String,byte[],int,int,java.security.ProtectionDomain)  
WARNING: Please consider reporting this to the maintainers of com.google.inject.internal.cglib.core.\$ReflectUtils\$1  
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations  
WARNING: All illegal access operations will be denied in a future release

#### 4.11 Comment the code inside the <configuration> tag

```

<configuration>
    <!-- <skipTests>true</skipTests> -->
    <excludes>
        | <exclude>AppTest.java</exclude>
    </excludes>
    <includes>
        | <include>MessageTest.java</include>
    </includes>
</configuration>

```

INFO: -----[ test ]-----  
INFO: --- maven-surefire-plugin:2.22.1:test (default-test) @ cms ...  
INFO: T E S T S  
INFO: Running com.example.cms.MessageTest  
INFO: Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.040 s - in com.example.cms.MessageTest  
INFO: Results:  
INFO: Tests run: 1, Failures: 0, Errors: 0, Skipped: 0  
INFO: BUILD SUCCESS  
INFO: -----  
INFO: Total time: 3.331 s  
INFO: Finished at: 2022-01-07T10:25:02Z  
INFO: erishant@gmail@ip-172-31-17-157:~/Downloads/cms\$

#### 4.12 To run a single test class **MessageTest** through the command prompt, use the below command to execute the **MessageTest**:

**mvn test -Dtest=MessageTest**

The screenshot shows the VS Code interface with the terminal tab active. The terminal window displays the following Maven command and its execution:

```
[INFO] --- maven-surefire-plugin:2.22.1:test (default-test) @ cms ---
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.046 s - in com.example.cms.MessageTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 3.331 s
[INFO] Finished at: 2022-01-07T10:25:02Z
[INFO]
```

The terminal also shows the command used: `erishant@gmail@ip-172-31-17-157:~/Downloads/cms$ mvn test -Dtest=MessageTest`.

4.13 To run a single test class **MessageTest** through the command prompt, use the below command:

```
mvn test -Dtest=MessageTest
```

The screenshot shows the VS Code interface with the terminal tab active. The terminal window displays the following Maven command and its execution:

```
[INFO] --- maven-surefire-plugin:2.22.1:test (default-test) @ cms ---
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.028 s - in com.example.cms.MessageTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 1.936 s
[INFO] Finished at: 2022-01-07T10:26:02Z
[INFO]
```

The terminal also shows the command used: `erishant@gmail@ip-172-31-17-157:~/Downloads/cms$ mvn test -Dtest=AppTest`.

4.14 Go to the **AppTest.java** file and write one more test case. Copy-paste the above test case and make changes as shown below:

Change name to **testMyPromoCode**

Change value to **0.25** and discount coupon name to **JUMBO**

The screenshot shows a Maven project structure for a 'cms' module. The 'src/test/java/com/example/cms' directory contains 'App.java', 'Message.java', 'AppTest.java', and 'MessageTest.java'. The 'pom.xml' file is also visible. In the terminal, a Maven build command has been run, resulting in a 'BUILD FAILURE' because the test methods 'testPromoCode()' and 'testMyPromoCode()' have not been implemented.

```

src > test > java > com > example > cms > AppTest.java > AppTest > testMyPromoCode()
10  {
11      /**
12      * Rigorous Test :-)
13      */
14      @Test
15      public void testPromoCode()
16      {
17          assertEquals(Double.valueOf(0.55), App.getDiscount("HELL055"));
18      }
19
20      @Test
21      public void testMyPromoCode()
22      {
23          assertEquals(Double.valueOf(0.25), App.getDiscount("JUMBO"));
24      }
25
[INFO] [INFO] BUILD FAILURE
[INFO] [INFO] Total time: 2.023 s
[INFO] [INFO] Finished at: 2022-01-07T10:26:31Z
[INFO] [ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:2.22.1:test (default-test) on project cms: There are test failures.
[ERROR] Please refer to /home/erishant@gmail/Downloads/cms/target/surefire-reports for the individual test results.
[ERROR] Please refer to dump files (if any exist) [date].dump, [date]-jvmrun[N].dump and [date].dumpstream.
[ERROR] > [Help 1]
[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]

```

#### 4.15 Use mvn test with method name in it, that is, -DTest, and specify the promo code

**mvn test -Dtest=AppTest#testMyPromoCode**

The screenshot shows the same Maven project structure and terminal window as the previous one. However, the terminal output now indicates that the test was successful, as it shows 'Tests run: 0, Failures: 0, Errors: 0, Skipped: 0'.

```

src > test > java > com > example > cms > AppTest.java > AppTest > testMyPromoCode()
10  {
11      /**
12      * Rigorous Test :-)
13      */
14      @Test
15      public void testPromoCode()
16      {
17          assertEquals(Double.valueOf(0.55), App.getDiscount("HELL055"));
18      }
19
20      @Test
21      public void testMyPromoCode()
22      {
23          assertEquals(Double.valueOf(0.25), App.getDiscount("JUMBO"));
24      }
25
[INFO] Results:
[INFO]
[INFO] Tests run: 0, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 2.032 s
[INFO] Finished at: 2022-01-07T10:31:29Z
[INFO]
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:2.22.1:test (default-test) on project cms: No tests were executed! (Set -DfailIfNoTests=false to ignore this 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
[ERROR] erishant@gmail@ip-172-31-17-157:/Downloads/cms$ mvn test -Dtest=AppTest#testMyPromoCode

```

#### 4.16 To test multiple classes at once, add multiple test names separated by a comma

**mvn test -Dtest=MessageTest, AppTest**

The screenshot shows a Java project named 'cms' in the Explorer view. The project structure includes 'src' with 'main/java/com/example/cms' containing 'App.java', 'Message.java', 'AppTest.java', and 'MessageTest.java'. The 'test' folder contains 'AppTest.class' and 'MessageTest.class'. The 'pom.xml' file is also present. In the terminal, a Maven test command is run, showing the output of the tests and a successful build.

```

[INFO] --- maven-surefire-plugin:2.22.1:test (default-test) @ cms ---
[INFO] 
[INFO] T E S T S
[INFO] -----
[INFO] Running com.example.cms.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.014 s -> in com.example.cms.AppTest
[INFO] Results:
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO] 
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 1.994 s
[INFO] Finished at: 2022-01-07T10:31:51Z
[INFO] 
erishant@gmail:~/ip-172-31-17-157:~/Downloads/cms$ mvn test -Dtest=MessageTest, AppTest

```

## Step 5: Work with the security manager

### 5.1 In the pom.xml file, add system property variables

The screenshot shows the 'pom.xml' file in the code editor. A new section for 'systemPropertyVariables' has been added to the configuration of the 'maven-surefire-plugin'. The terminal shows a Maven build command being run, but it fails due to an unknown lifecycle phase 'AppTest'.

```

<configuration>
    </configuration>
</plugin>
<plugin>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>2.22.1</version>
    <configuration>
        <!-- <skipTests>true</skipTests> -->
        <!-- <excludes>
            | <exclude>AppTest.java</exclude>
        </excludes>
        <includes>
            | <include>MessageTest.java</include>
        </includes>
        </configuration>
        <!-- <includes> -->
        <systemPropertyVariables>
            |
        </systemPropertyVariables>
    
```

```

[INFO] Building cms 1.0-SNAPSHOT
[INFO] [ jar ]
[INFO] 
[INFO] BUILD FAILURE
[INFO] 
[INFO] Total time: 0.160 s
[INFO] Finished at: 2022-01-07T10:32:27Z
[INFO] 
[ERROR] Unknown lifecycle phase "AppTest". You must specify a valid lifecycle phase or a goal in the format <plugin-prefix>:<goal> or <plugin-group-id>:<plugin-artifact-id>[:<plugin-version>]:<goal>. Available lifecycle phases are: validate, initialize, generate-sources, process-sources, generate-resources, process-resources, compile, process-classes, generate-test-sources, process-test-sources, generate-test-resources, process-test-resources, test-compile, process-test-classes, test, prepare-package, package, pre-integration-test, integration-test, post-integration-test, verify, install, deploy, pre-clean, clean, post-clean, pre-site, site, post-site, site-deploy. -> [Help]
[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.

```

### 5.2 Use the surefire.security.manager tag to work with older versions of unit tests:

<surefire.security.manager></surefire.security.manager>

The screenshot shows the VS Code interface with the Explorer, Editor, Problems, Output, Debug Console, and Terminal tabs. The Editor tab displays the pom.xml file with the following configuration:

```

<configuration>
    </configuration>
</plugin>
<plugin>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>2.22.1</version>
    <configuration>
        <!-- <skipTests>true</skipTests> -->
        <!-- <excludes>
            | <exclude>AppTest.java</exclude>
        </excludes>
        <includes>
            | <include>MessageTest.java</include>
        </includes> -->
        <systemPropertyVariables>
            | <surefire.security.manager>${surefire.security.manager}</surefire.security.manager>
        </systemPropertyVariables>
    
```

The Output tab shows Maven building the project and failing due to an undefined lifecycle phase:

```

[INFO] Building cms 1.0-SNAPSHOT
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time:  0.168 s
[INFO] Finished at: 2022-01-07T10:32:27Z
[INFO] -----
[ERROR] Unknown lifecycle phase "AppTest". You must specify a valid lifecycle phase or a goal in the format <plugin-prefix>:<goal> or <plugin-group-id>:<plugin-artifact-id>[:<plugin-version>]:<goal>. Available lifecycle phases are: validate, initialize, generate-sources, process-sources, generate-resources, process-resources, compile, process-classes, generate-test-sources, process-test-sources, generate-test-resources, process-test-resources, test-compile, process-test-classes, test, prepare-package, package, pre-integration-test, integration-test, post-integration-test, verify, install, deploy, pre-clean, clean, post-clean, pre-site, site, post-site, site-deploy. -> [Help 1]

```

### 5.3 Add a security manager in the App.java file

The screenshot shows the VS Code interface with the Explorer, Editor, Problems, Output, Debug Console, and Terminal tabs. The Editor tab displays the App.java file with the following code:

```

import java.util.ArrayList;
/*
 * Hello world!
 */
public class App {
    public static void main( String[] args )
    {
        SecurityManager
        System.out.println( "Search the candle, rather than cursing the darkness" );
        int a = 10;
        int b = 20;
    }
}

```

The Output tab shows Maven building the project and failing due to an undefined lifecycle phase:

```

[INFO] Building cms 1.0-SNAPSHOT
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time:  0.160 s
[INFO] Finished at: 2022-01-07T10:32:27Z
[INFO] -----
[ERROR] Unknown lifecycle phase "AppTest". You must specify a valid lifecycle phase or a goal in the format <plugin-prefix>:<goal> or <plugin-group-id>:<plugin-artifact-id>[:<plugin-version>]:<goal>. Available lifecycle phases are: validate, initialize, generate-sources, process-sources, generate-resources, process-resources, compile, process-classes, generate-test-sources, process-test-sources, generate-test-resources, process-test-resources, test-compile, process-test-classes, test, prepare-package, package, pre-integration-test, integration-test, post-integration-test, verify, install, deploy, pre-clean, clean, post-clean, pre-site, site, post-site, site-deploy. -> [Help 1]
[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.

```

### 5.4 Go to the pom.xml file and configure the API for JUnit 3, which is:

**java.lang.SecurityManager**

The screenshot shows the VS Code interface with a Maven project named 'CMS' open. The 'pom.xml' file is selected in the Explorer view. The terminal window at the bottom shows the following output:

```
[INFO] Building cms 1.0-SNAPSHOT
[INFO] -----
[INFO] BUILD FAILURE
[INFO] Total time: 0.160 s
[INFO] Finished at: 2022-01-07T10:32:27Z
[INFO]
[ERROR] Unknown lifecycle phase "AppTest". You must specify a valid lifecycle phase or a goal in the format <plugin-prefix>:<goal> or <plugin-group-id>:<plugin-artifact-id>[:<plugin-version>]:<goal>. Available lifecycle phases are: validate, initialize, generate-sources, process-sources, generate-resources, process-resources, compile, process-classes, generate-test-sources, process-test-sources, generate-test-resources, process-test-resources, test-compile, process-test-classes, test, prepare-package, package, pre-integration-test, integration-test, post-integration-test, verify, install, deploy, pre-clean, clean, post-clean, pre-site, site, post-site, site-deploy. -> [Help]
[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.
```

5.5 Use the **parallel** tag in configuration to execute the tests in parallel and have unlimited threads to run the configuration

**<parallel></parallel>**

The screenshot shows the VS Code interface with the same Maven project 'CMS'. The 'pom.xml' file is open, and the 'parallel' tag has been added to the 'configuration' section of the 'plugin' block:

```
<configuration>
  <skipTests>true</skipTests>
  <includes>
    <include>AppTest.java</include>
  </includes>
  <systemPropertyVariables>
    <surefire.security.manager>java.lang.SecurityManager</surefire.security.manager>
  </systemPropertyVariables>
</configuration>
<parallel></parallel>
```

The terminal window shows the same build failure message as the previous screenshot.

5.6 Add **methods** in the parallel tag and provide value for the **threadCount** as **10** to execute ten different tests

**<parallel>methods</parallel>**  
**<threadCount>10</threadCount>**

The screenshot shows a Java project structure in the Explorer view of a code editor. The project contains files like Message.java, MessageTest.java, App.java, AppTest.java, and pom.xml. The pom.xml file is open in the editor, showing configuration for the Surefire plugin. The configuration includes settings for skipTests, excludes, includes, systemPropertyVariables, parallel methods, and threadCount. The terminal view shows Maven building the project, with logs indicating a build failure due to an unknown lifecycle phase 'AppTest'. The terminal output ends with an error message about the lifecycle phase.

```

<configuration>
    <!-- <skipTests>true</skipTests> -->
    <!-- <excludes>
        | <exclude>AppTest.java</exclude>
    </excludes>
    <!-- <includes>
        | <include>MessageTest.java</include>
    </includes> -->
    <!-- <systemPropertyVariables>
        | <surefire.security.manager>java.lang.SecurityManager</surefire.security.manager>
    </systemPropertyVariables> -->
    <parallel>methods</parallel>
    <threadCount>10</threadCount>

```

```

[INFO] Building cms 1.0-SNAPSHOT
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 0.160 s
[INFO] Finished at: 2022-01-07T10:32:27Z
[INFO]
[ERROR] Unknown lifecycle phase "AppTest". You must specify a valid lifecycle phase or a goal in the format <plugin-prefix>:<goal> or <plugin-group-id>:<plugin-artifact-id>[:<plugin-version>]:<goal>. Available lifecycle phases are: validate, initialize, generate-sources, process-sources, generate-resources, process-resources, compile, process-classes, generate-test-sources, process-test-sources, generate-test-resources, process-test-resources, test-compile, process-test-classes, test, prepare-package, package, pre-integration-test, integration-test, post-integration-test, verify, install, deploy, pre-clean, clean, post-clean, pre-site, site, post-site, site-deploy. -> [Help 1]
[ERROR]

```

By following these steps, you have successfully configured the Surefire plugin for various roles. You have achieved the goal of running unit tests for your application using the test goal.