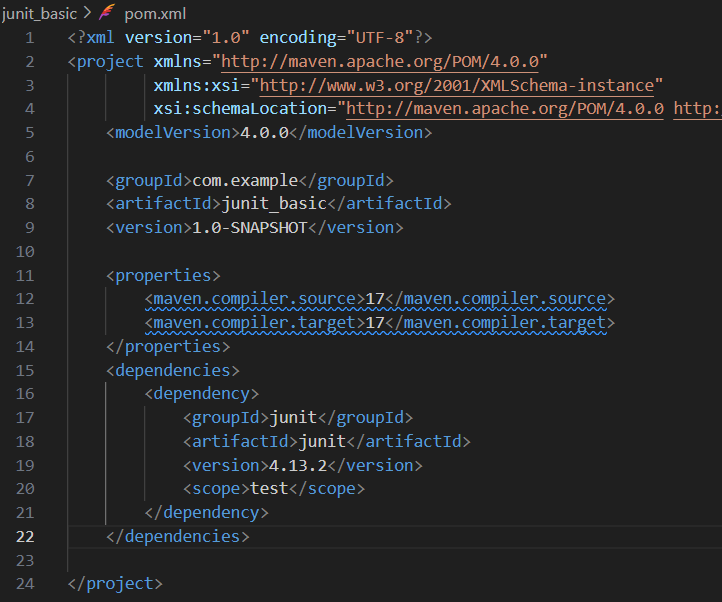
**JUNIT Testing Exercises**

Exercise 1:

* Open vscode
* In the command pallete type >java new project and select maven
* Type the project name and package name
* Add the dependencies in pom.xml

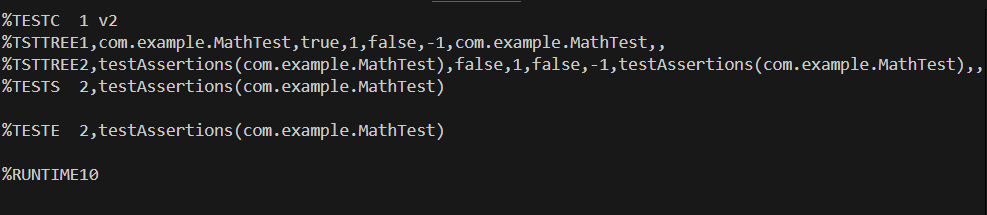
Output:



Exercise 3:

* Open vscode
* In the command pallete type >java new project and select maven
* Type the project name and package name
* Add the dependencies in pom.xml
* Create a test java file in test directory and add the given code and run the tests

Output:



Exercise 4:

* Open vscode
* In the command pallete type >java new project and select maven
* Type the project name and package name
* Add the dependencies in pom.xml

Code:

Main :

package com.example;

public class Maths {

    public int sum(int a,int b){

        return a+b;

    }

    public int prod(int a,int b){

        return a\*b;

    }

}

Test:

package com.example;

import org.junit.After;

import static org.junit.Assert.assertEquals;

import org.junit.Before;

import org.junit.Test;

public class MathsTest2 {

    Maths x;

    @Before

    public void setUp(){

        x=new Maths();

        System.out.println("Setting up");

    }

    @After

    public void tearDown(){

        x=null;

        System.out.println("tearing down");

    }

    @Test

    public void testSum(){

        assertEquals(12, x.sum(7, 5));

    }

    @Test

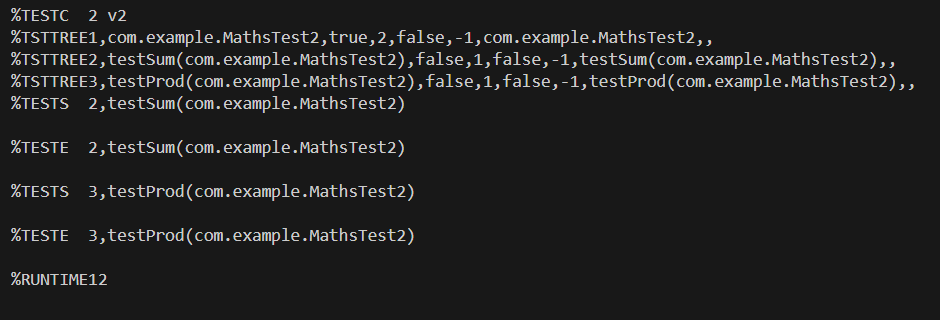
    public void testProd(){

        assertEquals(35, x.prod(7, 5));

    }

}

Output:



**Mockito Testing**

Exercise 1:

Code:

package eg;

import org.junit.Test;

import static org.mockito.Mockito.\*;

import static org.junit.Assert.\*;

public class MyServiceTest {

    @Test

    public void testExternalApiMockingAndStubbing() {

        ExternalApi mockApi = mock(ExternalApi.class);

        when(mockApi.getData()).thenReturn("Mock Data");

        MyService service = new MyService(mockApi);

        String result = service.fetchData();

        assertEquals("Mock Data", result);

    }

    @Test

    public void testExternalApiInteraction() {

        ExternalApi mockApi = mock(ExternalApi.class);

        MyService service = new MyService(mockApi);

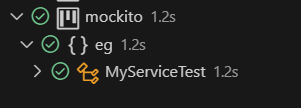
        service.fetchData();

        verify(mockApi).getData();

    }

}

Output:



Exercise 2:

Code:

package eg;

public class MyService {

    private ExternalApi api;

    public MyService(ExternalApi api) {

        this.api = api;

    }

    public String fetchData() {

        return api.getData();

    }

}

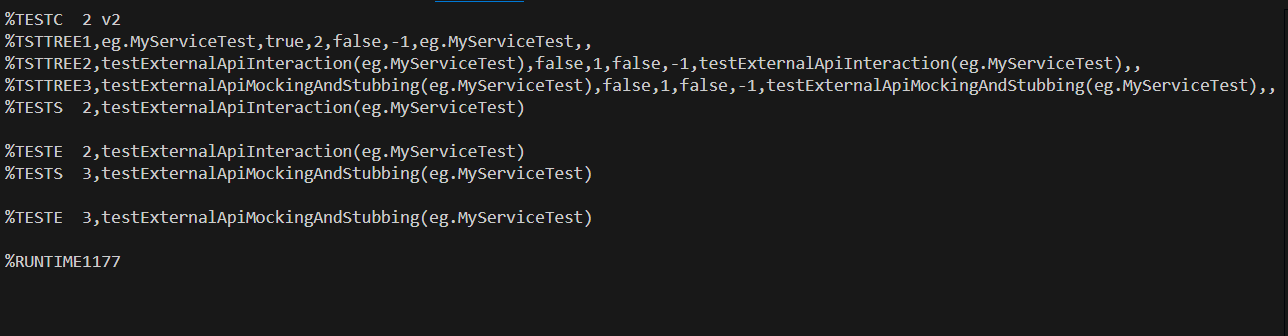
package eg;

public interface ExternalApi {

    String getData();

}

Output:



**SLFJ Testing**

Code:

<dependencies>

        <dependency>

            <groupId>org.slf4j</groupId>

            <artifactId>slf4j-api</artifactId>

            <version>1.7.30</version>

        </dependency>

        <dependency>

            <groupId>ch.qos.logback</groupId>

            <artifactId>logback-classic</artifactId>

            <version>1.2.3</version>

        </dependency>

    </dependencies>

package eg;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class LoggingExample {

    private static final Logger logger = LoggerFactory.getLogger(LoggingExample.class);

    public static void main(String[] args) {

        logger.error("This is an error message");

        logger.warn("This is a warning message");

    }

}

Output:

