**WEEK-2**

**TDD using JUnit5 and Mockito**

**1.Setting Up JUnit**

**pom.xml:**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>

    <artifactId>junit</artifactId>

    <version>1.0-SNAPSHOT</version>

    <name>junit</name>

    <url>http://www.example.com</url>

    <properties>

        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

        <maven.compiler.source>17</maven.compiler.source>

        <maven.compiler.target>17</maven.compiler.target>

    </properties>

    <dependencies>

        <!-- ✅ JUnit 5 dependency -->

        <dependency>

            <groupId>org.junit.jupiter</groupId>

            <artifactId>junit-jupiter</artifactId>

            <version>5.10.2</version>

            <scope>test</scope>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <!-- ✅ Compiler plugin -->

            <plugin>

                <artifactId>maven-compiler-plugin</artifactId>

                <version>3.11.0</version>

                <configuration>

                    <source>17</source>

                    <target>17</target>

                </configuration>

            </plugin>

            <!-- ✅ Surefire plugin for running JUnit 5 -->

            <plugin>

                <artifactId>maven-surefire-plugin</artifactId>

                <version>3.0.0-M9</version>

            </plugin>

        </plugins>

    </build>

</project>

**AppTest.java:**

package com.example;

import static org.junit.jupiter.api.Assertions.assertTrue;

import org.junit.jupiter.api.Test;

/\*\*

 \* Unit test for simple App.

 \*/

public class AppTest

{

    /\*\*

     \* Rigorous Test :-)

     \*/

    @Test

    public void shouldAnswerWithTrue()

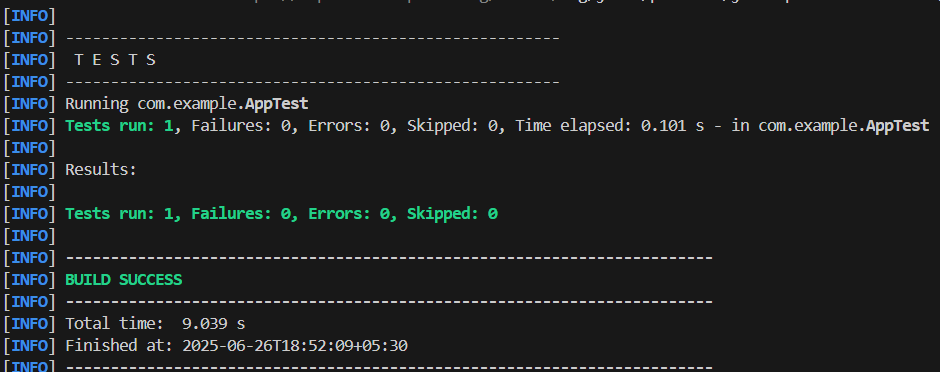
    {

        assertTrue( true );

    }

}

**Output:**



**2.Assertions in JUnit**

package com.example;

import static org.junit.jupiter.api.Assertions.assertArrayEquals;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.junit.jupiter.api.Assertions.assertFalse;

import static org.junit.jupiter.api.Assertions.assertNotEquals;

import static org.junit.jupiter.api.Assertions.assertNotNull;

import static org.junit.jupiter.api.Assertions.assertNull;

import static org.junit.jupiter.api.Assertions.assertTrue;

import org.junit.jupiter.api.Test;

/\*\*

 \* Unit test for simple App.

 \*/

public class AppTest

{

    /\*\*

     \* Rigorous Test :-)

     \*/

    @Test

    public void shouldAnswerWithTrue()

    {

        assertTrue( true );

    }

      @Test

    void testAssertEquals(){

        assertEquals(10,5+5);

    }

    @Test

    void testAssertNotEquals(){

        assertNotEquals(10,5\*3);

    }

    @Test

    void testAssertTrue(){

        assertTrue(4<10);

    }

     @Test

    void testAssertFalse(){

        assertFalse(5>10);

    }

    @Test

    void testAssertNull(){

        String name=null;

        assertNull(name);

    }

    @Test

    void testAssertNotNull(){

        String city="Hyderabad";

        assertNotNull(city);

    }

    @Test

    void testAssertArrayEquals(){

        int[] expected={1,2,3};

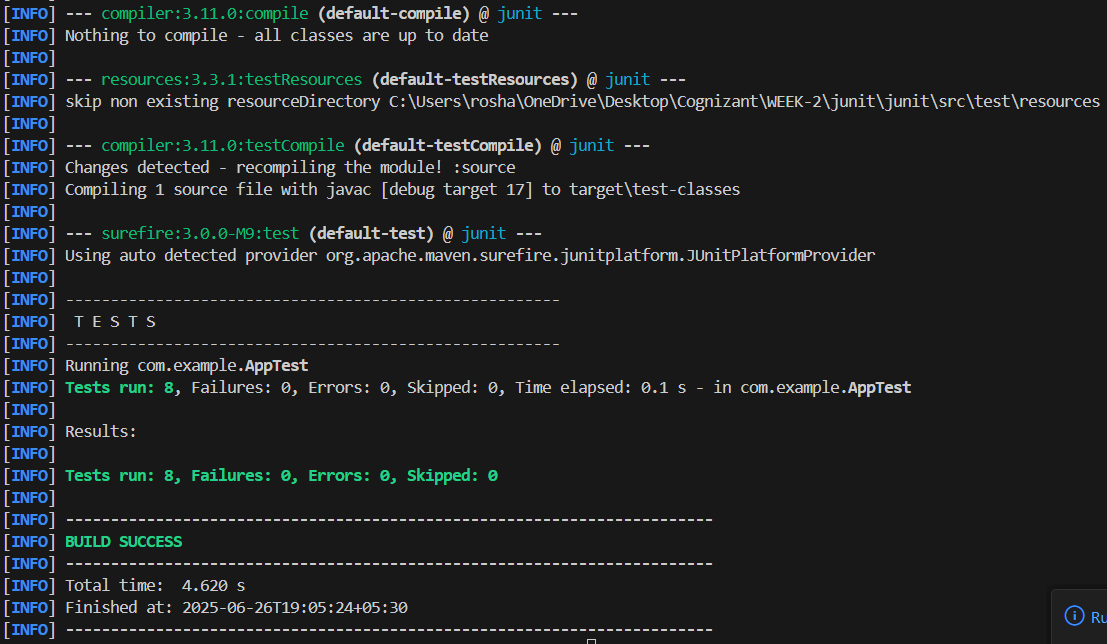
        int[] actual={1,2,3};

        assertArrayEquals(expected,actual);

    }

}

**Output:**

****

**3. Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit**

**Calculator.java**

package com.example;

public class Calculator {

    public int add(int a,int b){

        return a+b;

    }

    public int divide(int a,int b){

        if (b==0) throw new IllegalArgumentException("Cannot divide by zero");

        return a/b;

    }

}

**CalculatorTest.java**

package com.example;

import org.junit.jupiter.api.\*;

import static org.junit.jupiter.api.Assertions.\*;

public class CalculatorTest {

    private Calculator calc;

    //Setup

    @BeforeEach

    void setUp() {

        calc=new Calculator();

        System.out.println("Setup complete");

    }

    //Teardown

    @AfterEach

    void tearDown() {

        System.out.println("Teardown\n");

    }

    //Test 1

@Test

    void testAddition(){

        // Arrange

        int a=5;

        int b=7;

        // Act

        int result=calc.add(a,b);

        // Assert

        assertEquals(12,result);

        System.out.println("testAddition passed");

    }

    //Test 2

    @Test

    void testDivision(){

        // Arrange

        int a=10;

        int b=2;

        // Act

        int result=calc.divide(a, b);

        // Assert

        assertEquals(5, result);

        System.out.println("testDivision passed");

    }

    //Test 3

    @Test

    void testDivisionByZero() {

        // Arrange

        int a=10;

        int b=0;

        // Act + Assert

        Exception exception=assertThrows(IllegalArgumentException.class,() -> {

            calc.divide(a,b);

        });

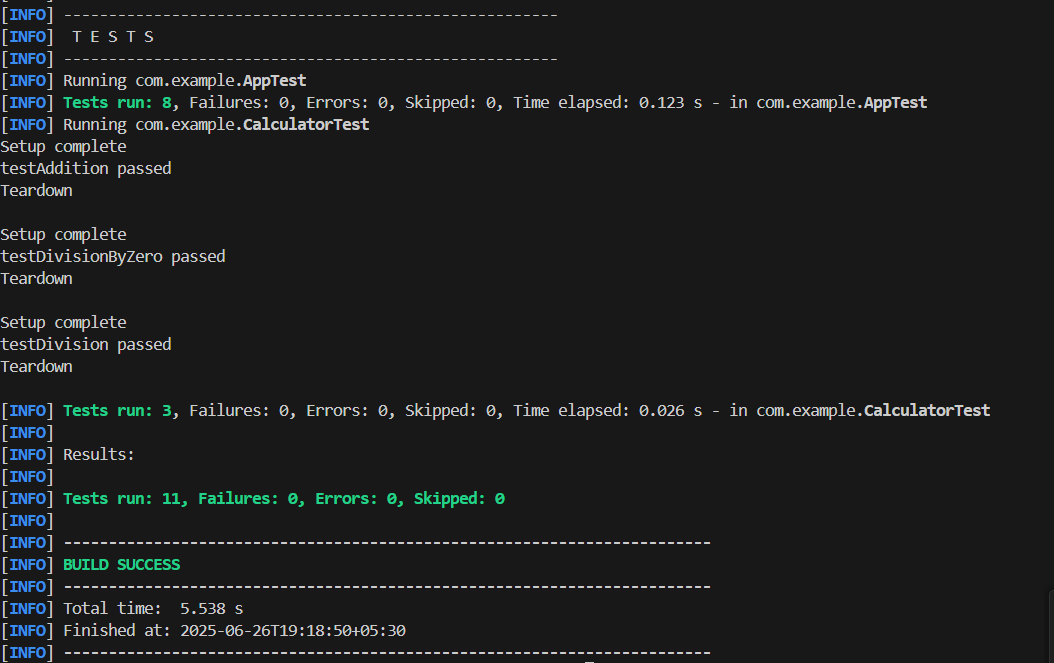
        assertEquals("Cannot divide by zero",exception.getMessage());

        System.out.println("testDivisionByZero passed");

    }

}

**Output:**



**4. Mocking and Stubbing**

**UserRepository.java**

package com.example;

public interface UserRepository {

    String findUserNameById(int id);

}

**UserService.java**

package com.example;

public class UserService {

    private UserRepository repo;

    public UserService(UserRepository repo){

        this.repo=repo;

    }

    public String getWelcomeMessage(int id){

        String name=repo.findUserNameById(id);

        return "Welcome, "+name;

    }

}

**UserServiceTest.java**

package com.example;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

import static org.mockito.Mockito.\*;

public class UserServiceTest {

    @Test

    void testGetWelcomeMessage(){

        //Mock

        UserRepository mockRepo=mock(UserRepository.class);

        //Stub

        when(mockRepo.findUserNameById(101)).thenReturn("abc");

        // Use mock in service

        UserService service=new UserService(mockRepo);

        // Assert

        String message=service.getWelcomeMessage(101);

        assertEquals("Welcome, abc",message);

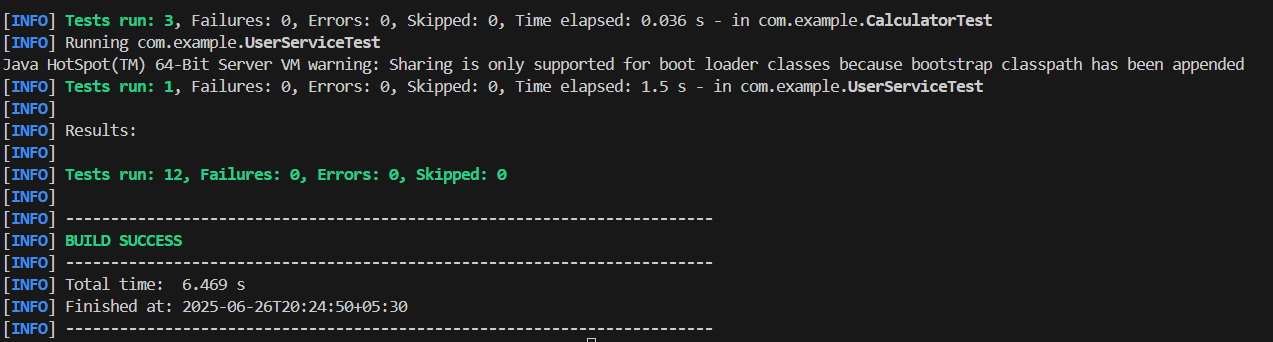
        //Verify interaction

        verify(mockRepo).findUserNameById(101);

    }

}

**Output:**

****

**5. Verifying Interactions**

**NotificationService.java**

package com.example;

public interface NotificationService {

    void sendEmail(String user, String message);

}

**UserManager.java**

package com.example;

public class UserManager {

    private NotificationService notificationService;

    public UserManager(NotificationService notificationService) {

        this.notificationService=notificationService;

    }

    public void registerUser(String username){

        if (username!=null && !username.isEmpty()) {

            notificationService.sendEmail(username,"Welcome!");

        }

    }

}

**UserManagerTest.java**

package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.ArgumentMatchers.anyString;

import static org.mockito.Mockito.\*;

public class UserManagerTest {

    @Test

    void testEmailSentOnRegistration(){

        // Arrange

        NotificationService mockNotification=mock(NotificationService.class);

        UserManager manager=new UserManager(mockNotification);

        // Act

        manager.registerUser("Ashritha");

        // Assert

        verify(mockNotification).sendEmail("Ashritha", "Welcome!");

    }

    @Test

    void testNoEmailSentForEmptyUser(){

        // Arrange

        NotificationService mockNotification=mock(NotificationService.class);

        UserManager manager=new UserManager(mockNotification);

        // Act

        manager.registerUser("");

        // Assert

        verify(mockNotification,never()).sendEmail(anyString(),anyString());

    }

    @Test

    void testEmailCalledOnce(){

        NotificationService mockNotification=mock(NotificationService.class);

        UserManager manager=new UserManager(mockNotification);

        manager.registerUser("Ashritha");

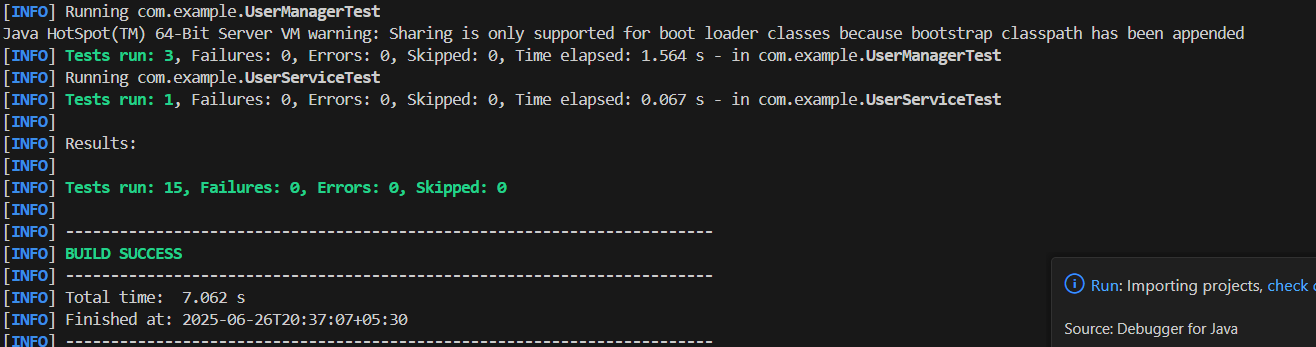
        // Check if it was called exactly once

        verify(mockNotification, times(1)).sendEmail("Ashritha", "Welcome!");

    }

}

**Output:**

****

**6.Logging Error Messages and Warning Levels**

**LoggerExample.java**

package com.example;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class LoggerExample {

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

    public void processUser(String userId){

        logger.info("Processing user with ID: {}", userId);

        if (userId==null || userId.isEmpty()){

            logger.warn("User ID is missing or empty");

        }

        try{

            simulateError();

        }catch (Exception e){

            logger.error("An exception occurred while processing user: {}",userId,e);

        }

    }

    private void simulateError(){

        throw new RuntimeException("Simulated error");

    }

}

**Main.java**

package com.example;

public class Main{

    public static void main(String[] args){

        LoggerExample logExample=new LoggerExample();

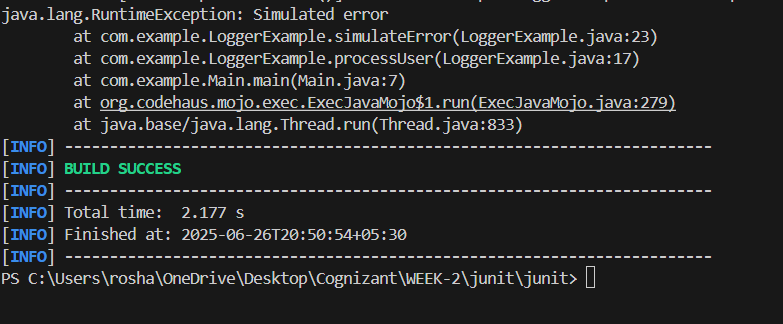
        logExample.processUser("Ashritha");

        logExample.processUser("");

    }

}

**Output:**

****