**Advanced JUnit Testing Exercises**

**Exercise 1: Parameterized Tests**

**EvenChecker.java**

public class EvenChecker {

    public static boolean isEven(int number) {

        return number % 2 == 0;

    }

}

**EvenCheckerTest.java**

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

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

import org.junit.jupiter.params.ParameterizedTest;

import org.junit.jupiter.params.provider.ValueSource;

public class EvenCheckerTest {

    @ParameterizedTest

    @ValueSource(ints = {12,1024,246,234,296,44})

    void testIsEven\_ShouldReturnTrueForEvenNumbers(int number) {

        assertTrue(EvenChecker.isEven(number), number + " should be even");

    }

    @ParameterizedTest

    @ValueSource(ints = {23,45,347,445,787,331})

    void testIsEven\_ShouldReturnFalseForOddNumbers(int number) {

        assertFalse(EvenChecker.isEven(number), number + " should be odd");

    }

}

**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 http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>

    <artifactId>demo</artifactId>

    <version>1.0-SNAPSHOT</version>

    <properties>

        <maven.compiler.source>21</maven.compiler.source>

        <maven.compiler.target>21</maven.compiler.target>

    </properties>

    <dependency>

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

    <artifactId>junit-jupiter-params</artifactId>

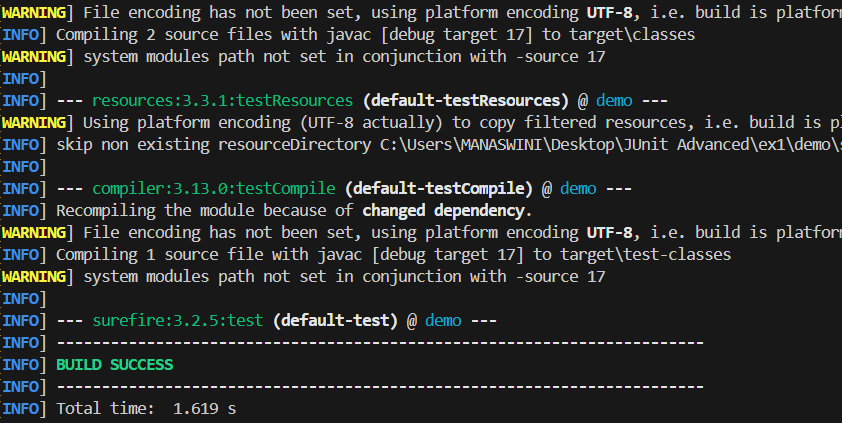
    <version>5.10.0</version>

    <scope>test</scope>

</dependency>

</project>

**Output:**

****

**Exercise 2: Test Suites and Categories**

**EvenChecker.java**

public class EvenChecker {

    public static boolean isEven(int number) {

        return number % 2 == 0;

    }

**}**

**OddChecker.java**

public class OddChecker {

    public static boolean isOdd(int number) {

        return number % 2 != 0;

    }

}

**EvenCheckerTest.java**

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

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

import org.junit.jupiter.params.ParameterizedTest;

import org.junit.jupiter.params.provider.ValueSource;

public class EvenCheckerTest {

    @ParameterizedTest

    @ValueSource(ints = {12,1024,246,234,296,44})

    void testIsEven\_ShouldReturnTrueForEvenNumbers(int number) {

        assertTrue(EvenChecker.isEven(number), number + " should be even");

    }

    @ParameterizedTest

    @ValueSource(ints = {23,45,347,445,787,331})

    void testIsEven\_ShouldReturnFalseForOddNumbers(int number) {

        assertFalse(EvenChecker.isEven(number), number + " should be odd");

    }

}

**OddCheckerTest.java**

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

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

import org.junit.jupiter.params.ParameterizedTest;

import org.junit.jupiter.params.provider.ValueSource;

public class OddCheckerTest {

    @ParameterizedTest

    @ValueSource(ints = {101,233,111,145,2467,115})

    void testIsOdd\_ShouldReturnTrueForOddNumbers(int number) {

        assertTrue(OddChecker.isOdd(number), number + " should be odd");

    }

    @ParameterizedTest

    @ValueSource(ints = {112,114,234,456,5678})

    void testIsOdd\_ShouldReturnFalseForEvenNumbers(int number) {

        assertFalse(OddChecker.isOdd(number), number + " should not be odd");

    }

}

**AllTest.java**

import org.junit.platform.suite.api.SelectClasses;

import org.junit.platform.suite.api.Suite;

@Suite

@SelectClasses({

    EvenCheckerTest.class,

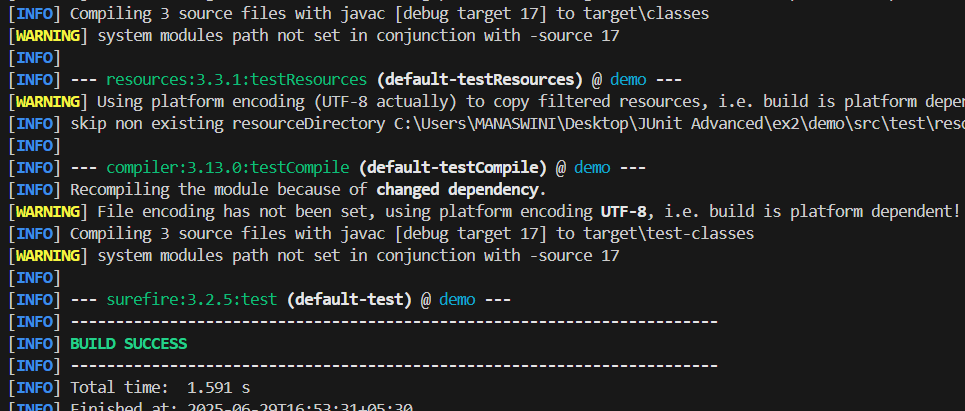
    OddCheckerTest.class

})

public class AllTest {

}

**Output:**

****

**Exercise 3: Test Execution Order**

**OrderedTest.java**

import org.junit.jupiter.api.Order;

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.TestMethodOrder;

import org.junit.jupiter.api.MethodOrderer;

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

@TestMethodOrder(MethodOrderer.OrderAnnotation.class)

public class OrderedTests {

    @Test

    @Order(1)

    void testA() {

        System.err.println("Running test A (1)");

        assertTrue(true);

    }

    @Test

    @Order(2)

    void testB() {

        System.out.println("Running test B (2)");

        assertTrue(true);

    }

    @Test

    @Order(3)

    void testC() {

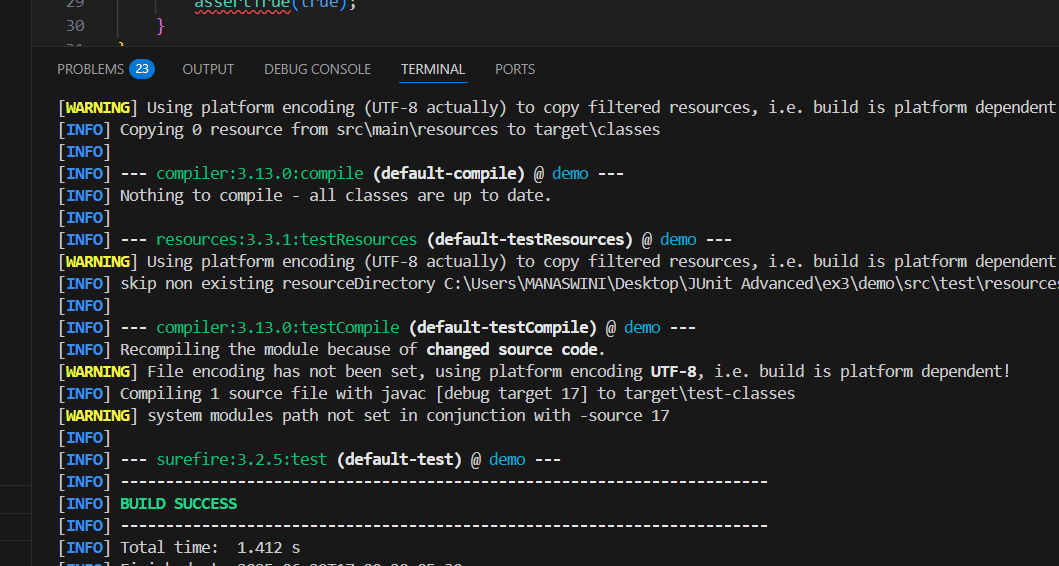
        System.out.println("Running test C (3)");

        assertTrue(true);

    }

}

**Output:**

****

**Exercise 4: Exception Testing**

**ExceptionThrower.java**

public class ExceptionThrower {

    public void throwException(String input) {

        if (input == null) {

            throw new IllegalArgumentException("Input cannot be null");

        }

    }

}

**ExceptionThrowerTest.java**

// ExceptionThrowerTest.java

import org.junit.jupiter.api.Test;

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

public class ExceptionThrowerTest {

    @Test

    void testThrowException\_WhenInputIsNull\_ShouldThrowIllegalArgumentException() {

        ExceptionThrower thrower = new ExceptionThrower();

        assertThrows(IllegalArgumentException.class, () -> {

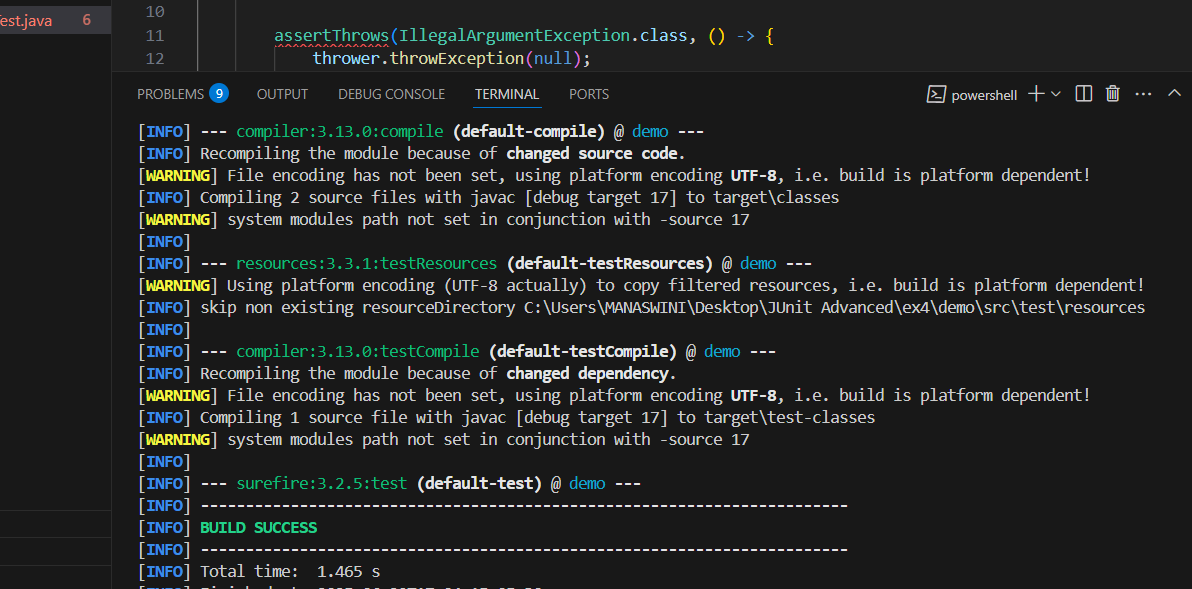
            thrower.throwException(null);

        });

    }

}

**Output:**



**Exercise 5: Timeout and Performance Testing**

**PerformanceTester.java**

public class PerformanceTester {

    public void performTask() {

        try {

            Thread.sleep(1000);

        } catch (InterruptedException e) {

            Thread.currentThread().interrupt(); // Restore interrupted status

        }

    }

}

**PerformanceTesterTest.java**

import org.junit.jupiter.api.Test;

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

import java.time.Duration;

public class PerformanceTesterTest {

    @Test

    void testPerformTaskCompletesWithin1Second() {

        PerformanceTester tester = new PerformanceTester();

        assertTimeout(Duration.ofSeconds(1), () -> {

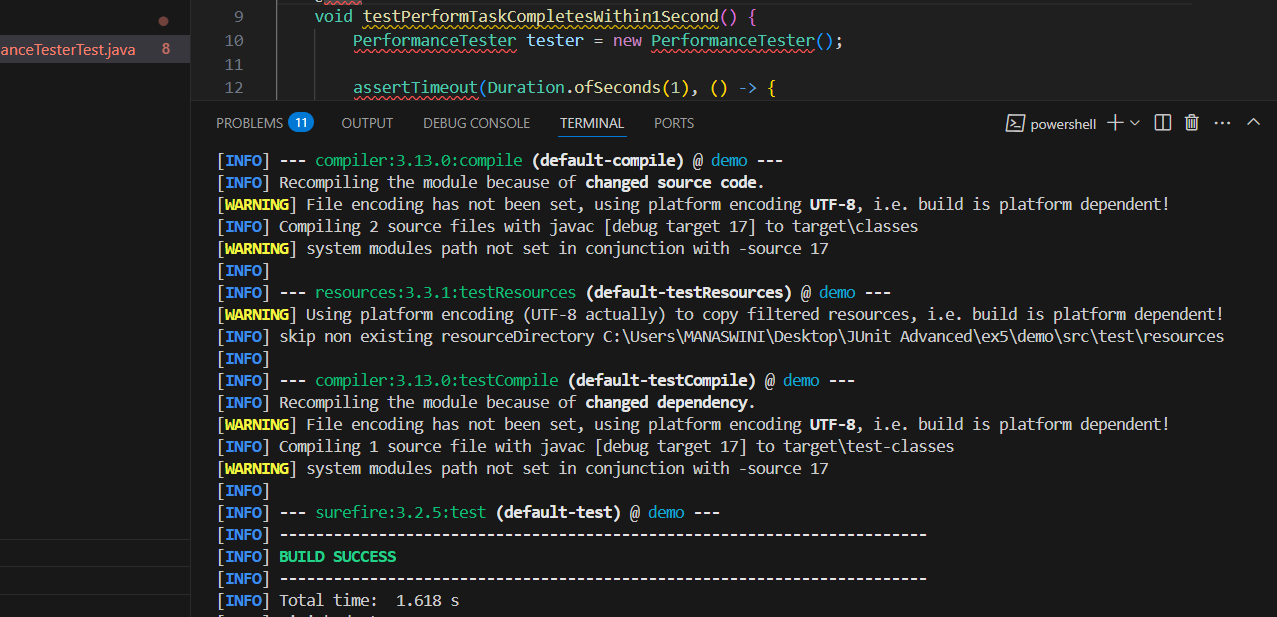
            tester.performTask();

        });

    }

}

**Output:**

****