**JUNIT Basic Testing Exercises**

**Exercise 1: Setting up Junit**

JUnit is a unit testing framework for Java programming language. JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks collectively known as xUnit, that originated with Junit.

**Step 1: Verify Java Installation in Your Machine**

First of all, open the console and execute a java command based on the operating system you are working on.

Let's verify the output for all the operating systems −

|  |  |
| --- | --- |
| **OS** | **Output** |
| Windows | java version "1.8.0\_101"  Java(TM) SE Runtime Environment (build 1.8.0\_101) |
| Linux | java version "1.8.0\_101"  Java(TM) SE Runtime Environment (build 1.8.0\_101) |
| Mac | java version "1.8.0\_101"  Java(TM) SE Runtime Environment (build 1.8.0\_101) |

If you do not have Java installed on your system, then download the Java Software Development Kit (SDK) from the following link [https://www.oracle.com](https://www.oracle.com/technetwork/java/javase/downloads/index.html). We are assuming Java 1.8.0\_101 as the installed version for this tutorial.

**Step 2: Set JAVA Environment**

Set the **JAVA\_HOME** environment variable to point to the base directory location where Java is installed on your machine. For example.

Append Java compiler location to the System Path.

|  |  |
| --- | --- |
| **OS** | **Output** |
| Windows | Append the string **C:\Program Files\Java\jdk1.8.0\_101\bin** at the  end of the system variable, **Path**. |
| Linux | export PATH = $PATH:$JAVA\_HOME/bin/ |
| Mac | not required |

Verify Java installation using the command **java -version** as explained above.

**Step 3: Download JUnit Archive**

Download the latest version of JUnit jar file from <https://junit.org/junit5/>. At the time of writing this tutorial, we have downloaded Junit-4.12.jar and copied it into C:\>JUnit folder.

|  |  |
| --- | --- |
| **OS** | **Archive name** |
| Windows | junit4.12.jar |
| Linux | junit4.12.jar |
| Mac | junit4.12.jar |

|  |  |
| --- | --- |
| **S.No** | **OS & Description** |
| 1 | **Windows**  Set the environment variable JUNIT\_HOME to C:\JUNIT |
| 2 | **Linux**  export JUNIT\_HOME = /usr/local/JUNIT |
| 3 | **Mac**  export JUNIT\_HOME = /Library/JUNIT |

**Step 4: Set JUnit Environment**

Set the **JUNIT\_HOME** environment variable to point to the base directory location where JUNIT jar is stored on your machine. Lets assuming we've stored junit4.12.jar in the JUNIT folder.

**Step 5: Set CLASSPATH Variable**

Set the **CLASSPATH** environment variable to point to the JUNIT jar location.

|  |  |
| --- | --- |
| **S.No** | **OS & Description** |
| 1 | **Windows**  Set the environment variable CLASSPATH to %CLASSPATH%;%JUNIT\_HOME%\  junit4.12.jar;.; |
| 2 | **Linux**  export CLASSPATH = $CLASSPATH:$JUNIT\_HOME/junit4.12.jar:. |
| 3 | **Mac**  export CLASSPATH = $CLASSPATH:$JUNIT\_HOME/junit4.12.jar:. |

**Step 6: Test JUnit Setup**

Create a java class file name TestJunit in **C:\>JUNIT\_WORKSPACE**

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class TestJunit {

@Test

public void testAdariable

Set the CLASSPATH environment variable to point to the JUNIT jar location.

**OS & Description**

1.Windows

Set the environment variable CLASSPATH to %CLASSPATH%;%JUNIT\_HOME%\junit4.12.jar;.;

2. Linux

export CLASSPATH = $CLASSPATH:$JUNIT\_HOME/junit4.12.jar:.

3.Mac

export CLASSPATH = $CLASSPATH:$JUNIT\_HOME/junit4.12.jar:.

d() {

String str = "Junit is working fine";

assertEquals("Junit is working fine",str);

}

}

Create a java class file name TestRunner in **C:\>JUNIT\_WORKSPACE** to execute test case(s).

import org.junit.runner.JUnitCore;

import org.junit.runner.Result;

import org.junit.runner.notification.Failure;

public class TestRunner {

public static void main(String[] args) {

Result result = JUnitCore.runClasses(TestJunit.class);

for (Failure failure : result.getFailures()) {

System.out.println(failure.toString());

}

System.out.println(result.wasSuccessful());

}

}

**Step 7: Verify the Result**

Compile the classes using **javac** compiler as follows −

C:\JUNIT\_WORKSPACE>javac TestJunit.java TestRunner.java

Now run the Test Runner to see the result as follows −

C:\JUNIT\_WORKSPACE>java TestRunner

Verify the output.

True

**Exercise 3:** Assertions in Junit

**Source Code:**

Main.java

public class Main {

public int square(int n) {

return n \* n;

}

public int cube(int n) {

return n \* n \* n;

}

}

MainTest.java

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

import org.junit.jupiter.api.Test;

class MathUtilsTest {

Main utils = new Main();

@Test

void testSquare() {

assertEquals(25, utils.square(5));

}

@Test

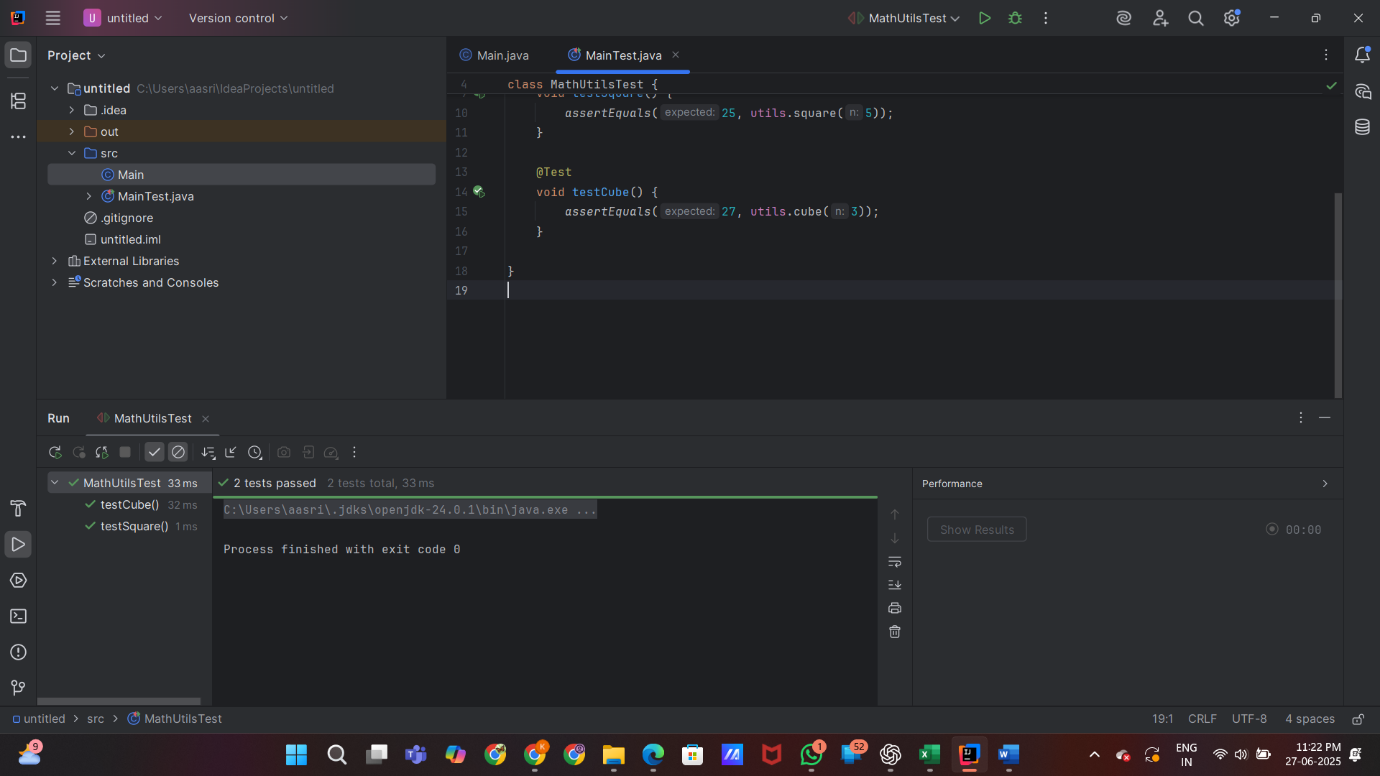
void testCube() {

assertEquals(27, utils.cube(3));

}

}

Output:

**Exercise 4**: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in Junit

**Source Code:**

Main.java

public class Main {

private double balance;

public Main(double initialBalance) {

this.balance = initialBalance;

}

public void deposit(double amount) {

balance += amount;

}

public void withdraw(double amount) {

if (amount <= balance) {

balance -= amount;

} else {

throw new IllegalArgumentException("Insufficient funds");

}

}

public double getBalance() {

return balance;

}

}

MainTest.java

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

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

public class MainTest {

private Main account;

@BeforeEach

void setUp() {

account = new Main(100.0);

}

@Test

void testDeposit() {

account.deposit(50.0);

assertEquals(150.0, account.getBalance(), 0.001);

}

@Test

void testWithdraw() {

account.withdraw(40.0);

assertEquals(60.0, account.getBalance(), 0.001);

}

@Test

void testWithdrawOverLimit() {

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

account.withdraw(200.0);

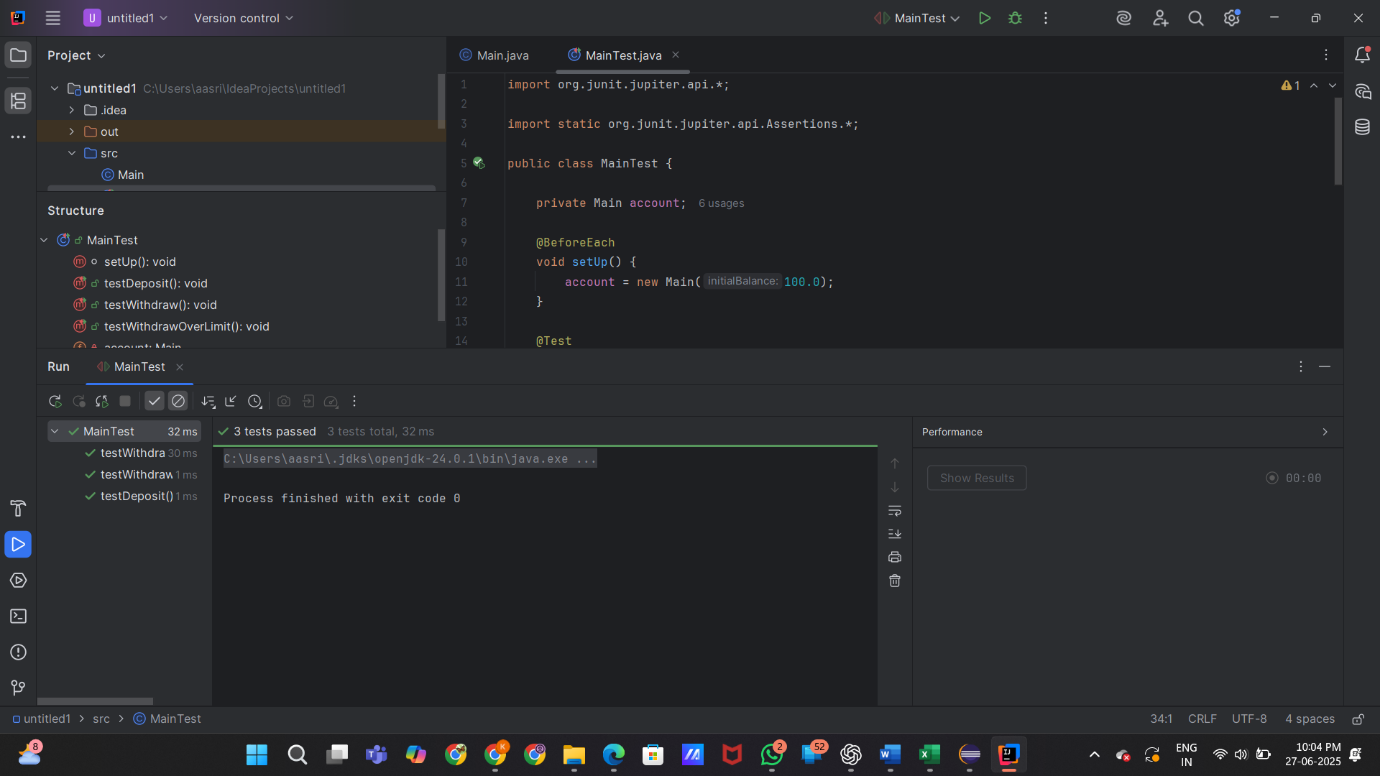
});

assertEquals("Insufficient funds", exception.getMessage());

}

}

**Output:**



**Exercise 1**: Logging Error Messages and Warning Levels

**Source Code:**

LoggingExample.java

import java.util.logging.Level;

import java.util.logging.Logger;

public class LoggingExample {

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

public void riskyOperation() {

try {

int result = 10 / 0;

} catch (ArithmeticException e) {

logger.severe("Division by zero error");

}

}

public static void main(String[] args) {

LoggingExample example = new LoggingExample();

example.riskyOperation();

}

}

LoggingExampleTest.java

import org.junit.jupiter.api.Test;

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

public class LoggingExampleTest {

@Test

void testRiskyOperationDoesNotThrow() {

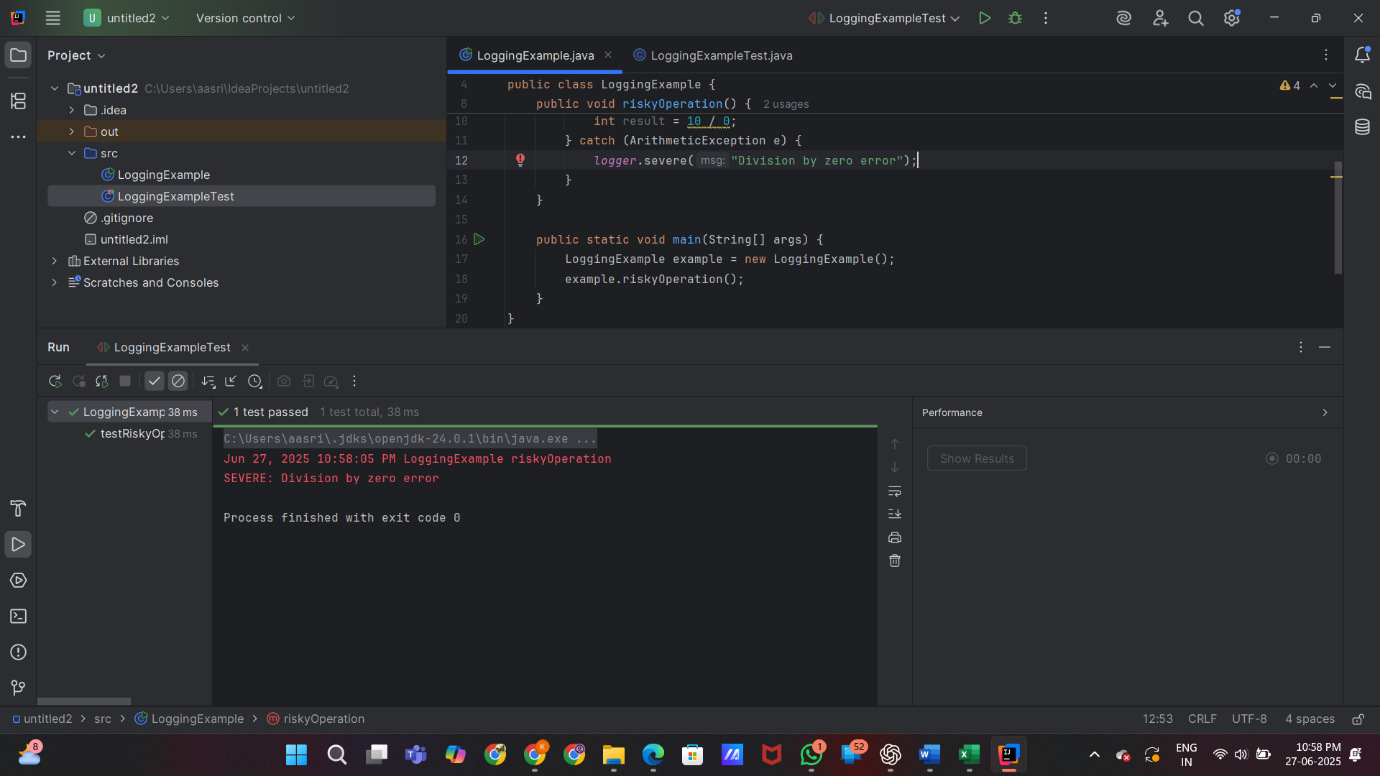
LoggingExample example = new LoggingExample();

assertDoesNotThrow(example::riskyOperation);

}

}

**Output:**

****