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. 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:

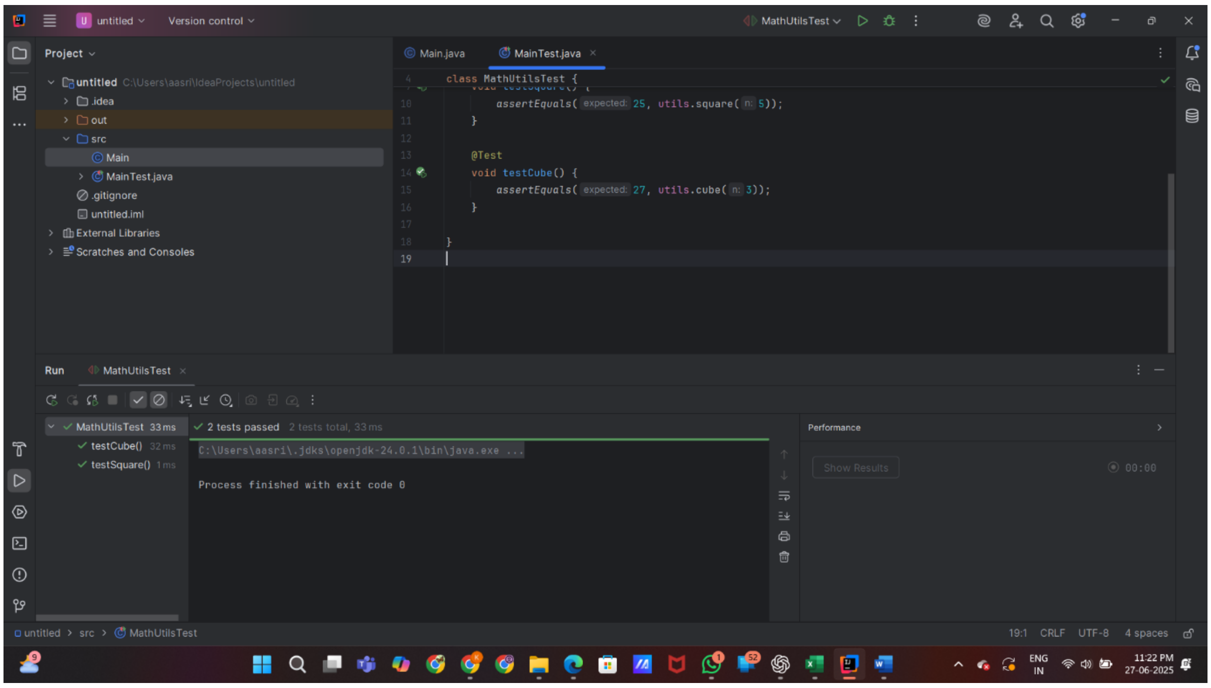
<u>Main.java</u>

```
public class Main {
   public int square(int n) {
      return n * n;
   }
   public int cube(int n) {
      return n * n * n;
   }
}
```

<u>MainTest.java</u>

```
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:



```
class MathUtilsTest {
        assertEquals( expected: 25, utils.square( n: 5));
    }

    @Test
    void testCube() {
        assertEquals( expected: 27, utils.cube( n: 3));
    }

}
```

```
✓ MathUtilsTest 33 ms      ✓ 2 tests passed 2 tests total, 33 ms
  ✓ testCube() 32 ms       C:\Users\aasri\.jdks\openjdk-24.0.1\bin\java.exe ...
  ✓ testSquare() 1 ms
                           Process finished with exit code 0
```

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

Source Code:

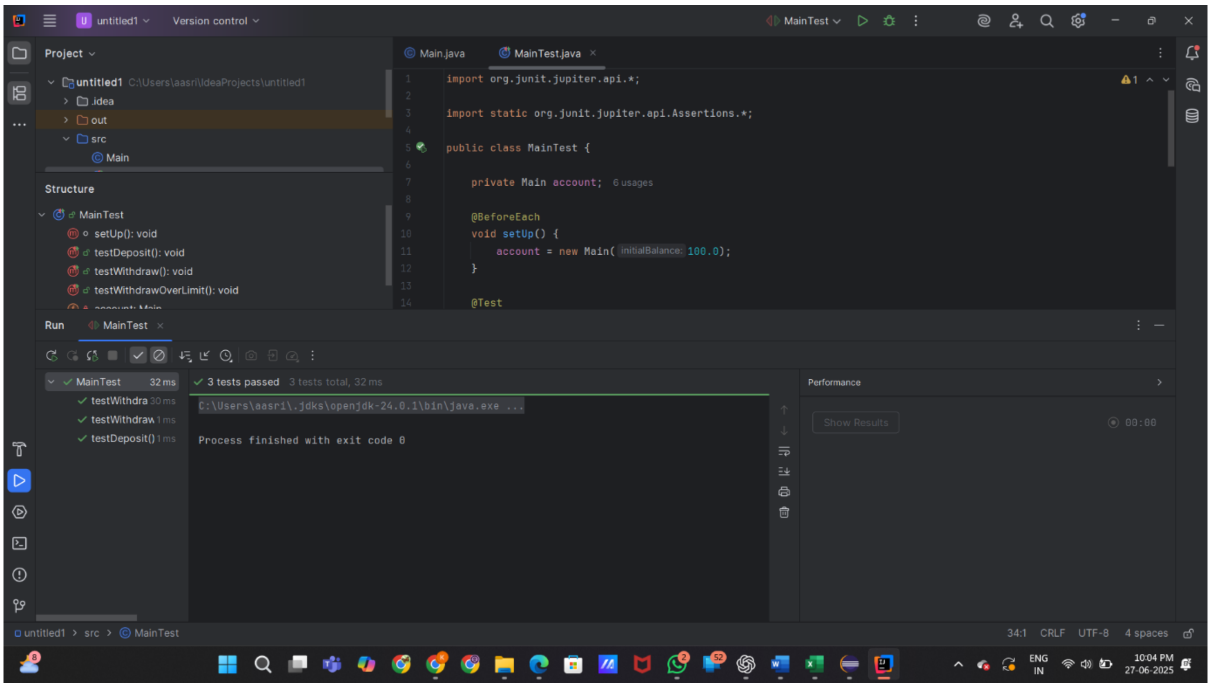
<u>Main.java</u>

```
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:



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

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

public class MainTest {

    private Main account;  6 usages

    @BeforeEach
    void setUp() {
        account = new Main( initialBalance: 100.0);
    }

    @Test
```

```
✓ MainTest 32 ms
    ✓ testWithdra 30 ms
    ✓ testWithdraw 1 ms
    ✓ testDeposit() 1 ms

✓ 3 tests passed  3 tests total, 32 ms
C:\Users\aasri\.jdks\openjdk-24.0.1\bin\java.exe ...

Process finished with exit code 0
```

Exercise 1: Logging Error Messages and Warning Levels

Source Code:

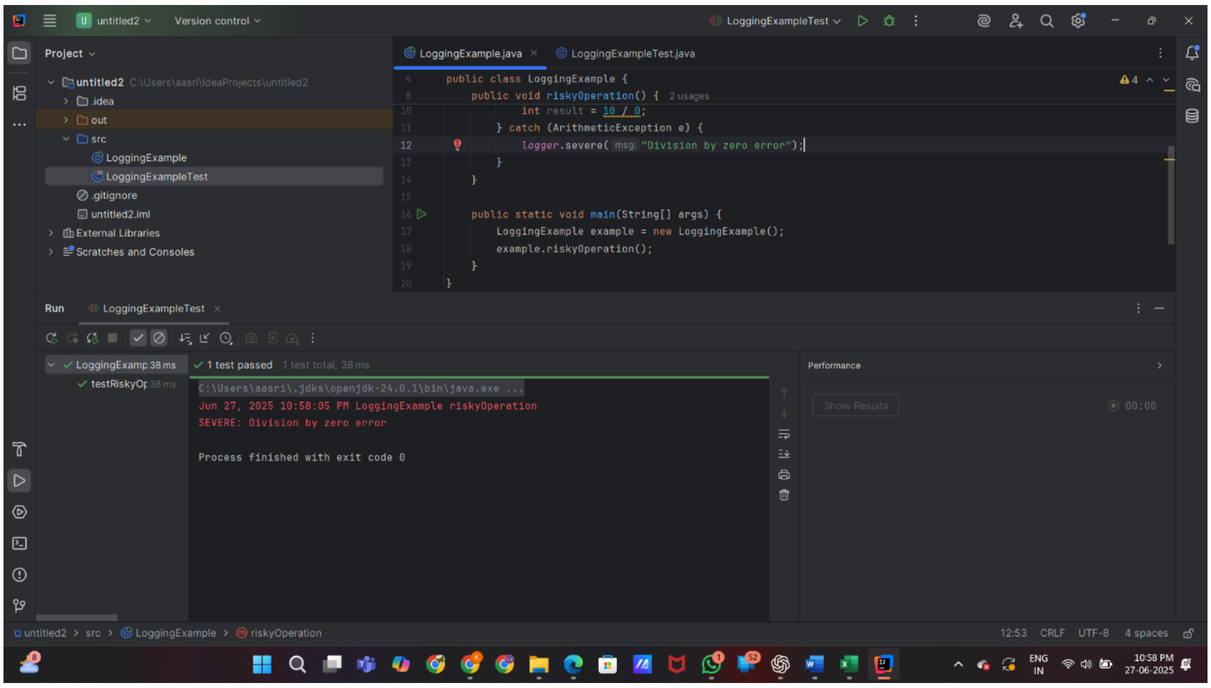
LoggingExample.java

```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

```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:



```java
public class LoggingExample {
    public void riskyOperation() {
            int result = 10 / 0;
        } catch (ArithmeticException e) {
            logger.severe( msg: "Division by zero error");
        }
    }

    public static void main(String[] args) {
        LoggingExample example = new LoggingExample();
        example.riskyOperation();
    }
}
```

✓ 1 test passed 1 test total, 38 ms

```
C:\Users\aasri\.jdks\openjdk-24.0.1\bin\java.exe ...
Jun 27, 2025 10:58:05 PM LoggingExample riskyOperation
SEVERE: Division by zero error

Process finished with exit code 0
```