BACHOLER OF COMPUTER APPLICATIONS

SEMESTER 4

DCA2202

JAVA PROGRAMMING

Unit 11

JUnit

Table of Contents

1. INTRODUCTION

This digital age is unfolding in many dimensions rapidly. The speed of progress and development has been boosted, increasing at an exponential rate. Programming or coding has become an essential part of development. Coding has made our everyday life much easier than before.

Java is one of the most used, simple programming languages that is used for creating, compiling, and debugging programs easily.

It is based on an object-oriented programming model, and it is much simpler to use and deploy.

STUDY NOTE

The original name of the Java programming language was Oak. But the Sun's marketing changed the name to Java after finding out that another computer company has already taken up Oak.

Java has evolved continuously with the rise in technology and has been used extensively by major companies like Google, Uber, Netflix, Spotify, Amazon, Pinterest, Instagram, and many more. Most websites and applications have their core programming in Java.

Why is Java invaluable to programmers?

The features elaborated below are one of the many reasons that make it stand out from the cluster of languages.

- There is no restriction to the programmers to write and run code on a single platform. Many virtual platforms are available for writing and running code online. Moreover, several IDEs are present that assist the programmers in writing code in a much easier way.
- Java is a simple language and does not have any complex features such as operator overloading, explicit memory allocation, multiple inheritances, and so on.
- Programming in Java is useful for programmers because it is safe and secure. It is not easy to run past the protective system of Java.
- Java is a reliable language. It runs an extensive check on each program or piece of code to make sure it is completely free of errors.

- Garbage collection, Exception Handling, and memory allocation are some of the best features of this programming language.
- For the maximum utilisation of CPU, Java allows multithreading, which means concurrent execution of more than two parts of the Java program.
- The programs that are created on Java can easily be distributed to other systems. This comes in handy when programmers want to create distributed applications.
- Programmers can develop and run codes online and can access them easily through web applications.
- Online forums, HTML forms, and many applications require a server-side application. Java is extremely useful for developing and coding these kinds of server-side applications.
- Many coders take the help of Java to solve complex programs.

What are Java frameworks?

To make coding applications and to solve problems easier, programmers often take the help of sets of prewritten code to save time and reduce the complexity of the programs. These are known as Java frameworks.

So, we can say that a framework is a platform that is used to develop software applications. We can say that it serves as the foundation for specific programming. Frameworks support several elements and can be accessed through an API that is already in the framework.

In the last unit, we talked about Collections. As we already learned, it is a framework in Java that assists the programmers in operations performed on data such as sorting, searching, insertion, deletion, and manipulation.

In this unit, we are going to learn about another framework of Java, JUnit. This unit is given a piece of complete and in-depth information about JUnit. You can take the help of this unit but remember, practice along with theory will be the best way to master this topic.

1.1 Learning Objectives

After studying this chapter, you will be able to:

- *Understand the Concept of JUnit*
- *Explain Unit Testing and JUnit*
- *Explain Writing Test*
- *Describe Testing Code in JUnit*
- *Elaborate JUnit Assertions*

2. Junit: Overview

In today's world, there is an unlimited number of software and applications that are present, and some of them are still being developed. With the increased demand for digitalising everything, there will be numerous software and application that will be created to make the complex processes much easier.

Providing a great user experience is extremely important, and for that, the concerned application or software must work and process smoothly, without any interruptions, errors, and bugs. To ensure this, testing is done so that the application of the software contains as few bugs as possible.

STUDY NOTE

In JUnit, you actually have the option to stop the execution of a certain test in the case when timeout exceeds the required limit. You can measure the actual time taken to run a code.

There are usually two types of unit testing:

- manual testing
- automated testing

It works fine; it is a small application, and not too many changes are needed. But the problem becomes bigger when the code is more complex and too lengthy. It is just not feasible to run the whole application through tests every time when we make a small change in the code.

In bigger and complex applications, manual unit testing becomes too tiresome for the programmer. In developing an application, it is not possible to tests small portions of code every time the programmer makes a change and compiles the application. For this purpose, automated testing was introduced.

What is Automated Testing?

Why do you code? To let the computer, do the work for you, right. You give certain instructions to the computer to perform a certain task at the time of a specific event. You have to do certain work to ensure that the code you write is error-free.

For that task, you again write a code, therefore give some instructions to the computer, that is, to test the entire application or code block by block every time there is a change, or the programmer compiles the code.

All in all, an automated test is a set of code written and compiled by a programmer to test the code block by block every time the programmer makes any small change in it.

This reduces the time and effort of the programmer and ensures that the application is bug-free too. It is very useful in cases where the programmer codes lengthy applications. Lengthy codes often require too much edit or changes. An automated test is perfect for such cases.

The theory and the concept of tests are elaborated further in this section. We will know how it is essential in developing error-free software. But first, let us dive into the rich content of JUnit and learn more about this popular framework for Java.

What is JUnit?

JUnit can be referred to as an open-source unit testing framework for Java programming. The developers can write or create tests for their software or applications using this framework. Erich Gamma and Kent Beck were responsible for the creation of such a useful platform.

From the name itself, unit testing, it can be deduced that it is used to test small portions of code. The developer has to write and test the unit test in the beginning. After this, they can continue and run all the blocks of code. It is considered successful if all the blocks of code pass through it.

In case the programmer has to make any changes, say like adding a piece of code, they have to re-execute all the portions of the code to make sure that the flow is not broken.

But do you understand testing? We will continue with our framework but let us first understand how it works.

What do you understand by testing?

Programmers have to create tests and run their code in small portions, but what are the tests? If we have to write a good and clean code, we should know what it is, or just mugging up things will do us no good.

For instance, let us assume that you are programming a good light sensing device that should know to blink twice when the light is off and blink once if the light is on. If it does not do so, we know there is an error in the code. What do we understand from this?

The code you are programming should be very specific and should meet the requirements to achieve the desired target or result. So, the code will be correct if it meets certain requirements.

To achieve this correctness of the code, we have to make sure that the code meets the specifications, and for that, we do the testing of code.

By the above example, I hope that we're clear on the requirements for testing. So, for testing the above code, it is imperative that the light sensor blinks twice in case of no light and blinks once in the presence of light. If it does so, it meets the specifications of the application, and the code is passed!

So, to sum it all up, the process goes like this. A programmer usually writes the code, compiles it, executes it, and then feeds it an input to ensure that it gives the desired or expected results. In our case, we will feed the input, say light is present, then the light sensor should blink once. This is the expected output, and the code is tested and passed.

When we inject or feed input to a program or code and receive inspected outputs, this process is known as black-box testing. The code or application that we are testing is referred to as black-box. The internals are hidden for the application or the black box.

Finally, we can say that we feed input to a black-box, the application or code, and then the received output or results are matched with the expected output. The tests of this measure are called unit tests. A small portion of the code is tested one at a time; that is why it is called unit tests.

Although unit tests are very popular and basic, this is not the only one. There is one more called integration test. This type of test interacts between several components and systems at once. But in this unit, we will be covering and focusing on unit testing only.

Now, in unit testing, we must test each section of code manually. If the programmer makes any change or addition to the code, they have first compiled it again. Then every portion is tested again to see that the flow is not broken, and the system or application is still giving the desired results.

The above process is true even for a very minute change. Testing the code every time is critical so that we may ensure that there is no bug in our code and the rest of the application is still working fine as before.

For instance, we make a small change in the code, and then we test it if the previous functions and portions of the program are still giving the same output. If the output comes out differently, then we know that there is a break in the flow. These types of bugs are called regressions.

To ensure that no previous features of the code are compromised, we run the tests of the code even if we have to make the smallest change. So, every time a programmer compiles the code, it is critical to run through the test to make sure it is still giving expected results.

2.1 Unit Testing And Junit

As we learned in the previous section, unit testing is the way of testing or verifying a code or an application block by block by creating another method or function. The term 'unit' has been defined long before the object-oriented programming case.

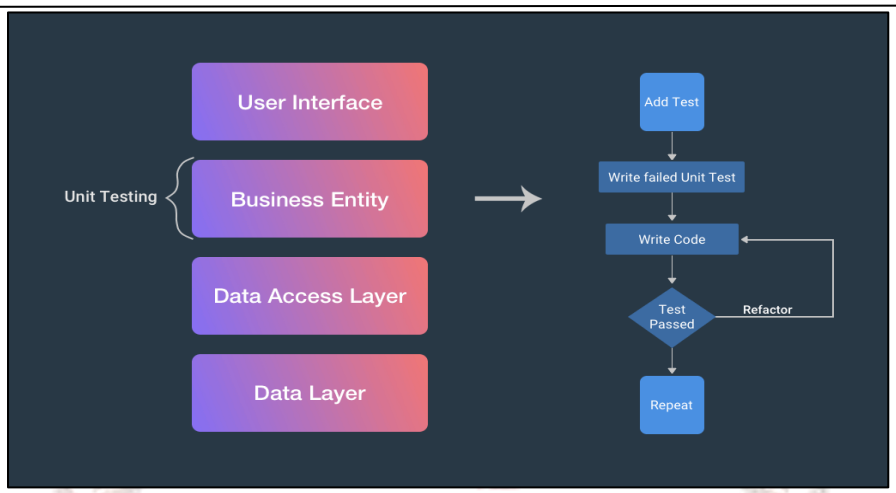
This unit testing is performed in the development phase of the application by the developers to ensure no bugs are present in the code. Unit testing is done by verifying code blocks to ensure that every block performs the function it is supposed to do.

STUDY NOTE

As unit testing is used to make sure that the application runs smoothly and without any error, some developers write the test first and code afterward. This is known as TDD (Test-Driven Development)

This is a good way to identify the errors. Usually, when you run a lengthy code, it becomes very hectic to locate the source of the error. In unit testing, by taking small chunks of code, the errors can be easily identified and removed, saving both the effort and time of the developer.

The unit testing extracts or pulls off the Java class or object. It is called unit testing because the code or the application created by the programmer is verified in small chunks or blocks manually whenever the programmer makes a small change or compiles the program.



***Source**: simform.com*

Fig 1: Unit Testing

We can understand why unit testing is so essential by going through the points given below.

Importance of Unit Testing

Whenever the coder adds or changes the code and compiles it, it is essential that the programmer verifies it by giving it certain inputs and then matching the output. It is common for an application to contain a regression bug if it is not tested to see that every feature gives the desired outputs. So, by unit testing, you can be ensured that there is no bug in your application, and it is completely fine and gives the desired output.

If the code contains bugs, then the same will reflect on the main application. The bugs can create a lot of errors and can hamper the functioning of the system. It will disrupt the whole function and render the system useless. Not to mention the harm it causes to the user's data.

On the other hand, a well-written bug-free code that has been tested and verified multiple times may lead to a great application that works seamlessly. Now bugs and errors free code will give a great user experience to the customer, and it will also enhance the functionality of the application.

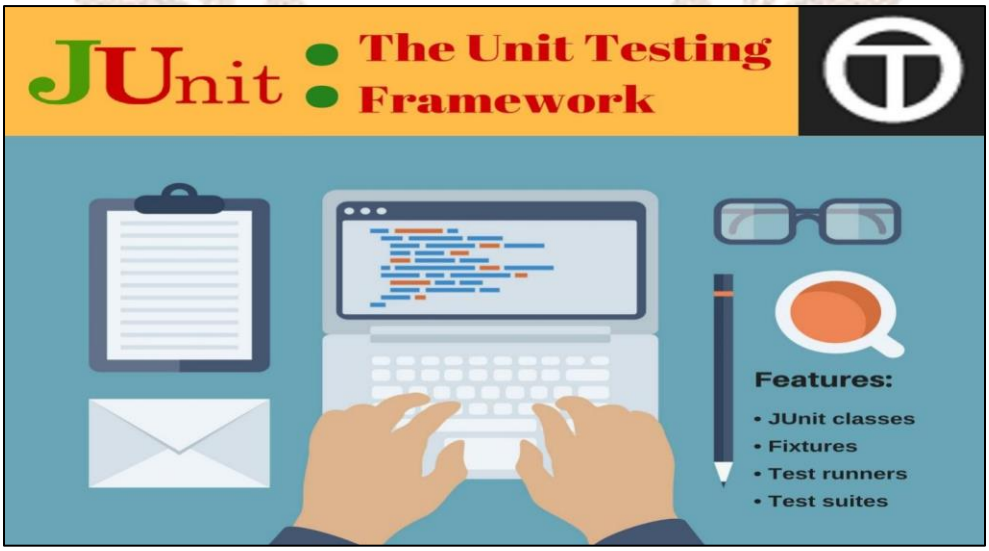
By identifying and removing the bugs in the development phase of the application, will helps to save a lot of money for the developers.

Workflow of unit testing

- The process of unit testing is simple and can be completed in a few steps.
- The first step that a developer has to take towards unit testing is creating test cases.
- The developer has to review and rewrite the production code multiple times to finalise the perfect code that satisfies the test cases.
- The developer then uses a fixed set of objects which serves as a baseline for testing out the software.
- The final step is to execute the codes.

JUnit

There are many tools for unit testing, but the most popular one is JUnit. It is a free tool that is used for testing the Java programming language. To identify the test methods, which one is appropriate or perfect, JUnit provides assertions for the same. JUnit verifies the test data at the beginning and then gets started with the input of blocks of code, one at a time.



Source: *testorigen.com*

Fig 2: JUnit

JUnit is an open-source project that consists of discrete elements. JUnit platform is one such element that serves as the base or the foundation level that enables the developers to launch different testing applications.

The JUnit 5 version contains several exciting innovations, intending to support new features in Java 8 and above and enable many different testing styles. It follows an advanced modular approach which in turn makes it easier to extend the APIs. It leverages features like lambda functions and added features like organizing, describing, and executing tests.

This is all about JUnit and unit testing. Now let us take a look at the process of writing test codes.

2.2 Writing Test

What is a test in JUnit?

A code has the sole purpose of testing. It can be defined as a method that is inside a class that has only one function to be used for testing. This class is called a test class. To create a method in the test class, we must use the notation '@test' to define that it is a test method.

This test method is used to verify the code or application that comes for testing. In order to check the output, whether it is the same as the expected result, we use the assert method that is provided by JUnit. This determines if the code is functioning as expected or not. The statements associated with the assert method are called assert statements.

STUDY NOTE

Whenever we are testing in JUnit, if by chance there is an error by our side, then JUnit will notify the developer of that invalid situation due to which no output is available.

If the output of the code does not match with the expected results, then the statements that show that the test is failed are defined in the assert statements. The statements should not be vague. Clear and direct messages ensure that the developer can identify the errors and remove or rectify them.

Naming Rules of the Test method and Test class

Whenever you are defining a test method or a test class, be very specific in what name you are assigning to each of them. Vague names can be confusing and will take more time and effort to run a simple code.

On the other hand, specific names will be able to give you a clear picture of what the test is and what should happen if the test fails or passes.

For example, in creating a test method name, it should tell you what to expect, so for a test that should create a list, the naming can be done in the following way:

testListWithLength20

This test method will tell you that by running that particular piece of code, a list should be generated if the test is passed.

JUnit annotations

Some of the annotations of JUnit5 in comparison to JUnit 4 are:

- **@Test:** It tells the computer, and the user that the method used is a test method.
- **@TestFactory:** denotes a method that's a test factory for dynamic tests
- **@Tag:** declares tags for filtering tests.
- **@BeforeEach:** This is used for the preparation of the test environment. It is executed before every test (previously @Before).
- **@AfterEach:** This is used for the activities that take place after the test, like cache cleanup, restoring the settings to default values, and so on. All this is executed after the test is completed (previously @After).
- **@RepeatedTest(<Number):** It works just like the test method but has one more feature. It repeats the test the number of times the developer mentioned. This annotation saves a lot of time for the developers.
- **@DisplayName("<Name>"):** The name can be used to explain what the test is supposed to do in a proper way. Proper space can be used to form a good sentence.
- **@AfterAll:** This refers to the method that is executed after all the tests defined are performed (previously @AfterClass). It can be used to clean up the cache or disconnecting for the database, etc.

- **@BeforeAll:** This refers to the method that is executed before all the tests defined are performed (previously @BeforeClass). It can be used to connecting to important databases, preparing for the test environment, etc.
- **@Nested:** The test class can be nested. It is used when a certain activity is to be performed.
- **@ExtendWith:** registers custom extensions.
- **@Disable:** disables a test class or method (previously @Ignore).

Self-Assessment Questions – 1

1. _____ is that language that is formed on an object-oriented programming model, and it is easy to established and utilised by programmers.
2. _____ framework is employed for evaluating codes of Java.
3. Testing is used to remove _____ bugs.
4. Manual testing is preferred over automated testing. [True/False]
5. Testing is the process of checking whether the application is giving desired output or not. [True/False]
6. Which of the following can be considered as the benefit of manual testing?
   A. Bugs can be easily found and removed
   B. The programmers spend more time refining their code
   C. Both A and B
   D. None of these

3. TESTING YOUR CODE

After everything is done perfectly, it is finally time to look into the steps to test the code. All the steps are in accordance with JUnit5:

- First, create a new and simple maven project.
- Give the artifact Id an appropriate name.
- Create and name a package in the Java and the test folder of the main folder.
- To use JUnit, you have to update your pom file. Add the group id, artifact id, and other necessary details to access JUnit.
- Update your project in the maven folder by selecting update project in the maven folder.
- Create a Java class in the package you created.
- Select MyClass in the Java editor and select the option of creating a new JUnit test case.
- Select the test directory and ensure that the test class is created.
- Select the test methods that you want to be created.
- To run your test, right-click on the test class and select the option Run-As.
- The successful and failed tests will be shown in the window.
- If a test fails, you can identify the issue and rectify it. Make sure to run it again so that the code is error and bug-free.

STUDY NOTE

An assert statement gives a Boolean value after comparing the actual and expected output. There are several methods that can be used for comparing the outputs.

These are the complete steps that are required to write a test and run it perfectly.

3.1 Junit Assertions

Assertions in this context mean to compare and present a fact. When we run a test, we give input to the system, and as expected, it produces an output. But we can decide if the output is correct or not only if we compare it with the expected results.

If the results are matched, then the test is passed; if the results are different, then that means the test is failed. The developer has to identify the errors and bugs and remove them. It is

considered better if the codes are run again through the system to make sure nothing is out of order.

In JUnit, we can use the assert class. There are several methods in the asset class that the developers can use for testing.

One of the methods is assertEquals. This method includes three parameters. You can display any message you want in case the test fails in the first parameter. The first parameter is optional. You can input the values you are expecting in the second parameter. The third parameter will have the actual value.

Other methods of the assert class include assertSame, assertTrue, assertNotEquals, assertNotNull, assertFalse, and assertNull.

Example:

```
String obj1="Junit";
String obj2="Junit";
assertEquals(obj1,obj2);
```

3.1.1 Sample logic to write an assertion

Expected Output:

True

Other than this there are several annotations that can be added to the codes like @Test, @Before, @AfterClass, @BeforeClass, and @Ignore.

Example:

Program to demonstrate usage of @After, @Before, and @Test by implementing division of 2 numbers

This is a Java file named MathProvider.java with a method called divide.

```
package unext.java_examples.JUnit_example1;

public class MathProvider {
	public MathProvider(){}
	public int divide(int firstNumber, int secondNumber)
	{
		return (firstNumber / secondNumber);
	}

}
```

3.1.2 MathProvider.java

The below code to specifies the Junit Test, MathProvider_Junit.java.

```
package unext.java_examples.JUnit_example1;

import static org.junit.Assert.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;


public class MathProvider_Junit {

   MathProvider provider;
   @Before
   public void setUp() {
        System.out.println("Before test, Creating MathProvider object.\n");
        provider = new MathProvider();
   }
   @Test
   public void divide()
   {
        System.out.println("Starting test " + new
Object(){}.getClass().getEnclosingMethod().getName());
       int firstNumber = 10;
       int secondNumber = 0;
       assertEquals(firstNumber / secondNumber, provider.divide(firstNumber, secondNumber));
        System.out.println("Ending test " + new
Object(){}.getClass().getEnclosingMethod().getName());
   }

    @After
    public void tearDown() {
         System.out.println("\nafter test");
    }
}
```

3.1.3: MathProvider_Junit.java

The below code is a Java file named TestRunner.java.

```
package unext.java_examples.JUnit_example1;

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) {
		//This result object has many methods and it is very useful
		//Type result and press dot, all the methods will display
		//This statement is to load all type of results in the result object
		Result result = JUnitCore.runClasses(MathProvider_Junit.class);
		//Here it is getting the run count from the result object
		System.out.println("Total number of tests " + result.getRunCount());
		//This is to get the failure count from the result object
		System.out.println("Total number of tests failed " +
result.getFailureCount());

		for(Failure failure : result.getFailures())
		{	//This will print message only in case of failure
			System.out.println(failure.getMessage());
		}
		//This will print the overall test result in boolean type
		System.out.println(result.wasSuccessful());
	}
}
```

Output:

```
Before test, Creating MathProvider object.

Starting test divide

after test
Total number of tests 1
Total number of tests failed 1
/ by zero
false
```

JUnit4 vs JUnit 5

JUnit4 has everything contained in a single file whereas JUnit5 is divided into 3 subparts: JUnit Platform, JUnit Jupiter, JUnit Vintage.

- **JUnit Platform:** To be able to launch JUnit tests, IDEs, build tools or plugins need to include and extend platform APIs. It defines the TestEngine API for developing new testing frameworks that run on the platform.
  It also provides a Console Launcher to launch the platform from the command line and build plugins for Gradle and Maven.
- **JUnit Jupiter:** It includes new programming and extension models for writing tests. It has all new JUnit annotations and TestEngine implementation to run tests written with these annotations.
- **JUnit Vintage**: Its primary purpose is to support running JUnit 3 and JUnit 4 written tests on the JUnit 5 platform. It's there is backward compatibility.

The below test class shows are an example to use the @test methods.

Program to demonstrate @Test method with an example

This is a Java file named MathProvider.java with a method called divide.

```
package unext.java_examples.Junit_example2;

public class MathProvider {

	public MathProvider(){}

	public int add(int firstNumber, int secondNumber)
	{
		return (firstNumber + secondNumber);
	}
	public int multiply(int firstNumber, int secondNumber)
	{
		return (firstNumber * secondNumber);
	}
	public int divide(int firstNumber, int secondNumber)
	{
		return (firstNumber / secondNumber);
	}

}
```

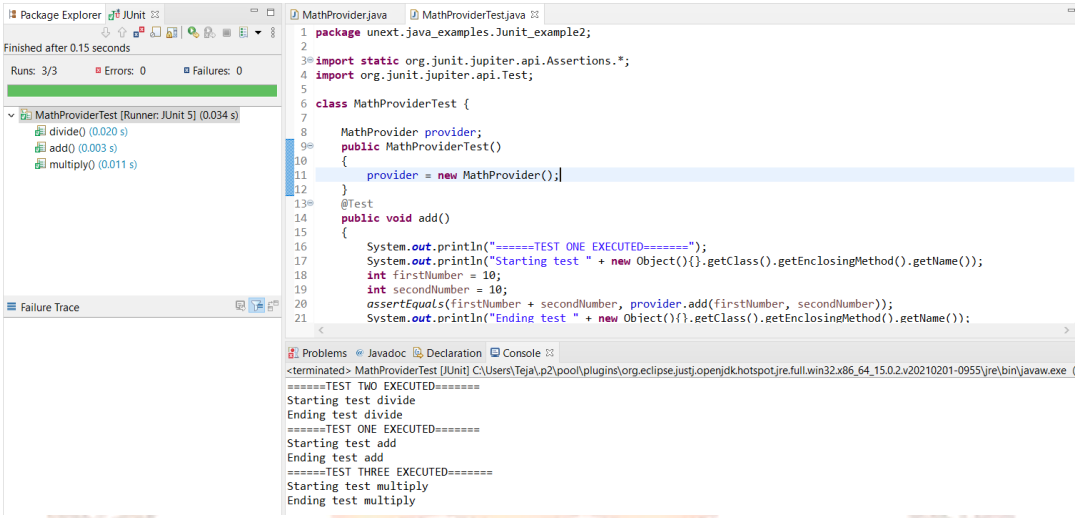
3.1.2 MathProvider.java

The below code, MathProviderTest.java, specifies the Junit Test for add, divide and multiply methods.

```
package unext.java_examples.Junit_example2;

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

class MathProviderTest {

	MathProvider provider;
	public MathProviderTest()
	{
		provider = new MathProvider();
	}
	@Test
	public void add()
	{
		System.out.println("======TEST ONE EXECUTED=======");
		System.out.println("Starting test " + new
Object(){}.getClass().getEnclosingMethod().getName());
		int firstNumber = 10;
		int secondNumber = 10;
		assertEquals(firstNumber + secondNumber, provider.add(firstNumber,
secondNumber));
		System.out.println("Ending test " + new
Object(){}.getClass().getEnclosingMethod().getName());
	}
    @Test
	public void divide()
	{
	System.out.println("======TEST TWO EXECUTED=======");
		System.out.println("Starting test " + new
Object(){}.getClass().getEnclosingMethod().getName());
	    int firstNumber = 10;
	    int secondNumber = 10;
		assertEquals(firstNumber / secondNumber, provider.divide(firstNumber,
secondNumber));
		System.out.println("Ending test " + new
Object(){}.getClass().getEnclosingMethod().getName());
	}
    @Test
	public void multiply()
	{
	System.out.println("======TEST THREE EXECUTED=======");
		System.out.println("Starting test " + new
Object(){}.getClass().getEnclosingMethod().getName());
	    int firstNumber = 10;
	    int secondNumber = 20;
		assertEquals(firstNumber * secondNumber, provider.multiply(firstNumber,
secondNumber));
		System.out.println("Ending test " + new
Object(){}.getClass().getEnclosingMethod().getName());
	}
}
```

3.1.3: MathProviderTest.java

The output for the above-mentioned code is



Activity 1

Suppose you are a software engineer working in X Ltd. You are required to perform testing for good software development. For that, you have to determine the requirements for the code that has to be written. Produce JUnit tests that examine the requirements. You must write code that executes the requirements, run JUnit tests, and demonstrate that the code passes the tests.

Self-Assessment Questions – 2

6. _____ present a fact after comparing the expected and the actual output according to the method used.
7. assertEquals has _____ parameters.
8. Select _____ in the Java editor to create new JUnit test case.
9. assertCheckTrue is a method in assert class. [True/False]
10. assertNotNull is a method in assert class. [True/False]
11. Which of the following is not a method in assert class?
    A. assertFalse
    B. assertCheckFalse
    C. assertSame
    D. assertNotEquals

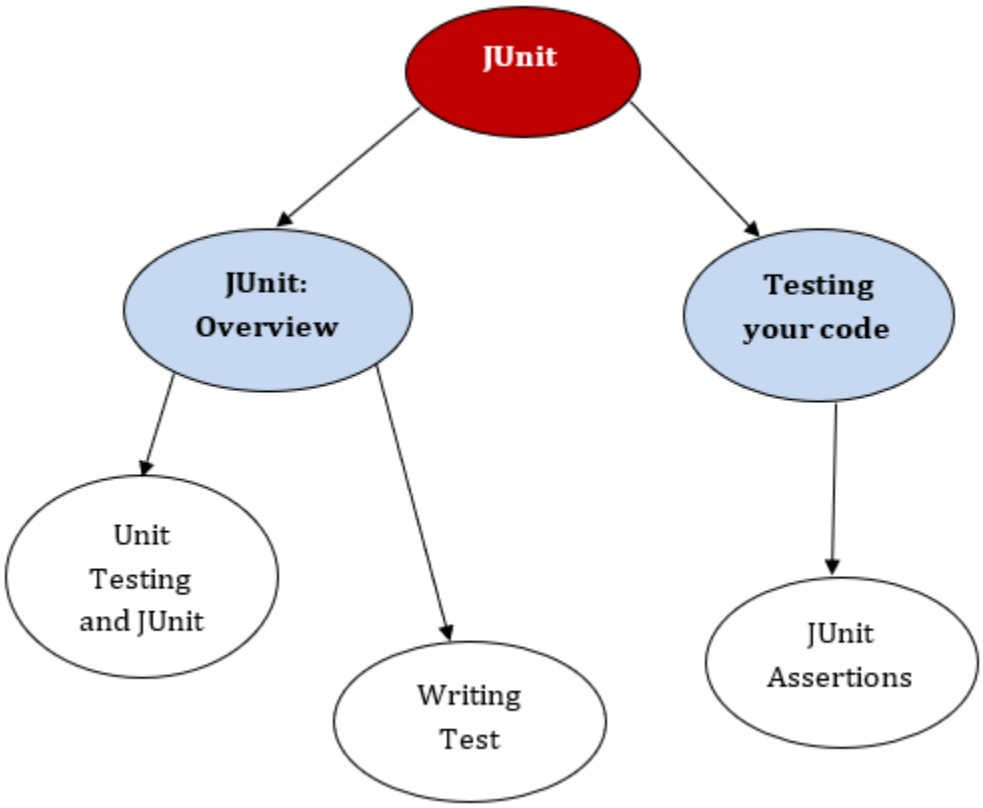


Fig 3: Conceptual Map

4. SUMMARY

- Java is based on an object-oriented programming model, and it is much simpler to use and deploy.
- Java is one of the most used, simple programming languages that is used for creating, compiling, and debugging programs easily.
- Java is a simple language and does not have any complex features such as operator overloading, explicit memory allocation, multiple inheritances, and so on.
- For the maximum utilisation of CPU, Java allows multithreading, which means concurrent execution of more than two parts of the Java program.
- A framework is a platform that is used to develop software applications.
- An individual unit of objects is referred to as a group or Collection.
- The data structures and algorithms used in the framework increase the efficiency and the speed of the program.
- Testing is done so that the application of the software contains as few bugs as possible.
- JUnit can be referred to as an open-source unit testing framework for Java programming.
- Unit testing is used to test small portions of code.
- Unit testing is done by verifying blocks of code to ensure that every block performs the function it is supposed to do.
- When we inject or feed input to a program or code, and we receive inspected outputs, this process is known as black-box testing.
- Every time there is a change made in the code, the programmer takes a small chunk of code at a time and tests it.
- Testing the code every time is critical so that we may ensure that there is no bug in our code and the rest of the application is still working fine as before.
- An automated test is a set of code written and compiled by a programmer to test the code block by block every time the programmer makes any small change in it.
- This test method is used to verify the code or application that comes for testing.
- Assertion presents a fact after comparing the expected and the actual output according to the method used.

5. GLOSSARY

Function: A set of actions that returns something to the programmer when called. It is also referred to complete segment of code that achieves a particular task.

Parameter: It is a particular form of variable that specifies an argument.

Method: A set of code executed when it is called. It is also a process linked with an object and a message.

Class: A group of objects is known as class. A class determines expandable program-code-template for producing objects.

Framework: A set of prewritten code that is used by programmers to create code efficiently is known as a framework.

Assertion: A statement that tells the correctness of any assumption taken beforehand. An assertion allows you to test your assumptions based on the program.

Object: Object is defined as an instance of a class and is a theoretical data type and actual element of the program.

6. CASE STUDY

JUNIT 5

The famous test framework of Java JUnit 5 is an unlock source project presented by GitHub. JUnit 5 is also called Jupiter, is the new release primary release of JUnit. JUnit 5 utilises interpretations to mark techniques as test techniques and to construct them.

It includes various discrete elements:

- **JUnit Platform:** basis layer which allows various testing frameworks to be introduced on the JVM.
- **JUnit Jupiter:** is the test framework of JUnit 5, which is introduced by the JUnit Platform.
- **JUnit Vintage:** miserable TestEngine, which operates older tests

JUnit 5 requires a minimum of Java 8 to run.

You have to create libraries to utilise JUnit 5 accessible for your test code. Now, let us understand the definition of test in JUnit. A JUnit test is a technique included in a class that is only be utilised for testing. This is known as the Test class. To mark a test method, interpret it with @Test interpretation. This method implements the code under test.

You can also utilise asset methods given by JUnit to examine the expected result compared to the actual result. This statement is known as assert statement.

Source: *vogella.com*

Discussion Questions:

1. Explain what is the role JUnit 5 test framework while executing the code for any software program?
2. Being a software programmer, explain how you will run JUnit 5 tests and, with context to that define JUnit tests in detail.

7. TERMINAL QUESTIONS

SHORT ANSWER QUESTIONS

Q1. What is testing?

Q2. What is JUnit?

Q3. What is the difference between manual and automated testing?

Q4. What is unit testing?

Q5. What is assertion?

LONG ANSWER QUESTIONS

Q1. Why is manual testing is still preferred?

Q2. What are some of the features of JUnit?

Q3. Write annotations of the JUnit5?

Q4. Explain JUnit5 architecture.

Q5. Why JUnit5 is better than previous versions?

8. ANSWERS

SELF ASSESSMENT QUESTIONS

1. Java
2. JUnit
3. Regression
4. False
5. True
6. C. Both A and B
7. Assertion
8. three
9. MyClass
10. False
11. True
12. B. assertCheckFalse

TERMINAL QUESTIONS

SHORT ANSWER QUESTIONS

Answer 1: Testing is the process of checking whether the application is giving desired output or not.

Answer 2: JUnit is a Java framework that is used to test codes.

Answer 3: Manual testing is done by developers manually, so it is time-consuming and takes much more effort than necessary. Automated testing is performed by testing tools, and once you write a code for testing, you save a lot of time and resources.

Answer 4: The process of a testing block by block of the code or application is called unit testing.

Answer 5: A statement that tells the correctness of any assumption taken beforehand is known as an assertion.

LONG ANSWER QUESTIONS

Answer 1: Manual testing is preferred due to following reasons:

- Bugs can be easily found and removed.
- The programmers spend more time refining their code.
- New cases can be executed very swiftly.
- Much cheaper than other types of testing because they require more investment in the beginning.
- It is preferred for short codes and bugs that can be easily detected rather than writing and code to run it.
- Manual testing supports ad-hoc testing.

Answer 2: Some of the features of JUnit are:

- It is an open-source framework of Java.

- JUnit provides annotation support for test cases.
- JUnit also gives assertion classes and methods for testing codes

Answer 3: Some of the annotations of JUnit5:

- **@Test:** It tells the computer, and the user that the method used is a test method.
- **@TestFactory:** denotes a method that's a test factory for dynamic tests
- **@Tag:** declares tags for filtering tests.
- **@BeforeEach:** This is used for the preparation of the test environment. It is executed before every test (previously @Before).
- **@AfterEach:** This is used for the activities that take place after the test, like cache cleanup, restoring the settings to default values, and so on. All this is executed after the test is completed (previously @After).
- **@RepeatedTest(<Number):** It works just like the test method but has one more feature. It repeats the test the number of times the developer mentioned. This annotation saves a lot of time for the developers.
- **@DisplayName("<Name>"):** The name can be used to explain what the test is supposed to do in a proper way. Proper space can be used to form a good sentence.
- **@AfterAll:** This refers to the method that is executed after all the tests defined are performed (previously @AfterClass). It can be used to clean up the cache or disconnecting for the database, etc.
- **@BeforeAll:** This refers to the method that is executed before all the tests defined are performed (previously @BeforeClass). It can be used to connecting to important databases, preparing for the test environment, etc.
- **@Nested:** The test class can be nested. It is used when a certain activity is to be performed.
- **@ExtendWith:** registers custom extensions.
- **@Disable:** disables a test class or method (previously @Ignore).

Answer 4: JUnit4 has everything contained in a single file whereas JUnit 5 is composed of several different modules from three different sub-projects:

JUnit 5 = JUnit Platform + JUnit Jupiter + JUnit Vintage

- **JUnit Platform:** To be able to launch JUnit tests, IDEs, build tools or plugins need to include and extend platform APIs. It defines the TestEngine API for developing new testing frameworks that run on the platform. It also provides a Console Launcher to launch the platform from the command line and build plugins for Gradle and Maven.
- **JUnit Jupiter:** It includes new programming and extension models for writing tests. It has all new JUnit annotations and TestEngine implementation to run tests written with these annotations.
- **JUnit Vintage:** Its primary purpose is to support running JUnit 3 and JUnit 4 written tests on the JUnit 5 platform. It's there is backward compatibility.

Answer 5: JUnit5 is built with three subparts: JUnit Platform, JUnit Jupiter, JUnit Vintage.

These are some of the reasons it should be favored by programmers.

- It has multiple libraries that can be used in the programs when necessary.
- It can also use more extensions when needed.
- It has built-in systems Maven and Gradle that makes programs much easier.
- It has several new features like test extensions, new assertions, dynamic tests and much more.

9. SUGGESTED BOOKS AND E-REFERENCES

BOOKS:

- Andy Hunt. (2003). Pragmatic Unit Testing in Java with JUnit. 1st edition. O'Reilly Publisher.
- Gary Gregory. (2010). JUnit in Action. 2nd edition. Manning Publications.
- Shekhar Gulati. (2017). Java Unit Testing with JUnit 5: Test-Driven Development with JUnit 5. 1st edition. Apress Publisher.
- Boni Garcia. (2017). Mastering Software Testing with JUnit 5: Comprehensive guide to Develop High-Quality Java Applications. 1st editionn. Packt Publishing.

E-REFERENCES:

- Java Zone, viewed on 25th July 2021, <https://dzone.com/articles/an-introduction-to-JUnit>
- Testing your code, viewed on 25th July 2021, <https://dzone.com/articles/an-introduction-to-JUnit>
- JUnit Assertions, viewed on 26th July 2021, <https://www.guru99.com/junit-assert.html>