Introduction to Unit Testing in Java

Student's Name

Department, Institutional Affiliation

Course Number and Name

Instructor's Name

Due Date

Introduction to Unit Testing in Java

In Java, unit testing is the testing of units of codes, such as a class or a method. Every unit is tested separately to assess whether it works as intended. As a result, development is faster with unit testing in Java.

JUnit is a Java framework for performing unit testing. According to Vasquez 2018, JUnit is commonly used and has been of importance in the creation of test-driven development. It is a member of the family of xUnit that comprises different testing frameworks. JUnit tests all units of all components of a Java program. Therefore, JUnit makes it easy to assert the accuracy of a unit of code in Java.

To perform unit testing using JUnit, a testing class is created with a @Test notation. According to Vogel 2021, unit testing using JUnit in Java is performed in a Test class and the method executes the code under test. The assert method is used and is provided by the JUnit framework by default. Inside the method, an expected value is defined, and the value is compared to the actual value returned by the function being assessed. Different messages are defined by programmers that show if the tests fail. Therefore, messages should be meaningful to easily identify and fix errors.

Demonstration of Unit testing in Java

Java Class Containing Method to be Unit Tested

```
public final class Math {

    public static int add(int first, int second) {
        return firstNumber + secondNumber;
    }

    public static double divide(int dividend, int divisor) {
        if (divisor == 0)
            throw new IllegalArgumentException("Cannot divide by zero (0).");

        return dividend / divisor;
    }
}
```

Figure 1 Static methods to be unit tested

JUnit Function to Test the Add and Divide Methods

```
public final class MathTests {
    private Math math;

    @Test
    public void add_TwoPlusTwo_ReturnsFour() {
        final int expected = 4;

        final int actual = math.add(2, 2);

        Assert.assertEquals(actual, expected);
    }

    @Test
    public void divide_TenDividedByFive_ReturnsTwo() {
        final double expected = 2.0;

        final double actual = math.divide(10, 5);

        Assert.assertEquals(actual, expected);
    }
}
```

Figure 2 Testing the add function with 2 and 2 as parameters and divide function with 10 and 5 as parameters.

References

Vasquez, C. (2018, April 22). *Introduction to unit testing with Java*. DEV Community. https://dev.to/chrisvasqm/introduction-to-unit-testing-with-java-2544

Vogel, L. (2021, July 31). *JUnit 5 tutorial - Learn how to write unit tests*. Eclipse, Android and Java training and support. https://www.vogella.com/tutorials/JUnit/article.html