BASIC OVERVIEW OF TESTING

TESTING:- ***the process of evaluating and verifying that a software product. The benefits of testing include preventing bugs, reducing development costs and improving performance.***

WHY WE NEED TESTING:-

- ***To identify defects***
- ***To reduce flaws in the component or system***
- ***Increase the overall quality of the system***
- ***Unit testing uses module approach due to that any part can be tested without waiting for completion of another parts testing.***

BENEFITS OF UNIT TESTING:-

- ***The testing is important since it discovers defects/bugs before the delivery to the client, which guarantees the quality of the software.***
- ***It makes the software more reliable and easy to use.***
- ***Thoroughly tested software ensures reliable and high-performance software operation.***

EXAMPLE:-

- **In amazon we are trying to add the product to wish list but its directly redirecting to payment option.**
- **We are trying to pay the order using net banking but its redirecting to credit card payment option**

UNIT TESTING TOOLS:-

- ✓ ***NUnit***
- ✓ ***JUnit***
- ✓ ***PHPunit***
- ✓ ***Parasoft Jtest***
- ✓ ***EMMA***

JUNIT:-

JUnit 5 = JUnit Platform + JUnit Jupiter + JUnit Vintage

JUnit Platform:-

- ✓ ***It defines the TestEngine API for developing new testing frameworks that runs on the platform.***

- ✓ ***Launches testing frameworks on the JVM***

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:-

- ✓ ***Provides support to execute previous JUnit version 3 and 4 tests on this new platform***

Annotations for Junit testing:-

Annotations plays a important role in Junit .junit provides a annotations to make unit testing more reliable

`@BeforeEach:-` The annotated method will be run before each test method in the test class.

`@AfterEach:-` The annotated method will be run after each test method in the test class.

`@BeforeAll:-` The annotated method will be run before all test methods in the test class. This method must be static.

`@AfterAll:-` The annotated method will be run after all test methods in the test class. This method must be static.

`@Test:-` It is used to mark a method as junit test

`@DisplayName:-` Used to provide any custom display name for a test class or test method

`@Disable:-` It is used to disable or ignore a test class or method from test suite.

`@Tag:-` Mark test methods or test classes with tags for test discovering and filtering

```java
import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Disabled;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;

public class AnnotationExample {

    @BeforeAll
    static void beforeAll() {
        System.out.println("**--- Executed once before all test methods in this class ---**");
    }

    @BeforeEach
    void beforeEach() {
        System.out.println("**--- Executed before each test method in this class ---**");
    }

    @Test
    void testMethod1() {
        System.out.println("**--- Test method1 executed ---**");
    }

    @DisplayName("Test method2 with condition")
    @Test
    void testMethod2() {
        System.out.println("**--- Test method2 executed ---**");
    }

    @Test
    @Disabled("implementation pending")
    void testMethod3() {
        System.out.println("**--- Test method3 executed ---**");
    }

    @AfterEach
    void afterEach() {
        System.out.println("**--- Executed after each test method in this class ---**");
    }

    @AfterAll
    static void afterAll() {
        System.out.println("**--- Executed once after all test methods in this class ---**");
    }


}
```

JUnit Assertions:-

Every test method must be evaluated against condition to true using assertions so that the test can continue to execute. JUnit Jupiter assertions are kept in the org.junit.jupiter.api. Assertions class. All of the methods are static.

Assertion	Description
assertEquals(expected, actual)	Fails when expected does not equal actual
assertFalse(expression)	Fails when expression is not false
assertNull(actual)	Fails when actual is not null
assertNotNull(actual)	Fails when actual is null
assertAll()	Group many assertions and every assertion is executed even if one or more of them fails
assertTrue(expression)	Fails if expression is not true
assertThrows()	Class to be tested is expected to throw an exception

EXAMPLE PROGRAMS FOR JUNIT:-

```
public class EvenOdd {
    public static int Even(int a)
    {

        return a%2;
    }
    static int Square(int a)
    {
        return a*a;
    }
    static int fact(int a)
    {
        int i,fact=1;

        for(i=1;i<=a;i++){
            fact=fact*i;
        }
        return fact;
    }
}
```

```java
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;
class EvenOddTest {

    @Test
    void even() {
        assertEquals(1,EvenOdd.Even(5));
        assertEquals(0,EvenOdd.Even(6));
        assertEquals(1,EvenOdd.Even(21));
        assertEquals(0,EvenOdd.Even(8));
    }
    @Test
    void square()
    {
        assertEquals(24,EvenOdd.Square(4));
    }
    @Test
    void fact()
    {
        assertEquals(120,EvenOdd.fact(5));
        assertEquals(3628800,EvenOdd.fact(10));
        assertEquals(24,EvenOdd.fact(4));
        assertEquals(6,EvenOdd.fact(3));
    }

}
```

WE CREATED 3 METHODS

- ➔ **EVEN**
- ➔ **FACT**
- ➔ **SQUARE**

And we r performing testing to ensure errors and quality of the programming..

EXAMPLE 2:-

Performing testing on array

We created a method➔ findMin which returns the min value of the array

So to check program quality and bugs we r prforming testing

```java
public class TestAssignmentt {
```

```
    public int findMin(int[] array) {
        if (!(array.length > 0)) {
            throw new IllegalArgumentException("Input array is empty");
        }

        int min = Integer.MAX_VALUE;
        for (int i = 0; i < array.length; i++) {
            if (array[i] <= min)
                min = array[i];
        }

        return min;
    }
}
```

PERFORMING TESTING

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

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

class TestAssignmenttTest {

    @Test
    void testFindMin() {
        TestAssignmentt msao = new TestAssignmentt();
        int[] array = {10, 2, 3, 10, 1, 0, 2, 3, 16, 0, 2};
        assertEquals(0, msao.findMin(array));

    }
}
```