Experiment No. 12

Aim : - Develop test cases for the project using **White Box testing (JUnit).**

Theory : -

White Box Testing

The box testing approach of software testing consists of black box testing and white box testing. We are discussing here white box testing which also known as glass box is testing, structural testing, clear box testing, open box testing and transparent box testing. It tests internal coding and infrastructure of a software focus on checking of predefined inputs against expected and desired outputs. It is based on inner workings of an application and revolves around internal structure testing. In this type of testing programming skills are required to design test cases. The primary goal of white box testing is to focus on the flow of inputs and outputs through the software and strengthening the security of the software. The term 'white box' is used because of the internal perspective of the system. The clear box or white box or transparent box name denote the ability to see through the software's outer shell into its inner workings. Developers do white box testing. In this, the developer will test every line of the code of the program. The developers perform the White-box testing and then send the application or the software to the testing team, where they will perform the black box testing and verify the application along with the requirements and identify the bugs and sends it to the developer. The developer fixes the bugs and does one round of white box testing and sends it to the testing team. Here, fixing the bugs implies that the bug is deleted, and the particular feature is working fine on the application.

Using this method, Software Engineer can derive test cases that:

o Guarantee that all independent paths within a module have been exercised at least once

o Exercise all logical decisions on their true and false sides,

o Execute all loops at their boundaries and within their operational bounds

o Exercise internal data structures to ensure their validity.

- What are test cases?

A test case is exactly what it sounds like: a test scenario measuring functionality across a set of actions or conditions to verify the expected result. They apply to any software application, can use manual testing or an automated test, and can make use of test case management tools.

A test case is a set of actions performed on a system to determine if it satisfies software requirements and functions correctly. The purpose of a test case is to determine if different features within a system are performing as expected and to confirm that the system satisfies all related

standards, guidelines and customer requirements. The process of writing a test case can also help reveal errors or defects within the system.

A test case document includes test steps, test data, preconditions and the post conditions that verify requirements. Test cases define what must be done to test a system, including the steps executed in the system, the input data values that are entered into the system and the results that are expected throughout test case execution. Using test cases allows developers and testers to discover errors that may have occurred during development or defects that were missed during ad hoc tests.

The benefits of an effective test case include:

o Guaranteed good test coverage. o Reduced maintenance and software support costs. o Reusable test cases.

o Confirmation that the software satisfies end-user requirements. o Improved quality of software and user experience.

o Higher quality products lead to more satisfied customers.

o More satisfied customers will increase company profits.

Overall, writing and using test cases will lead to business optimization. Clients are more satisfied, customer retention increases, the costs of customer service and fixing products decreases, and more reliable products are produced, which improves the company's reputation and brand image.

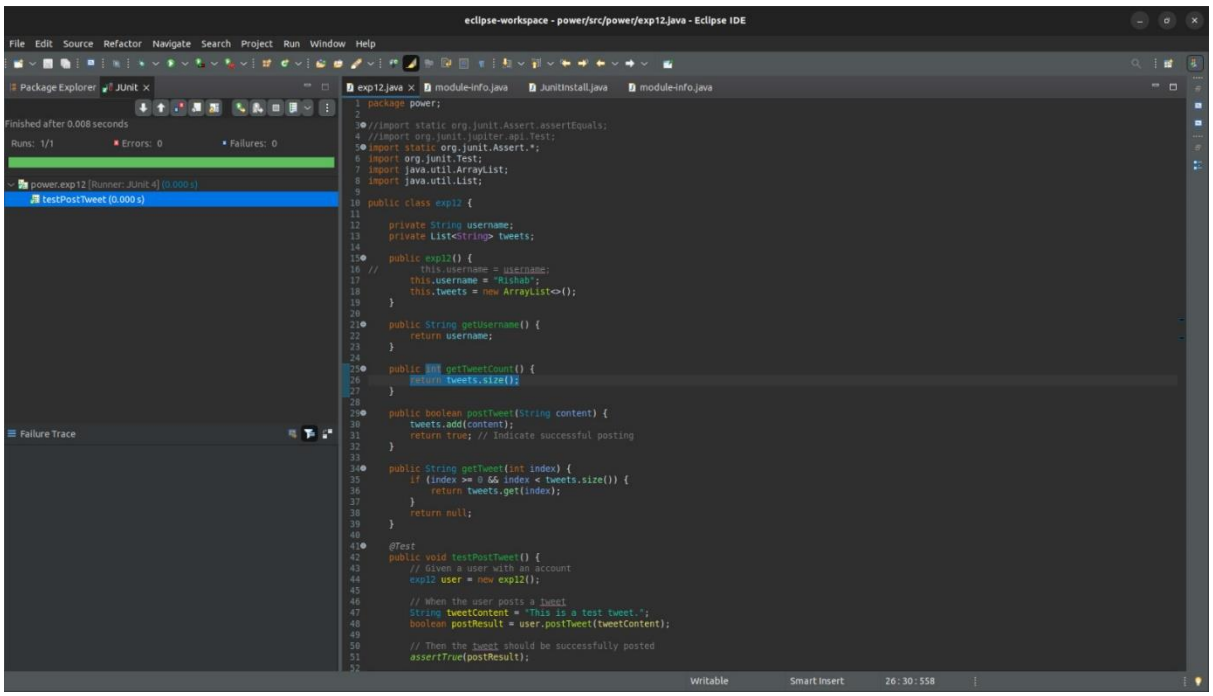
Code:-

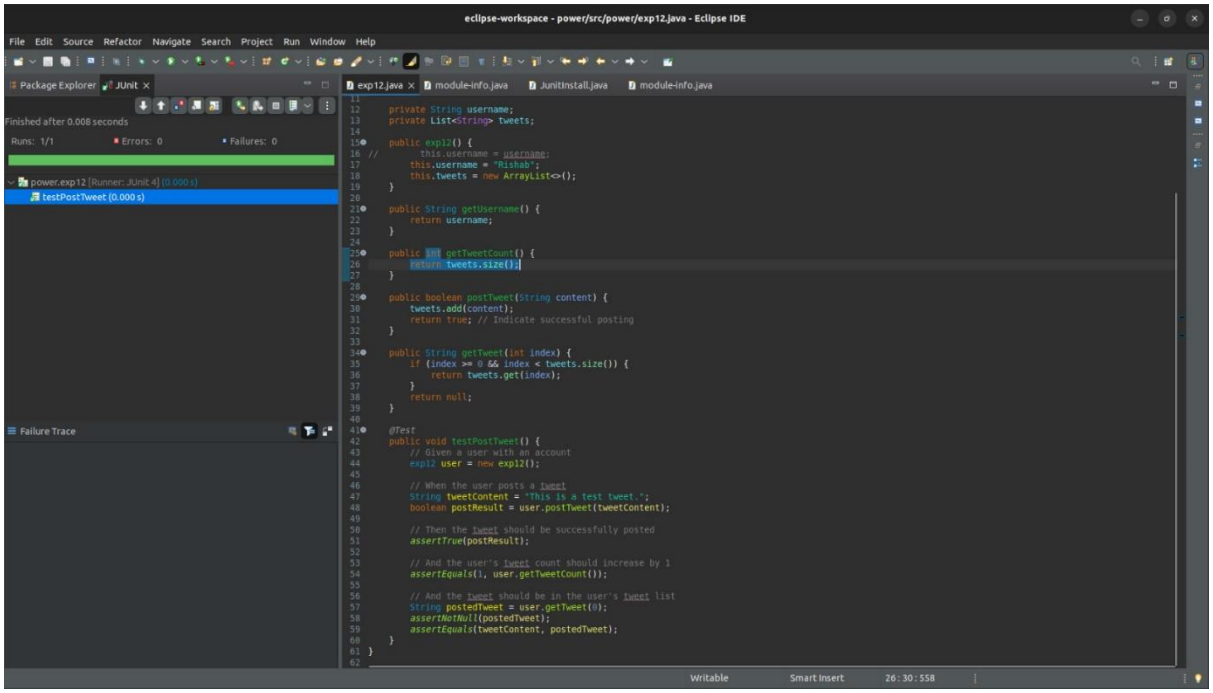
```
package power;


//import static org.junit.Assert.assertEquals;

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

import static org.junit.Assert.*;

import org.junit.Test;

import java.util.ArrayList;

import java.util.List;


public class exp12 {


    private String username;

    private List<String> tweets;
```

```
    public exp12() {
//        this.username = username;
            this.username = "Rishab";
        this.tweets = new ArrayList<>();
    }

    public String getUsername() {
        return username;
    }

    public int getTweetCount() {
        return tweets.size();
    }

    public boolean postTweet(String content) {
        tweets.add(content);
        return true; // Indicate successful posting
    }

    public String getTweet(int index) {
        if (index >= 0 && index < tweets.size()) {
            return tweets.get(index);
        }
        return null;
    }

    @Test
    public void testPostTweet() {
        // Given a user with an account
        exp12 user = new exp12();
```

```
        // When the user posts a tweet
        String tweetContent = "This is a test tweet.";
        boolean postResult = user.postTweet(tweetContent);

        // Then the tweet should be successfully posted
        assertTrue(postResult);

        // And the user's tweet count should increase by 1
        assertEquals(1, user.getTweetCount());

        // And the tweet should be in the user's tweet list
        String postedTweet = user.getTweet(0);
        assertNotNull(postedTweet);
        assertEquals(tweetContent, postedTweet);
    }
}
```

Output : -





Conclusion : -

Thus, we developed the desired test cases for our existing project using white box testing.