**Software Engineering Lab Experiment No. 8**

**Aim:** Perform White box testing for the project

**Objectives:**

1. Code Coverage: Ensure that all parts of the code are executed at least once during testing.
2. This includes validating the accuracy and security of the login process and verifying the correctness of account transactions such as fund transfers, balance inquiries, and transaction history.

**Requirements:**

1. Computer with internet access.
2. Sample software project or problem statement for requirements analysis.
3. Word processing software for creating the lab report.
4. IDE for running project

**Concept:**

* **White box testing** is based on the understanding of the internal structure of the code. Test cases are designed to exercise different paths through the code, including loops, branches, and conditions. The goal is to ensure that the code behaves correctly under various scenarios.

**Tools:**

1. **JUnit**: A widely used testing framework for Java that supports the creation and execution of unit tests.
2. **JaCoCo**: A Java Code Coverage Library that shows how much of your code is covered by tests.
3. **SonarQube**: An open-source platform for continuous inspection of code quality.
4. **OWASP** Dependency-Check: A tool for identifying project dependencies and checking if there are any known, publicly disclosed, vulnerabilities.

**Steps:** Steps for White Box Testing

1. **Code Review:** Understand the codebase through code review to identify potential areas of interest and risk.:

**1.** **Successful Login:**

- Scenario: User provides correct username and password.

- Expected Result: The login should succeed.

**2. Failed Login - Incorrect Password:**

- Scenario: User provides a correct username but an incorrect password.

- Expected Result: The login should fail.

**3. Failed Login - Invalid Username:**

- Scenario: User provides an invalid username.

- Expected Result: The login should fail.

**4. Failed Login - Empty Password:**

- Scenario: User provides a valid username but an empty password.

- Expected Result: The login should fail.

**5. Failed Login - Empty Username:**

- Scenario: User provides an empty username.

- Expected Result: The login should fail.

**6. Deposit Positive Amount:**

- Scenario: User deposits a positive amount into the account.

- Expected Result: The account balance should increase.

**7. Withdraw Positive Amount:**

- Scenario: User withdraws a positive amount from the account with sufficient funds.

- Expected Result: The account balance should decrease.

**8. Withdraw Insufficient Funds:**

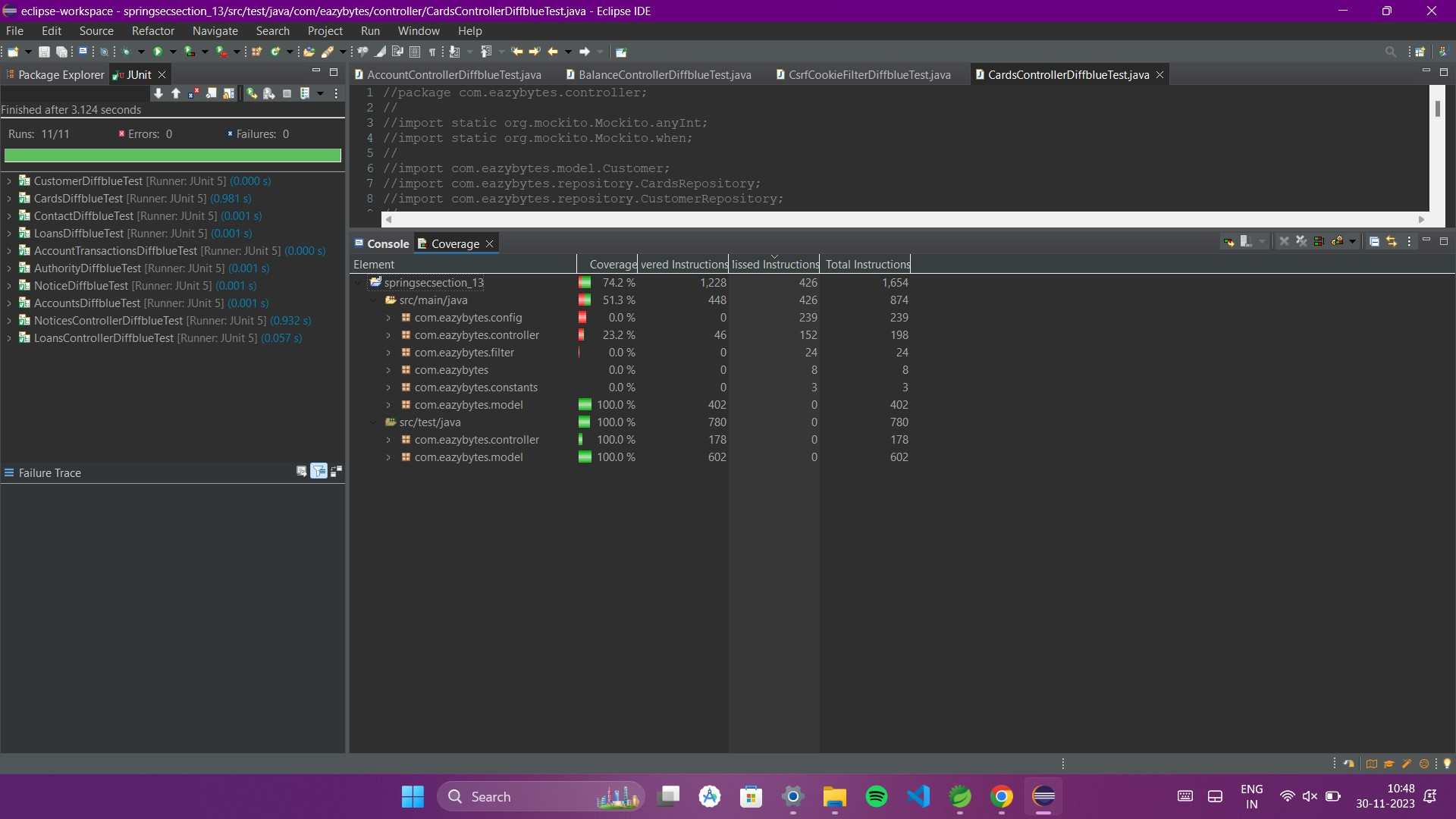
- Scenario: User attempts to withdraw more than the account balance.

- Expected Result: An `InsufficientFundsException` should be thrown.

1. **Unit Testing:** Write and execute unit tests for individual functions or methods. Use tools like JUnit for Java.

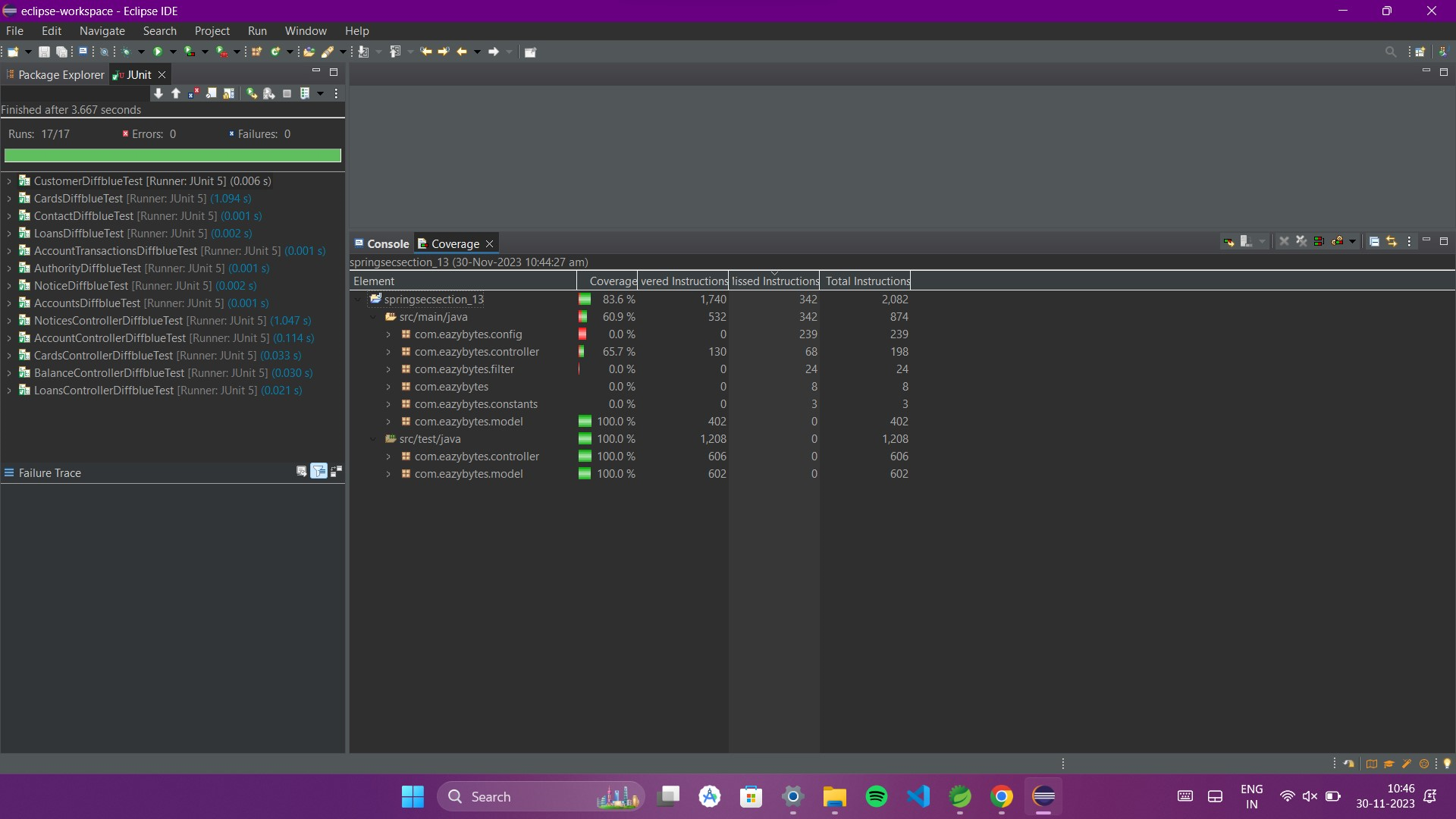
- Total tests run = 11

Code coverage = 74.2%



- Total tests run = 17

Code coverage = 83.6%



**Conclusion:**

We learnt that white box testing is crucial for ensuring the reliability, security, and performance of a Java project. By testing the internal logic and structure of the code, before they impact the functionality or security of the application.

**References:**

- <https://www.softwaretestinghelp.com/black-box-testing/>

- https://birdeatsbug.com/blog/black-box-testing