Week 2 – JUnit, Mockito, and SLF4J Hands-on Exercises

# Exercise 1: Setting Up JUnit

## Objective:

Set up a Java Maven project with JUnit dependency, create and run a sample unit test.

## Steps Performed:

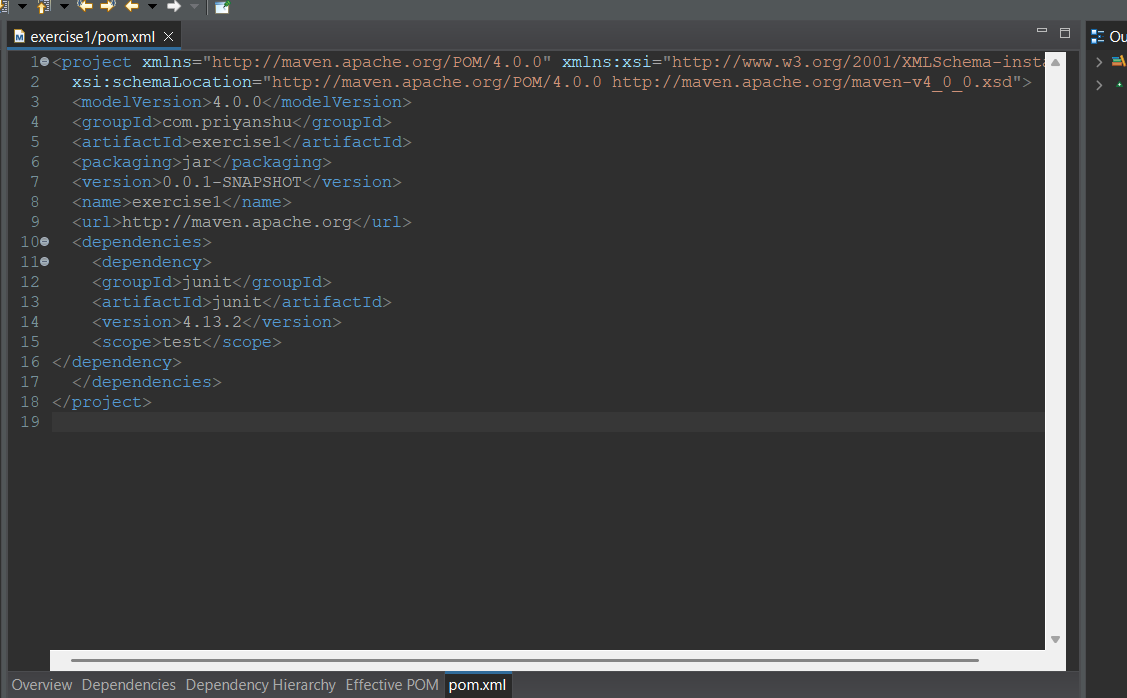
- Created Maven project using Eclipse (maven-archetype-quickstart).

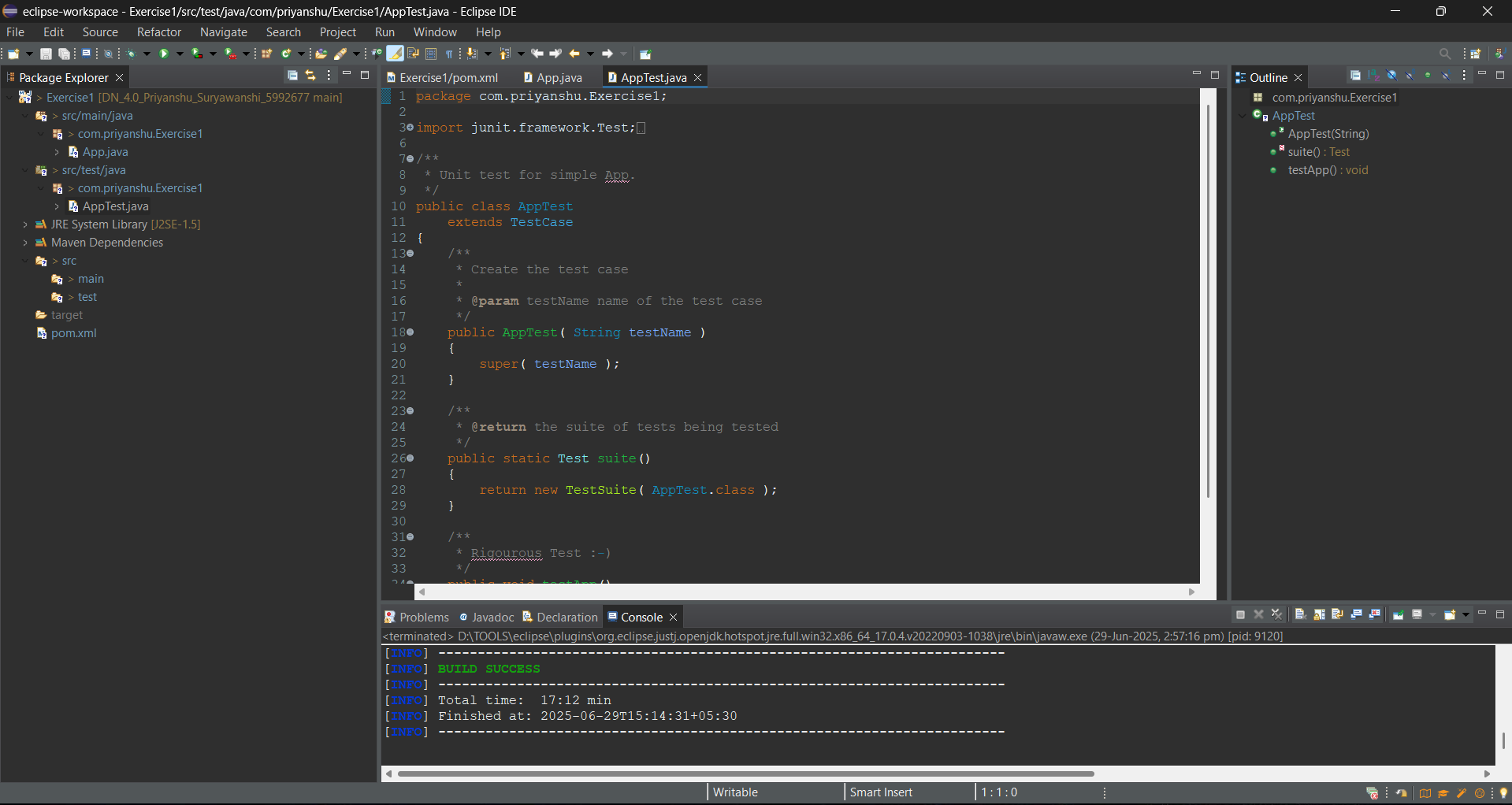
- Configured `pom.xml` with JUnit 4.13.2 dependency.

- Created `AppTest.java` under `src/test/java`.

- Ran test successfully with `BUILD SUCCESS`.

## Output :

 -



# Exercise 3: Assertions In Junit

We need to use different assertions in JUnit to validate your test results.

Steps:

1. Created a Maven project using JUnit 5.

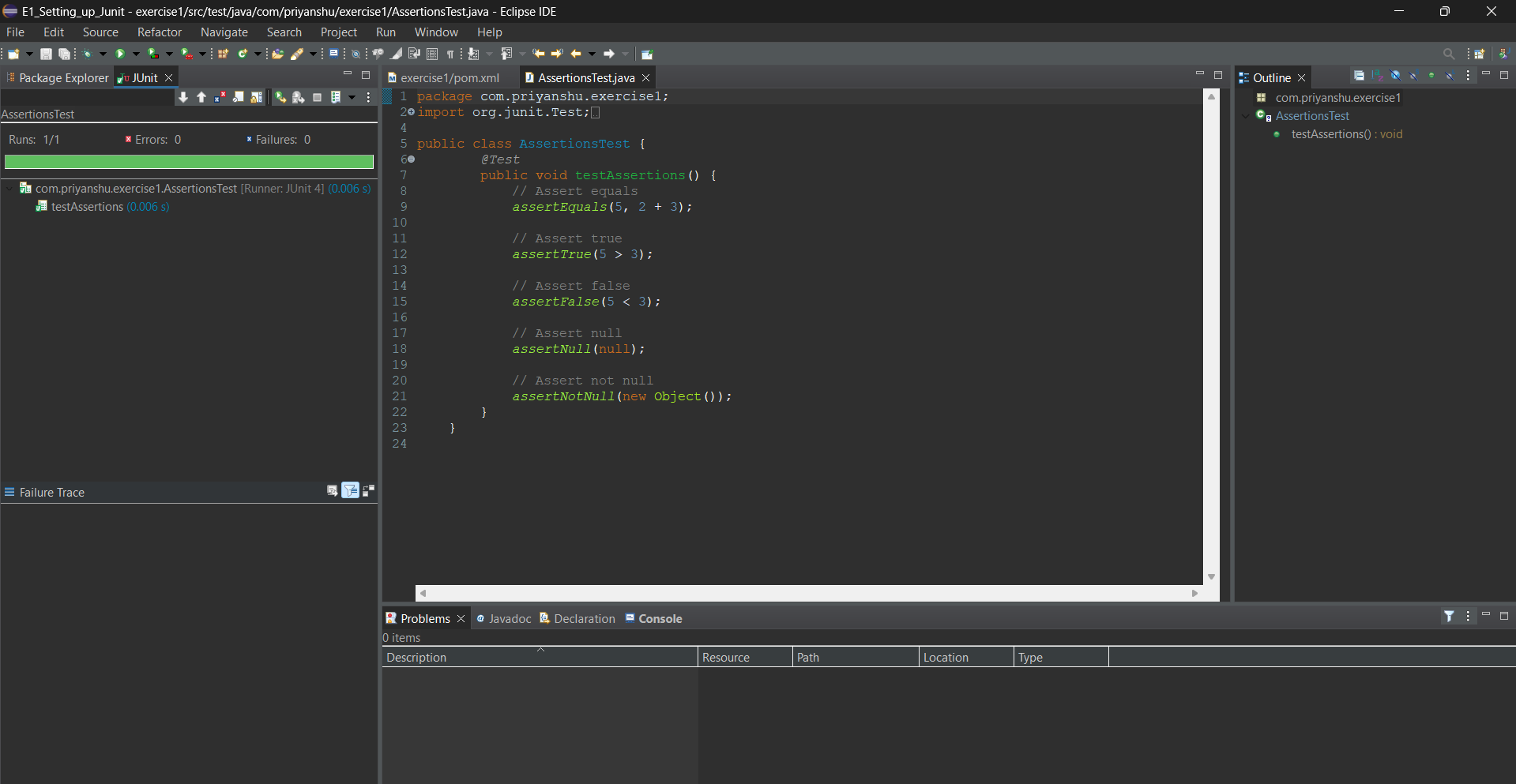
2. Added JUnit 5 dependency in pom.xml.

3. Created a test class named AssertionsTest.java in src/test/java.

4. Implemented the following assertions: assertEquals, assertTrue, assertFalse, assertNull, assertNotNull.

5. Ran the test and verified all tests passed.

## Output :



# Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit

Scenario: You need to organize your tests using the Arrange-Act-Assert (AAA) pattern and use setup and teardown methods.

**Code of BankAccount.java:**

package com.priyanshu.exercise1;

public class BankAccount {

private double balance;

public BankAccount(double initialBalance) {

this.balance = initialBalance;

}

public void deposit(double amount) {

balance += amount;

}

public void withdraw(double amount) {

balance -= amount;

}

public double getBalance() {

return balance;

}

}

**Code of BankAccountTest.java:**

package com.priyanshu.exercise1;

import org.junit.\*;

import static org.junit.Assert.\*;

public class BankAccountTest {

private BankAccount account;

*@Before*

public void setUp() {

System.***out***.println("Setting up test...");

account = new BankAccount(1000); }

// Teardown method

*@After*

public void tearDown() {

System.***out***.println("Cleaning up...");

account = null;

}

// Test 1: Deposit (AAA Pattern)

*@Test*

public void testDeposit() {

// Arrange

double amount = 500;

// Act

account.deposit(amount);

// Assert

*assertEquals*(1500, account.getBalance(), 0.001);

}

// Test 2: Withdraw (AAA Pattern)

*@Test*

public void testWithdraw() {

// Arrange

double amount = 200;

// Act

account.withdraw(amount);

// Assert

*assertEquals*(800, account.getBalance(), 0.001);

}

}

# Output :