**JUnit Testing Exercises**

Exercise 1: Setting Up JUnit Scenario: You need to set up JUnit in your Java project to start writing unit tests.

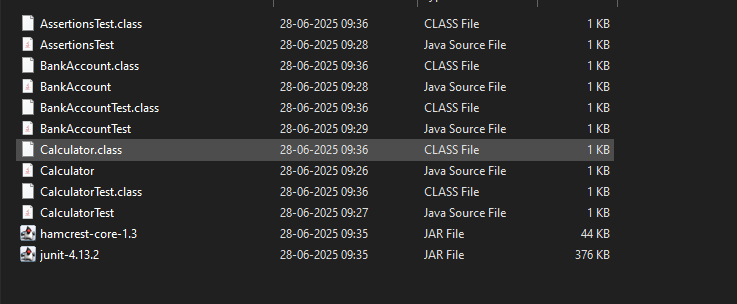
Steps:

1. Create a new Java project in your IDE (e.g., IntelliJ IDEA, Eclipse).

2. Add JUnit dependency to your project. If you are using Maven, add the following to your pom.xml: junit junit 4.13.2 test

3. Create a new test class in your project.

**Output:**



**Exercise 2: Writing Basic JUnit Tests**

Scenario: You need to write basic JUnit tests for a simple Java class.

Code:

Calculator.java

**public** **class** Calculator {

**public** **int** add(**int** a, **int** b) {

**return** a **+** b;

    }

**public** **int** subtract(**int** a, **int** b) {

**return** a **-** b;

    }

**public** **int** multiply(**int** a, **int** b) {

**return** a **\*** b;

    }

**public** **int** divide(**int** a, **int** b) {

**if** (b **==** 0) **throw** **new** IllegalArgumentException("Division by zero");

**return** a **/** b;

    }

}

CalculatorTest;

**import** **static** **org.junit.Assert.\***;

**import** **org.junit.Test**;

**public** **class** CalculatorTest {

**Calculator** calc **=** **new** Calculator();

    @**Test**

**public** **void** testAdd() {

        assertEquals(5, calc.add(2, 3));

    }

    @**Test**

**public** **void** testSubtract() {

        assertEquals(1, calc.subtract(4, 3));

    }

    @**Test**

**public** **void** testMultiply() {

        assertEquals(6, calc.multiply(2, 3));

    }

    @**Test**(expected **=** IllegalArgumentException.class)

**public** **void** testDivideByZero() {

        calc.divide(5, 0);

    }

    @**Test**

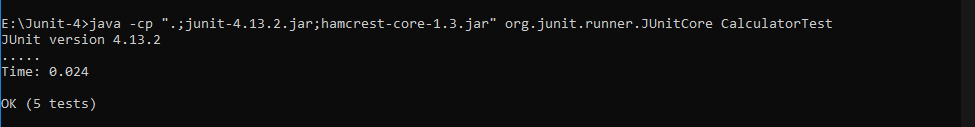
**public** **void** testDivide() {

        assertEquals(2, calc.divide(6, 3));

    }

}

Output:



**Exercise 3: Assertions in JUnit**

Scenario: You need to use different assertions in JUnit to validate your test results.

Code:

Assertion.java

**import** **static** **org.junit.Assert.\***;

**import** **org.junit.Test**;

**public** **class** AssertionsTest {

    @**Test**

**public** **void** testAssertions() {

*// Assert equals*

        assertEquals(5, 2 **+** 3);

*// Assert true*

        assertTrue(5 **>** 3);

*// Assert false*

        assertFalse(5 **<** 3);

*// Assert null*

        assertNull(**null**);

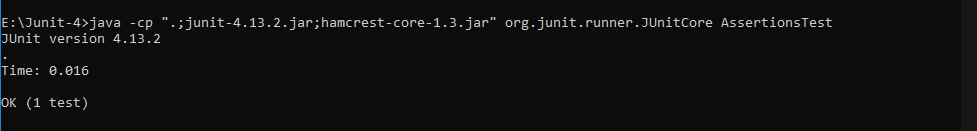
*// Assert not null*

        assertNotNull(**new** Object());

    }

}

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:

**public** **class** BankAccount {

**private** **int** balance **=** 0;

**public** **void** deposit(**int** amount) {

**if** (amount **>** 0)

            balance **+=** amount;

    }

**public** **void** withdraw(**int** amount) {

**if** (amount **<=** balance)

            balance **-=** amount;

    }

**public** **int** getBalance() {

**return** balance;

    }

}

Output:

