**Setting Up Junit:**

**Greet.java**

public class Greet{

    public String says(){

        return "Hello";

    }

}

**GreetTest.java**

import org.junit.\*;

public class GreetTest {

    @Test

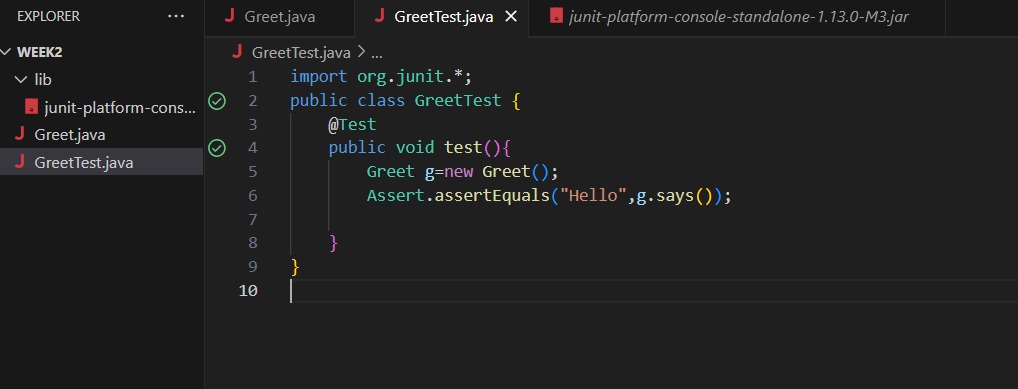
    public void test(){

        Greet g=new Greet();

        Assert.assertEquals("Hello",g.says());

}

}



**Assertions in JUnit**

**AssertionsTest.java**

import org.junit.jupiter.api.Test;

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

public class AssertionsTest {

    @Test

    public void testAssertions() {

        assertEquals(5, 2 + 3);

        assertTrue(5 > 3);

        assertFalse(5 < 3);

        assertNull(null);

        assertNotNull(new Object());

    }

}



**Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in Junit**

**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;

    }

}

**CalculatorTest.java**

import org.junit.jupiter.api.\*;

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

public class CalculatorTest {

private Calculator calculator;

    @BeforeEach

    void setUp() {

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

        calculator = new Calculator();

    }

    @AfterEach

    void tearDown() {

        System.out.println("Tearing down");

        calculator = null;   }

  @Test

    void testAddition() {

          int a = 5;

        int b = 3;

int result = calculator.add(a, b);

        assertEquals(8, result);

    }

    @Test

    void testSubtraction() {

        int a = 10;

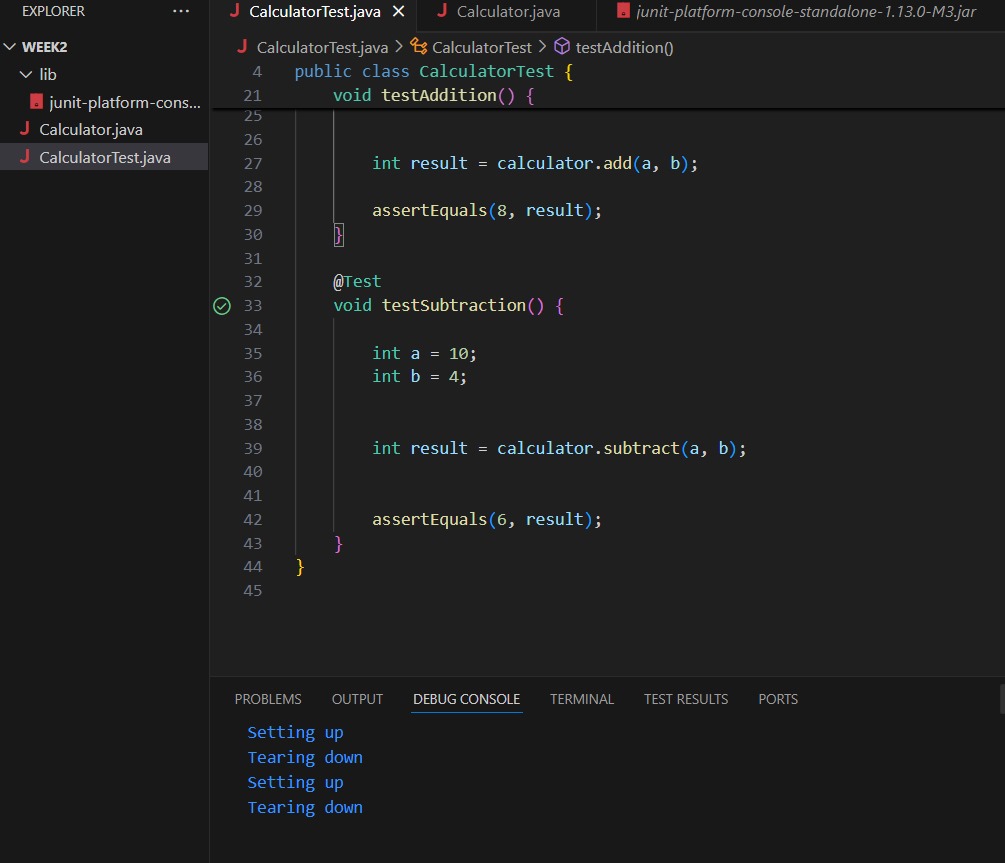
        int b = 4;

        int result = calculator.subtract(a, b);

        assertEquals(6, result);

    }

}



**Mocking and Stubbing**

**Scenario: You need to test a service that depends on an external API. Use Mockito to mock the external API and stub its methods.**

**ExternalApi.java**

public interface ExternalApi {

    String getData();

}

**MyService.java**

public class MyService {

    private ExternalApi api;

public MyService(ExternalApi api) {

        this.api = api;

    }

    public String fetchData() {

        return api.getData();

    }

}

**MyServiceTest.java**

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

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

public class MyServiceTest {

@Test

    public void testExternalApi() {

        ExternalApi mockApi = mock(ExternalApi.class);

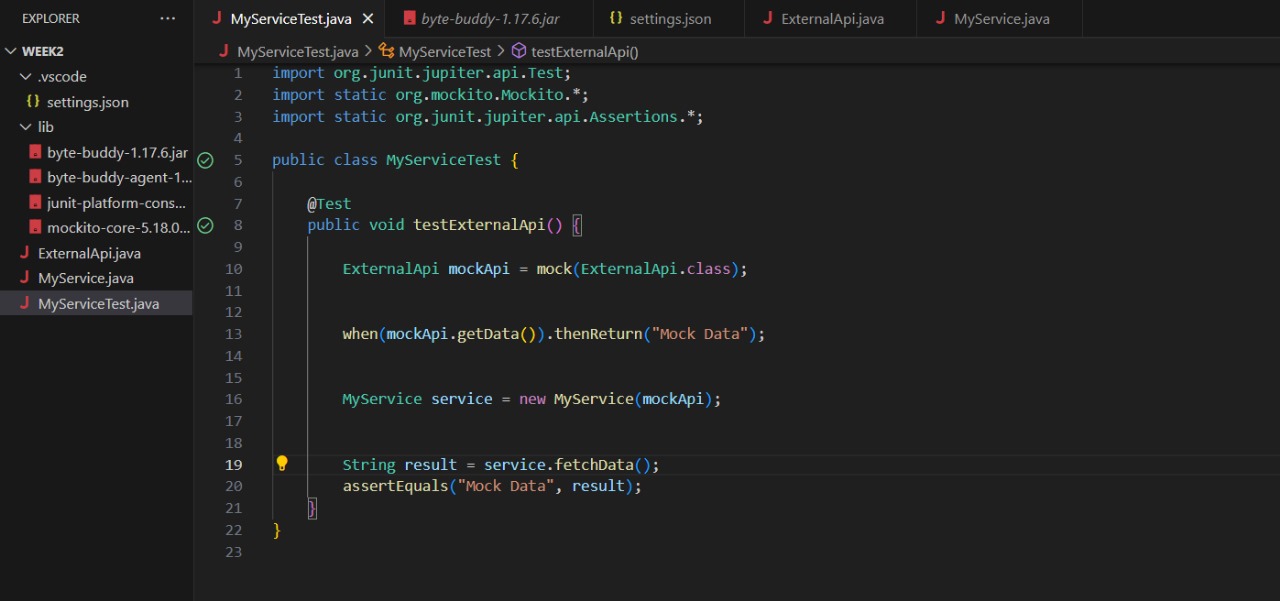
  when(mockApi.getData()).thenReturn("Mock Data");

        MyService service = new MyService(mockApi);

String result = service.fetchData();

        assertEquals("Mock Data", result);

    }}



**Exercise 2: Verifying Interactions**

**Scenario: You need to ensure that a method is called with specific arguments.**

**External.java**

public interface ExternalApi {

    String getData();

}

**MyService.java**

public class MyService {

    private ExternalApi api;

public MyService(ExternalApi api) {

        this.api = api;

    }

    public String fetchData() {

        return api.getData();

    }

}

**MyServiceTest.java**

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

    @Test

    public void testVerifyInteraction() {

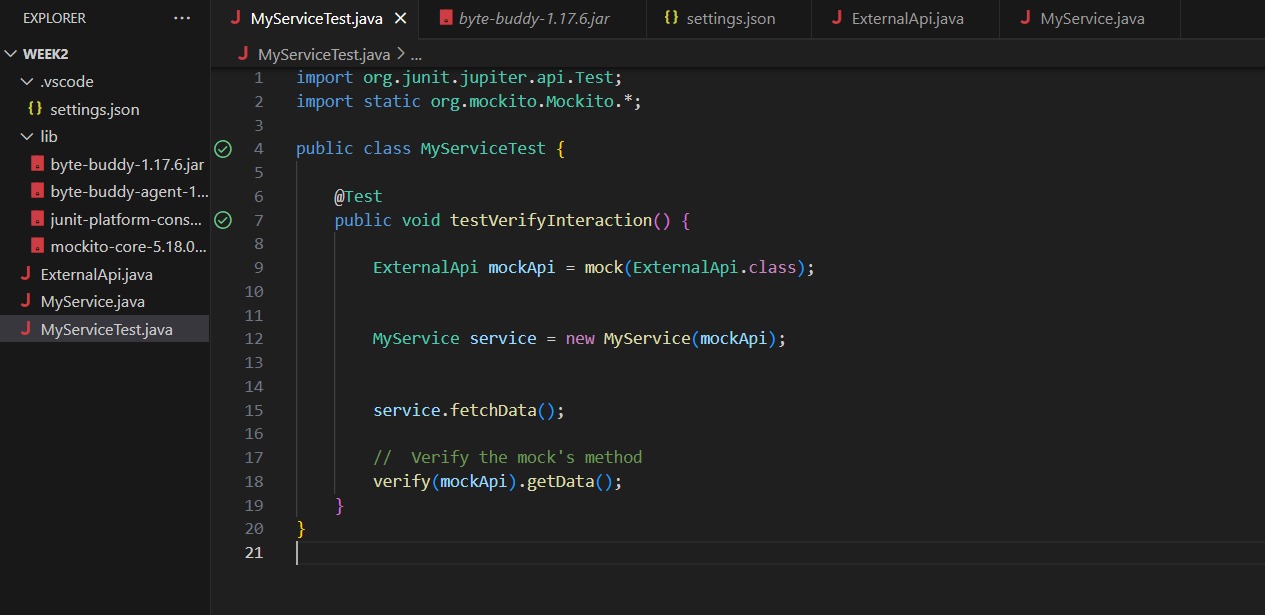
        ExternalApi mockApi = mock(ExternalApi.class);

        MyService service = new MyService(mockApi);

        service.fetchData();

        verify(mockApi).getData();

    }}



**Exercise 1: Logging Error Messages and Warning Levels**

**Task: Write a Java application that demonstrates logging error messages and warning levels using SLF4J.**

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class LoggingExample {

    private static final Logger logger = LoggerFactory.getLogger(LoggingExample.class);

public static void main(String[] args) {

        logger.error("This is an error message");

        logger.warn("This is a warning message");

    }}

