**Advanced JUnit Testing Exercises**

**Exercise 1: Parameterized Tests Scenario:**

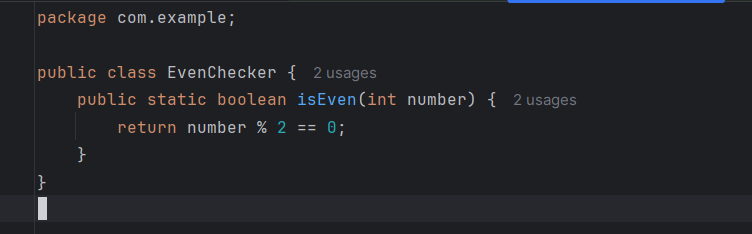
You want to test a method that checks if a number is even. Instead of writing multiple test cases, you will use parameterized tests to run the same test with different inputs.

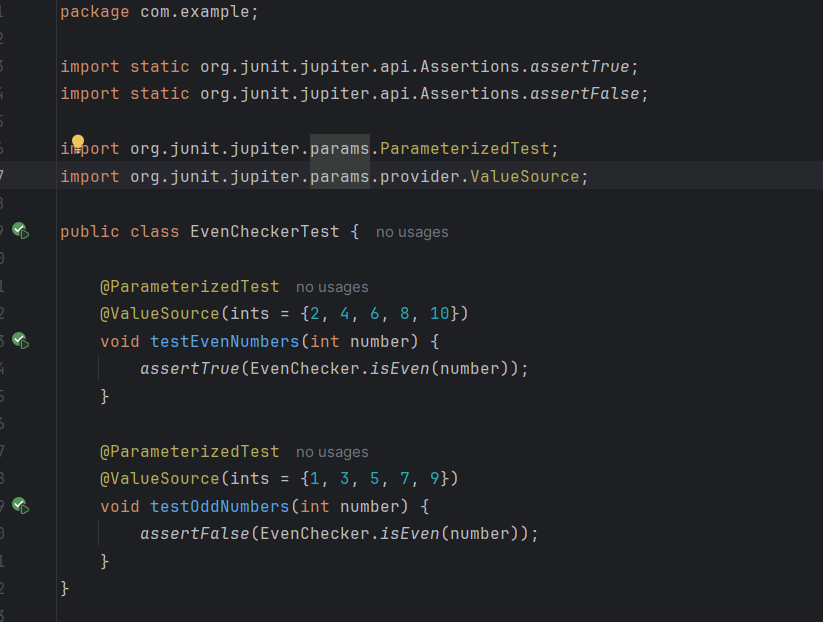
Steps: 1. Create a new Java class `EvenChecker` with a method `isEven(int number)`.

2. Write a parameterized test class `EvenCheckerTest` that tests the `isEven` method with different inputs.

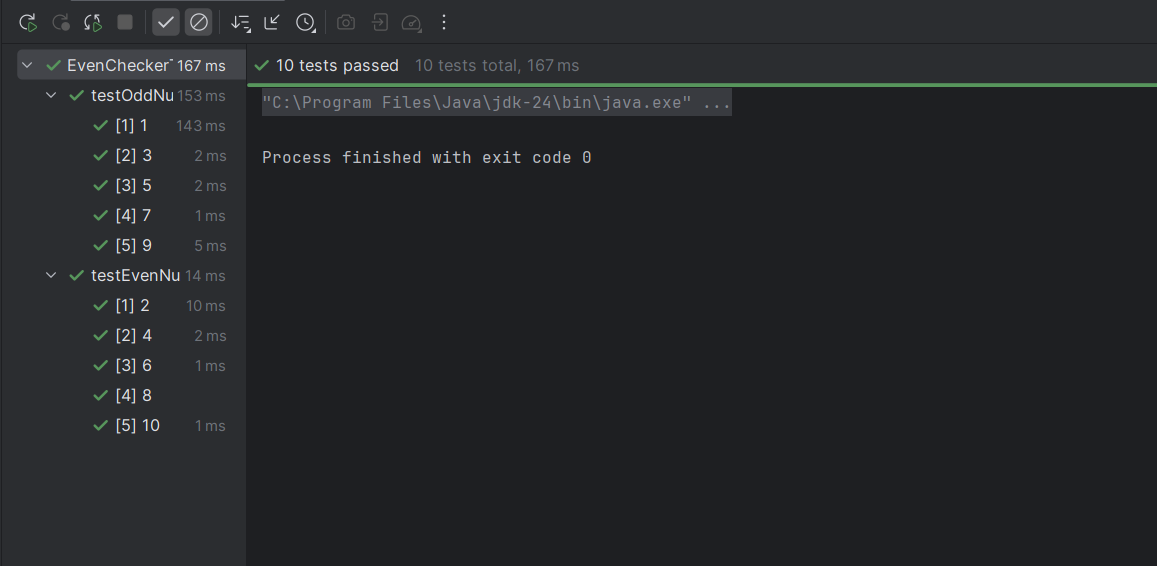
3. Use JUnit's `@ParameterizedTest` and `@ValueSource` annotations.

**SOLUTION:**

****

****

**OUTPUT:**

****

**Exercise 2: Test Suites and Categories Scenario:**

You want to group related tests into a test suite and categorize them.

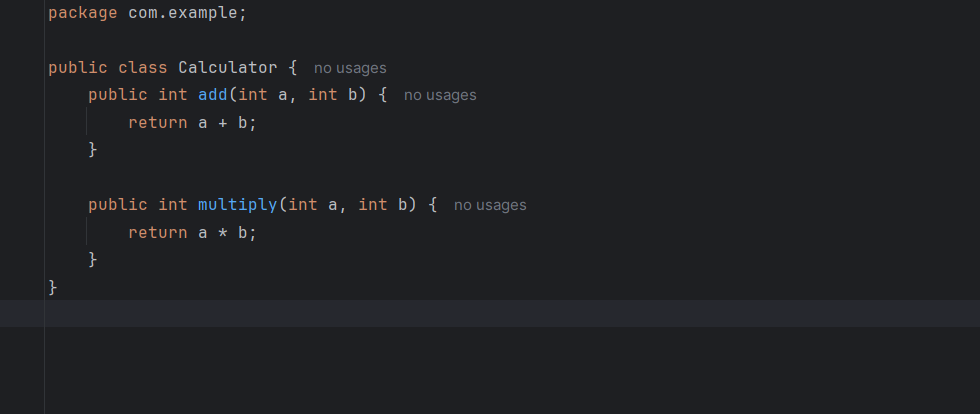
Steps: 1. Create a new test suite class `AllTests`.

2. Add multiple test classes to the suite.

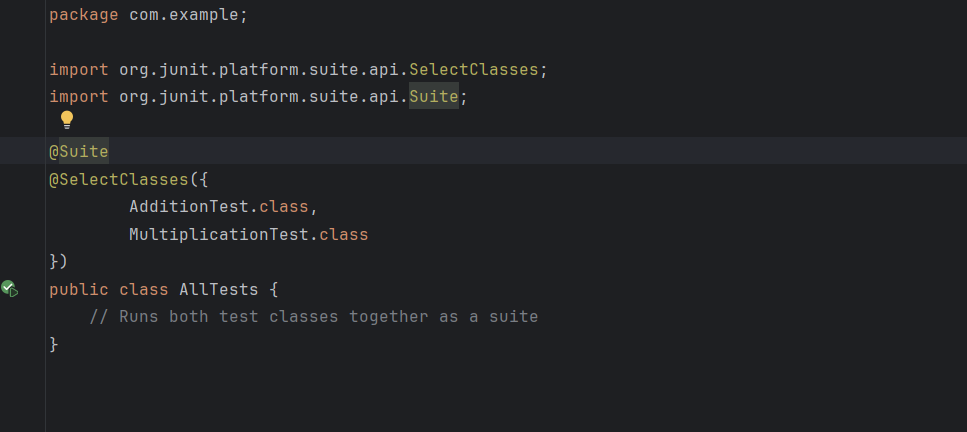
3. Use JUnit's `@Suite` and `@SelectClasses` annotations.

**SOLUTION:**

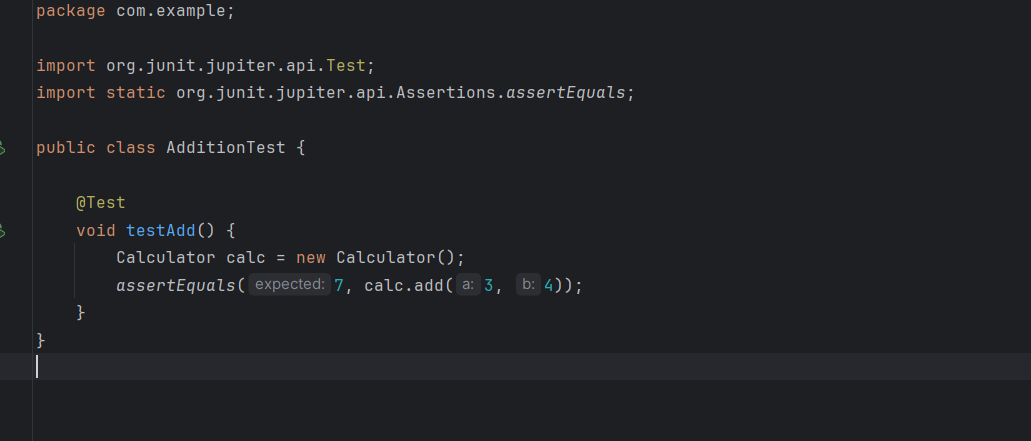
**Calculator.java:**

****

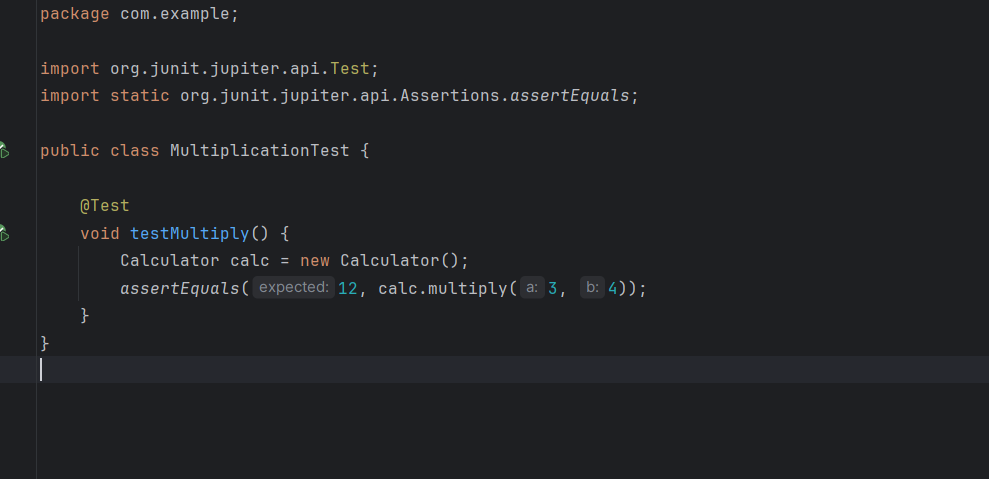
**AllTests.java:**

****

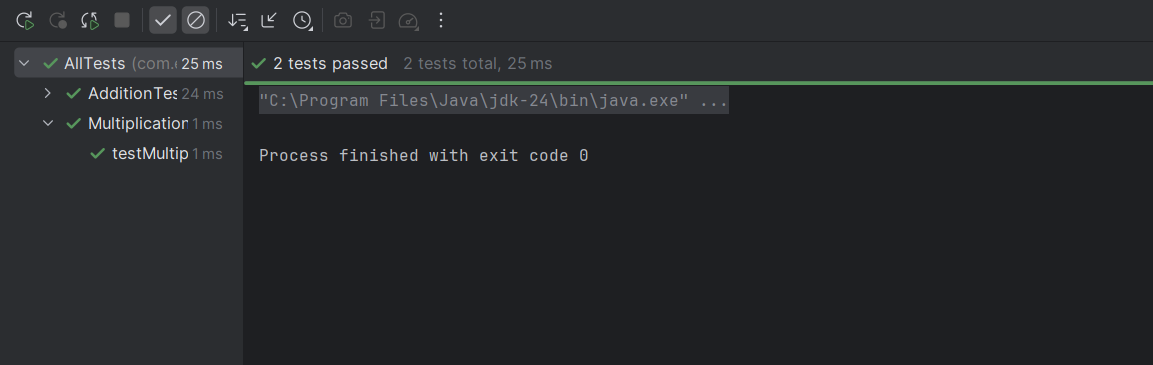
**AdditionTest.java:**

****

**MultiplicationTest.java:**

****

**OUTPUT:**

****

**Exercise 3: Test Execution Order**

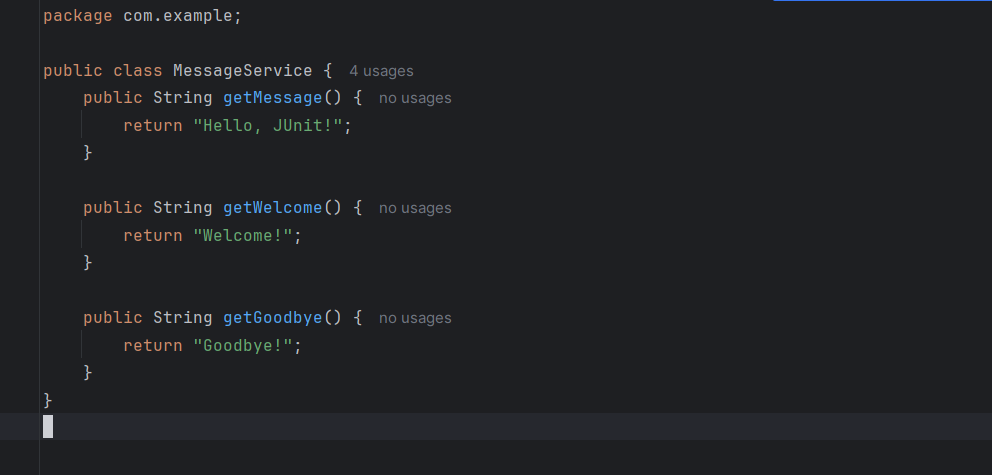
Scenario: You want to control the order in which tests are executed.

Steps: 1. Create a test class `OrderedTests`.

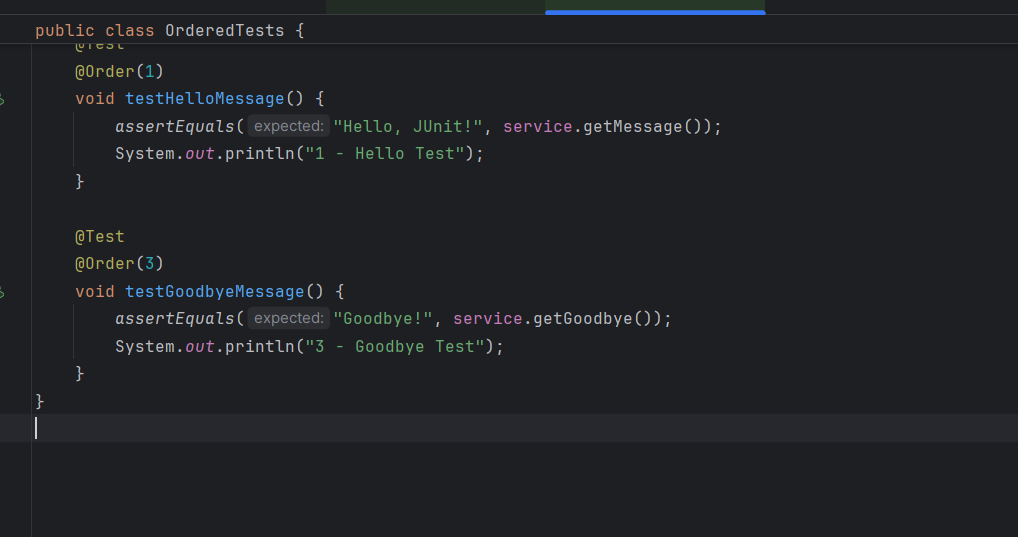
2. Use JUnit's `@TestMethodOrder` and `@Order` annotations.

**SOLUTION:**

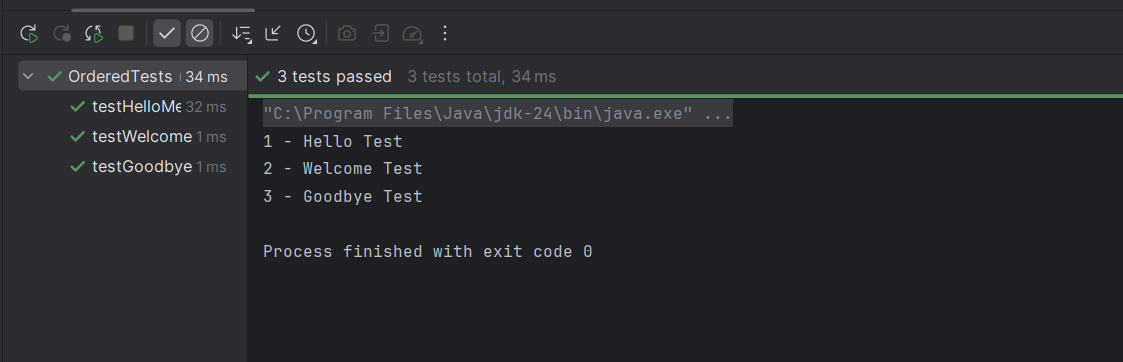
**MessageService.java:**

****

**OrderedTests.java:**

****

**SOLUTION:**

****

**Exercise 4: Exception Testing**

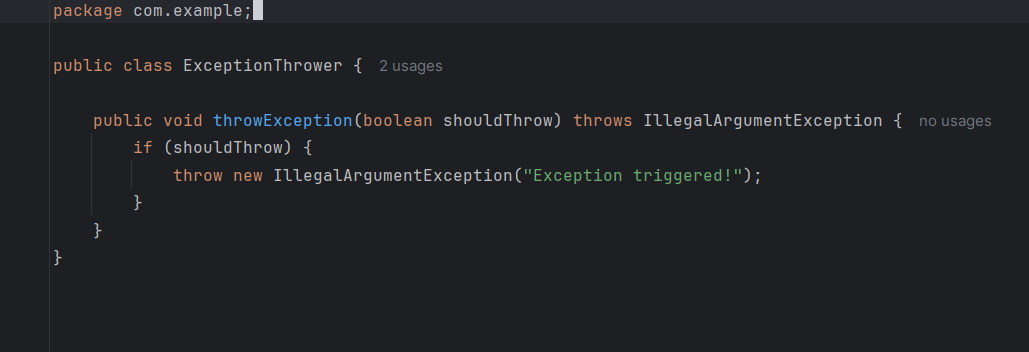
Scenario: You want to test that a method throws the expected exception.

Steps: 1. Create a class `ExceptionThrower` with a method `throwException`.

2. Write a test class `ExceptionThrowerTest` that tests the method for the expected exception.

**SOLUTION:**

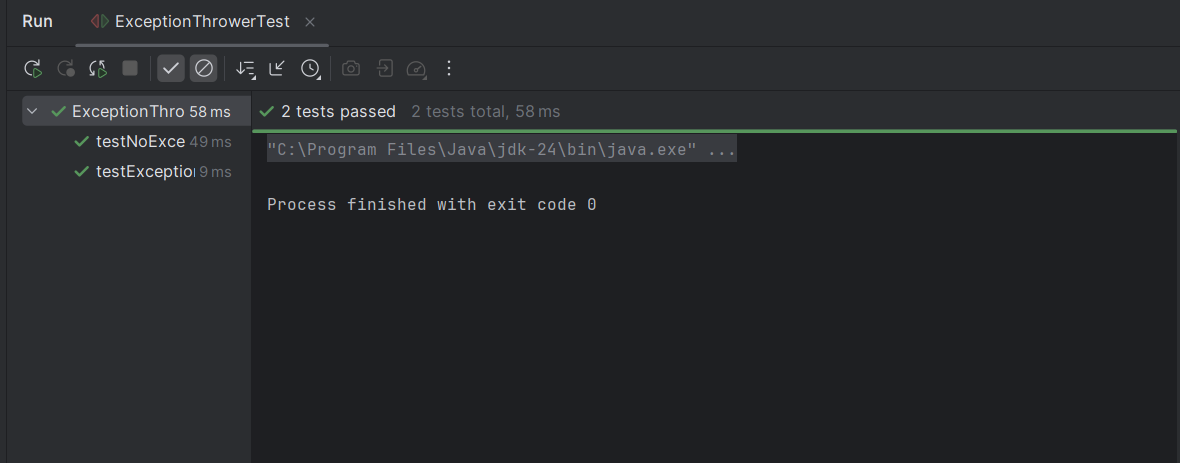
**ExceptionThrower.java:**

****

**ExceptionThrowerTests.java:**

****

**SOLUTION:**

****

**Exercise 5: Timeout and Performance Testing**

Scenario: You want to ensure that a method completes within a specified time limit.

Steps: 1. Create a class `PerformanceTester` with a method `performTask`.

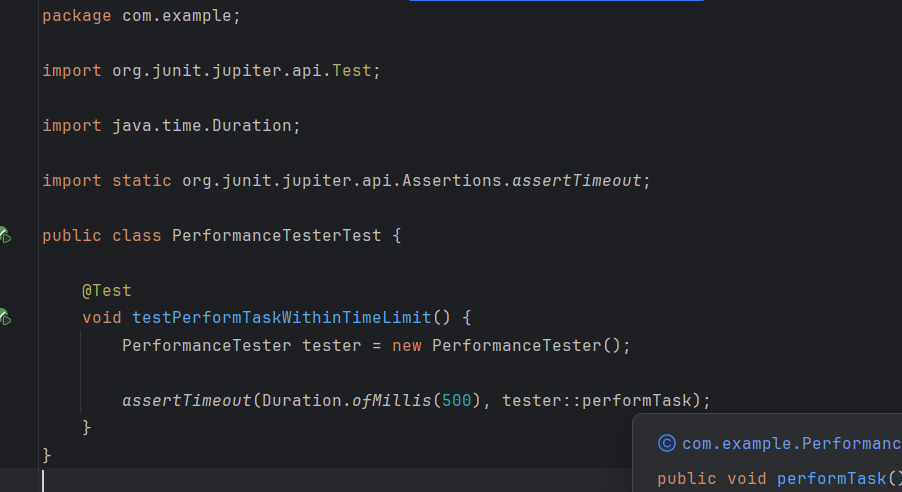
2. Write a test class `PerformanceTesterTest` that tests the method for timeout.

**SOLUTION:**

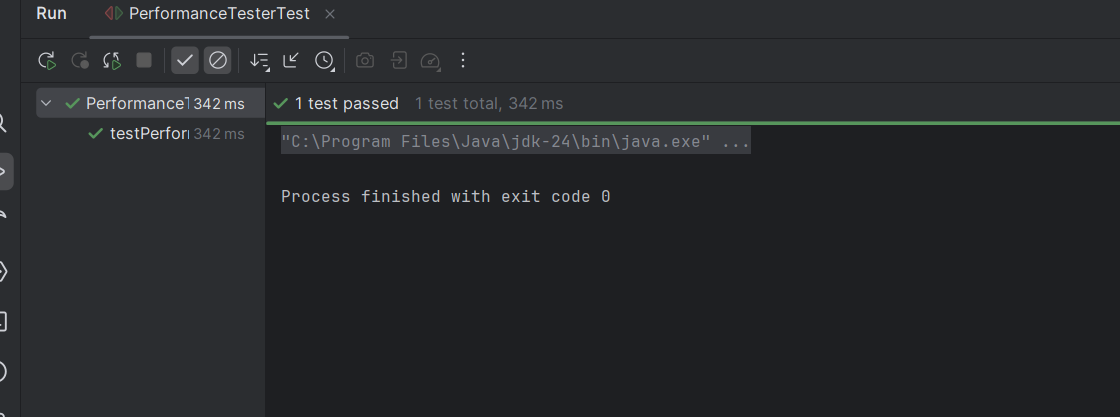
**PerformanceTester.java:**

****

**PerformanceTesterTest.java:**

****

**OUTPUT:**

****