

# Assignment 08

## Unit test

### 1.1. MỤC ĐÍCH VÀ NỘI DUNG

- Trước tiên, người học được hướng dẫn thực hành sử dụng Eclipse IDE, JUnit Framework và thực hành thiết kế các Testcases. Sau đó, người học sẽ được học cách tạo và sinh Javadoc tự động với Eclipse
- Trong phần Unit test, này người học sẽ được hướng dẫn về:
  - Các bước thiết kế TestCase cho UnitTest
  - Tạo TestCase cho các class bằng Eclipse và Junit
  - Tiếp cận phương pháp Test Driven Development (TDD) trong quá trình xây dựng phần mềm
- Tiếp theo người học sẽ được học về cách refactoring code. Người học sẽ được hướng dẫn nhận diện một class hoặc method cần được refactoring và sau đó sẽ thực hành refactoring một class
- Sau bài học người học sẽ có thể áp dụng những kiến thức mình đã được học vào bài tập về nhà cũng như Capstone Project

### 1.2. CHUẨN BỊ

- Người học cần cài đặt sẵn Eclipse và môi trường Java11 trên máy cá nhân
- Clone project từ: <https://github.com/leminhnguyen/AIMS-Student>

### 1.3. NỘI DUNG CHI TIẾT

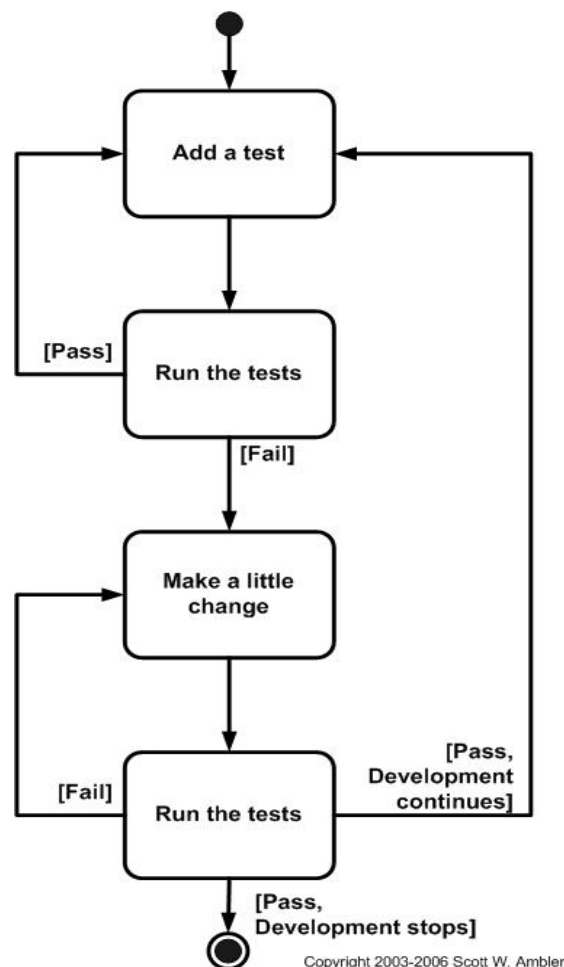
#### 1.3.1. Kiểm thử đơn vị

##### 1.3.1.1. *Bắt đầu với Kiểm thử đơn vị*

- Test là một phần nằm trong phần mềm phát triển, thực thi các phần khác nhau của phần mềm và xác nhận kết quả có đúng như mong đợi (expected result) hoặc thực thi đúng theo luồng các sự kiện mong muốn (behavior testing)
- Unit Test là các đoạn code hoặc chương trình được viết bởi lập trình viên nhằm kiểm tra hành vi hoặc trạng thái của một chức năng cụ thể trong PM
- Unit Test nhắm tới một đơn vị nhỏ của phần mềm như method hoặc class. Những phụ thuộc bên ngoài nên loại bỏ ra khỏi unit test, thay vào đó sử dụng mock được tạo ra bởi các test framework
- Unit Test không phù hợp cho việc testing cho những thành phần có giao diện người dùng phức tạp hoặc có sự giao tiếp giữa các thành phần khác nhau trong chương trình.

### 1.3.2. Làm quen với TDD (test driven development)

- TDD là quá trình phát triển phần mềm dựa trên việc các yêu cầu chức năng của phần mềm được chuyển thành các testcase trước khi phần mềm được phát triển hoàn thiện và theo dõi quá trình phát triển phần mềm bằng cách kiểm thử phần mềm với các testcase đó. Hay nói một cách đơn giản là chúng ta sẽ tiến hành viết test trước khi code và điều này trái ngược với việc một phần mềm được hoàn thiện rồi mới bắt đầu viết test



- Các bước thực hiện TDD
  - + **B1 - Add a Test:** bước đầu tiên trước khi bắt đầu phát triển một tính năng mới sẽ là viết test cho chức năng cần test (thông thường test ban đầu sẽ fail do có thể class cần Test chưa được viết)
  - + **B2 - Run the Tests:** chạy các đoạn test đã viết
  - + **B3 - Make A Litte Change:** tiến hành viết code hoặc cập nhật chức năng để có thể vượt qua các Tests
  - + **B4 - Run the Tests:** thực hiện chạy lại các test, nếu như failed thì ta cần quay lại cập nhật code và chạy lại test cho đến khi pass. Sau khi vượt qua các test thì ta bắt đầu lặp lại quá trình từ đầu

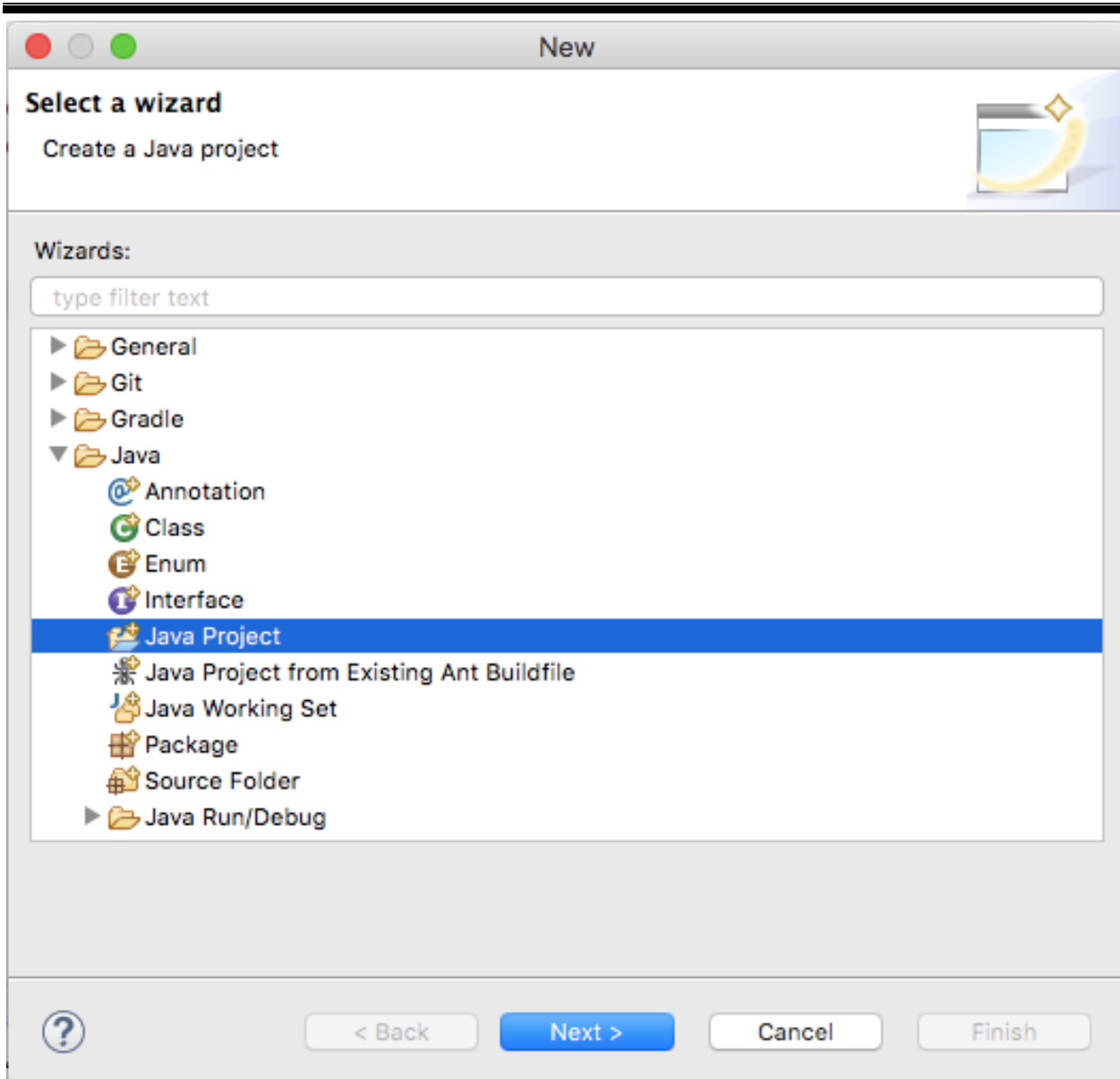
### **1.3.3. Làm quen với JUNIT**

- Junit5 là phiên bản thứ 5 của Junit, một framework sử dụng annotation để nhận diện ra các phương thức test và Junit là một phần mềm mã nguồn mở
- Các phiên bản của Junit đi kèm theo với Eclipse, vì vậy chúng ta không cần phải cài đặt thủ công
- Chi tiết về bước add Junit5 vào trong Eclipse sẽ được trình bày chi tiết ở bước tiếp theo
- Junit nhận diện các phương thức cần test bằng các annotation (bắt đầu bằng @), một vài annotation được sử dụng phổ biến trong Junit5
  - + @Test: Biểu thị một phương thức test (Test Method)
  - + @DisplayName: Khai báo tên cho Test Class hoặc Test Method
  - + @BeforeEach: phương thức được thực thi trước khi bắt đầu mỗi Test Method
  - + @AfterEach: phương thức được thực thi sau khi chạy xong mỗi Test Method
  - + @BeforeAll: phương thức được thực thi trước khi tất cả các Test Method được thực hiện (VD: connect tới DB,..)
  - + @AfterAll: phương thức được thực thi sau khi tất cả các Test Method được thực hiện (VD: đóng connection tới DB,...)
- Để tìm hiểu chi tiết hơn và cụ thể hơn về Junit5, người học có thể tham khảo theo các link sau
  - + <https://www.journaldev.com/20834/junit5-tutorial>
  - + <https://junit.org/junit5/docs/current/user-guide/>

### **1.3.4. Thực hành cơ bản với JUnit**

#### **Bài 1.1 Mục tiêu: Xác nhận lại việc cài đặt JUnit và Java đã thành công**

**STEP1: Mở Eclipse, Tạo mới project tên TestJUnit ==> Finish**



The screenshot shows the 'New Java Project' dialog box in Eclipse. The title bar says 'New Java Project'. Below the title bar, it says 'Create a Java Project' and 'Create a Java project in the workspace or in an external location.' There is a folder icon on the right. The 'Project name' field contains 'TestJUnit'. The 'Use default location' checkbox is checked. The 'Location' field shows the path '/Users/ThanDieu/eclipse-workspace/TestJUnit' with a 'Browse...' button next to it. Under the 'JRE' section, the 'Use an execution environment JRE:' radio button is selected, and the dropdown menu shows 'JavaSE-1.8'. Other options include 'Use a project specific JRE:' (set to 'Java SE 8 [1.8.0\_51]') and 'Use default JRE (currently 'Java SE 8 [1.8.0\_51]')' with a 'Configure JREs...' link. The 'Project layout' section has 'Create separate folders for sources and class files' selected, with a 'Configure default...' link. The 'Working sets' section has 'Add project to working sets' unchecked, a 'New...' button, and a 'Working sets:' dropdown menu with a 'Select...' button. At the bottom, there is a question mark icon, '< Back', 'Next >', 'Cancel', and 'Finish' buttons.

**New Java Project**

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name:

☒ Use default location

Location:

JRE

☒ Use an execution environment JRE:

☐ Use a project specific JRE:

☐ Use default JRE (currently 'Java SE 8 [1.8.0\_51]') [Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

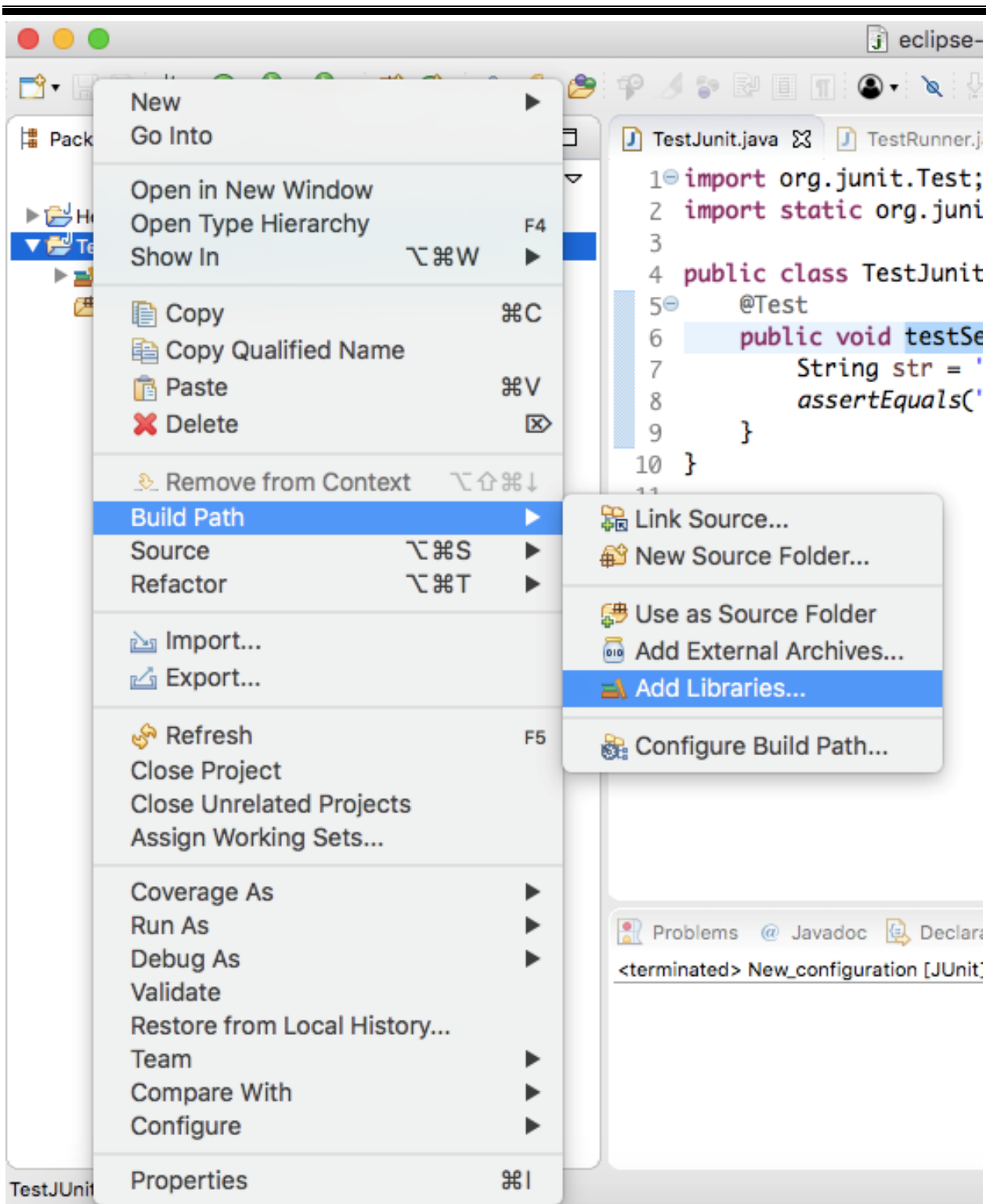
Working sets

☐ Add project to working sets

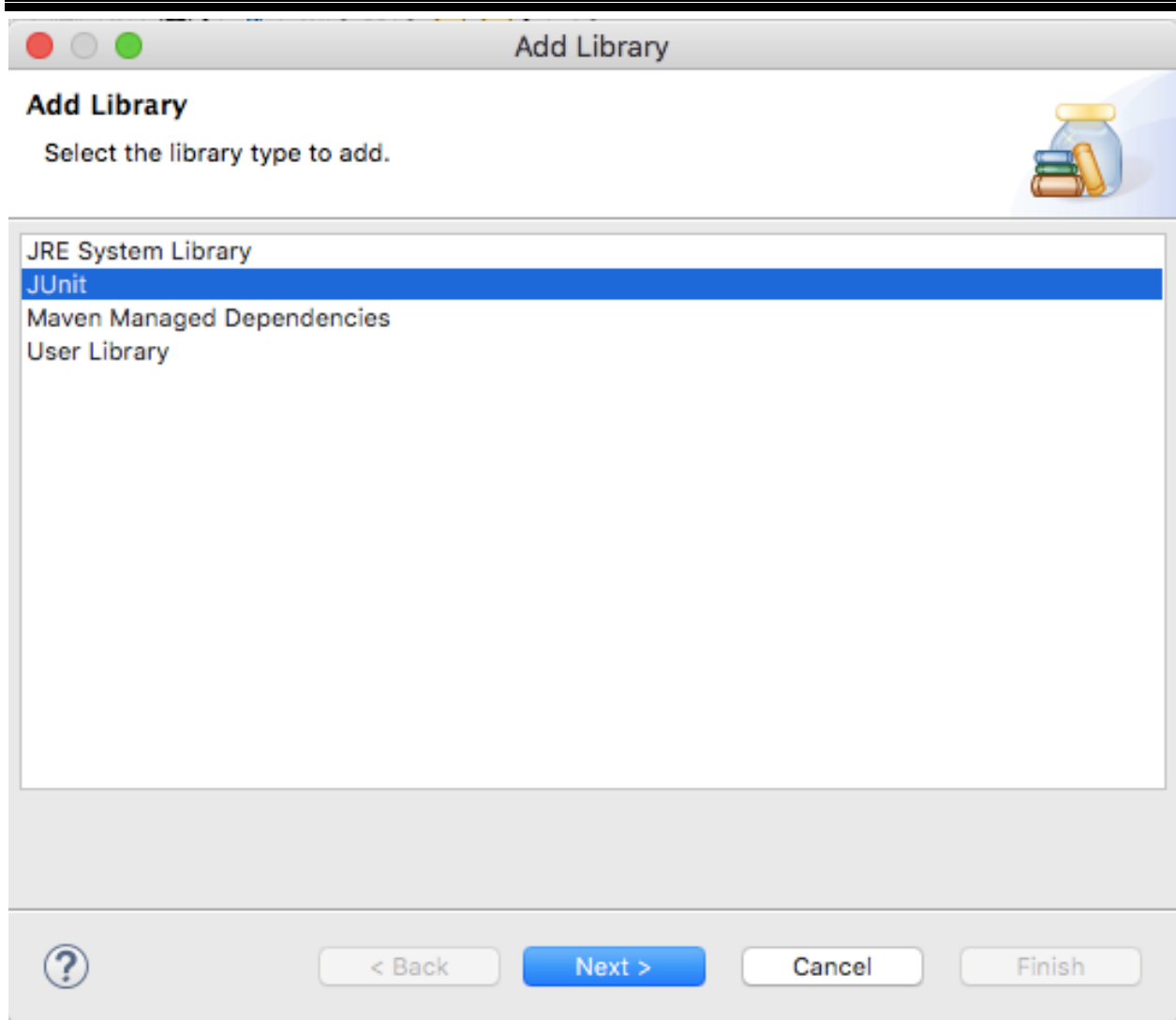
Working sets:

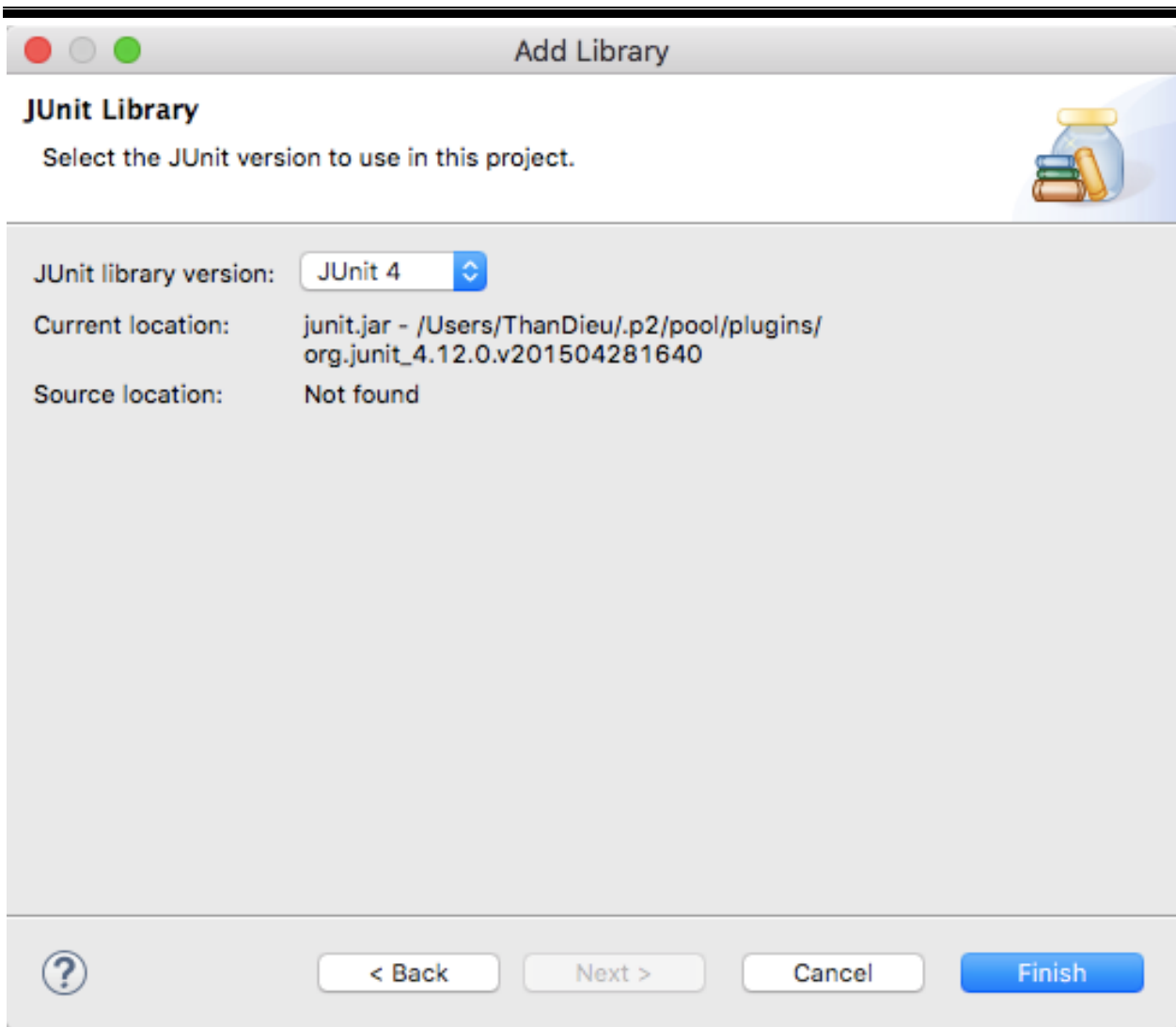
## STEP 2: Thêm thư viện Junit cho projec

Chuột phải vào project TestJUnit ==> Build Path ==> Add Library



**Chọn Junit sau đó chọn Junit 4 rồi Finish**





### STEP 3: Tạo mới class java tên TestJUnit

Chuột phải vào thư mục src của project TestJUnit ==> Chọn New ==> Chọn Class



**New Java Class**

**Java Class**

The use of the default package is discouraged.

Source folder:

Package:  (default)

☐ Enclosing type:

---

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)

☐ Constructors from superclass

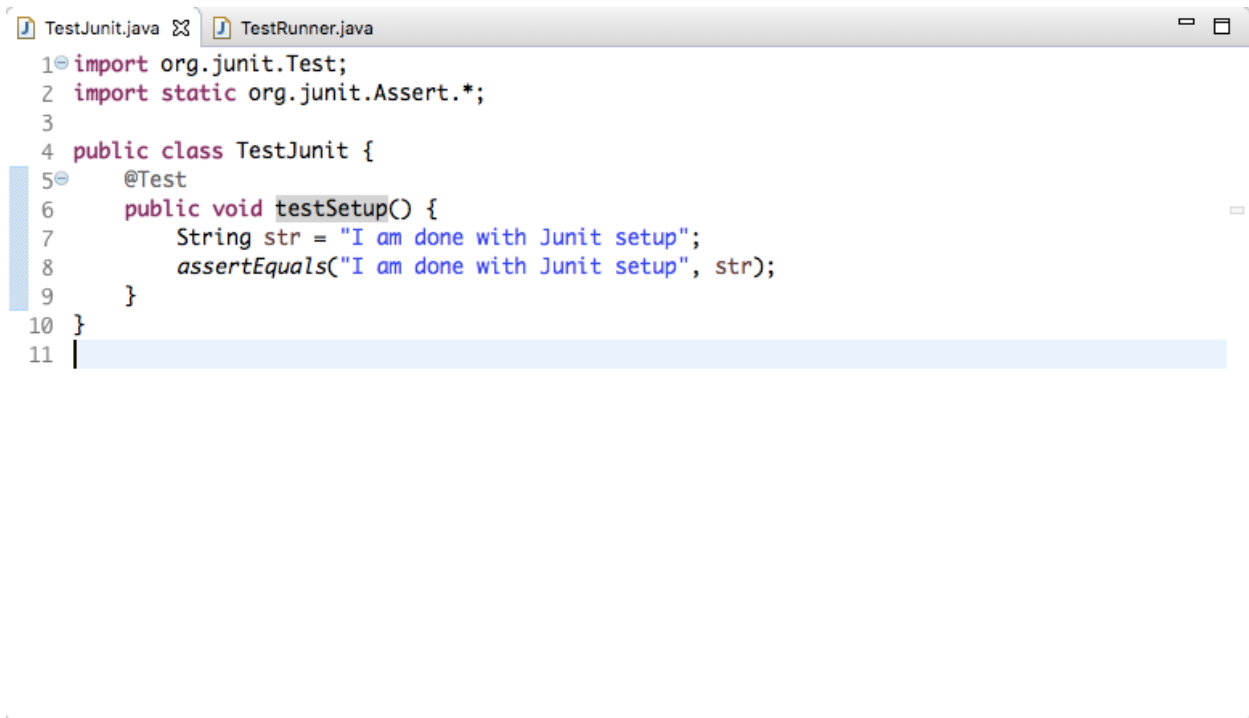
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

**Đặt tên class là TestJUnit ==> Finish**

**Step 4: Trong class TestJUnit.java, chúng ta sẽ setup một test case đơn giản**



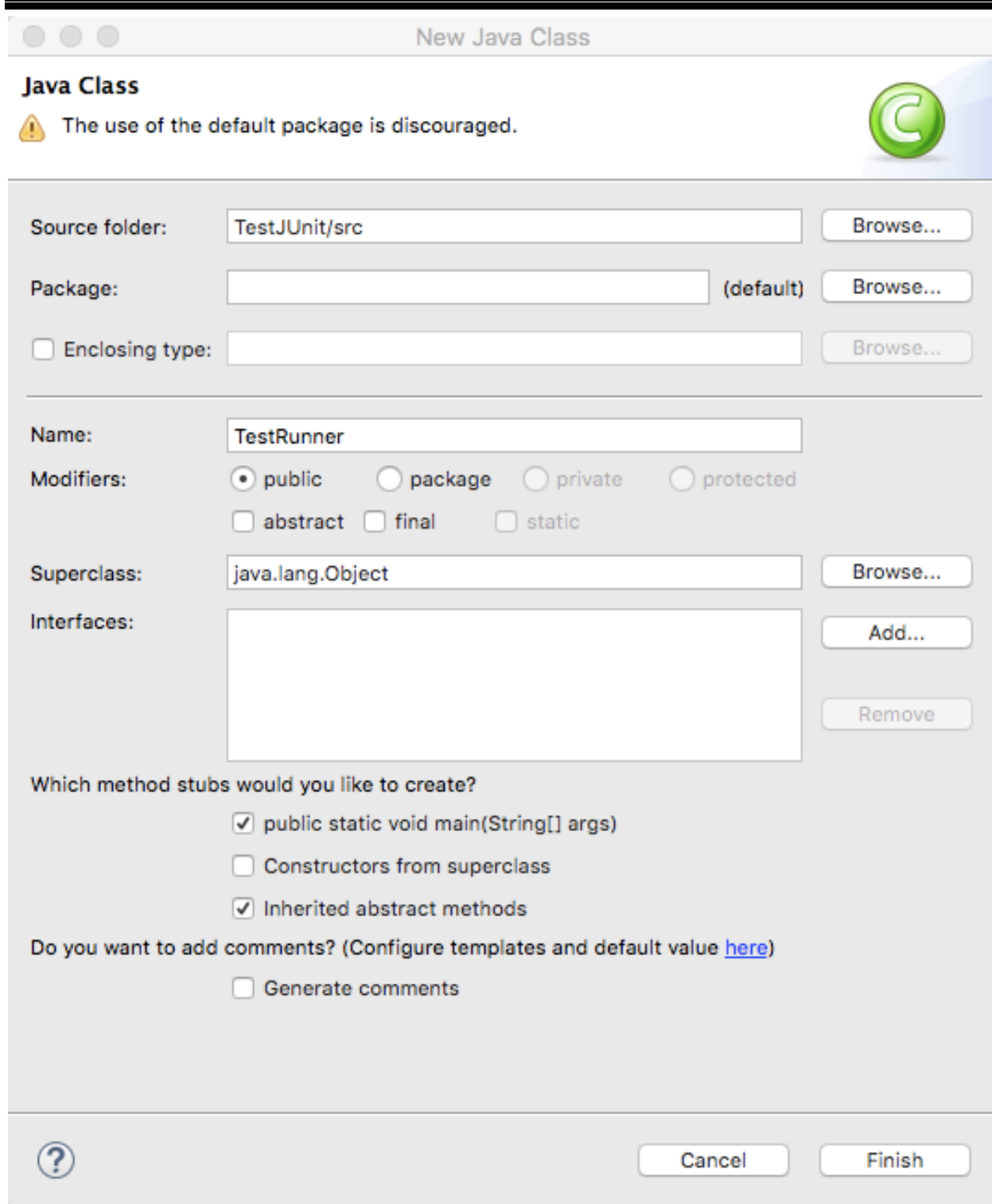
```
1 import org.junit.Test;
2 import static org.junit.Assert.*;
3
4 public class TestJUnit {
5     @Test
6     public void testSetup() {
7         String str = "I am done with Junit setup";
8         assertEquals("I am done with Junit setup", str);
9     }
10 }
11
```

Lưu ý việc thêm đủ các package của Junit và sử dụng annotation `@Test` của Junit trước method `testSetup()` của chúng ta. Annotation này sẽ thông báo cho Junit biết đây là một unit test.

#### Step 5: Tạo mới class TestRunner để chạy test case đã tạo ở Step 4


Chuột phải vào thư mục src ==> New Class ==> Đặt tên class mới là TestRunner

Chọn `public static void main (String[] args)` để tự sinh sẵn hàm `main()` trong class này.



**New Java Class**

**Java Class**

 The use of the default package is discouraged.

Source folder:

Package:

☐ Enclosing type:

---

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?


☒ public static void main(String[] args)

☐ Constructors from superclass

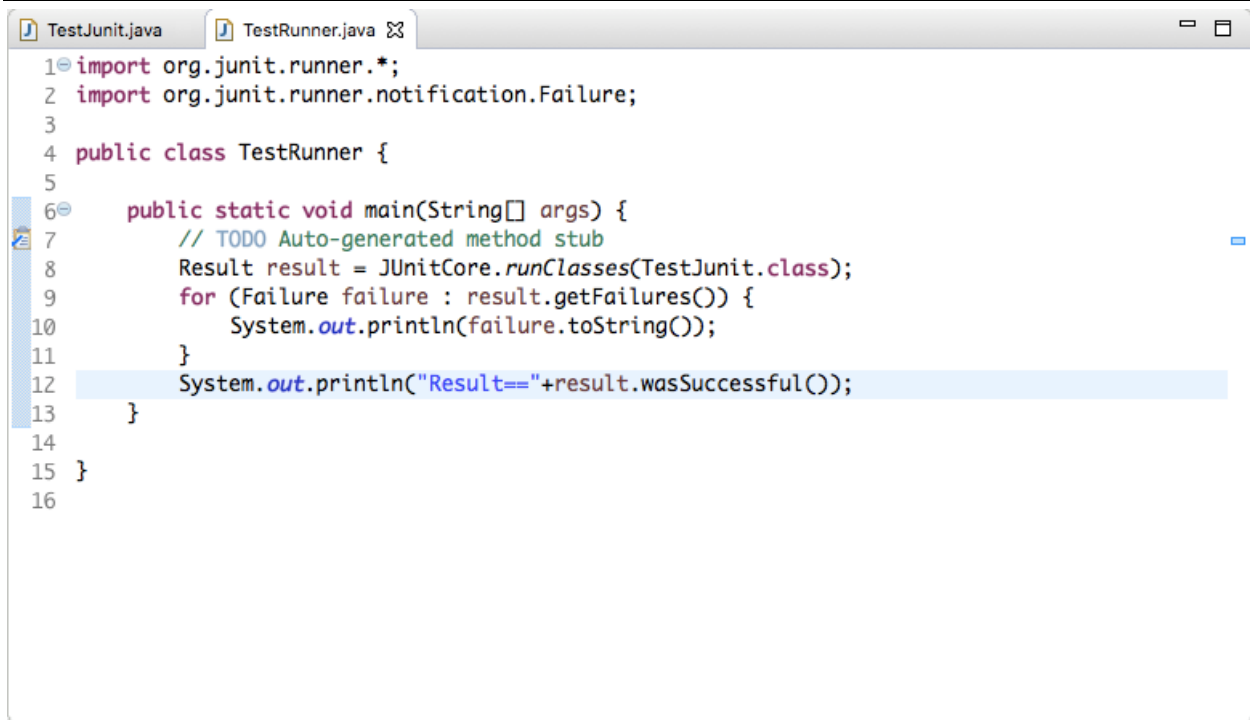
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments



Trong hàm main() của class này, chúng ta sẽ chạy test case có trong TestJUnit.java



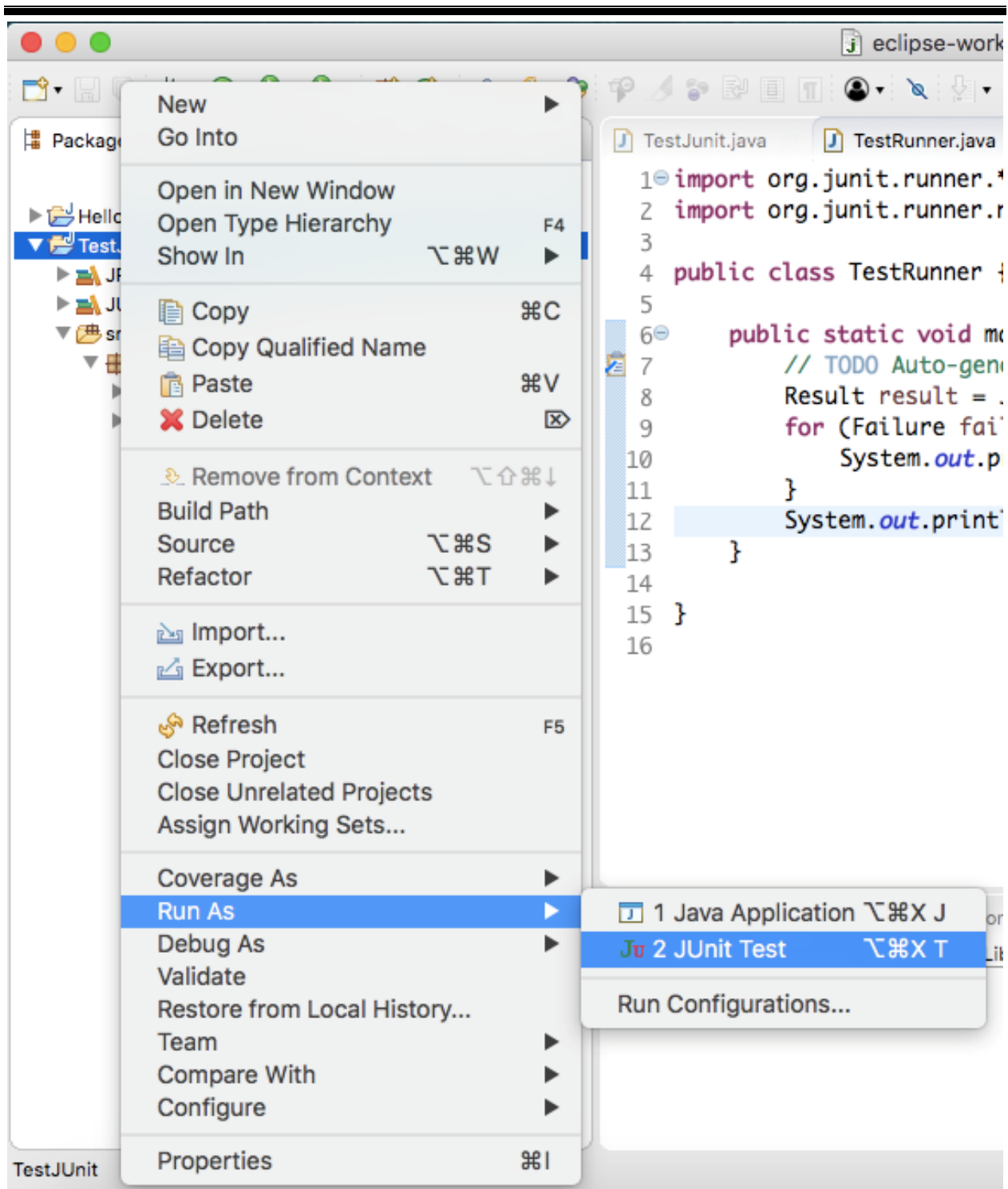
```
1 import org.junit.runner.*;
2 import org.junit.runner.notification.Failure;
3
4 public class TestRunner {
5
6     public static void main(String[] args) {
7         // TODO Auto-generated method stub
8         Result result = JUnitCore.runClasses(TestJUnit.class);
9         for (Failure failure : result.getFailures()) {
10             System.out.println(failure.toString());
11         }
12         System.out.println("Result==" + result.wasSuccessful());
13     }
14
15 }
16
```

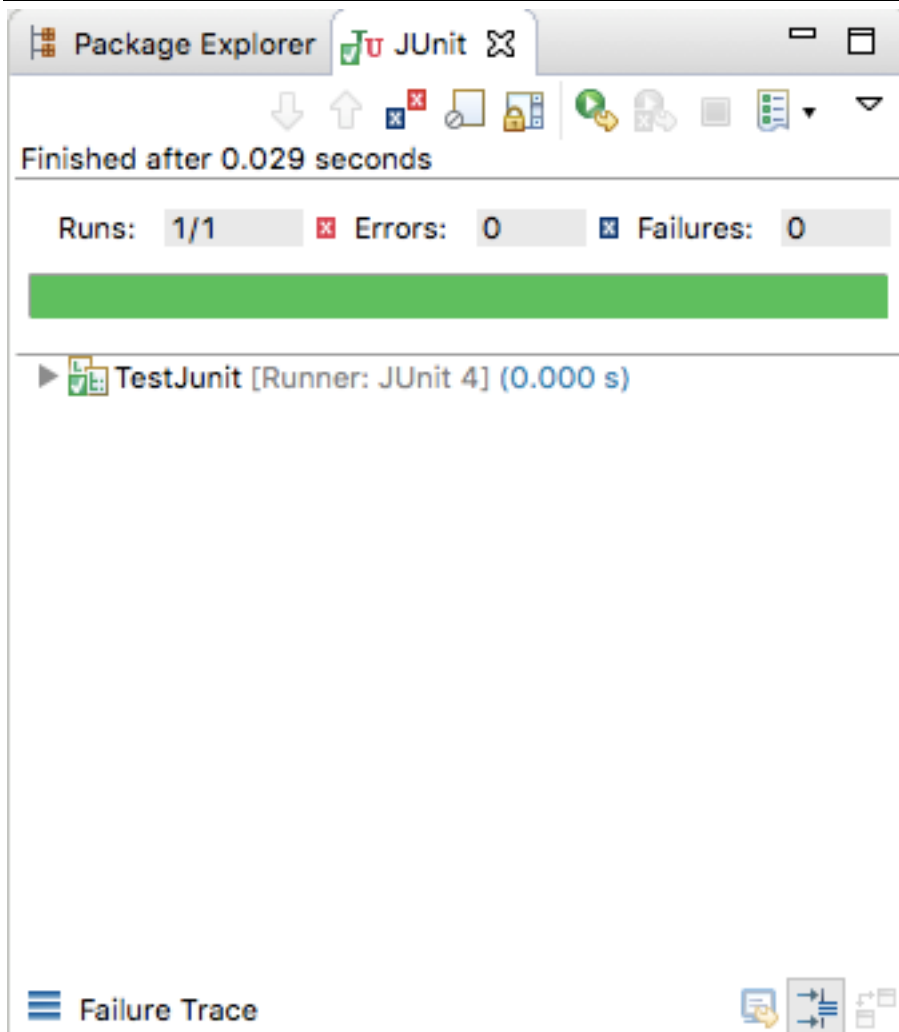
Chúng ta sử dụng object Result để chứa tất cả các unit test có trong test case TestJUnit.

Những test bị fail sẽ được lấy về từ hàm getFailures của class Result.

### Step 6: Chạy chương trình review kết quả

Chuột phải vào project TestJUnit ==> Run As ==> Junit Test





Test case đã chạy thành công và không có failure.

Note: Project này cho phép chúng ta bước đầu làm quen với môi trường Java và Junit tạo test unit để xác nhận việc cài đặt Java và Junit cũng như môi trường phát triển Eclipse đã hoàn tất.

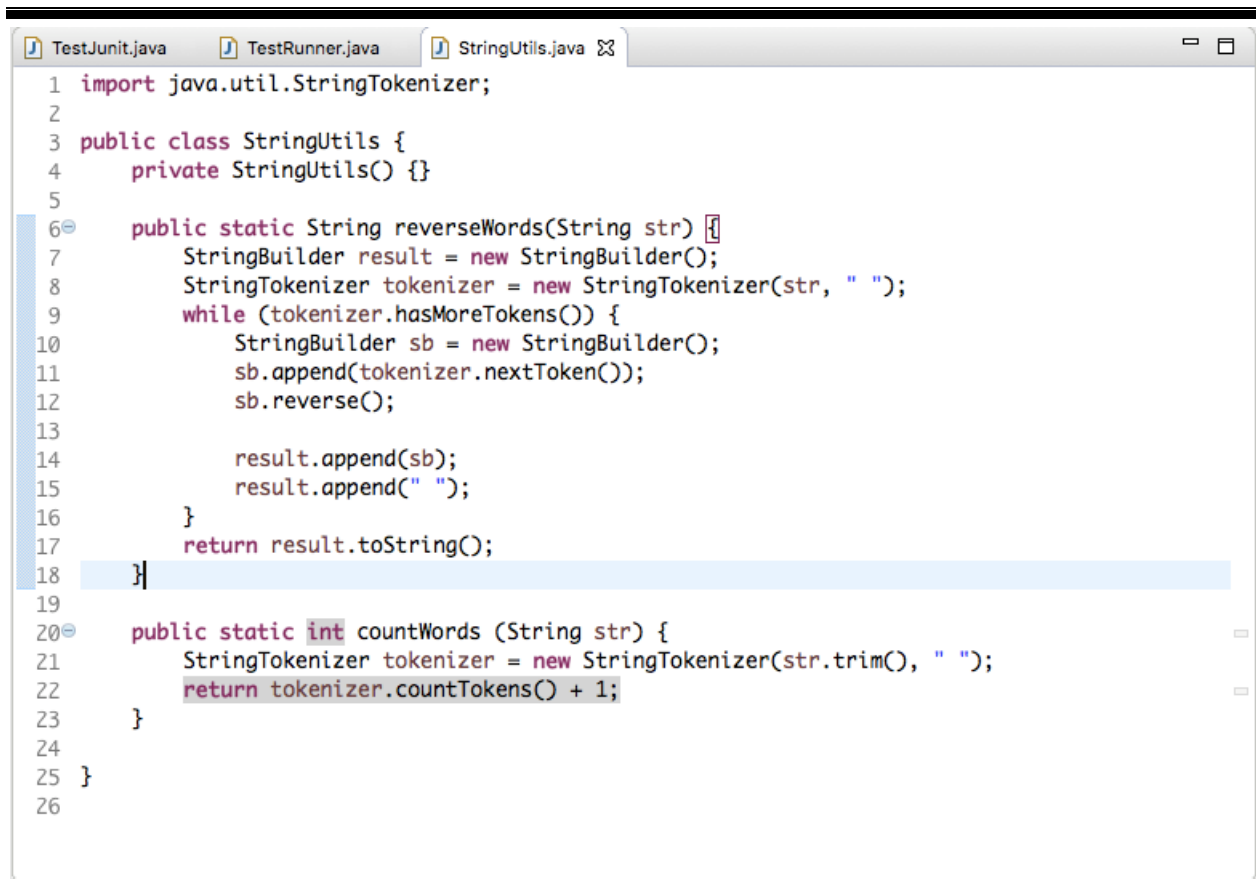
### **Bài 1.2 Mục tiêu: Thực hành viết nhiều hơn một test case cho class cần test**

#### **Step 1: Tạo mới project tên Utils**

#### **Step 2: Add Junit vào class path của project giống bài trước**

#### **Step 3: Tạo mới class trong thư mục src, đặt tên là StringUtils**

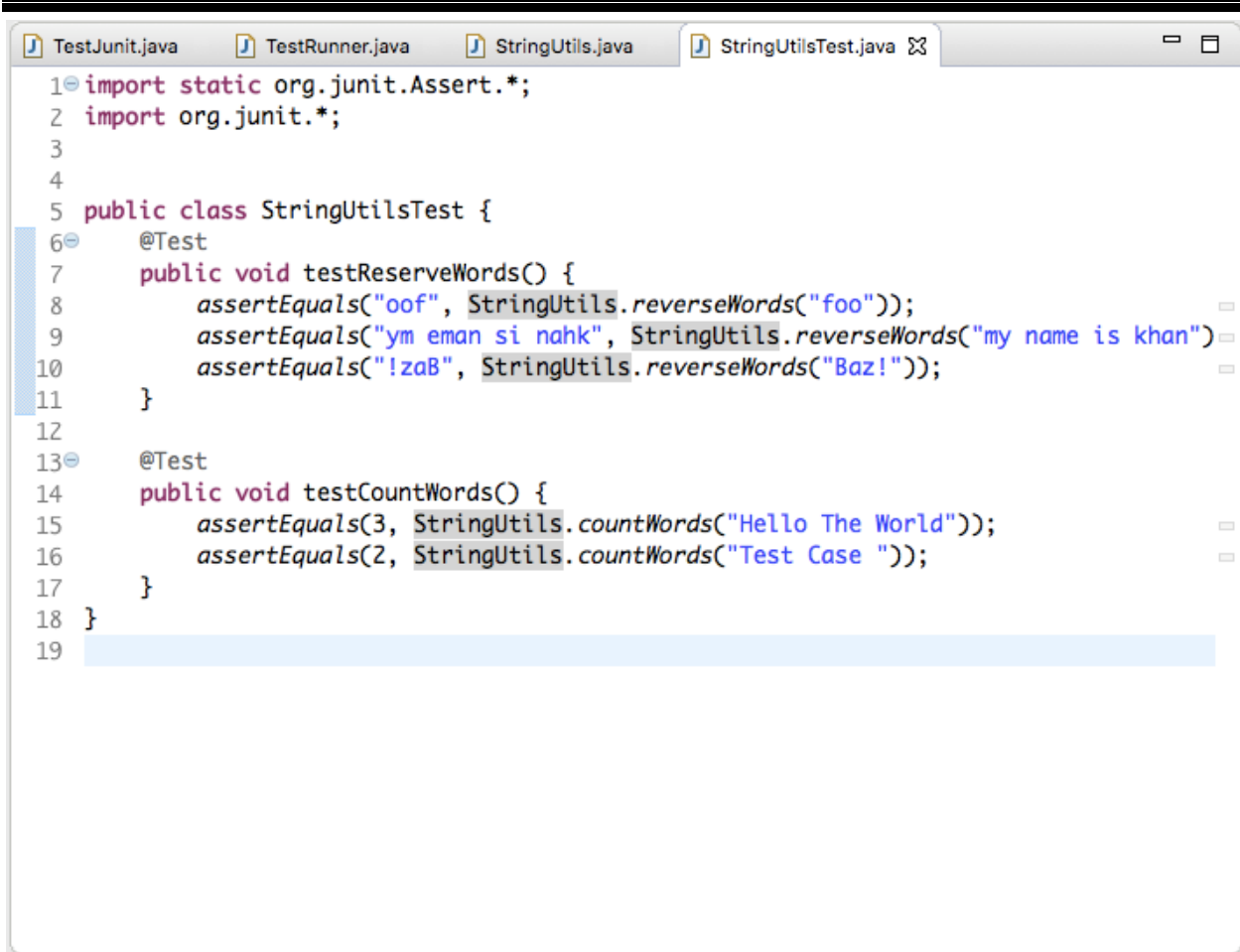
Trong class StringUtils, chúng ta tạo ra 2 phương thức: `reverseWords(String str)` để đảo ngược word và `countWords(String str)` để đếm số từ trong chuỗi. Giả thiết rằng chuỗi nhập vào các từ cách nhau bằng 1 dấu cách.



```
1 import java.util.StringTokenizer;
2
3 public class StringUtils {
4     private StringUtils() {}
5
6     public static String reverseWords(String str) {
7         StringBuilder result = new StringBuilder();
8         StringTokenizer tokenizer = new StringTokenizer(str, " ");
9         while (tokenizer.hasMoreTokens()) {
10             StringBuilder sb = new StringBuilder();
11             sb.append(tokenizer.nextToken());
12             sb.reverse();
13
14             result.append(sb);
15             result.append(" ");
16         }
17         return result.toString();
18     }
19
20     public static int countWords (String str) {
21         StringTokenizer tokenizer = new StringTokenizer(str.trim(), " ");
22         return tokenizer.countTokens() + 1;
23     }
24 }
25
26
```

#### Step 4: Tạo test case cho class StringUtils

Trong thư mục src tạo mới class tên StringUtilsTest, trong class này chúng ta sẽ tạo ra hai unit test tương ứng với hai hàm của class StringUtils.

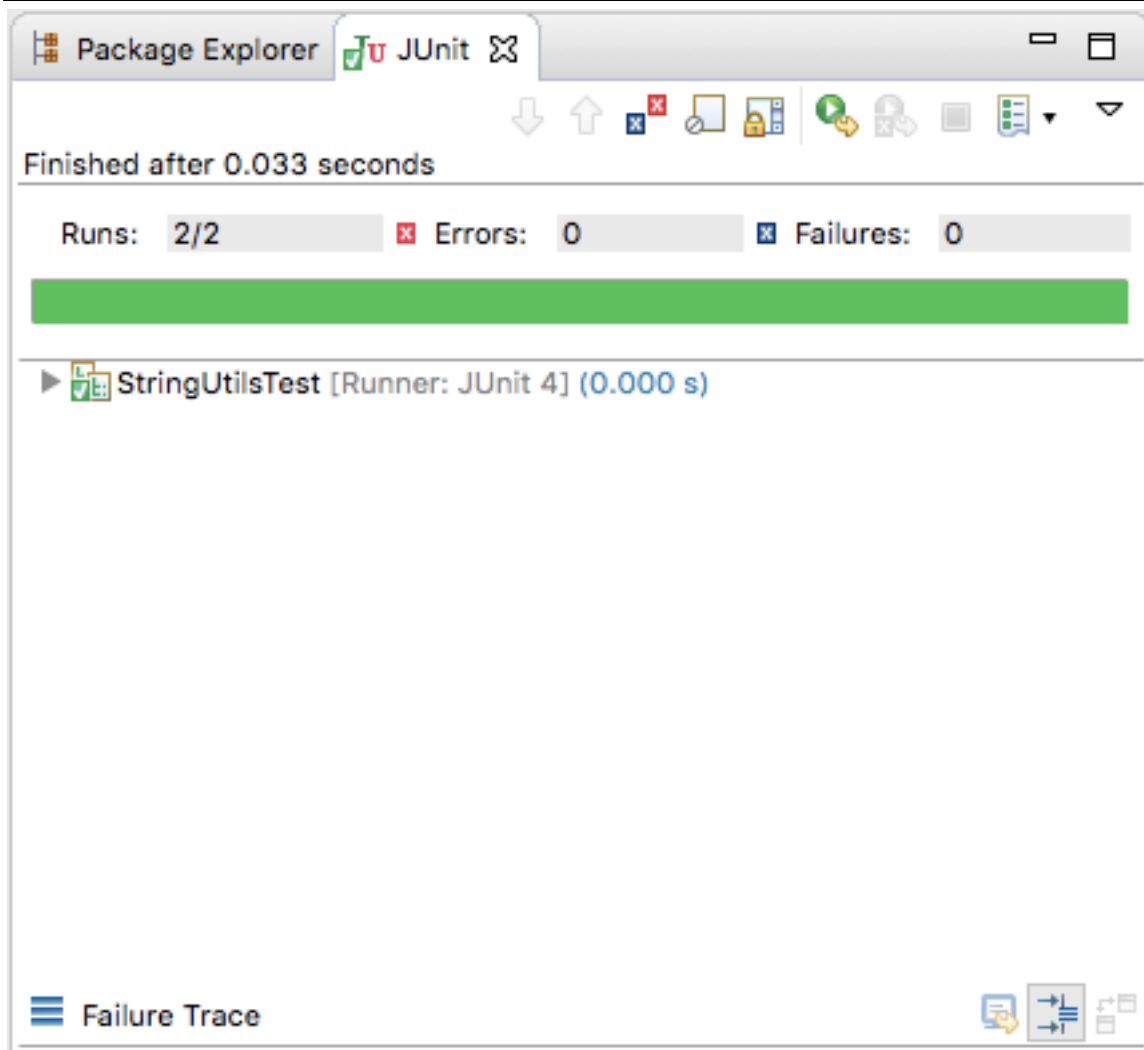


```
1 import static org.junit.Assert.*;
2 import org.junit.*;
3
4
5 public class StringUtilsTest {
6     @Test
7     public void testReverseWords() {
8         assertEquals("oof", StringUtils.reverseWords("foo"));
9         assertEquals("ym eman si nahk", StringUtils.reverseWords("my name is khan"));
10        assertEquals("!zaB", StringUtils.reverseWords("Baz!"));
11    }
12
13    @Test
14    public void testCountWords() {
15        assertEquals(3, StringUtils.countWords("Hello The World"));
16        assertEquals(2, StringUtils.countWords("Test Case "));
17    }
18 }
19
```

### Step 5: Chạy test case

Chuột phải vào class StringUtilsTest ==> Chọn Run As ==> Chọn Junit Test





### **Bài 1.3 Mục tiêu: Tham số hoá các test case để test cho nhiều trường hợp của dữ liệu**

Trong JUnit, chúng ta có thể đưa tham số vào các unit test bằng 2 cách:

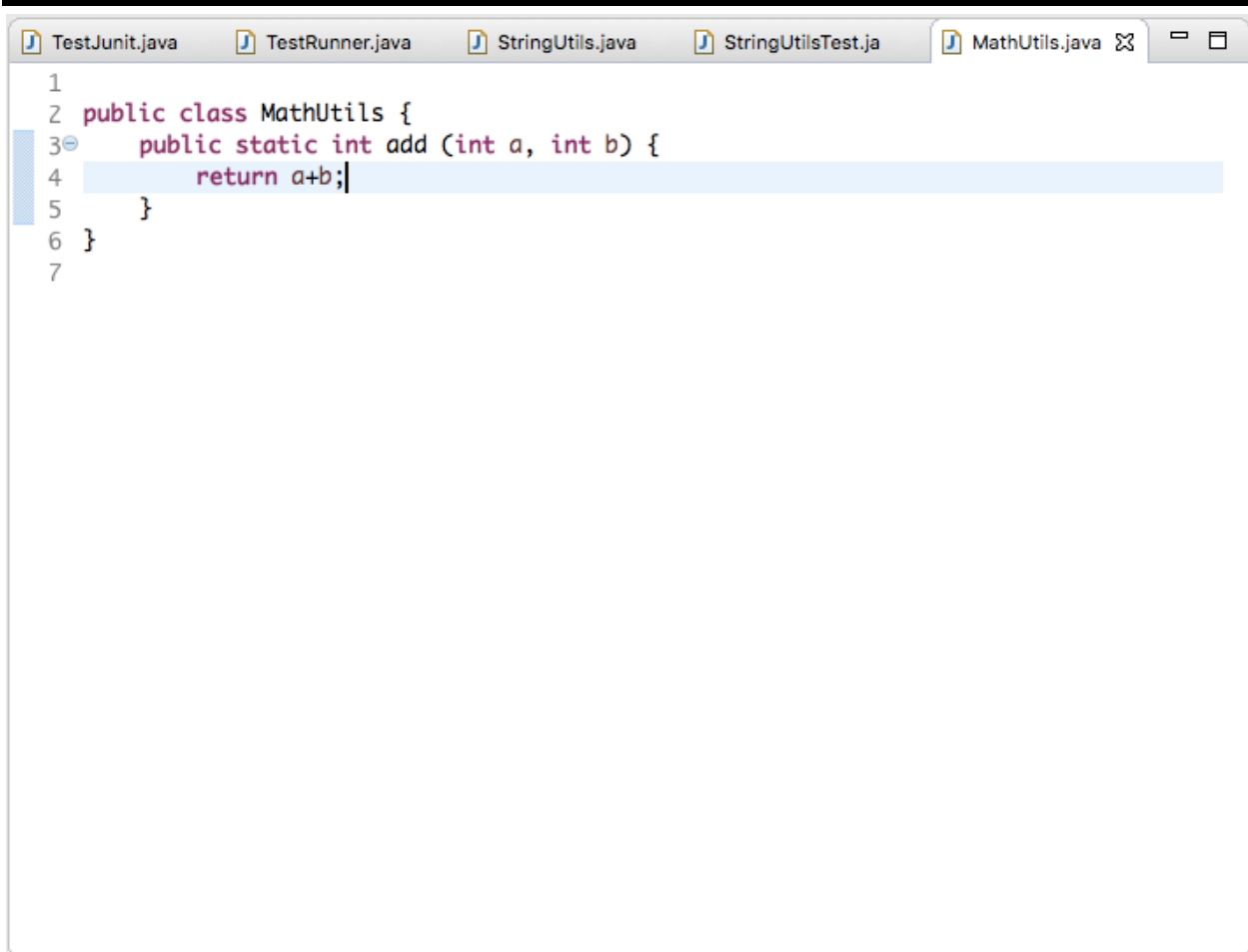
1. Thông qua phương thức khởi tạo (Constructor)
2. Sử dụng filed injection với annotation `@Parameter`

#### **Step 1: Tạo mới project tên Parameterizing**

#### **Step 2: Add Junit vào class path của project**

#### **Step 3: Tạo mới class tên MathUtils**

Trong class MathUtils, tạo một phương thức static đơn giản trả về tổng của hai số nguyên bất kì. Chúng ta sẽ sử dụng class này để tạo các test case với Junit.



```
1
2 public class MathUtils {
3     public static int add (int a, int b) {
4         return a+b;
5     }
6 }
7
```

#### Step 4: Parameterized với Junit thông qua phương thức khởi tạo Constructor

##### Tạo mới class ParameterizedTest

Trong class này chúng ta sẽ sử dụng annotation `@RunWith` của junit để chỉ cho Junit biết rằng test case này sẽ được chạy cùng với class nào. Cụ thể là `@RunWith(value = Parameterized.class)` của chính Junit.

Chúng ta sẽ thêm cấu hình tham số thông qua phương thức khởi tạo của class `ParameterizedTest`. Tham số sẽ là 1 array gồm 3 phần tử, tương ứng với number 1 (a), number 2 (b) và tổng.

```

StringUtils.jav StringUtilsTest MathUtils.java ParameterizedTe »_2
1 import static org.junit.Assert.*;
2
3 import java.util.Arrays;
4 import java.util.Collection;
5
6 import org.junit.Test;
7 import org.junit.runner.RunWith;
8 import org.junit.runners.Parameterized;
9 import org.junit.runners.Parameterized.Parameters;
10
11 @RunWith(value=Parameterized.class)
12 public class ParameterizedTest {
13
14     private int number1;
15     private int number2;
16     private int sum;
17
18     public ParameterizedTest(int number1, int number2, int sum) {
19         this.number1 = number1;
20         this.number2 = number2;
21         this.sum = sum;
22     }
23
24     @Parameters(name = "{index}: testAdd({0}+{1}) = {2}")
25     public static Collection<Object[]> data() {
26         return Arrays.asList(new Object[][] {
27             {1,1,2},
28             {2,2,4},
29             {8,-2,6},
30             {-7,8,1},
31             {-9,-1,-10}
32         });
33     }
34
35     @Test
36     public void test_addTwoNumbers() {
37         assertEquals(sum, MathUtils.add(number1, number2));
38     }
39 }
40
41

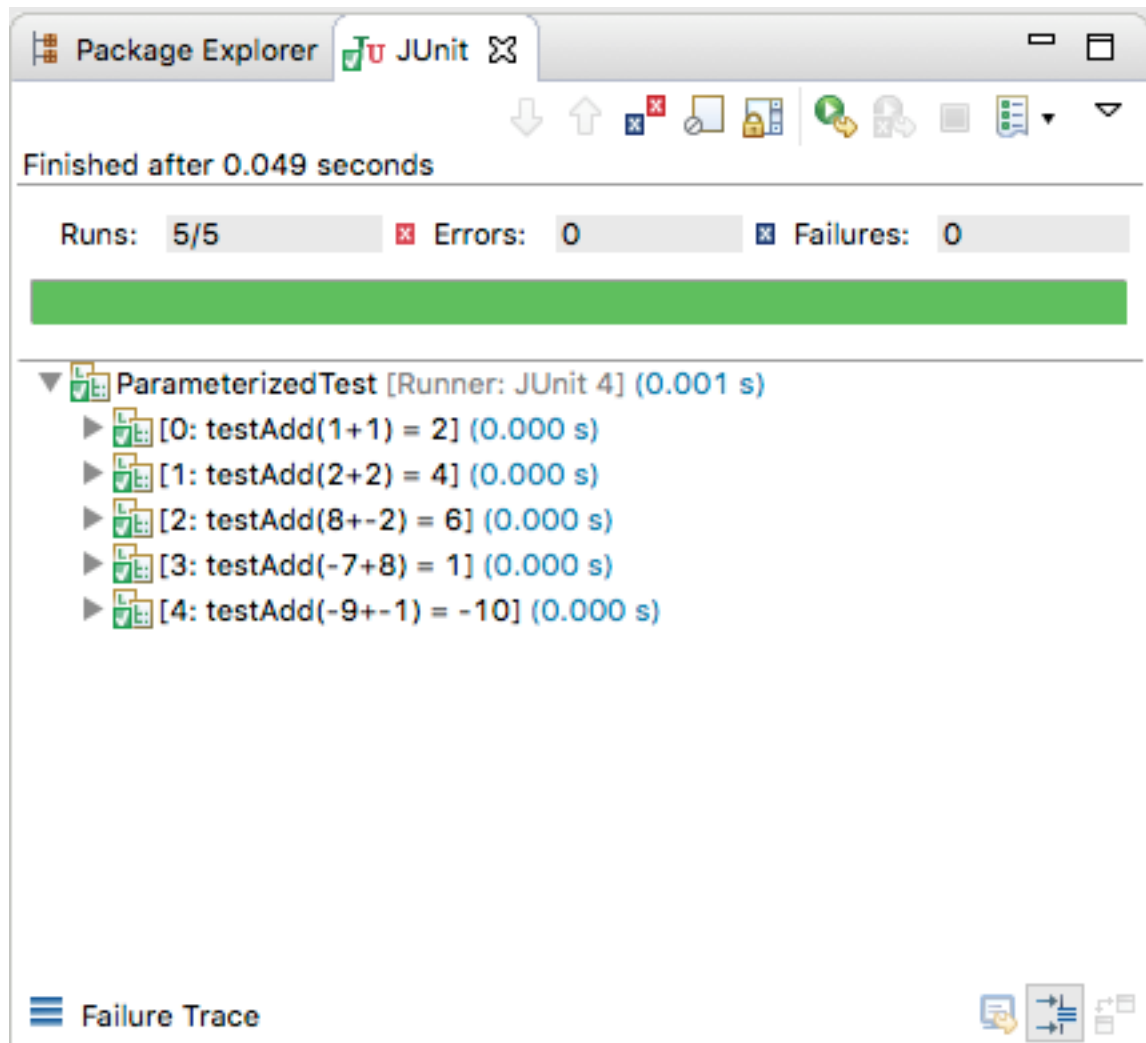
```

Lines từ 18-22: chúng ta có phương thức khởi tạo của class ParameterizedTest. Trong phương thức này chúng ta gán các giá trị tham số truyền vào (còn gọi là arguments) cho các thuộc tính của class ParameterizedTest (có 3 thuộc tính tương ứng với 2 số hạng và tổng của chúng).

Line 24: chúng ta sử dụng annotations @Parameters để định dạng dữ liệu của tham số, thuộc tính name của annotation này là không bắt buộc, ở đây chúng ta để name là một chuỗi biểu diễn chuỗi sẽ được in ra console khi từng đối tượng dữ liệu (lưu trong collection) được test.

### Step 5: Chạy test case review kết quả.

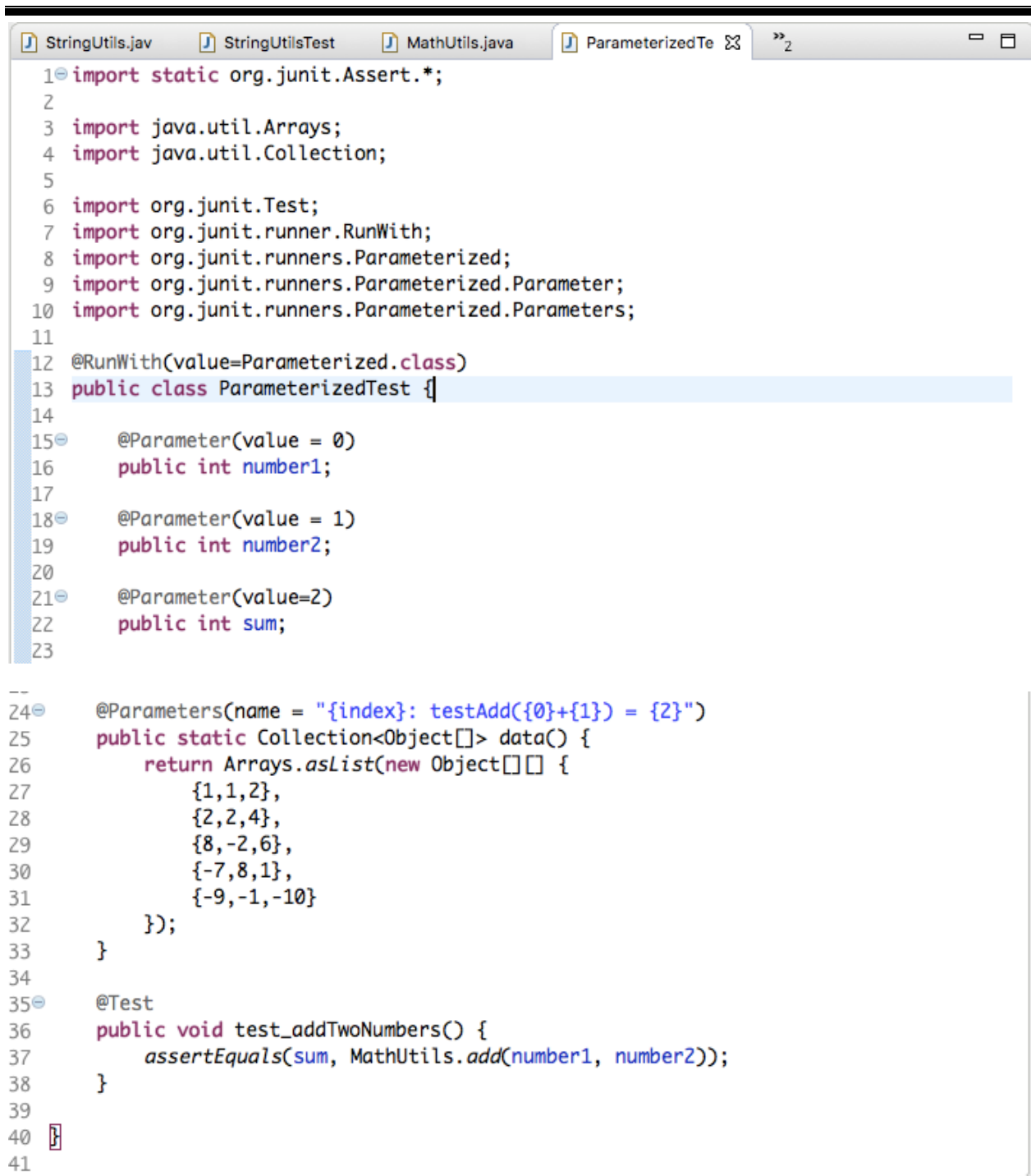
Chuột phải vào class ParameterizedTest ==> Chọn Run As ==> Chọn JUnit Test



Chúng ta sẽ quan sát thấy 5 test case với các đầu vào dữ liệu khác nhau được thực hiện.

### Step 6: Sửa lại class ParameterizedTest để thực hiện tham số hoá test case theo cách thứ 2 (Field Injection)

Chúng ta xoá đi phương thức khởi tạo của class này, thay vì thế sẽ sử dụng injection @Parameter của Junit cho từng thuộc tính của class. Chú ý rằng các thuộc tính này phải được chuyển từ private sang public.



```

1 import static org.junit.Assert.*;
2
3 import java.util.Arrays;
4 import java.util.Collection;
5
6 import org.junit.Test;
7 import org.junit.runner.RunWith;
8 import org.junit.runners.Parameterized;
9 import org.junit.runners.Parameterized.Parameter;
10 import org.junit.runners.Parameterized.Parameters;
11
12 @RunWith(value=Parameterized.class)
13 public class ParameterizedTest {
14
15     @Parameter(value = 0)
16     public int number1;
17
18     @Parameter(value = 1)
19     public int number2;
20
21     @Parameter(value=2)
22     public int sum;
23
24     @Parameters(name = "{index}: testAdd({0}+{1}) = {2}")
25     public static Collection<Object[]> data() {
26         return Arrays.asList(new Object[][] {
27             {1,1,2},
28             {2,2,4},
29             {8,-2,6},
30             {-7,8,1},
31             {-9,-1,-10}
32         });
33     }
34
35     @Test
36     public void test_addTwoNumbers() {
37         assertEquals(sum, MathUtils.add(number1, number2));
38     }
39
40
41

```

Chạy lại test case chúng ta thu được cùng kết quả như Step 5.

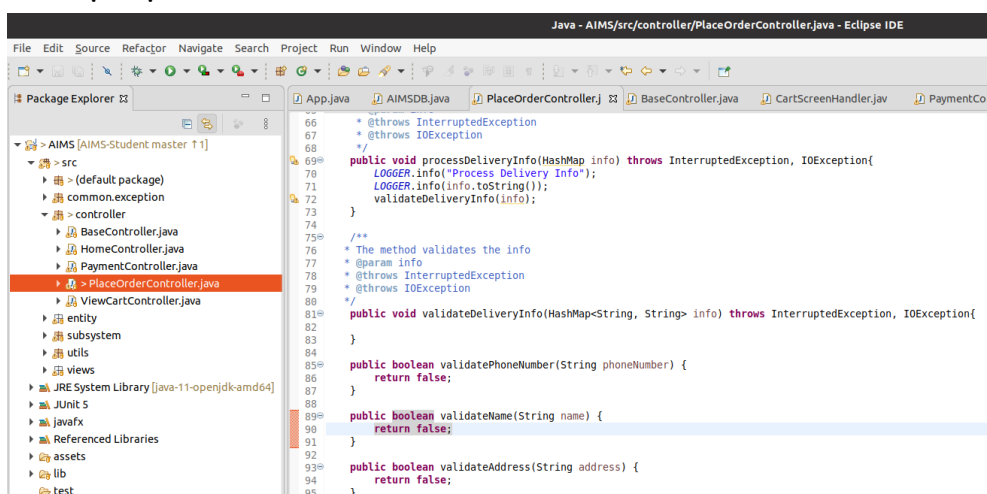
### 1.3.5. Thực hành thiết kế unit tests theo hướng TDD

- a) Clone project từ: <https://github.com/leminhnguyen/AIMS-Student>
- b) Đặc tả yêu cầu
  - Trong phần này chúng ta sẽ thực hành việc thiết kế các testcases cho phương thức *validateDeliveryInfo* nằm trong controller *PlaceOrder*

- Input đầu vào của phương thức *validateDeliveryInfo* sẽ là thông tin người dùng nhập vào như: *name, phone, address*
- Trong đó:

S	Tên tham số	Yêu cầu
1	name	chỉ bao gồm chữ cái, không chứa ký tự đặc biệt, không được phép null
2	phone	chỉ bao gồm chữ số, độ dài 10 ký tự và bắt đầu là số 0
3	address	không được phép null, không chứa ký tự đặc biệt

- Công việc chúng ta cần phải thực hiện là thiết kế unit tests dựa trên đặc tả và xây dựng phương thức *validateDeliveryInfo* theo quá trình TDD
- Chúng ta có thể tách nhỏ *validateDeliveryInfo* thành 3 phương thức nhỏ hơn là *validateAdress*, *validateName* và *validatePhoneNumber* (ban đầu các phương thức này đều empty bởi vì chúng ta cần xây dựng testcase trước khi bắt đầu implement thực sự



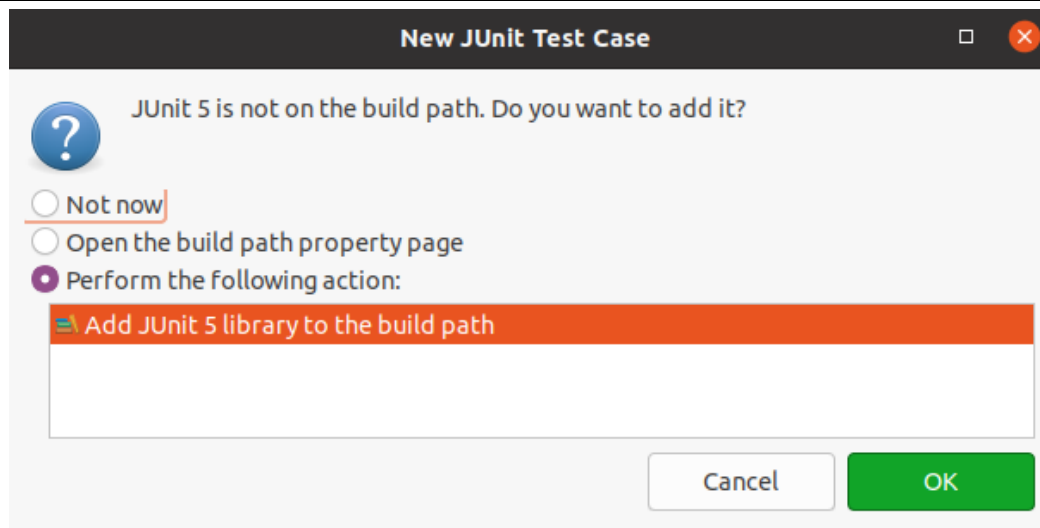
### c) Tạo UnitTest bằng Eclipse

- Tạo một package tên controller ở trong folder test, sau này khi tiến hành tạo các testcase cho các class khác thì chúng ta nên tuân theo cấu trúc của class đó ở trong src folder
- Tiếp theo chúng ta sẽ tiến hành xây dựng các class Test cho các phương thức này bằng Junit5. Click chuột phải vào Project -> New -> Junit Testcase. Sau đó điền các thông tin cần thiết như:
  - + **Source folder:** là nơi các class Test được tạo ra, chúng ta sẽ đặt các testcase ở trong folder test
  - + **Package:** là package ở trong folder test mà chúng ta muốn đặt các test case

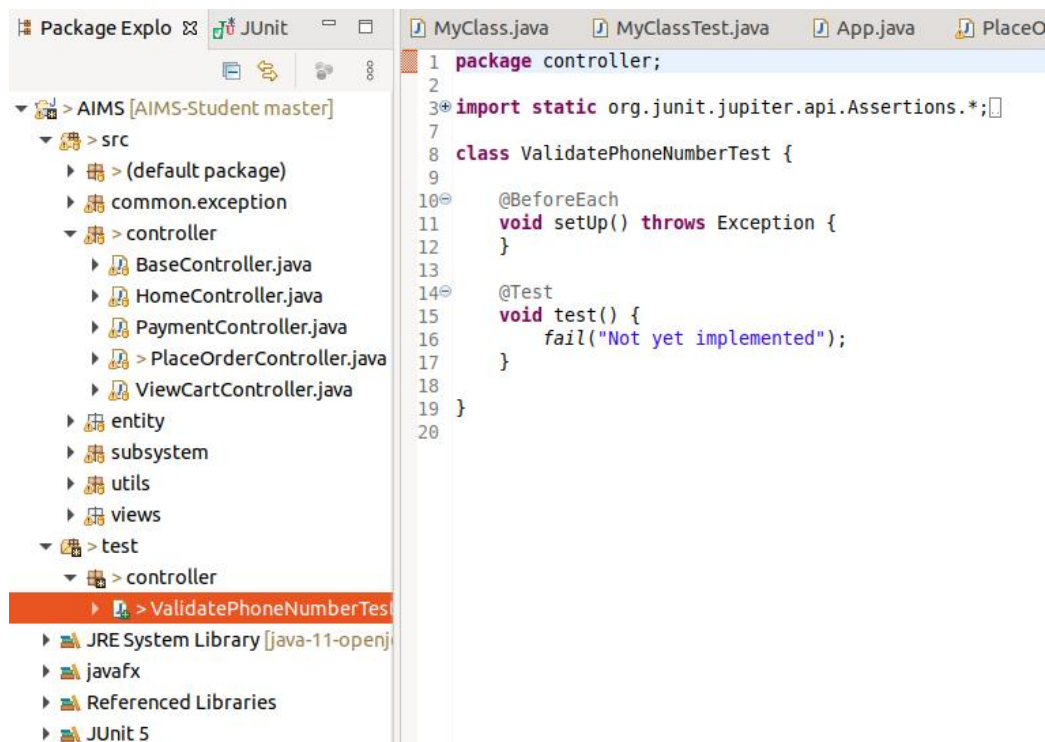
- + **Name:** là tên của class Test
- + **Class Under Test:** là tên class mà chúng ta đang cần test

The screenshot shows the 'New JUnit Test Case' dialog box. At the top, it says 'JUnit Test Case' and provides instructions: 'Select the name of the new JUnit test case. You have the options to specify the class under test and on the next page, to select methods to be tested.' Below this, there are three radio buttons: 'New JUnit 3 test', 'New JUnit 4 test', and 'New JUnit Jupiter test' (which is selected). The 'Source folder:' field is set to 'AIMS/test' with a 'Browse...' button. The 'Package:' field is set to 'controller' with a 'Browse...' button. The 'Name:' field is set to 'ValidatePhoneNumberTest'. The 'Superclass:' field is set to 'java.lang.Object' with a 'Browse...' button. Under 'Which method stubs would you like to create?', there are four checkboxes: 'setUpBeforeClass()' (unchecked), 'tearDownAfterClass()' (unchecked), 'setUp()' (checked), and 'tearDown()' (unchecked). There is also an unchecked checkbox for 'constructor'. Below this, it asks 'Do you want to add comments? (Configure templates and default value [here](#))' with an unchecked checkbox for 'Generate comments'. The 'Class under test:' field is set to 'controller.PlaceOrderController' with a 'Browse...' button. At the bottom, there are four buttons: a help icon (?), '< Back', 'Next >', 'Cancel', and a green 'Finish' button.

- Sau đó click Finish, nếu như project chưa có Junit thì Eclipse sẽ hiện lên popup yêu cầu add Junit -> Click ok



- Sau khi hoàn thiện xong bước ở trên ta sẽ có kết quả như sau



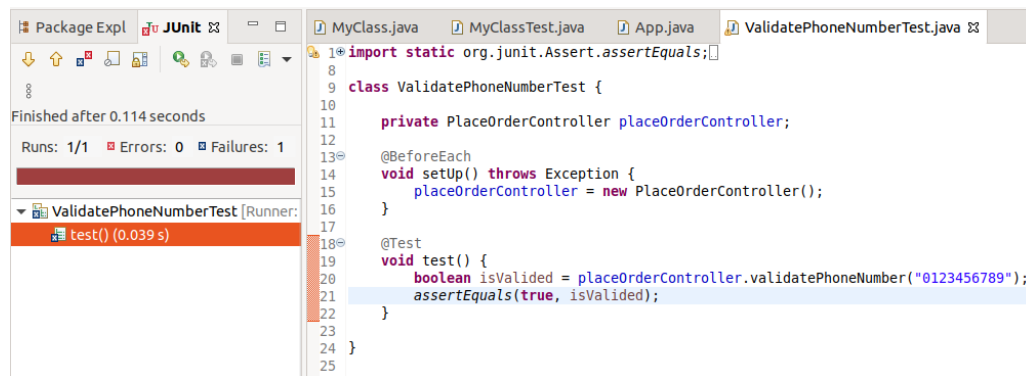
- Đến đây chúng ta đã học được cách tạo thành công testcase cho một class, và nếu như bấm run thì tất nhiên là kết quả sẽ failed bởi vì chúng ta chưa implement bất kỳ phần code cần test nào

#### d) Thực hành xây dựng các phương thức theo TDD

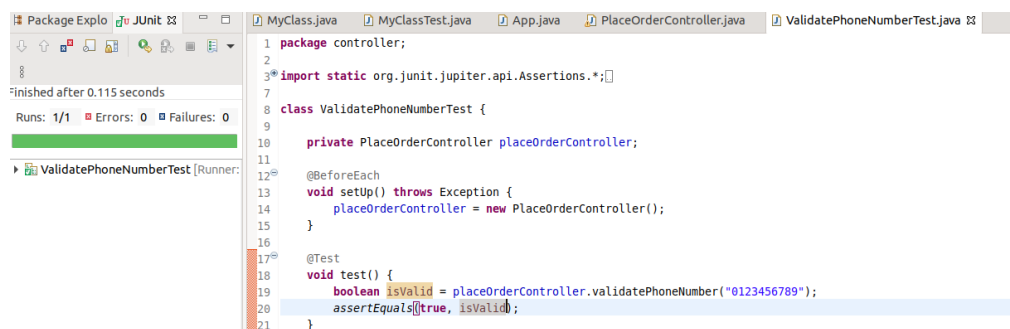
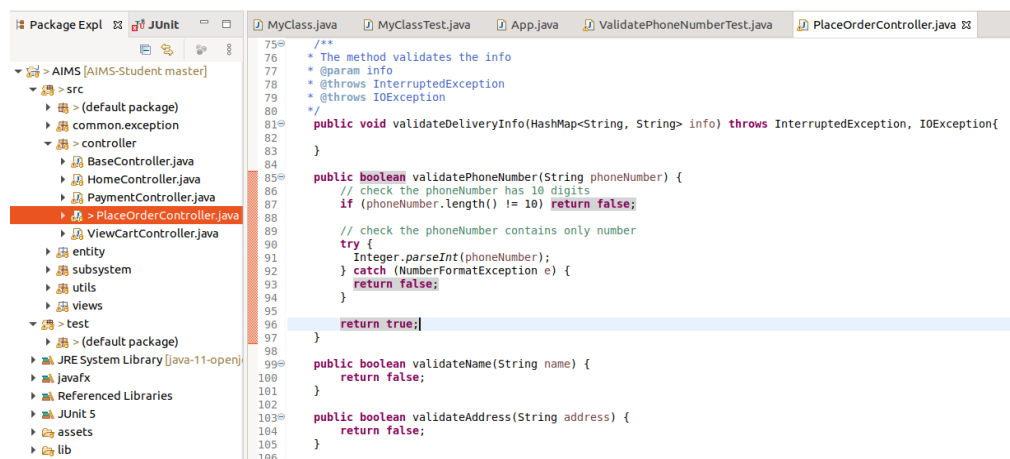
- Bây giờ chúng ta sẽ bắt tay vào quá trình TDD, nếu như bây giờ chúng ta khởi tạo một đối tượng của PlaceOrderController ở trong class ValidatePhoneNumber và test phương thức validatePhoneNumber với một số điện thoại hợp lệ (VD:



0123456789) thì chúng ta sẽ bị failed, lý do là code ban đầu của chúng ta đang mặc định return false)

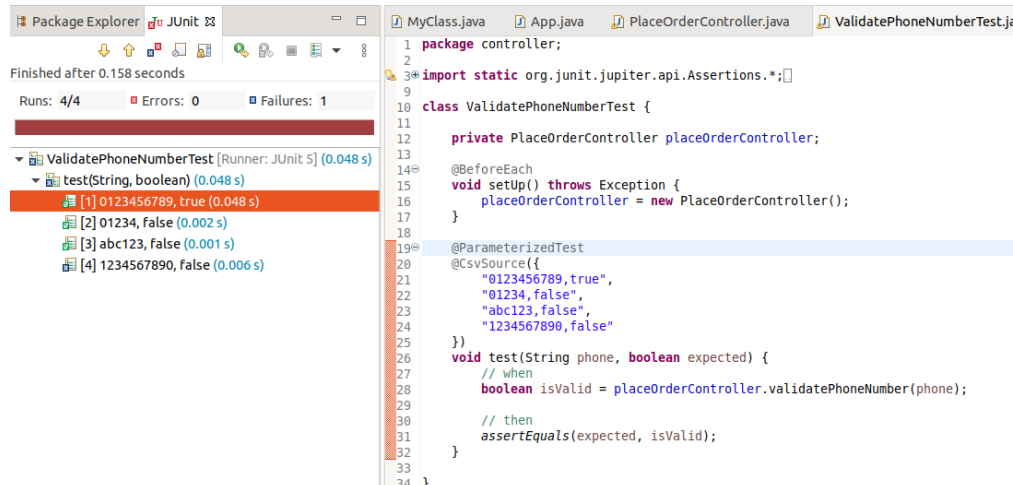


- Vì vậy chúng ta sẽ quay lại class PlaceOrder và implement phương thức validatePhoneNumber. Số điện thoại chỉ được bao gồm 10 ký tự số vậy nên ta sẽ implement, và nếu chúng ta quay lại để chạy lại testcase thì chúng ta sẽ pass

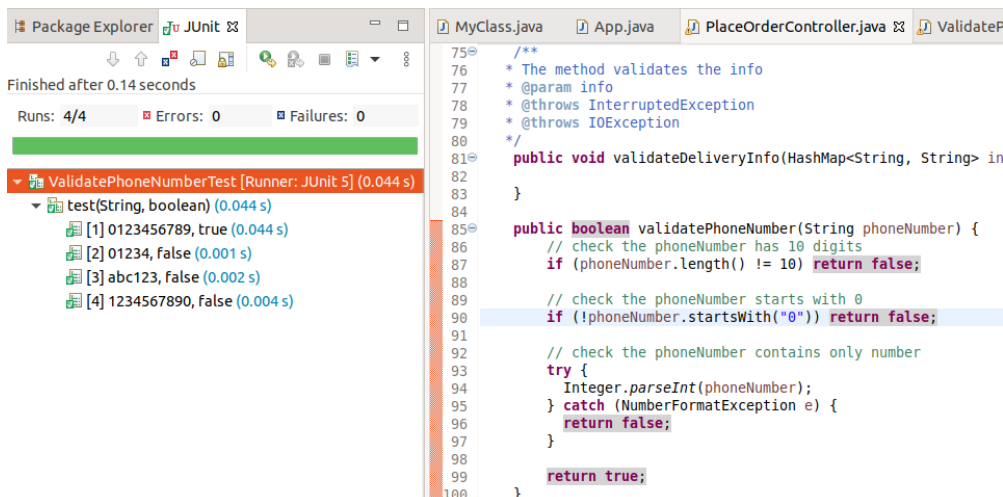


- Chúng ta có thể tối ưu dữ liệu test bằng cách đưa vào một list các cặp dữ liệu - kết quả mong đợi (input - expected outcome). Chúng ta có thể làm được điều này bằng

## cách sử dụng annotation @Parameterized và @CsvSource

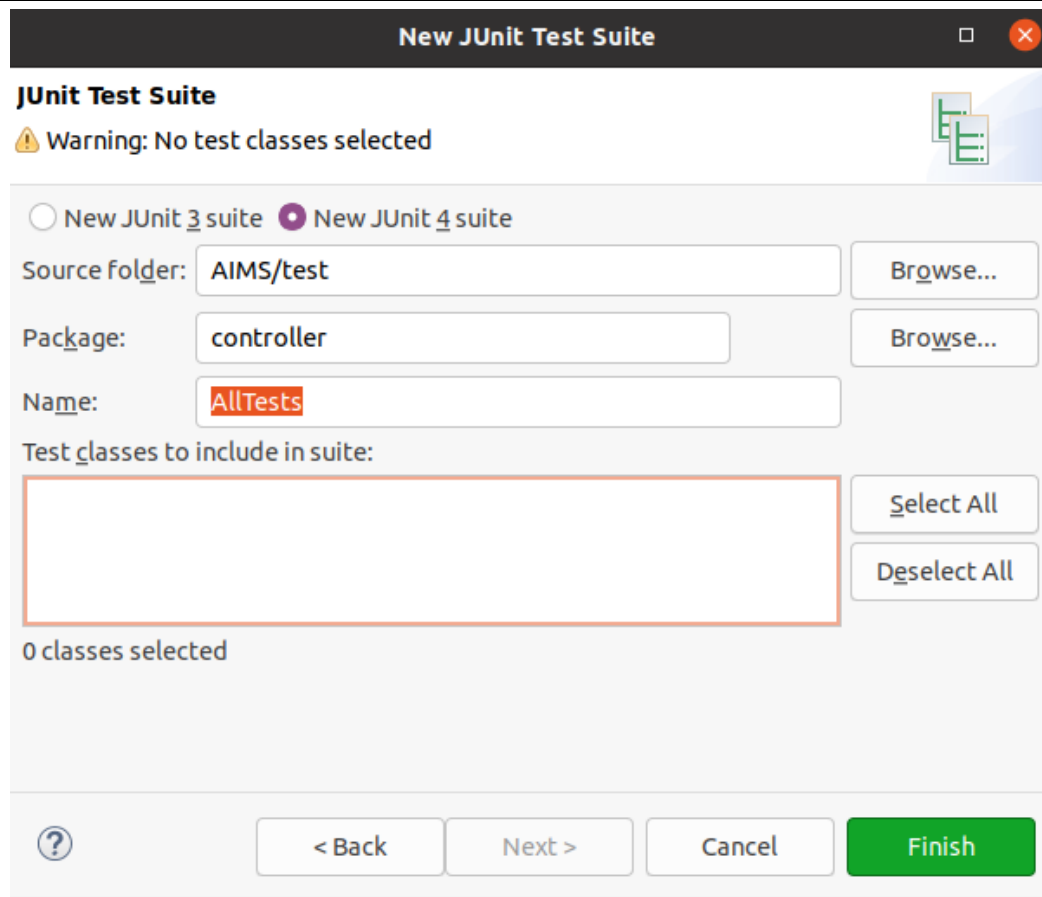


- Từ kết quả trên ta có thể thấy chúng ta bị failed một testcase "1234567890,false", mong muốn của chúng ta này là phương thức validatePhoneNumber sẽ trả về false do số điện thoại này không bắt đầu bởi 0, tuy nhiên trong trường hợp này chúng ta đang bị sai và phải quay lại phương thức validatePhoneNumber để điều chỉnh lại. Sau đó ta sẽ có kết quả là cả 4 test case đều pass

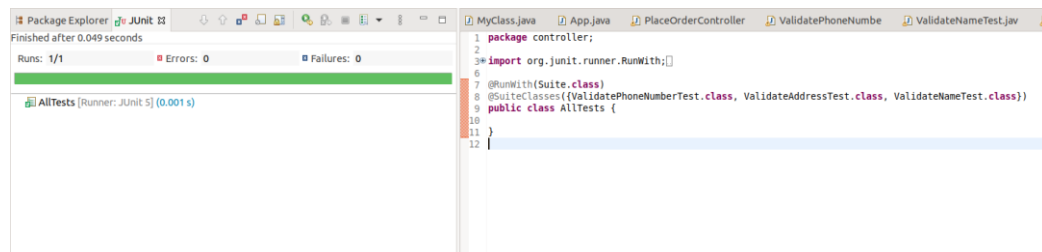


### e) Thực hành tạo Test Suite

- Người học tiếp tục thực hành làm cho 2 phương thức còn lại là *validateAddress* và *validateName* dựa vào bảng đặc tả input đầu vào
- Sau khi hoàn thiện chúng ta sẽ có 3 class Test, và chúng ta có thể tiến hành tạo Test Suite. Test Suite là một tập các testcases có liên quan đến một nghiệp vụ nào đó. Click chuột phải vào project -> New -> Test Suite



- Sau đó chúng ta thêm các class cần test vào class AllTests, sau đó bấm Run và xem kết quả



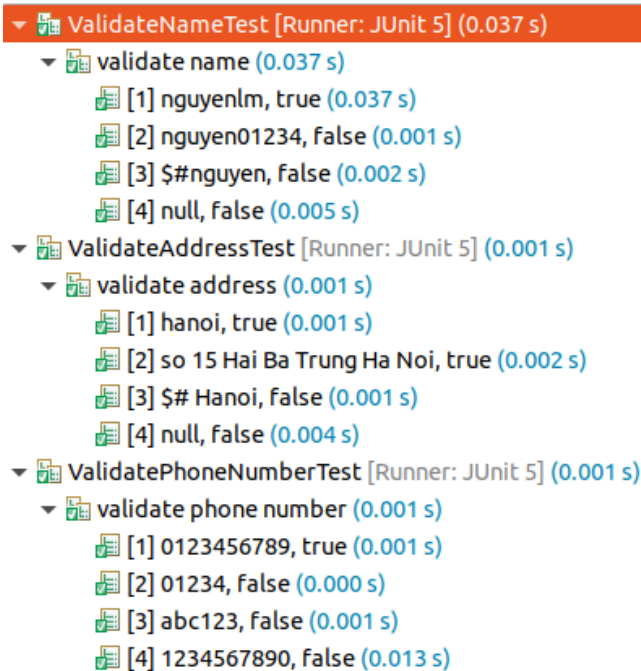


















- Nếu muốn nhìn kết quả một cách trực quan hơn, thì thay vì tạo test suite chúng ta có thể click vào trong folder test/controller sau đó bấm Run thì Eclipse sẽ chạy hết tất cả các testcase nằm ở bên trong và show kết quả của từng TestCase một cách rất trực quan

Finished after 0.174 seconds

Runs: 13/13

Errors: 0

Failures: 0

- 
- The screenshot displays the JUnit test results in an IDE. At the top, a status bar indicates 'Finished after 0.174 seconds', 'Runs: 13/13', 'Errors: 0', and 'Failures: 0'. Below this, a green progress bar is shown. The main area lists the test classes and their methods, each with a green icon indicating success and the execution time in parentheses. The classes are 'ValidateNameTest', 'ValidateAddressTest', and 'ValidatePhoneNumberTest', each with a 'validate' method. The 'ValidateNameTest' method has four test cases: '[1] nguyenlm, true (0.037 s)', '[2] nguyen01234, false (0.001 s)', '[3] \$#nguyen, false (0.002 s)', and '[4] null, false (0.005 s)'. The 'ValidateAddressTest' method has four test cases: '[1] hanoi, true (0.001 s)', '[2] so 15 Hai Ba Trung Ha Noi, true (0.002 s)', '[3] \$# Hanoi, false (0.001 s)', and '[4] null, false (0.004 s)'. The 'ValidatePhoneNumberTest' method has four test cases: '[1] 0123456789, true (0.001 s)', '[2] 01234, false (0.000 s)', '[3] abc123, false (0.001 s)', and '[4] 1234567890, false (0.013 s)'.
- ▼  ValidateNameTest [Runner: JUnit 5] (0.037 s)
    - ▼  validate name (0.037 s)
      -  [1] nguyenlm, true (0.037 s)
      -  [2] nguyen01234, false (0.001 s)
      -  [3] \$#nguyen, false (0.002 s)
      -  [4] null, false (0.005 s)
    - ▼  ValidateAddressTest [Runner: JUnit 5] (0.001 s)
      - ▼  validate address (0.001 s)
        -  [1] hanoi, true (0.001 s)
        -  [2] so 15 Hai Ba Trung Ha Noi, true (0.002 s)
        -  [3] \$# Hanoi, false (0.001 s)
        -  [4] null, false (0.004 s)
      - ▼  ValidatePhoneNumberTest [Runner: JUnit 5] (0.001 s)
        - ▼  validate phone number (0.001 s)
          -  [1] 0123456789, true (0.001 s)
          -  [2] 01234, false (0.000 s)
          -  [3] abc123, false (0.001 s)
          -  [4] 1234567890, false (0.013 s)

## Bài tập nhóm:

Xây dựng các test case cho bài tập lớn môn học

- Các bước thực hiện:
  - 1. Thực hiện kiểm thử đơn vị (unit test) cho phần mã nguồn mỗi thành viên thực hiện
  - 2. Thực hiện kiểm thử các chức năng sau khi đã hoàn thiện
  - 3. Xây dựng tài liệu kiểm thử
- Yêu cầu nộp bài:
  - Tổng hợp các kết quả kiểm thử (project unit test, tài liệu kiểm thử,...)
  - Nộp bài vào thư mục 05-UnitTest trên thư mục Google Drive mà thầy đã tạo

# HẾT