

Bài tập tuần 11

Thực hành kiểm thử đơn vị với JUnit

Mục tiêu

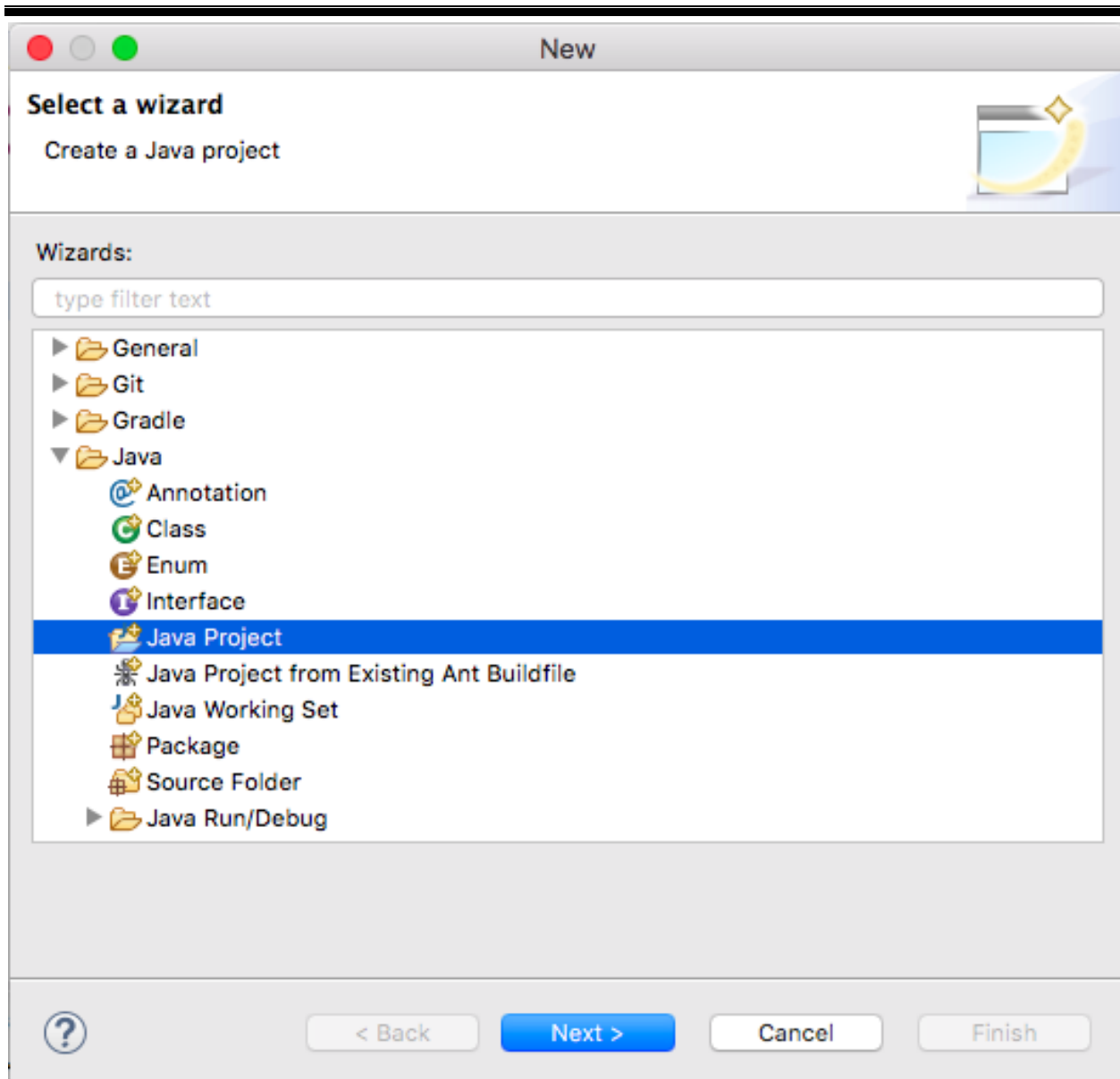
- Cài đặt Eclipse có tích hợp sẵn JUnit
- Thực hành với JUnit, tạo một test case đơn giản cho một đoạn chương trình

Phần I Bài tập step by step

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



JUnit Test

The screenshot shows the 'New Java Project' dialog box in Eclipse. The title bar says 'New Java Project'. Below the title bar, there's a section 'Create a Java Project' with a subtext 'Create a Java project in the workspace or in an external location.' and a folder icon. The 'Project name' field is filled with 'TestJUnit'. The 'Use default location' checkbox is checked. The 'Location' field shows the path '/Users/ThanDieu/eclipse-workspace/TestJUnit' with a 'Browse...' button. The 'JRE' section has three radio buttons: 'Use an execution environment JRE:' (selected), 'Use a project specific JRE:', and 'Use default JRE (currently 'Java SE 8 [1.8.0_51]')'. The first option has a dropdown menu showing 'JavaSE-1.8'. The second option has a dropdown menu showing 'Java SE 8 [1.8.0_51]'. There is a 'Configure JREs...' link. The 'Project layout' section has two radio buttons: 'Use project folder as root for sources and class files' and 'Create separate folders for sources and class files' (selected). There is a 'Configure default...' link. The 'Working sets' section has a checkbox 'Add project to working sets' and a 'New...' button. Below it, the 'Working sets:' field is empty with a dropdown arrow, and there is a 'Select...' button. At the bottom, there are buttons for '< Back', 'Next >', 'Cancel', and 'Finish'.

New Java Project

Create a Java Project

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

Project name: TestJUnit

☒ Use default location

Location: /Users/ThanDieu/eclipse-workspace/TestJUnit [Browse...](#)

JRE

☒ Use an execution environment JRE: JavaSE-1.8

☐ Use a project specific JRE: Java SE 8 [1.8.0_51]

☐ 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

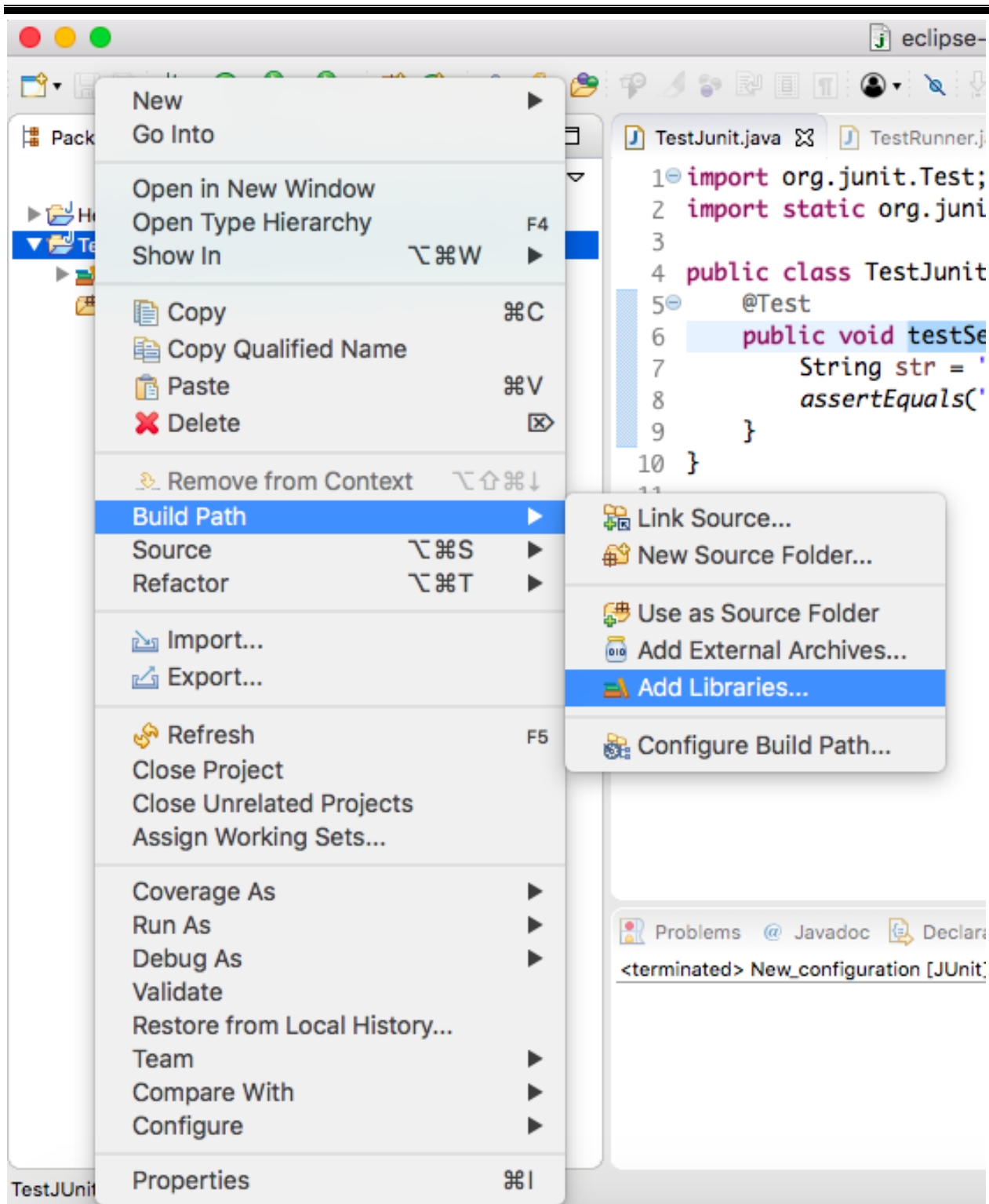
☐ Add project to working sets [New...](#)

Working sets: [Select...](#)

[? < Back Next > Cancel Finish](#)

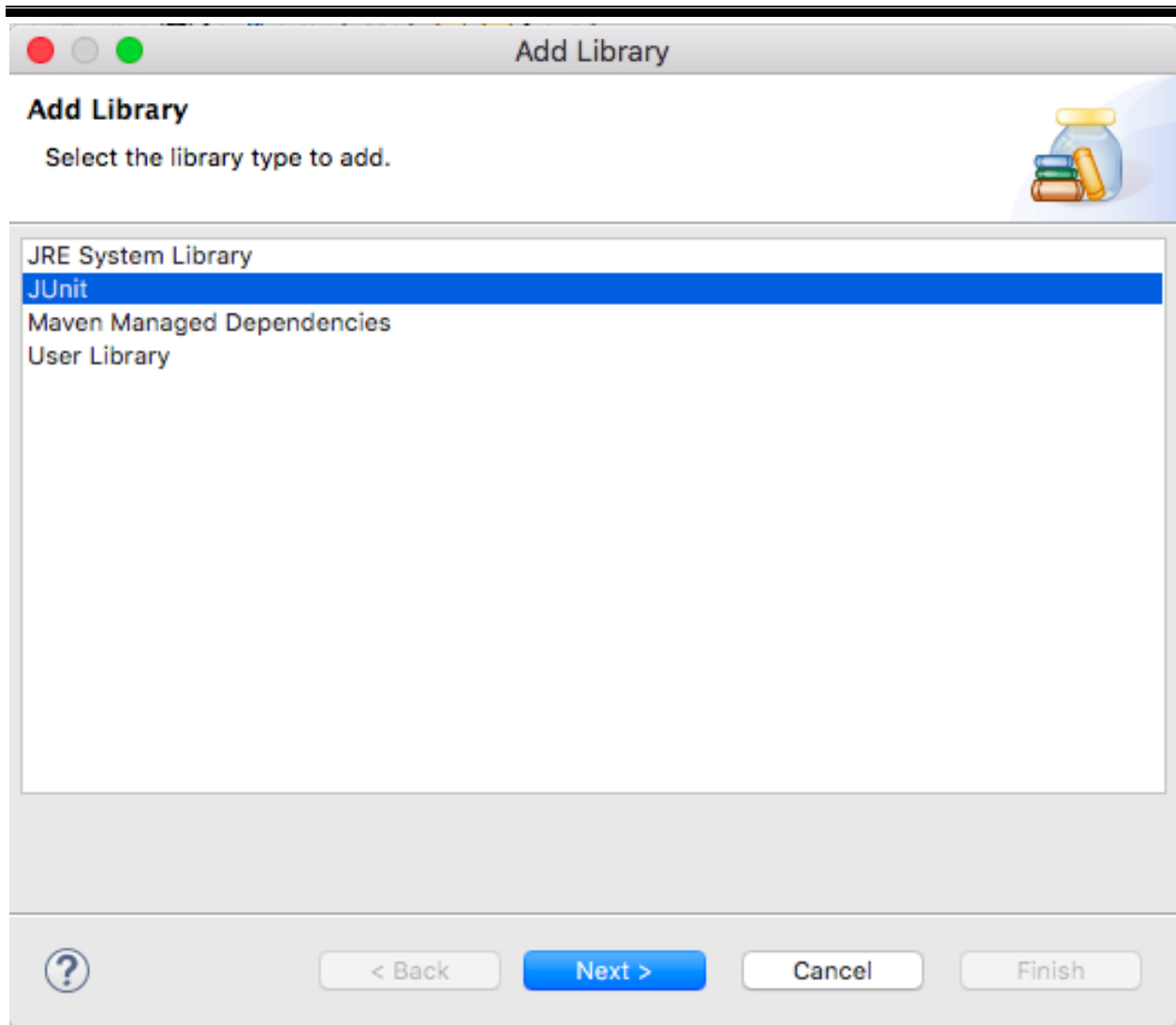
STEP 2: Thêm thư viện Junit cho projec

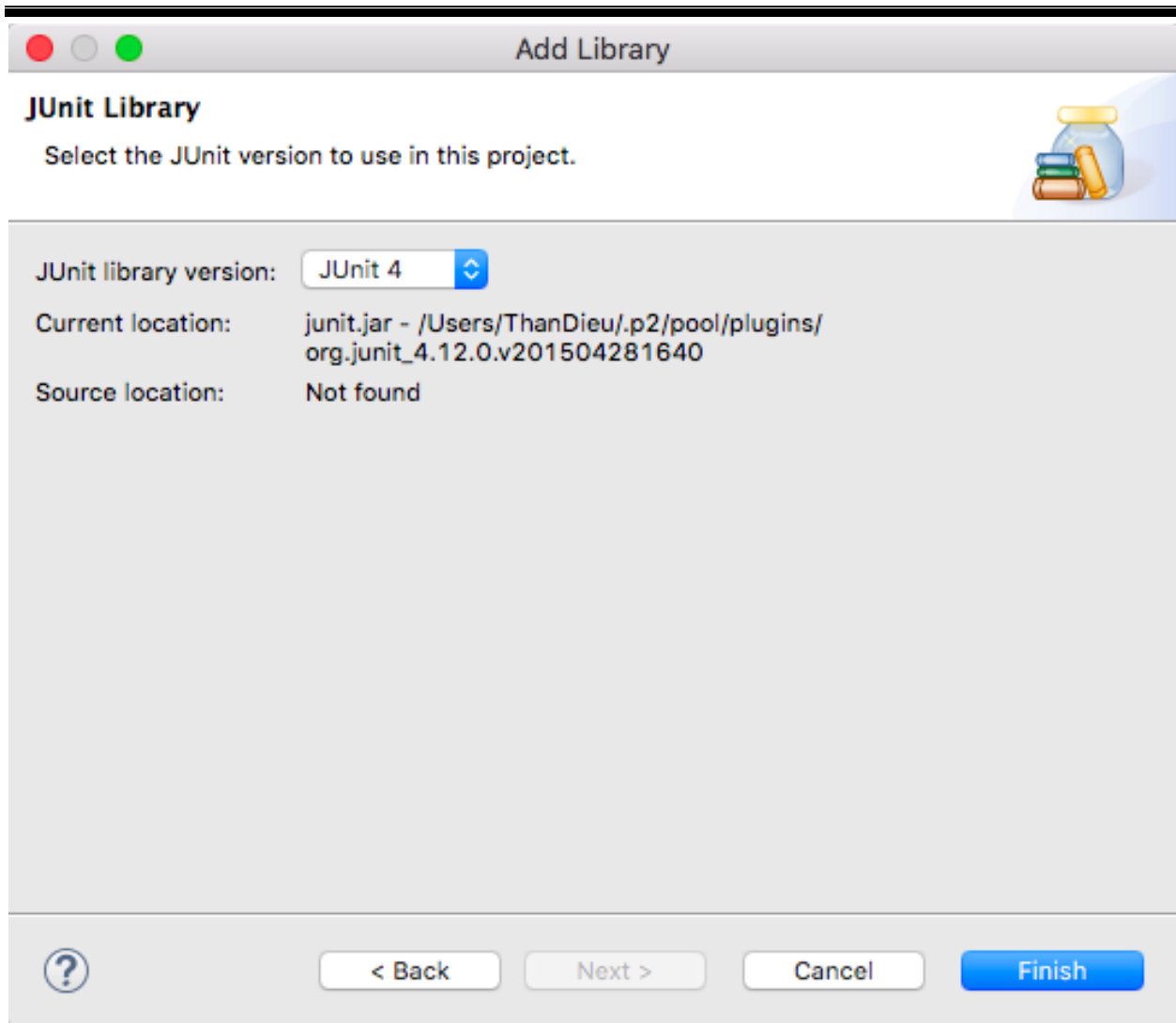
Chuột phải vào project TestJUnit ==> Build Path ==> Add Library



Chọn Junit sau đó chọn Junit 4 rồi Finish

JUnit Test





STEP 3: Tạo mới class java tên TestJUnit

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

JUnit Test

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

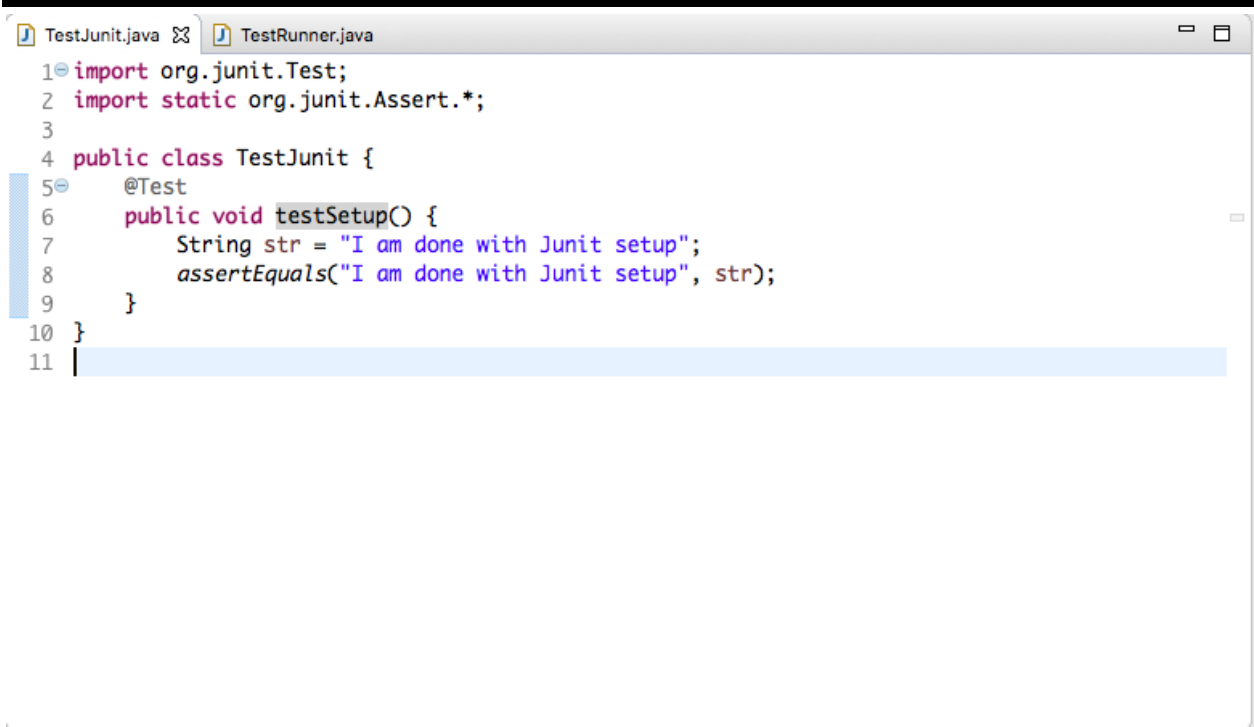
☒ Inherited abstract methods

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

☐ Generate comments

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.

JUnit Test

here)' with an unchecked 'Generate comments' checkbox. At the bottom are a help icon, 'Cancel', and 'Finish' buttons."/>

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)

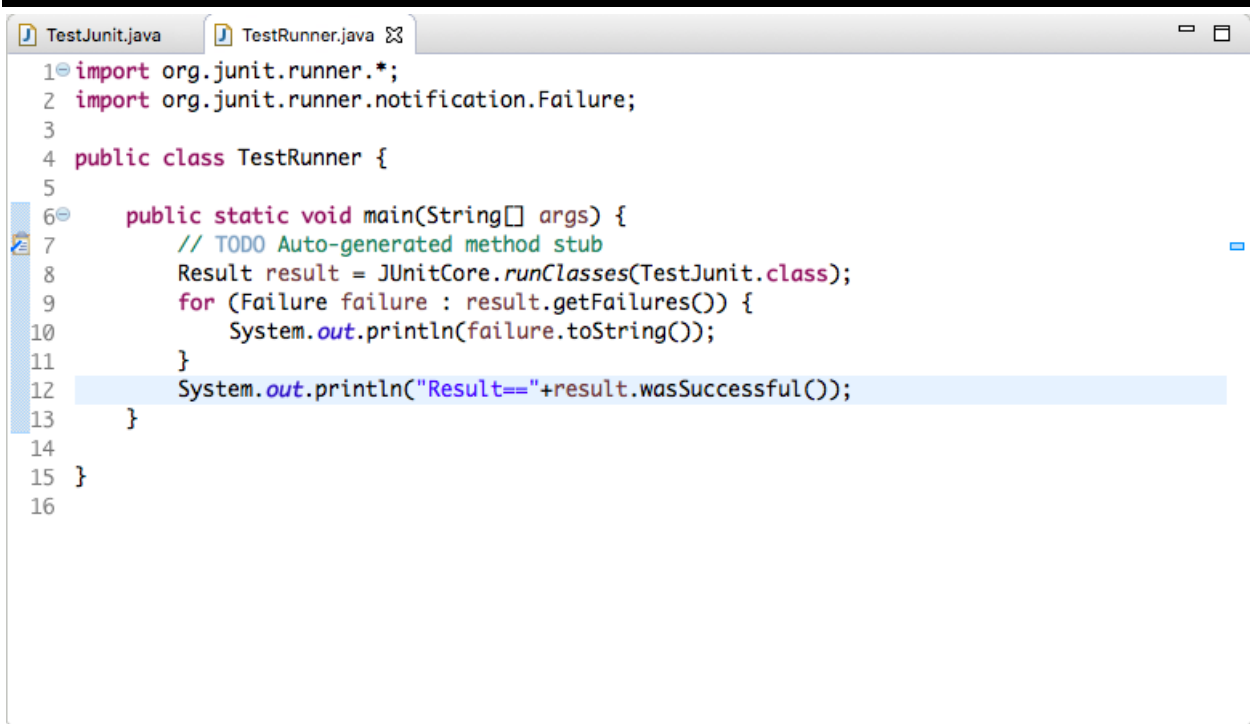
☐ Constructors from superclass

☒ 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
```

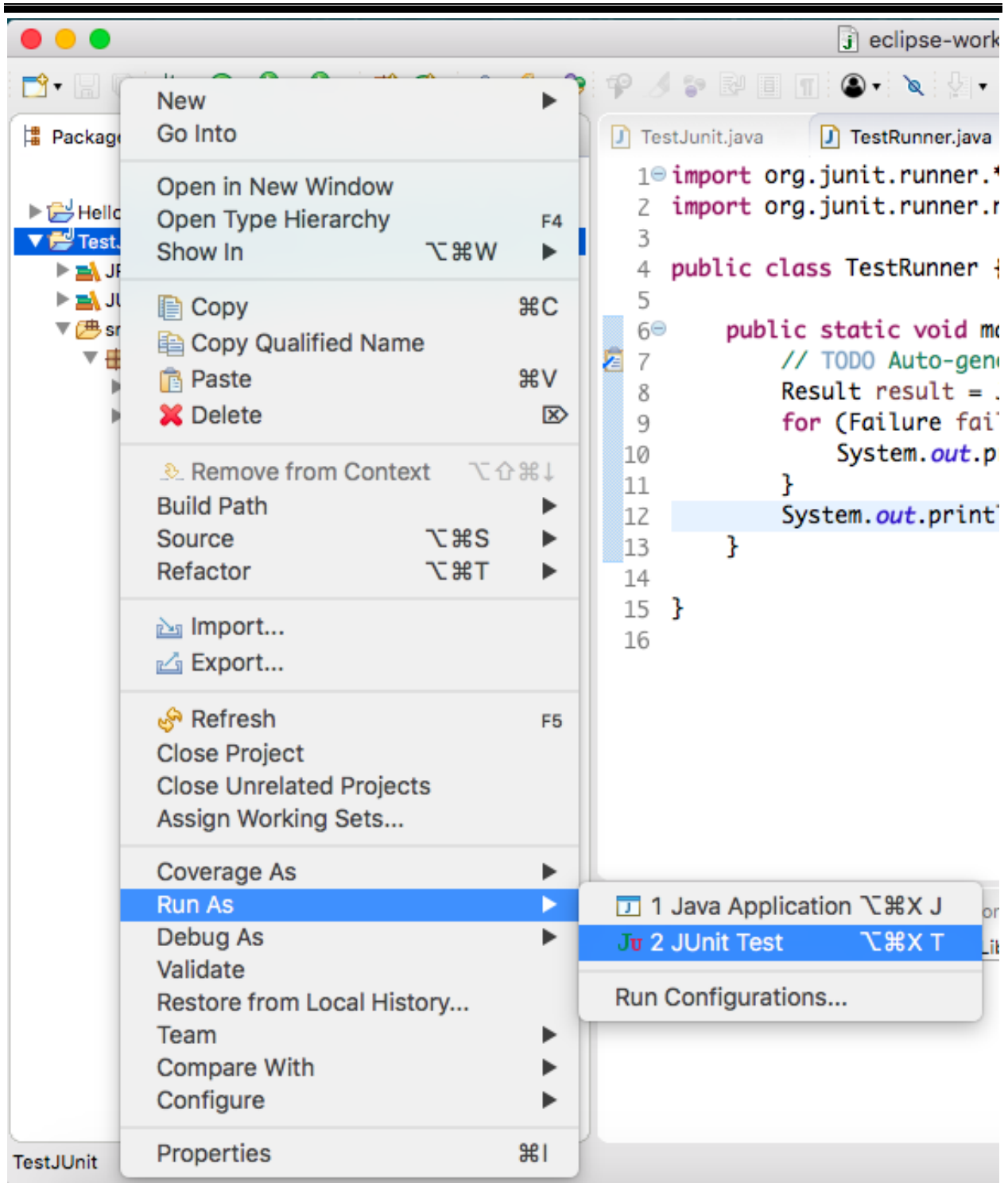
Chúng ta sử dụng object Result để chứa tất cả các unit test có trong test case TestJUnit.

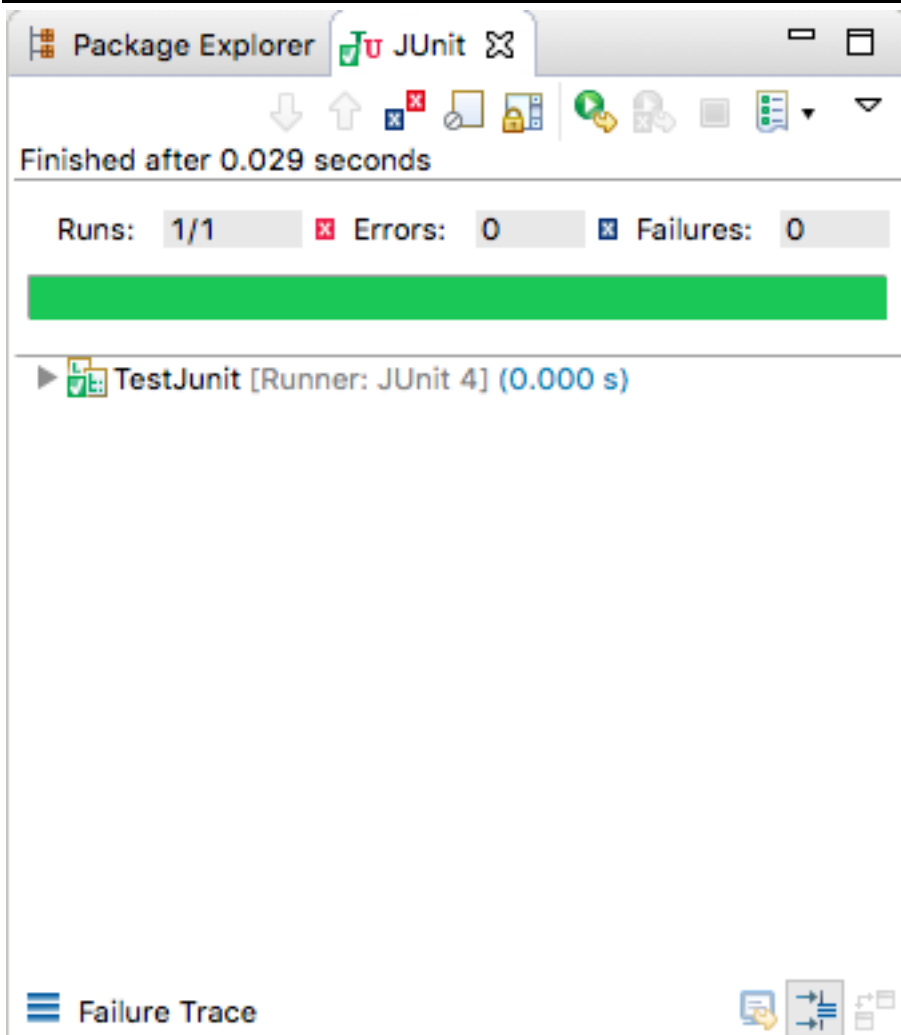
Những test bị fail sẽ được lấy về từ hàm getFailures của class Result.

Step 6: Chạy chương trình review kết quả

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

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

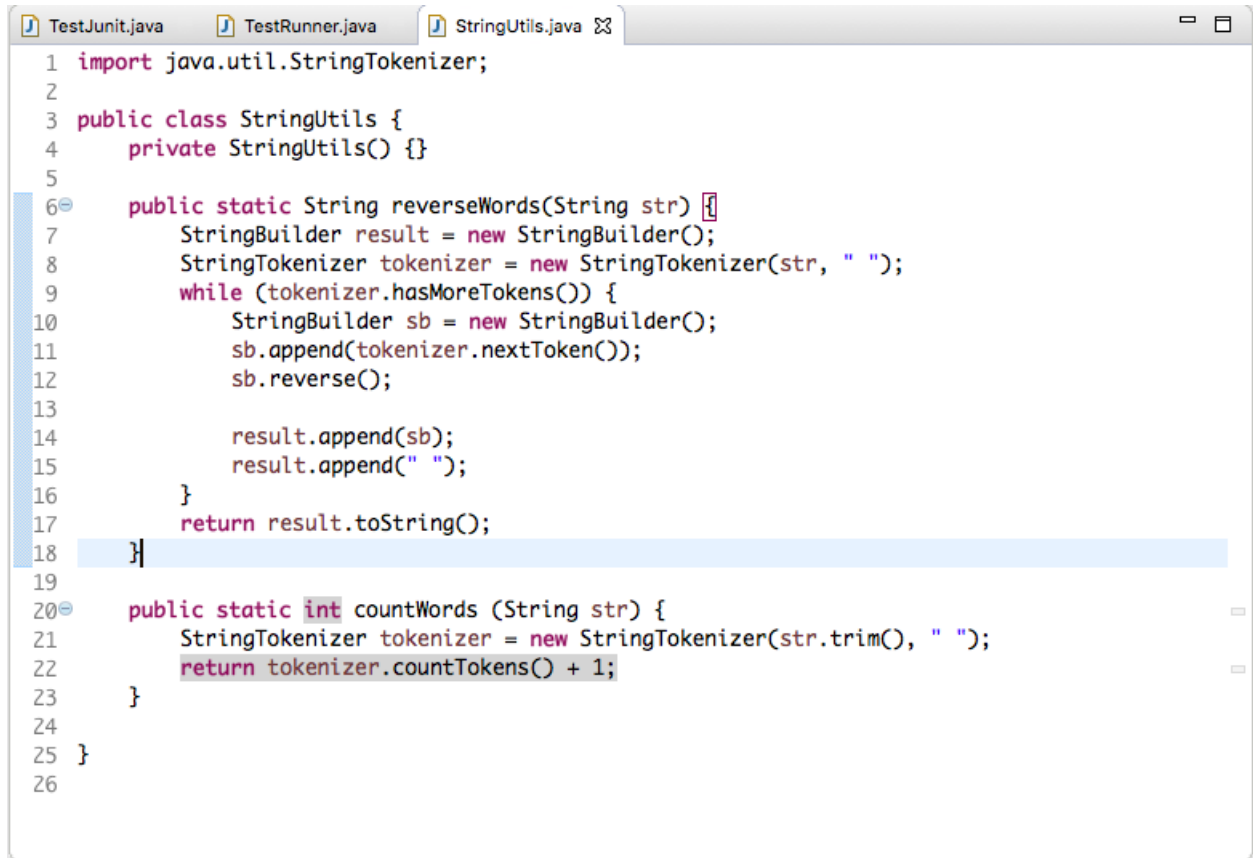
Step 1: Tạo mới project tên Utils

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

JUnit Test

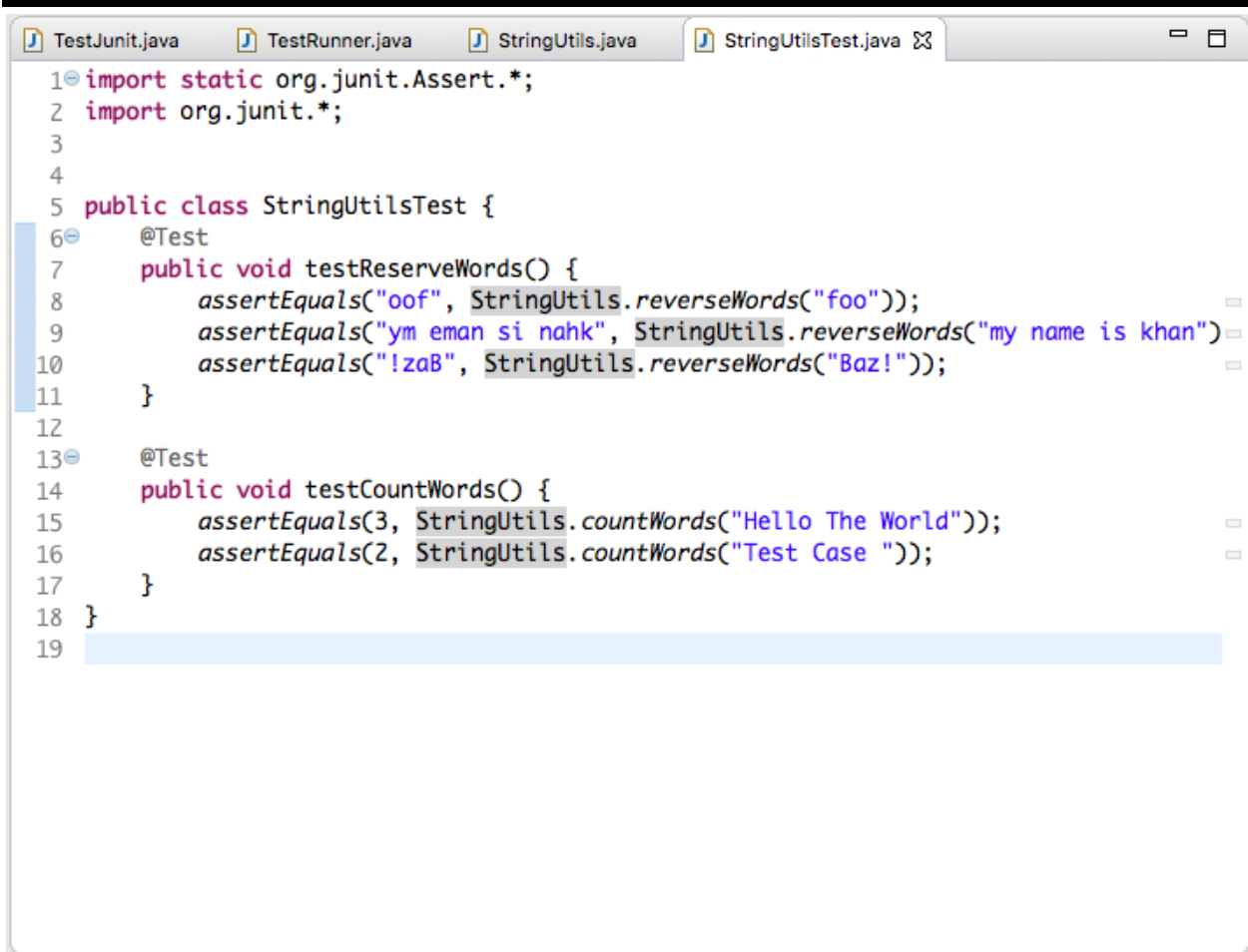
Trong class StringUtils, chúng ta tạo ra 2 phương thức: reverseWords(String str) để đảo ngược word và countWords(String str) để đếm số từ trong chuỗi. Giả thiết rằng chuỗi nhập vào các từ cách nhau bằng 1 dấu cách.



```
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.

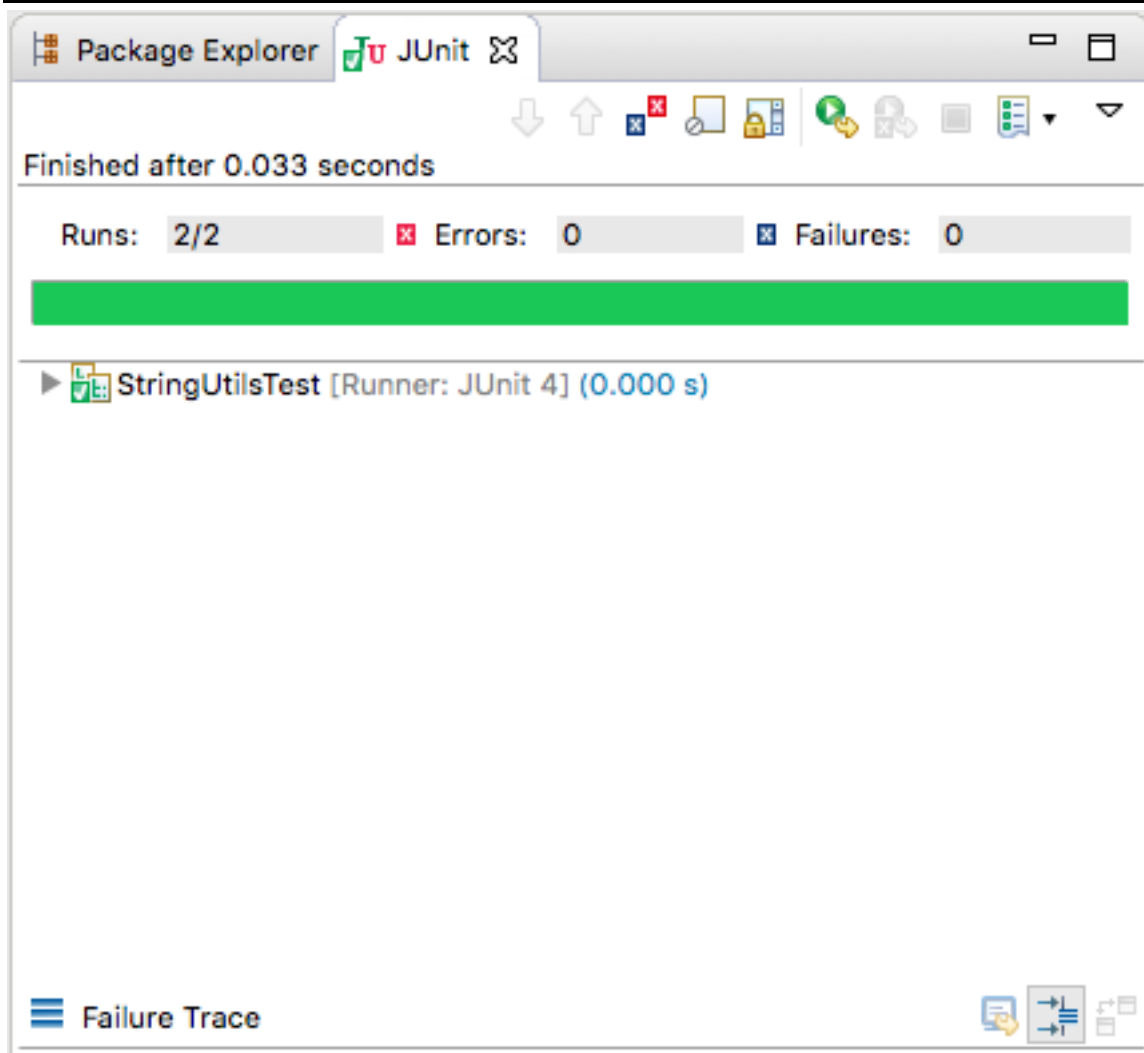


```
1 import static org.junit.Assert.*;
2 import org.junit.*;
3
4
5 public class StringUtilsTest {
6     @Test
7     public void testReverseWords() {
8         assertEquals("oof", StringUtils.reverseWords("foo"));
9         assertEquals("ym eman si nahk", StringUtils.reverseWords("my name is khan"));
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

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.

JUnit Test

```
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
```

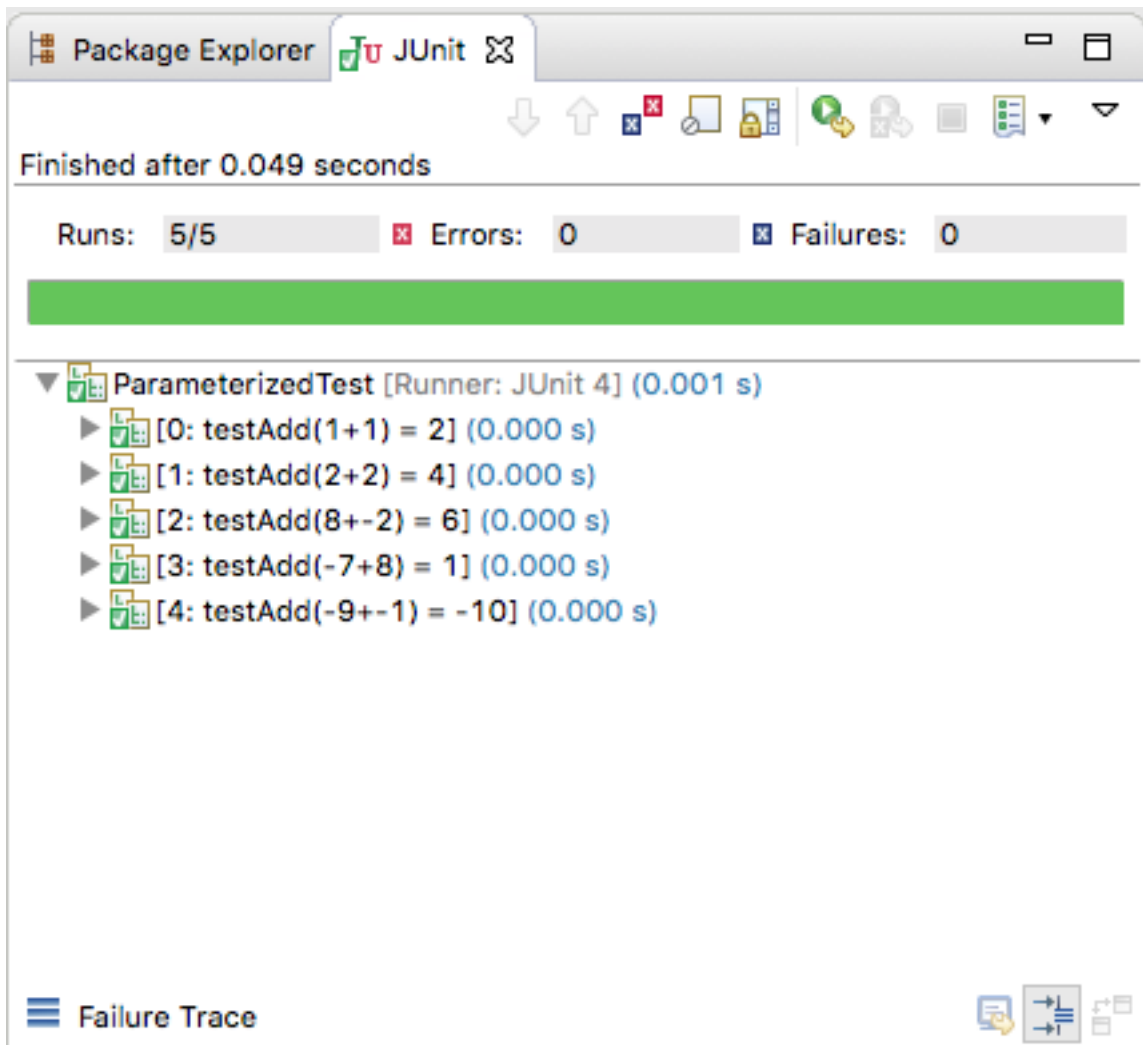
Lines từ 18-22: chúng ta có phương thức khởi tạo của class ParameterizedTest. Trong phương thức này chúng ta gán các giá trị tham số truyền vào (còn gọi là arguments) cho các thuộc tính của class ParameterizedTest (có 3 thuộc tính tương ứng với 2 số hạng và tổng của chúng).

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.

JUnit Test

```
StringUtils.java  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.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.

Phần bài tập tự thực hành:

Thay vì sử dụng traditional style như trên, các bạn hãy sử dụng modern style của junit (sử dụng `assertThat` và hàm `is()`).

HẾT