SD2 - Worksheet 1 - 6%

GRIFFITH COLLEGE DUBLIN

Student name:	**Carlos David Urra Cabello**					
Student number:	**3125350**					
Faculty:	**Computing Science**					
Course:	**BSCH/BSCO/EXCH**			**Stage/year:**	**2**	
Subject:	**Software Development 2**					
Study Mode:	Full time	☑		Part-time		
Lecturer Name:	**Gemma Deery**					
Assignment Title:	**Worksheet 1**					
Date due:	**19/02/2025**					
Date submitted:	**19/02/2025**					

Plagiarism disclaimer:

I understand that plagiarism is a serious offence and have read and understood the college policy on plagiarism. I also understand that I may receive a mark of zero if I have not identified and properly attributed sources which have been used, referred to, or have in any way influenced the preparation of this assignment, or if I have knowingly allowed others to plagiarise my work in this way.

I hereby certify that this assignment is my own work, based on my personal study and/or research, and that I have acknowledged all material and sources used in its preparation. I also certify that the assignment has not previously been submitted for assessment and that I have not copied in part or whole or otherwise plagiarised the work of anyone else, including other students.

Signed: Carlos Urra **Date:** 19/02/2025

Please note: **Students** MUST **retain a hard / soft copy of** ALL assignments as well as a receipt issued and signed by a member of Faculty as proof of submission.

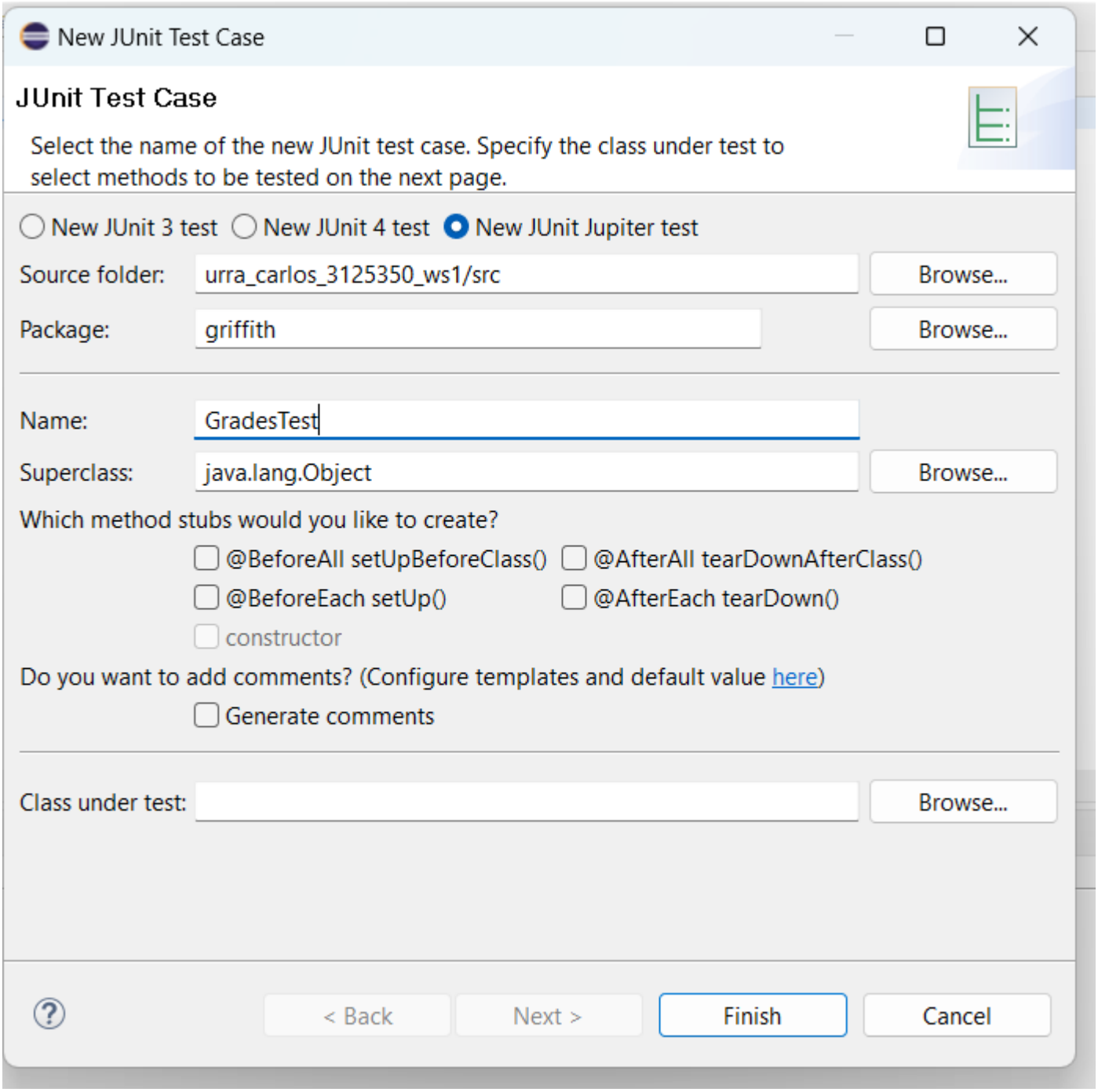
Please do not delete the questions.

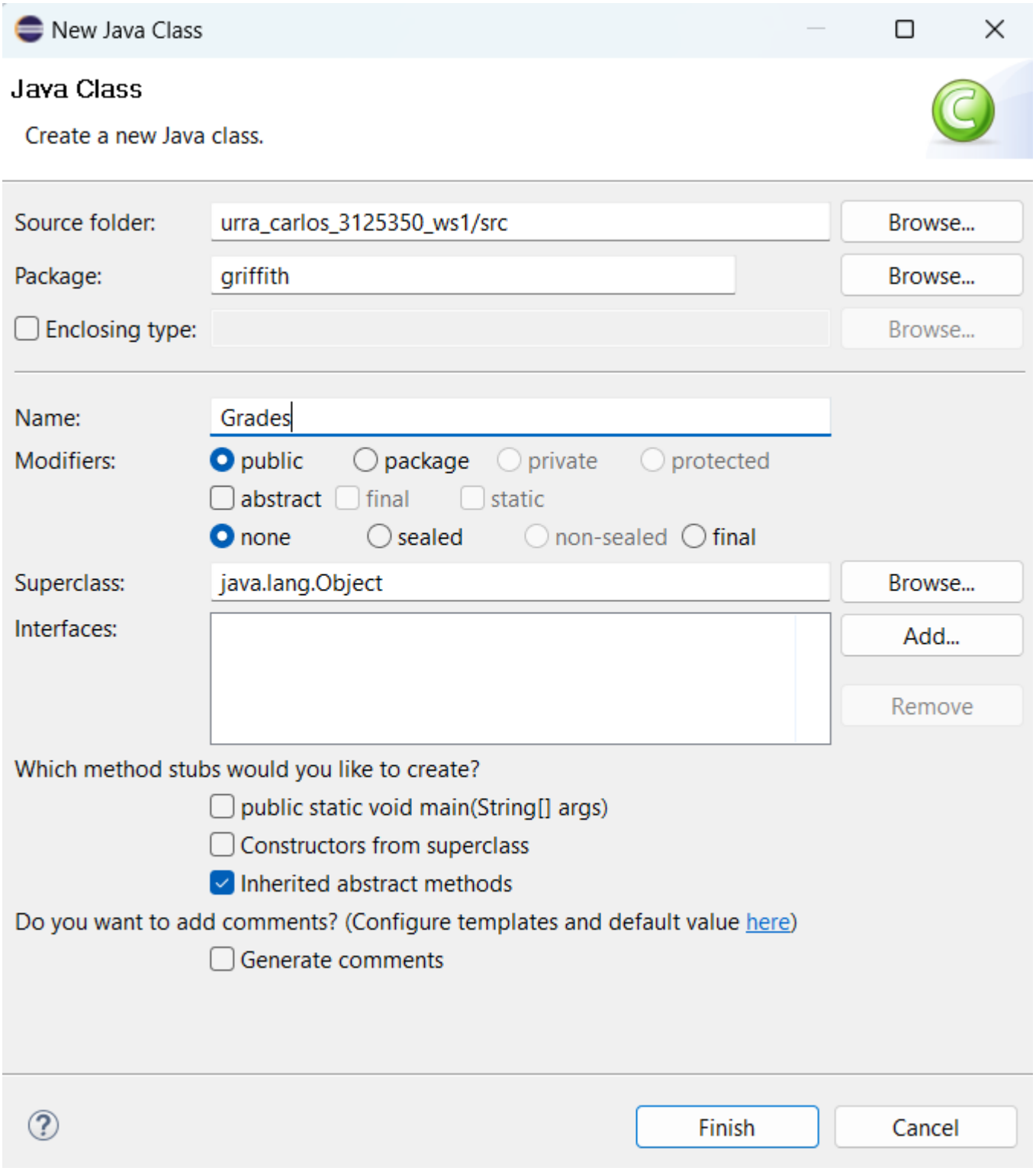
For each question insert your answer below the question

Task 1

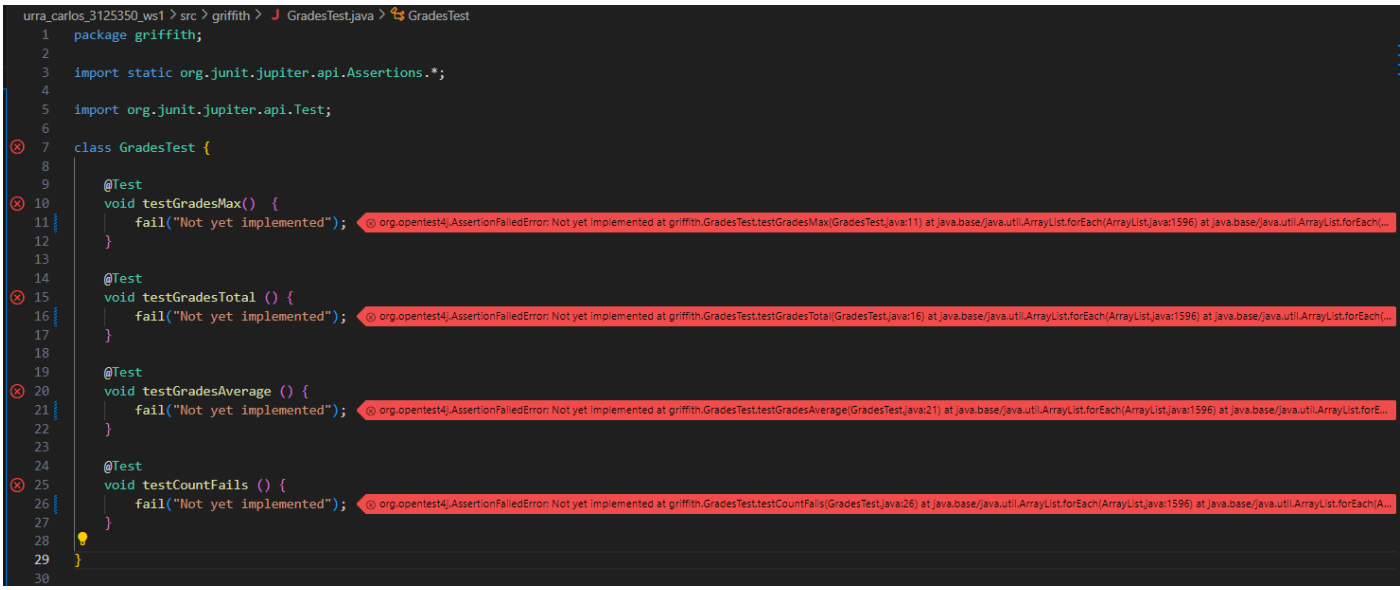
Part 1

- Create a JUnit test file called “GradesTest.java” and a java file “Grades.java”




New Java Class
Java Class
Create a new Java class.
Source folder:
urra_carlos_3125350_ws1/src
Browse...
Package:
griffith
Browse...
Enclosing type:
Browse...
Name:
Grades
Modifiers:
public
package
private
protected
abstract
final
static
none
sealed
non-sealed
final
Superclass:
java.lang.Object
Browse...
Interfaces:
Add...
Remove
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
Finish
Cancel

- In your JUnit test add the following four stub methods: testGradesMax(), testGradesTotal (), testGradesAverage (), and testCountFails ().



- In the regular java file add in the following stub methods:
  - int gradesMax(int[] grades)
  - int gradesTotal (int[] grades) // get sum of array
  - double gradesAverage(int[] grades) // get average of array
  - int countFails(int[] grades, int minGrade) // count how many grades < minGrade

urra_carlos_3125350_ws1 > src > griffith > Grades.java > Grades > countFails(int[], int)

```java
package griffith;

public class Grades {

    int gradesMax(int[] grades) {
        return 0;

    }

    static int gradesTotal (int[] grades) {
        return 0;

    }

    static double gradesAverage(int[] grades) {
        return 0.0;

    }

    static int countFails(int[] grades, int minGrade) {
        return 0;

    }
}

```

Part 2

- Write test cases for each of the unit test methods that call the appropriate methods in the java file.

urra_carlos_3125350_ws1 > src > griffith > J GradesTest.java > ...

```java
package griffith;

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

import org.junit.jupiter.api.Test;

class GradesTest {

    @Test
    void testGradesMax()  {
        int grade1 = 101 ;
        assertFalse(grade1 > 100, "The grade should not exceed 100");  ⊗ true != false

        int grade2 = 100;
        assertTrue(grade2 <= 100);

        int grade3 = -100;
        assertTrue(grade3 < 0,"Grades cannot be negative");
    }

    @Test
    void testGradesTotal () {
        int[] grades1 = {60, 75, 80};
        assertEquals(215,Grades.gradesTotal(grades1),"Total should be 215");  ⊗ 0 != 215

        int[] grades2 = {60, 75, -10};
        assertEquals(125,Grades.gradesTotal(grades2),"Total is 125 but there cannot be negative grades");

        int[] grades3 = {0};
        assertEquals(0, Grades.gradesTotal(grades3),"Total is 0, but there cannot be an empty array");
    }
```

urra_carlos_3125350_ws1 > src > griffith > J GradesTest.java > ...

```java
class GradesTest {

    @Test
    void testGradesAverage () {
        int[] grades1 = {60, 75, 80};
        assertEquals(71.66, Grades.gradesAverage(grades1),"The average should be 71.66");  ⊗ 0.0 != 71.66

        int[] grades2 = {60, 75, -10};
        assertEquals(41.66, Grades.gradesAverage(grades2),"The average should be 41.66, but grades cannot be negative");

        int[] grades3 = {};
        assertEquals(0.0, Grades.gradesAverage(grades3),"The average should be 0, but the array cannot be empty");
    }

    @Test
    void testCountFails () {
        int[] grades1 = {30, 45, 60, 20, 50};
        assertEquals(2, Grades.countFails(grades1, minGrade:40),"Fail count should 2");  ⊗ 40 != 2

        int[] grades2 = {55, 60, 75, 80, 90};
        assertEquals(0, Grades.countFails(grades2, minGrade:40),"Fail count should be 0");

        int[] grades3 = {30, 39, 20, 10, 15};
        assertEquals(5, Grades.countFails(grades3, minGrade:40),"Fail count should be 5");
    }

}
```

Part 3

- Implement the gradesMax. Once its implemented test if it passes the unit test. If it passes implement the gradesTotal method, Continue until all methods are implemented. You may only move onto the next implementation once each unit test has been satisfied.

Method: gradesMax(int[] grades) **Unit Test: testGradeMax()**

```java
static int gradesMax(int[] grades) {
    if (grades.length == 0) throw new IllegalArgumentException(s:"Grade array cannot be empty.");
    int max = grades[0];
    for (int grade : grades) {
        if (grade > max) {
            max = grade;
        }
    }
    return max;

}
```

urra_carlos_3125350_ws1 > src > griffith > GradesTest.java > ...

```java
package griffith;

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

class GradesTest {

    @Test
    void testGradesMax() {
        int[] grades1 = {80, 90, 70}; // Normal values
        int[] grades2 = {-10, 0, -5}; // Includes negatives
        int[] grades3 = {100, 50, 75}; // Includes max boundary

        assertEquals(90, Grades.gradesMax(grades1), "Max grade should be 90");
        assertEquals(0, Grades.gradesMax(grades2), "Max grade should be 0");
        assertEquals(100, Grades.gradesMax(grades3), "Max grade should be 100");
    }

    @Test
    void testGradesTotal() {
        int[] grades1 = {80, 90, 70}; // Normal case
        int[] grades2 = {10, 20, 30}; // Small values
        int[] grades3 = {}; // Edge case: empty array

        assertEquals(240, Grades.gradesTotal(grades1), "Total should be 240");
        assertEquals(60, Grades.gradesTotal(grades2), "Total should be 60");
```

0 != 240

Method: gradesTotal (int[] grades) **Unit Test: testGradesTotal()**

```java
static int gradesTotal (int[] grades) {
    if (grades.length == 0) throw new IllegalArgumentException(s:"Grade array cannot be empty.");
    int total = 0;
    for (int grade : grades) {
        total += grade;
    }
    return total;

}
```

```java
    @Test
    void testGradesTotal() {
        int[] grades1 = {80, 90, 70}; // Normal case
        int[] grades2 = {10, 20, 30}; // Small values
        int[] grades3 = {}; // Edge case: empty array

        assertEquals(240, Grades.gradesTotal(grades1), "Total should be 240");
        assertEquals(60, Grades.gradesTotal(grades2), "Total should be 60");

        // Check for empty array exception
        assertThrows(IllegalArgumentException.class, () -> {
            Grades.gradesTotal(grades3);
        }, "Should throw exception for empty array");
    }

    @Test
    void testGradesAverage() {
        int[] grades1 = {80, 90, 70}; // Normal case
        int[] grades2 = {10, 20, 30}; // Small values
        int[] grades3 = {}; // Edge case: empty array

        assertEquals(80.0, Grades.gradesAverage(grades1), 0.01, "Average should be 80.0");  0.0 != 80.0
        assertEquals(20.0, Grades.gradesAverage(grades2), 0.01, "Average should be 20.0");
```

Method: gradesAverage(int[] grades) **Unit Test: testGradesAverage()**

```
26
27      static double gradesAverage(int[] grades) {
28          if (grades.length == 0) throw new IllegalArgumentException(s:"Grade array cannot be empty.");
29          return (double) gradesTotal(grades) / grades.length;
30
31      }
32
```

```
33
34      @Test
35      void testGradesAverage() {
36          int[] grades1 = {80, 90, 70}; // Normal case
37          int[] grades2 = {10, 20, 30}; // Small values
38          int[] grades3 = {}; // Edge case: empty array
39
40          assertEquals(80.0, Grades.gradesAverage(grades1), 0.01, "Average should be 80.0");
41          assertEquals(20.0, Grades.gradesAverage(grades2), 0.01, "Average should be 20.0");
42
43          // Check for empty array exception
44          assertThrows(IllegalArgumentException.class, () -> {
45              Grades.gradesAverage(grades3);
46          }, "Should throw exception for empty array");
47      }
48
49      @Test
50      void testCountFails() {
51          int[] grades1 = {30, 45, 60, 20, 50}; // Some fails
52          int[] grades2 = {55, 60, 80, 75}; // No fails
53          int[] grades3 = {10, 20, 30, 39, 29}; // All fails
```

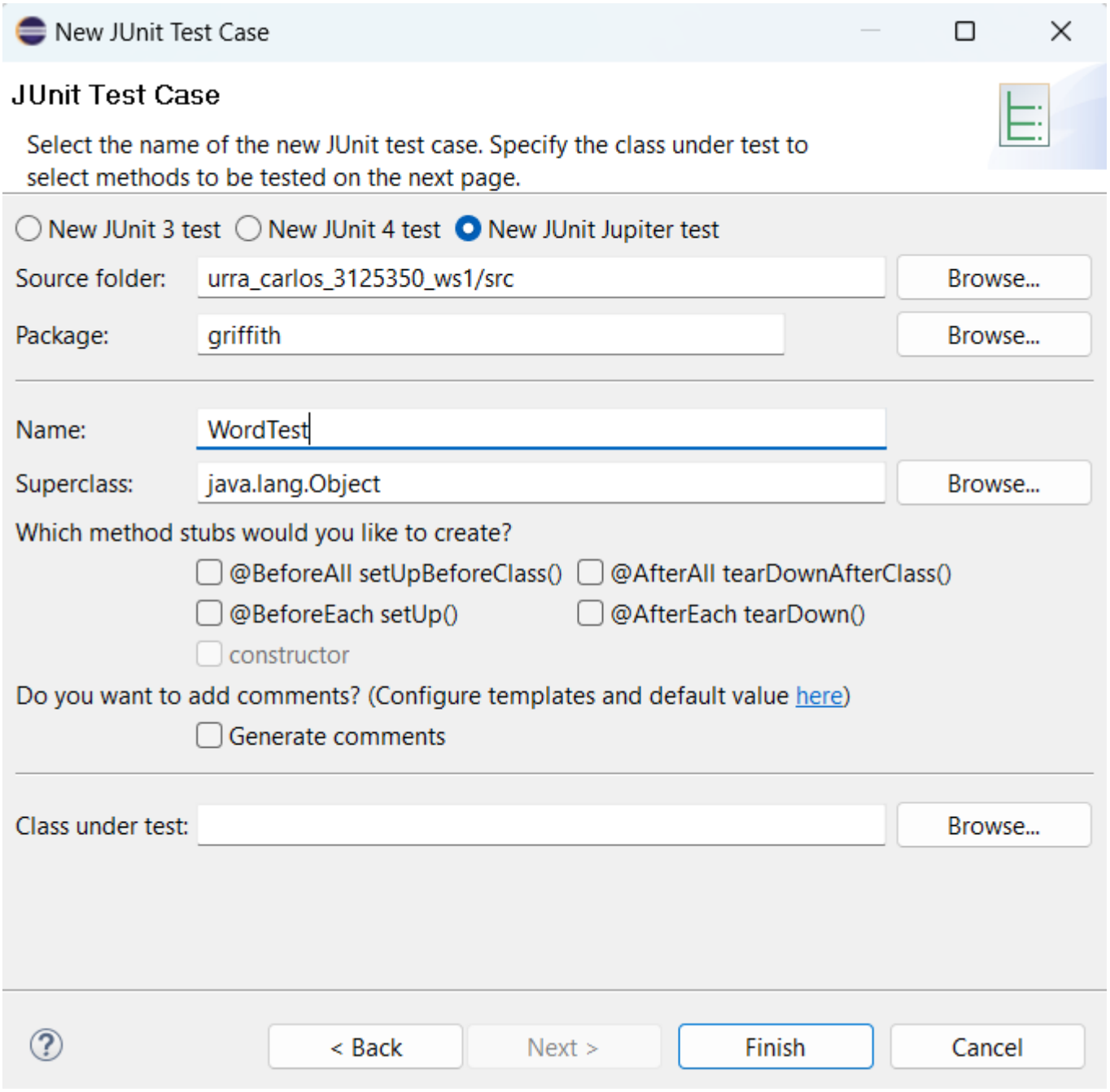
Method: countFails(int[] grades) **Unit Test: testCountFails()**

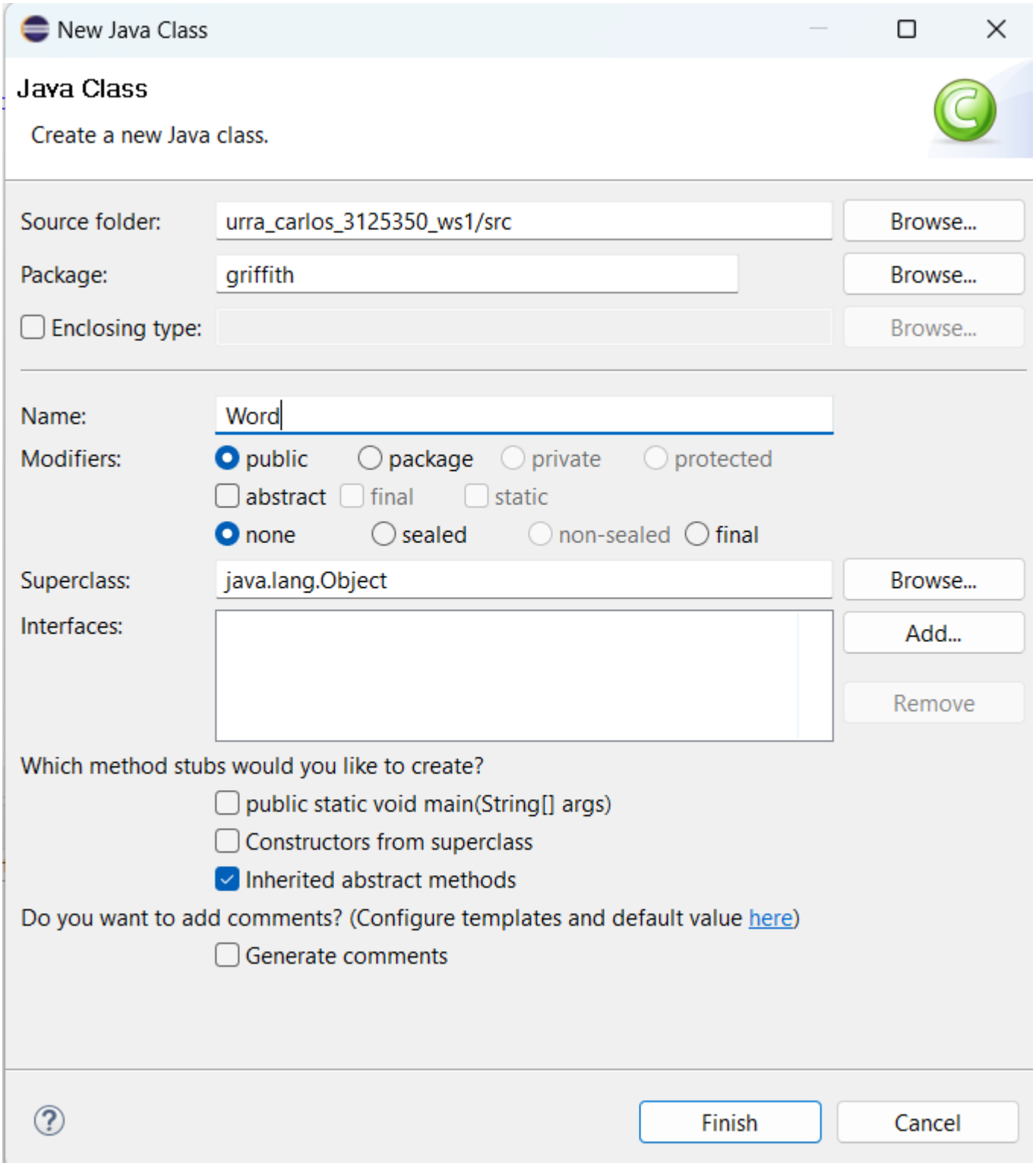
```java
    static int countFails(int[] grades, int minGrade) {
        if (grades.length == 0) throw new IllegalArgumentException(s:"Grade array cannot be empty.");
        int count = 0;
        for (int grade : grades) {
            if (grade < minGrade) {
                count++;
            }
        }
        return count;

    }
}
```

```java
    @Test
    void testCountFails() {
        int[] grades1 = {30, 45, 60, 20, 50}; // Some fails
        int[] grades2 = {55, 60, 80, 75}; // No fails
        int[] grades3 = {10, 20, 30, 39, 29}; // All fails

        assertEquals(2, Grades.countFails(grades1, minGrade:40), "Fail count should be 2");
        assertEquals(0, Grades.countFails(grades2, minGrade:40), "Fail count should be 0");
        assertEquals(5, Grades.countFails(grades3, minGrade:40), "Fail count should be 5");
    }
}
```

Task 2

Part 1


New JUnit Test Case
JUnit Test Case
Select the name of the new JUnit test case. Specify the class under test to select methods to be tested on the next page.
New JUnit 3 test
New JUnit 4 test
New JUnit Jupiter test
Source folder:
urra_carlos_3125350_ws1/src
Browse...
Package:
griffith
Browse...
Name:
WordTest
Superclass:
java.lang.Object
Browse...
Which method stubs would you like to create?
@BeforeAll setUpBeforeClass()
@AfterAll tearDownAfterClass()
@BeforeEach setUp()
@AfterEach tearDown()
constructor
Do you want to add comments? (Configure templates and default value here)
Generate comments
Class under test:
Browse...
< Back
Next >
Finish
Cancel


New Java Class
Java Class
Create a new Java class.
Source folder:
urra_carlos_3125350_ws1/src
Browse...
Package:
griffith
Browse...
Enclosing type:
Browse...
Name:
Word
Modifiers:
public
package
private
protected
abstract
final
static
none
sealed
non-sealed
final
Superclass:
java.lang.Object
Browse...
Interfaces:
Add...
Remove
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
Finish
Cancel

urra_carlos_3125350_ws1 > src > griffith > Word.java > Word

```java
package griffith;

public class Word {

    static boolean contains(char symbol) {
        return false;

    }

    int length() {
        return 0;

    }

    char[] getLetters() {
        return null;

    }

}
```

urra_carlos_3125350_ws1 > src > griffith > WordTest.java > ...

```java
package griffith;

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

import org.junit.jupiter.api.Test;

class WordTest {

    @Test
    void testContains()  {
        fail("Not yet implemented");
    }

    @Test
    void testLength()  {
        fail("Not yet implemented");
    }

    @Test
    void testNotNull()  {
        fail("Not yet implemented");
    }

}
```

Part 2

- Write test cases for each of the unit test methods that call the appropriate methods in the java file.

```
6
7   class WordTest {
8
9       @Test
10      void testContains()  {
11          Word word = new Word (new char[] {'C', 'a', 'r', 'l', 'o', 's'});
12          assertTrue(word.contains(symbol:'c'), "The word should contain the letter 'c'");   ⊗ false != true
13          assertTrue(word.contains(symbol:'a'), "Word should contain 'a'");
14          assertTrue(word.contains(symbol:'s'), "Word should contain 's'");
15      }
16
17
18      @Test
19      void testLength() {
20          Word word1 = new Word(new char[] {'H'});
21          assertEquals(1, word1.length(), "Length should be 1");   ⊗ 0 != 1
22
23          Word word2 = new Word(new char[] {'U', 'r', 'r', 'a'});
24          assertEquals(4, word2.length(), "Length should be 4");
25
26          Word word3 = new Word(new char[] {});
27          assertEquals(0, word3.length(), "Length should be 0");
28      }
29
30       @Test
31      void testNotNull() {
32          Word word = new Word(new char[] {'c', 'a', 'r', 'l', 'o', 's'});
33          assertNotNull(word.getLetters(), "Letters array should not be null");   ⊗ org.opentest4j.AssertionFailedError: Letters array should not be null =...
34      }
35  }
36
```

Part 3

Method: contains() **Unit test: testContains()**

```
10      public boolean contains(char symbol) {
11          for (char letter : letters) {
12              if (letter == symbol) {
13                  return true;
14              }
15          }
16          return false;
17      }
```

```java
class WordTest {

    @Test
    void testContains() {
        Word word = new Word(new char[] {'c', 'a', 'r', 'l', 'o', 's'});
        assertTrue(word.contains(symbol:'c'), "Word should contain 'c'");
        assertTrue(word.contains(symbol:'a'), "Word should contain 'a'");
        assertTrue(word.contains(symbol:'s'), "Word should contain 's'");
    }

    @Test
    void testLength() {
        Word word1 = new Word(new char[] {'H'});
        assertEquals(1, word1.length(), "Length should be 1");  ⊗ 0 != 1

        Word word2 = new Word(new char[] {'H', 'e', 'l', 'l', 'o'});
        assertEquals(5, word2.length(), "Length should be 5");

        Word word3 = new Word(new char[] {});
        assertEquals(0, word3.length(), "Length should be 0");
    }
```

Method: length() **Unit test: testLength()**

```java
    @Test
    void testLength() {
        Word word1 = new Word(new char[] {'H'});
        assertEquals(1, word1.length(), "Length should be 1");

        Word word2 = new Word(new char[] {'H', 'e', 'l', 'l', 'o'});
        assertEquals(5, word2.length(), "Length should be 5");

        Word word3 = new Word(new char[] {});
        assertEquals(0, word3.length(), "Length should be 0");
    }


    @Test
    void testNotNull() {
        Word word = new Word(new char[] {'c', 'a', 'r', 'l', 'o', 's'});
        assertNotNull(word.getLetters(), "Letters array should not be null");
    }
}
```

Method: getLetters() **Unit Test: testNotNull()**

```
22
23      public char[] getLetters() {
24          return letters;
25      }
26  }
```

```
30      @Test
31      void testNotNull() {
32          Word word = new Word(new char[] {'c', 'a', 'r', 'l', 'o', 's'});
33          assertNotNull(word.getLetters(), "Letters array should not be null");
34      }
35  }
36
```

Link to my GitHub repository:

https://github.com/Carlos-0620/SD2_New_Repository/tree/06e743c7a7ce8f6897c4b5548136edce54643ca0/urra_carlos_3125350_ws1