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**				

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

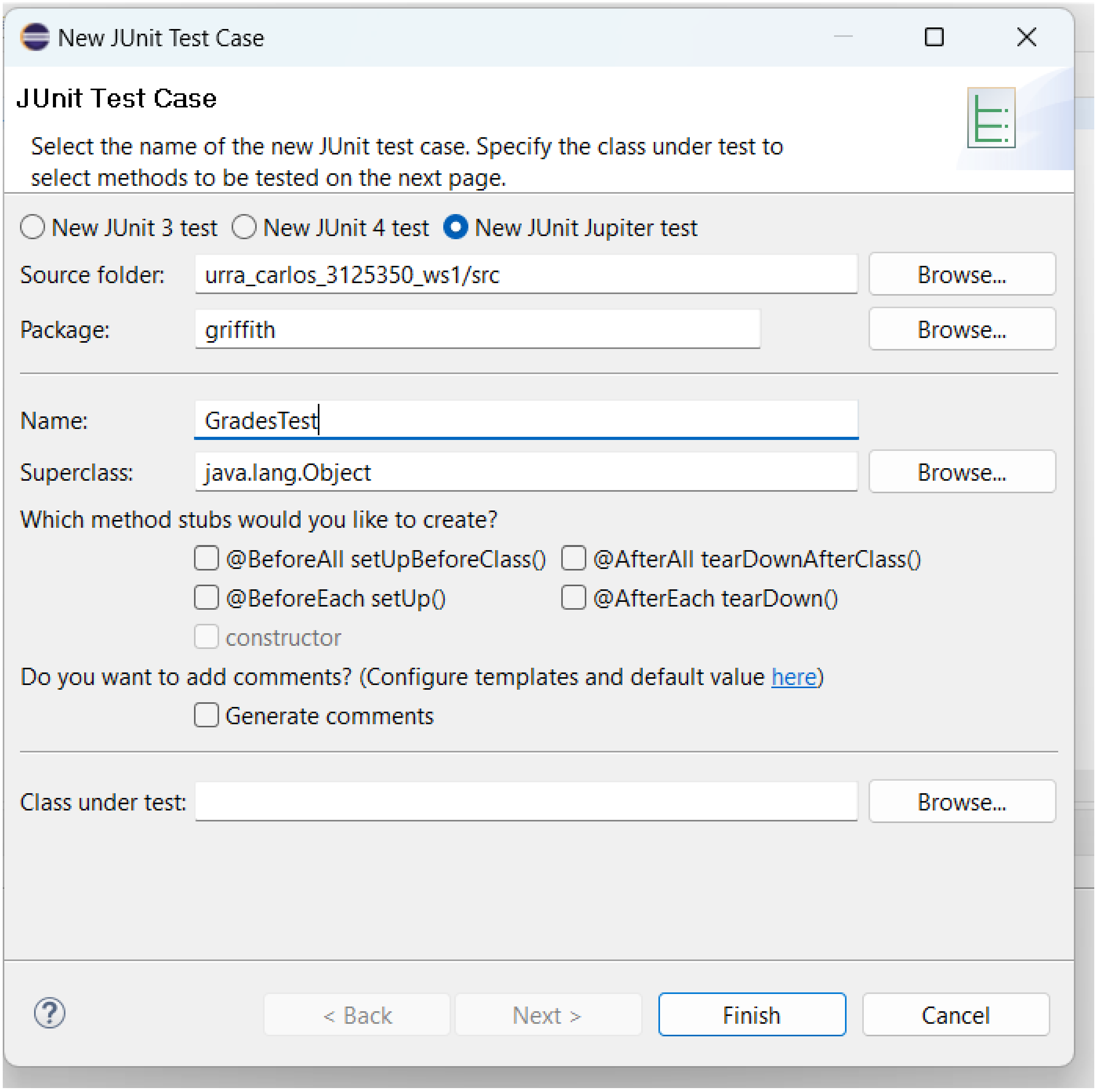
Please do not delete the questions.

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 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

Package: griffith

Name: GradesTest

Superclass: java.lang.Object

Which method stubs would you like to create?

☐ @BeforeAll setUpBeforeClass() ☐ @AfterAll tearDownAfterClass()

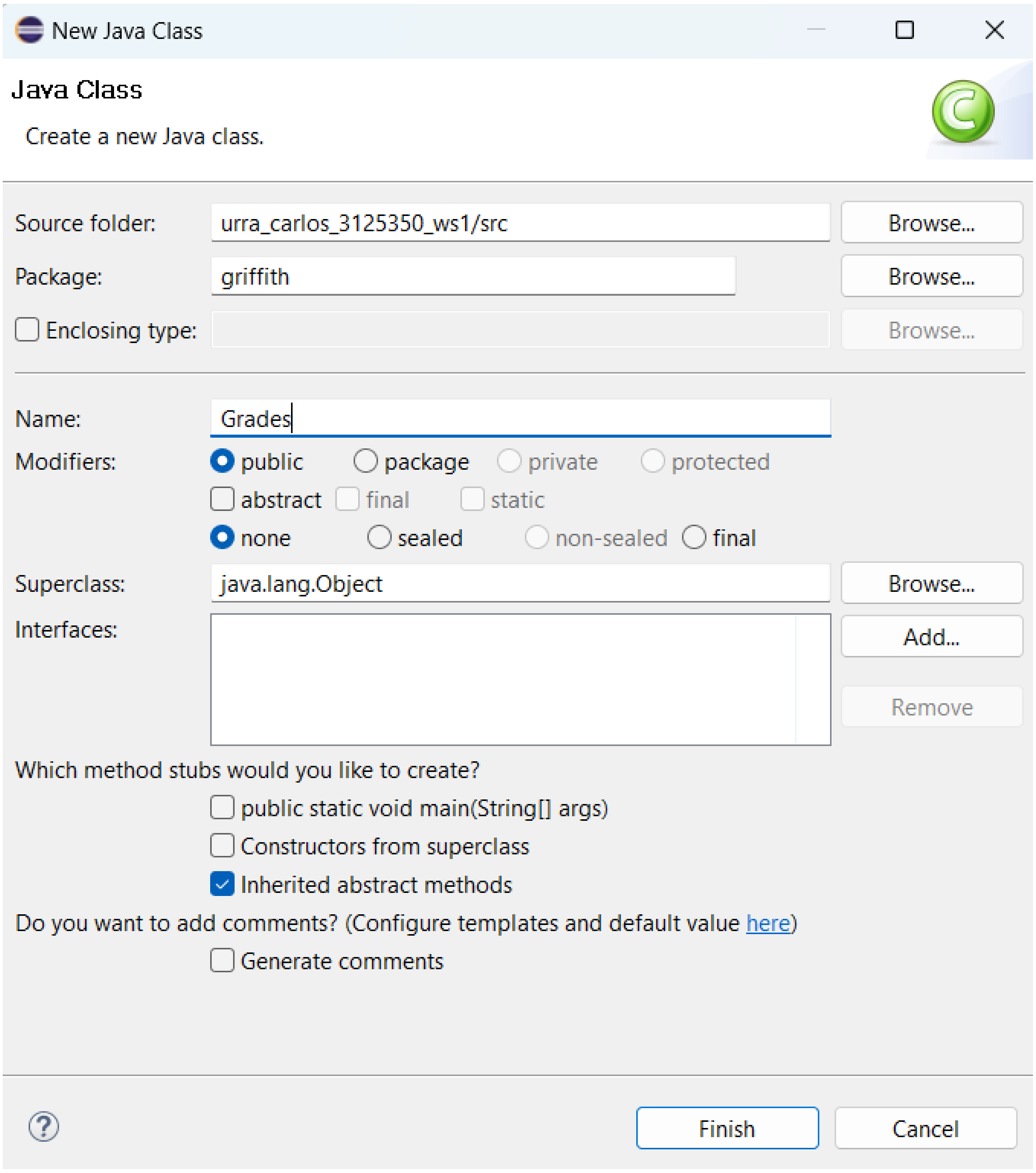
☐ @BeforeEach setUp() ☐ @AfterEach tearDown()

☐ constructor

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

☐ Generate comments

Class under test:



Java Class

Create a new Java class.

Source folder: urra_carlos_3125350_ws1/src

Package: griffith

[ ] Enclosing type:

Name: Grades

Modifiers: [✓ public] package private protected

[ ] abstract [ ] final [ ] static

[✓ none] sealed non-sealed final

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

[ ] public static void main(String[] args)

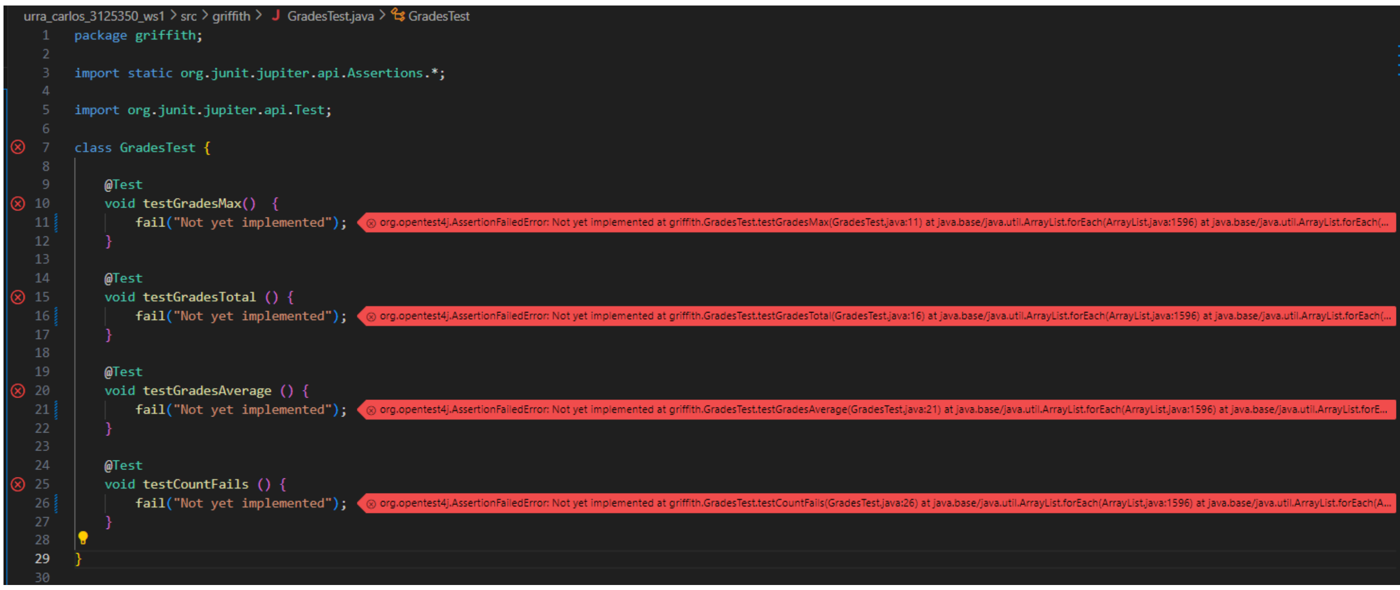
[ ] Constructors from superclass

[✓] Inherited abstract methods

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

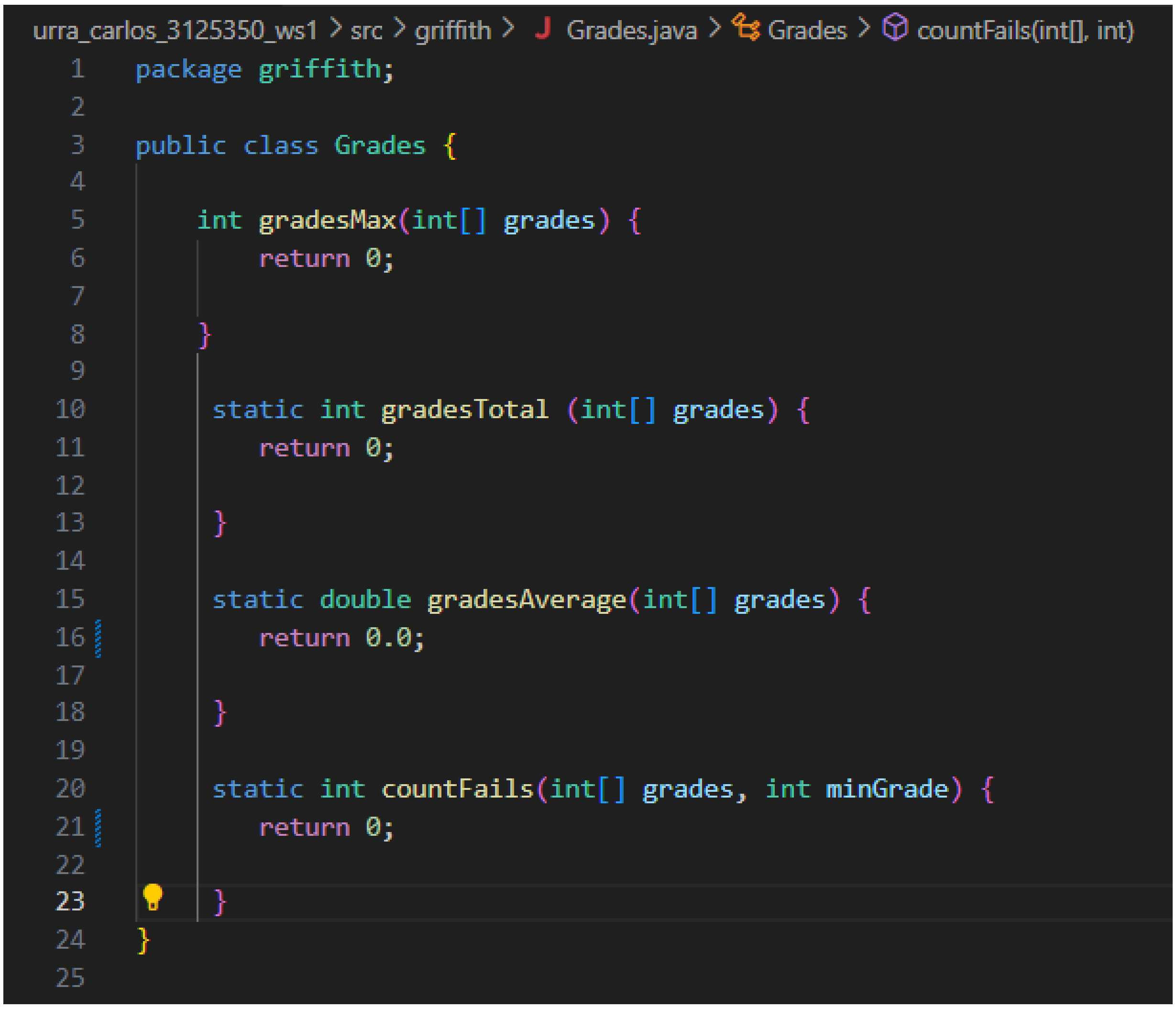
[ ] Generate comments

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



```
urra_carlos_3125350_ws1 > src > griffith > GradesTest.java > GradesTest
package griffith;

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

import org.junit.jupiter.api.Test;

class GradesTest {

    @Test
    void testGradesMax()  {
        fail("Not yet implemented");   org.opentest4j.AssertionFailedError: Not yet implemented at griffith.GradesTest.testGradesMax(GradesTest.java:11) at java.base/java.util.ArrayList.forEach(ArrayList.java:1596) at java.base/java.util.ArrayList.forEach(...
    }

    @Test
    void testGradesTotal () {
        fail("Not yet implemented");   org.opentest4j.AssertionFailedError: Not yet implemented at griffith.GradesTest.testGradesTotal(GradesTest.java:16) at java.base/java.util.ArrayList.forEach(ArrayList.java:1596) at java.base/java.util.ArrayList.forEach(...
    }

    @Test
    void testGradesAverage () {
        fail("Not yet implemented");   org.opentest4j.AssertionFailedError: Not yet implemented at griffith.GradesTest.testGradesAverage(GradesTest.java:21) at java.base/java.util.ArrayList.forE...
    }

    @Test
    void testCountFails () {
        fail("Not yet implemented");   org.opentest4j.AssertionFailedError: Not yet implemented at griffith.GradesTest.testCountFails(GradesTest.java:26) at java.base/java.util.ArrayList.forEach(ArrayList.java:1596) at java.base/java.util.ArrayList.forEach(A...
    }

}
```

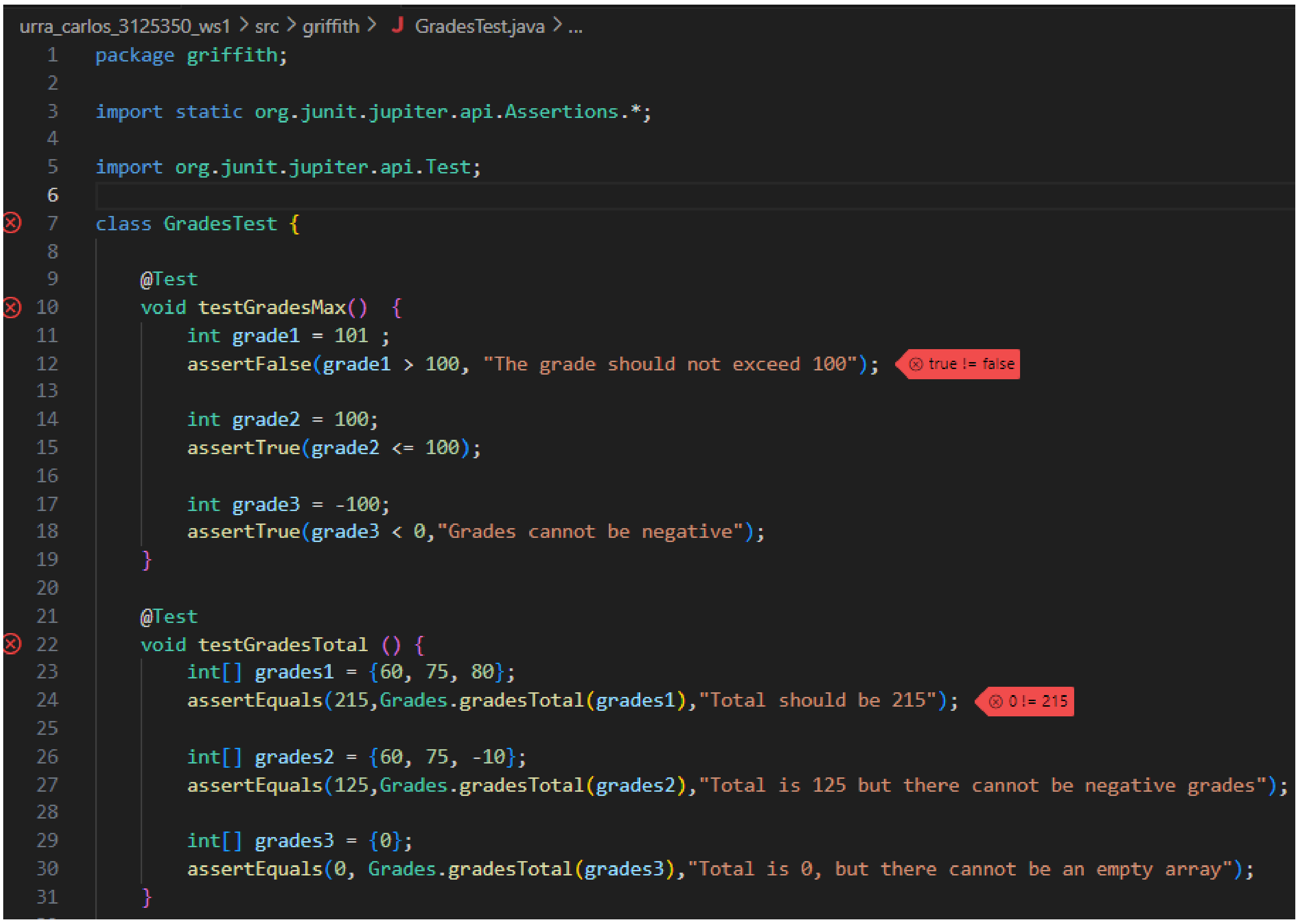
- In the regular java file add in the following stub methods:
    - int gradesMax(int[] grades)
    - int gradesTotal (int[] grades) // get sum of array
    - double gradesAverage(int[] grades) // get average of array
    - int countFails(int[] grades, int minGrade) // count how many grades < minGrade


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;

     }
}
```

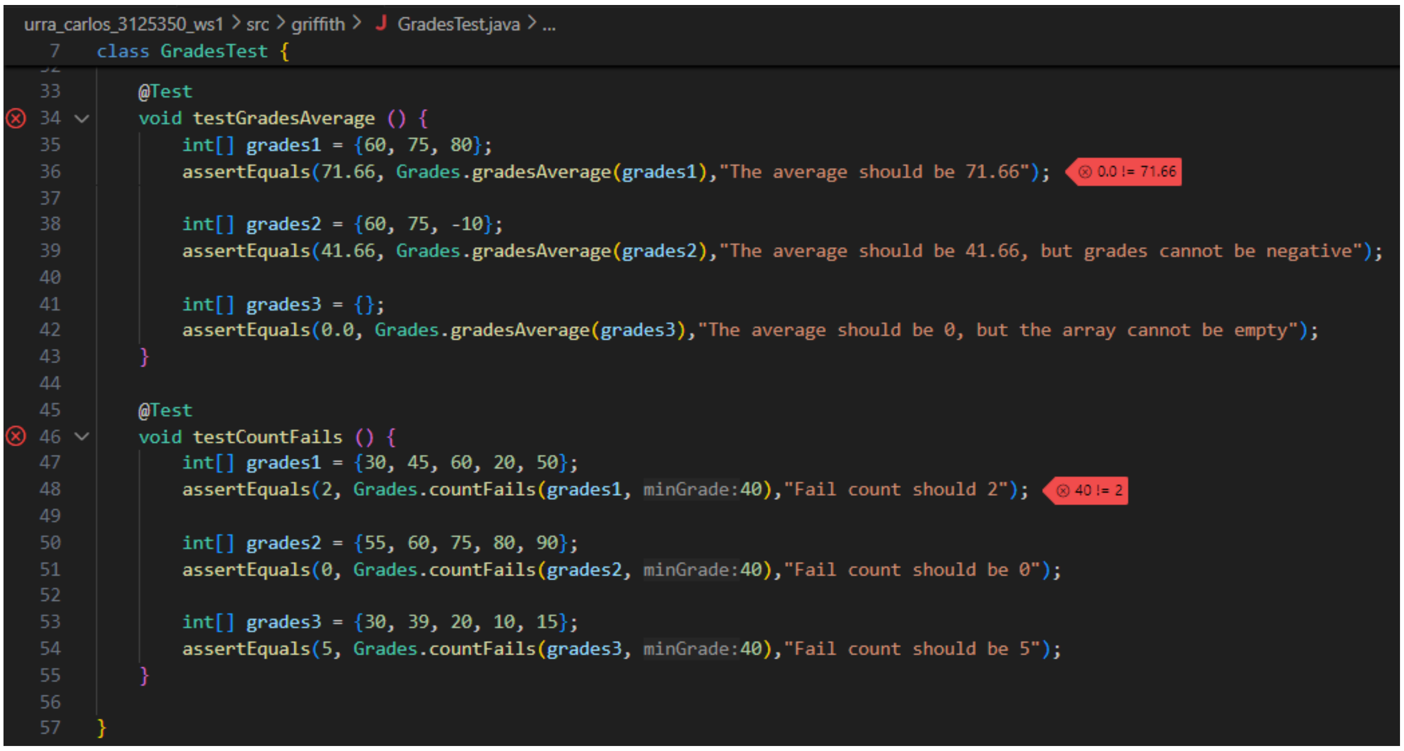
Part 2

- 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 > 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");
    }
```



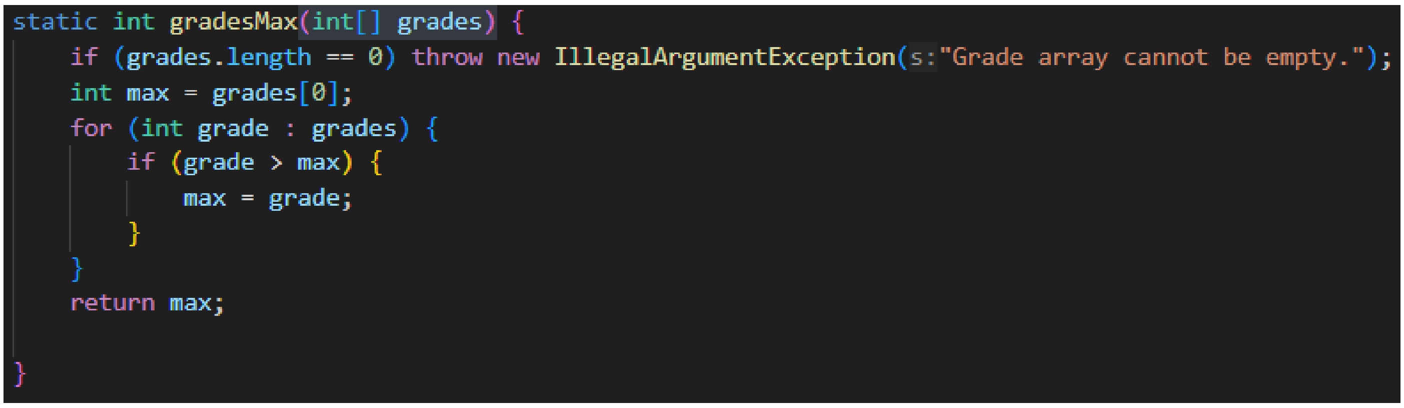
urra_carlos_3125350_ws1 > src > griffith > GradesTest.java > ...

```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");
    }

}
```

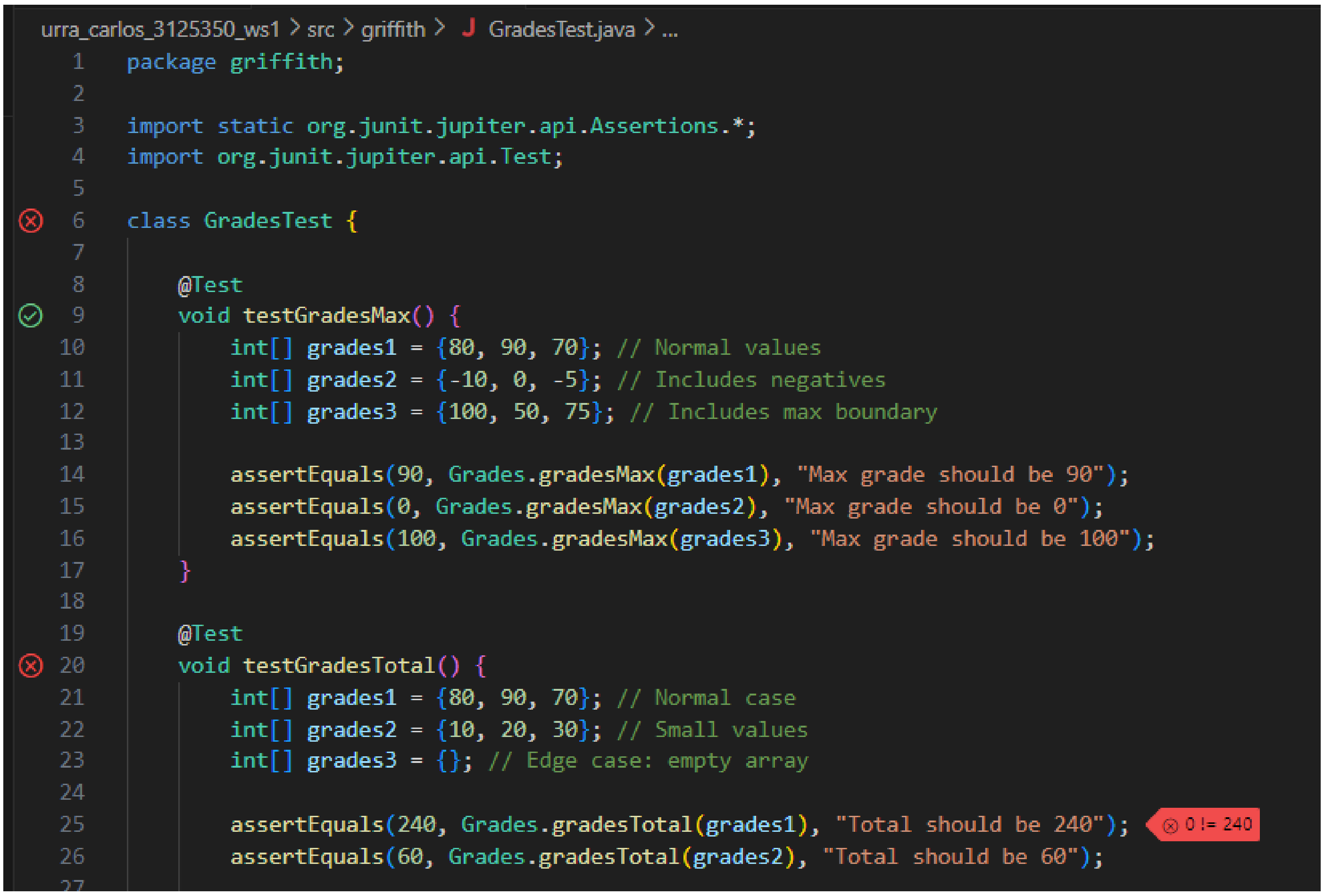
Part 3

- 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;

}
```

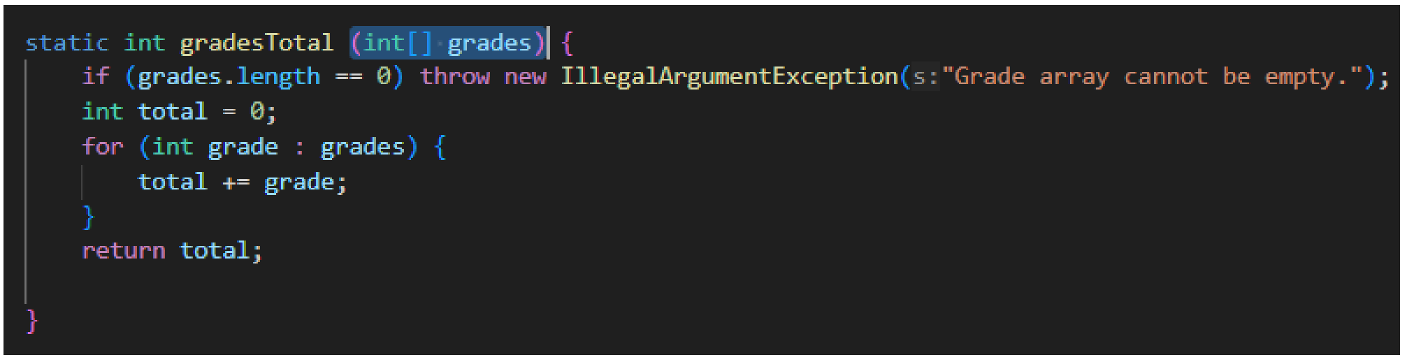


urra_carlos_3125350_ws1 > src > griffith > GradesTest.java > ...

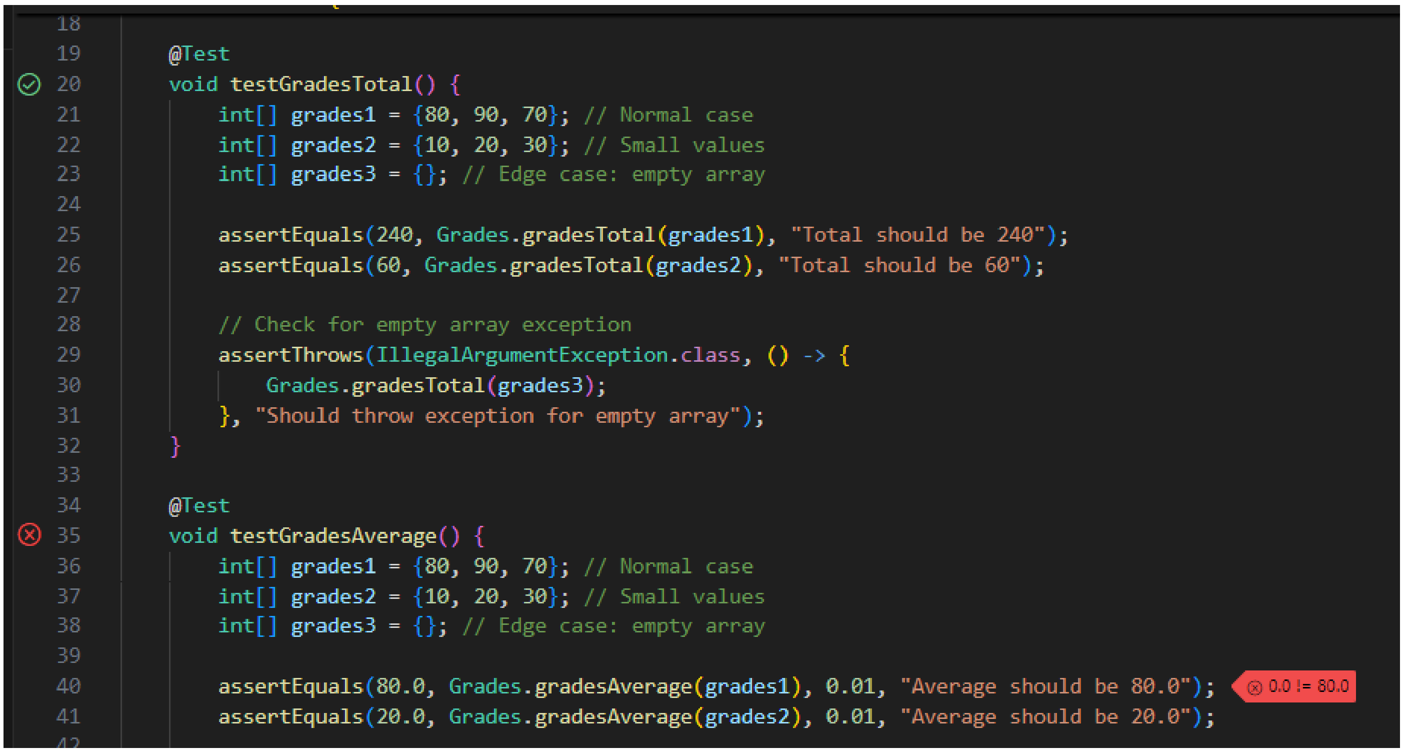
```java
package griffith;

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");  0 != 240
        assertEquals(60, Grades.gradesTotal(grades2), "Total should be 60");
```

SD2 - Worksheet 1 - 6%

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



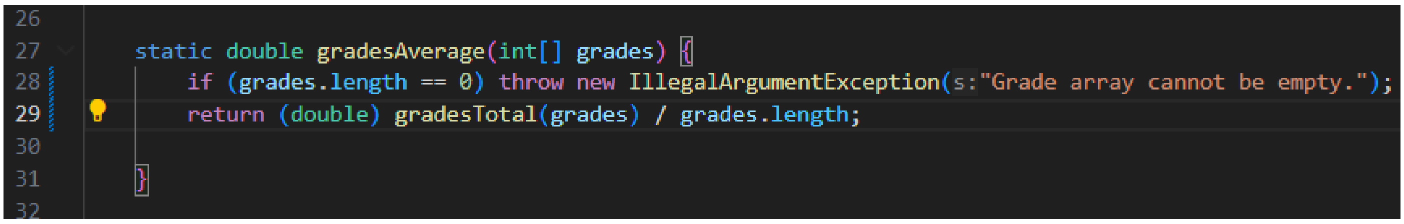
```java
static int gradesTotal (int[] grades) {
    if (grades.length == 0) throw new IllegalArgumentException(s:"Grade array cannot be empty.");
    int total = 0;
    for (int grade : grades) {
        total += grade;
    }
    return total;

}
```



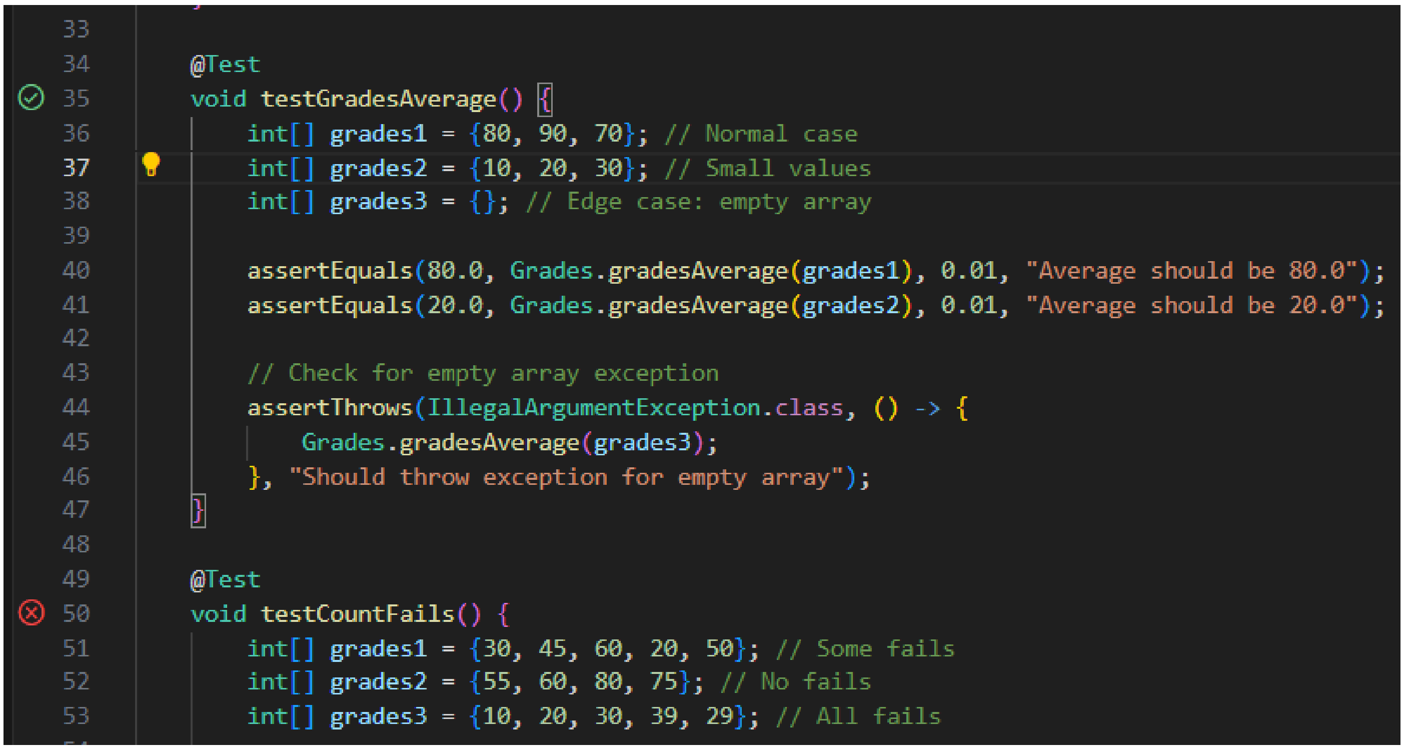
```java
    @Test
    void testGradesTotal() {
        int[] grades1 = {80, 90, 70}; // Normal case
        int[] grades2 = {10, 20, 30}; // Small values
        int[] grades3 = {}; // Edge case: empty array

        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");
```

Carlos Urra - 3125350

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

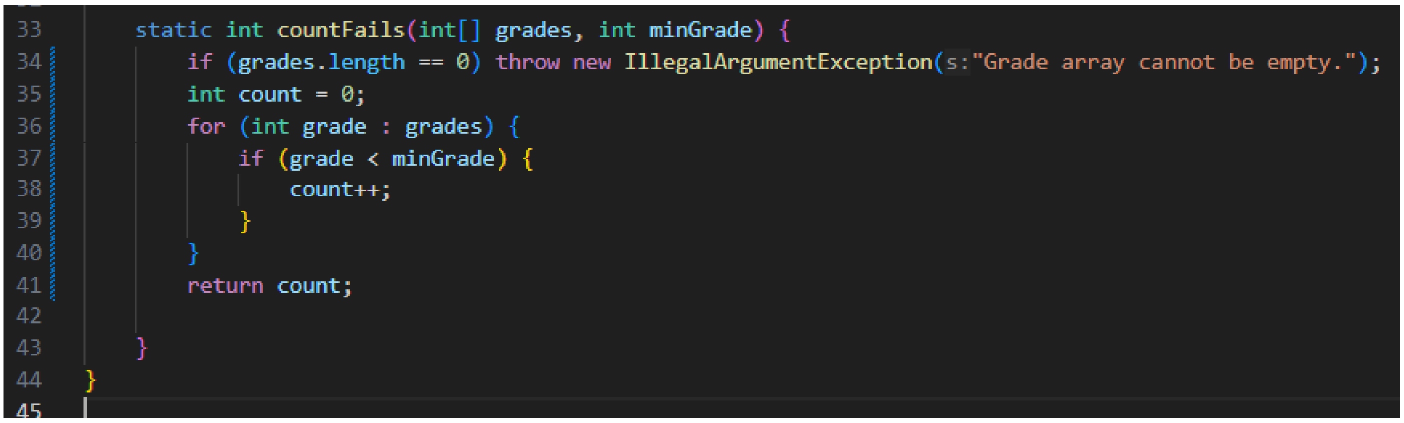


```java
    static double gradesAverage(int[] grades) {
        if (grades.length == 0) throw new IllegalArgumentException(s:"Grade array cannot be empty.");
        return (double) gradesTotal(grades) / grades.length;

    }
```

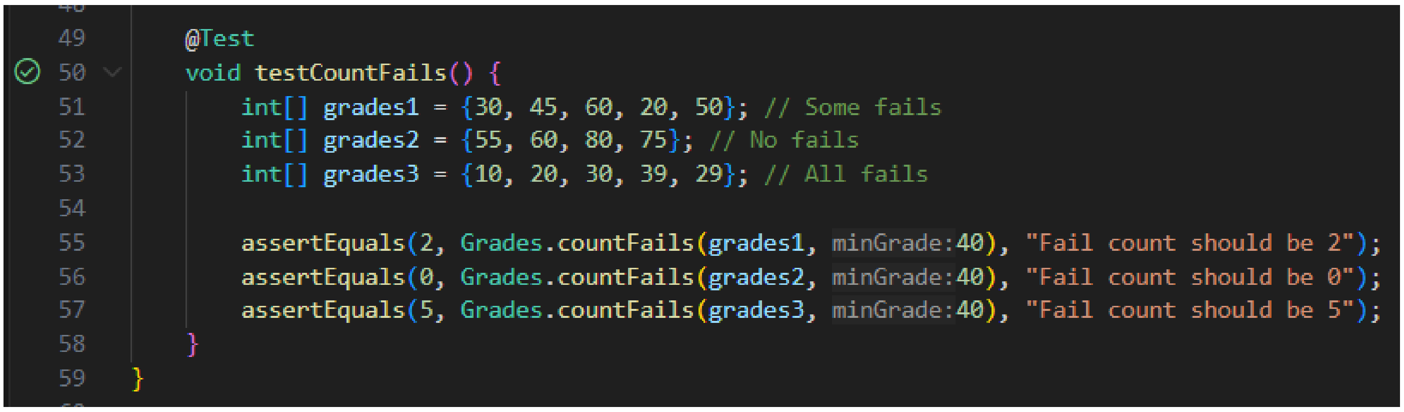


```java

    @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");
        assertEquals(20.0, Grades.gradesAverage(grades2), 0.01, "Average should be 20.0");

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

    @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
```

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



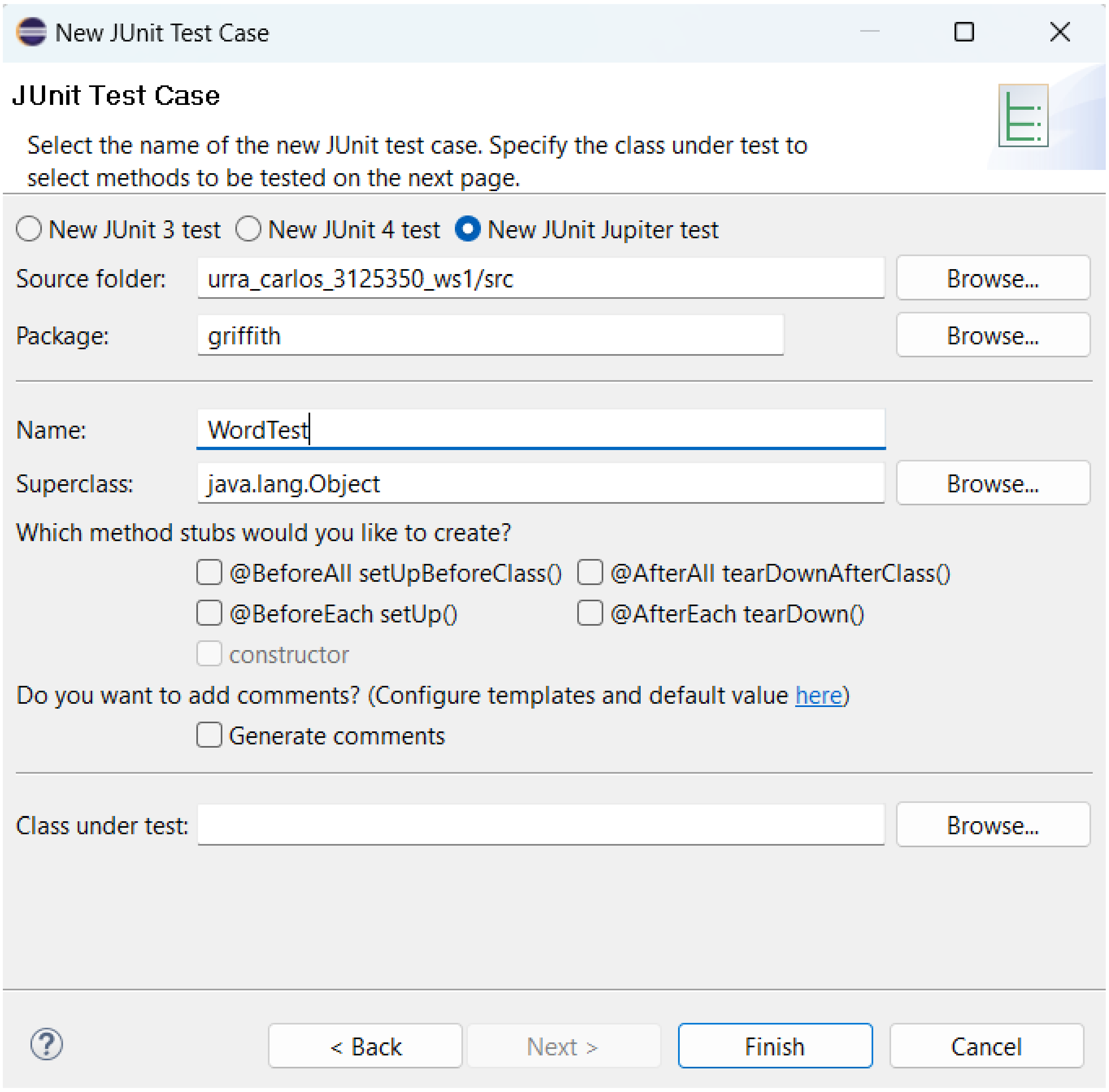
```
    static int countFails(int[] grades, int minGrade) {
        if (grades.length == 0) throw new IllegalArgumentException(s:"Grade array cannot be empty.");
        int count = 0;
        for (int grade : grades) {
            if (grade < minGrade) {
                count++;
            }
     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

    }
}
```



```
    @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
Package:	griffith
Name:	WordTest
Superclass:	java.lang.Object

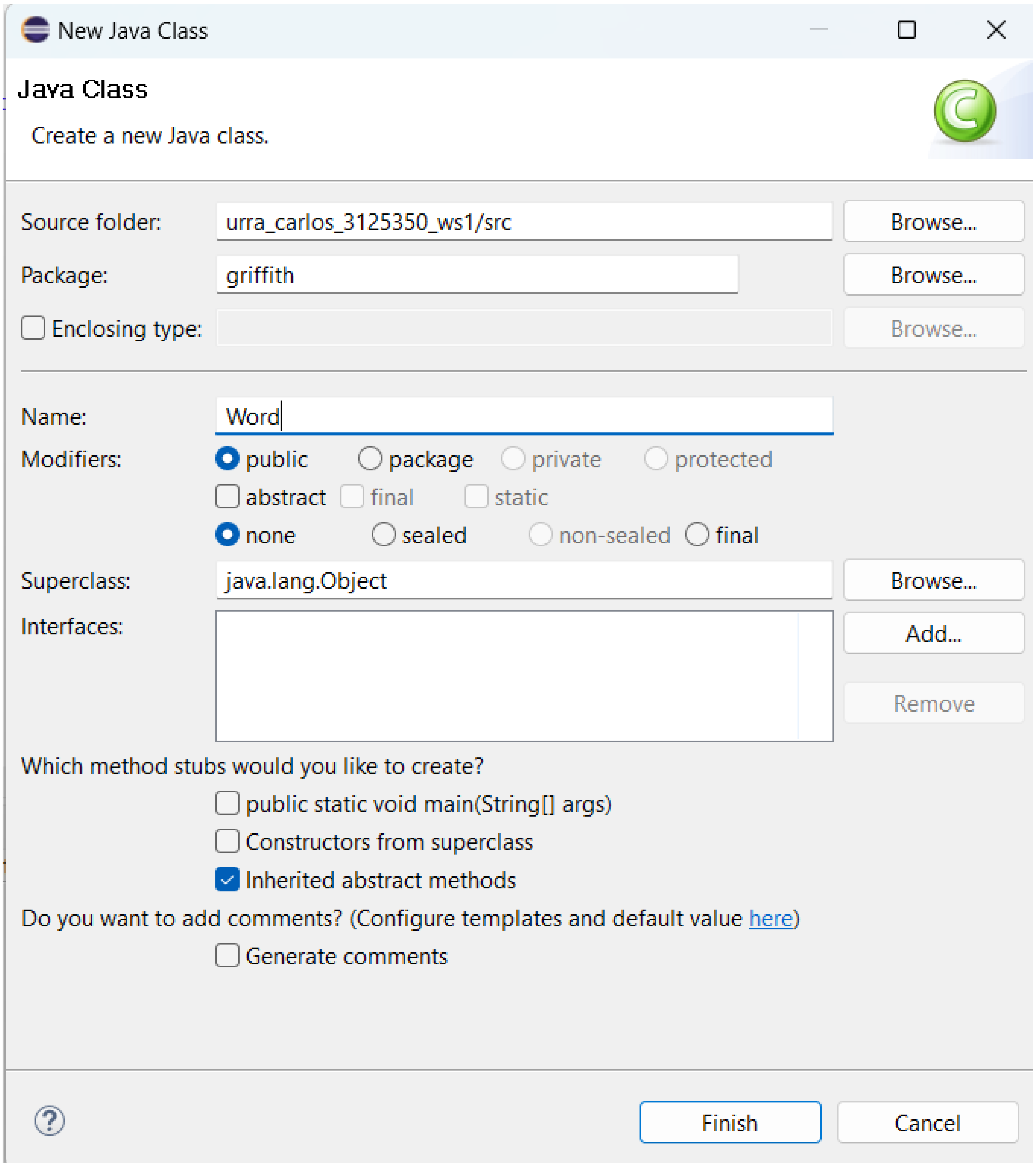
Which method stubs would you like to create?

- [ ] @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:



New Java Class

Java Class

Create a new Java class.

Source folder: urra_carlos_3125350_ws1/src

Package: griffith

Enclosing type:

Name: Word

Modifiers: [✓ public] package private protected

abstract final static

[✓ none] sealed non-sealed final

Superclass: java.lang.Object

Interfaces:

Which method stub would you like to create?

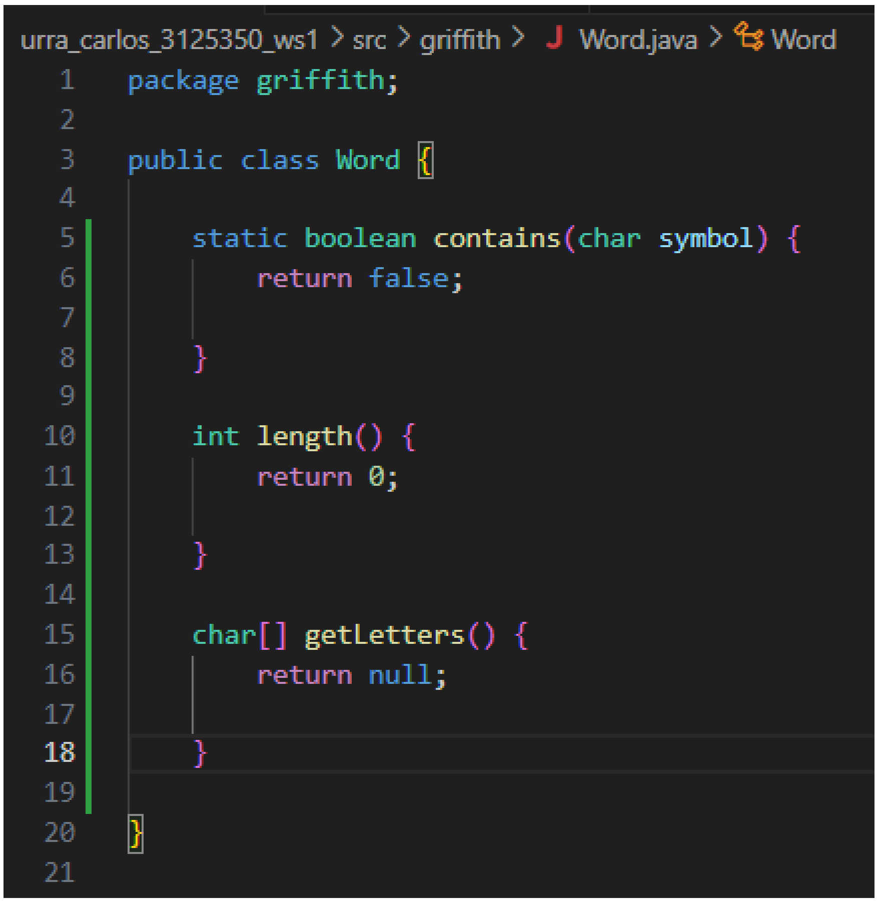
public static void main(String[] args)

Constructors from superclass

[✓ Inherited abstract methods]

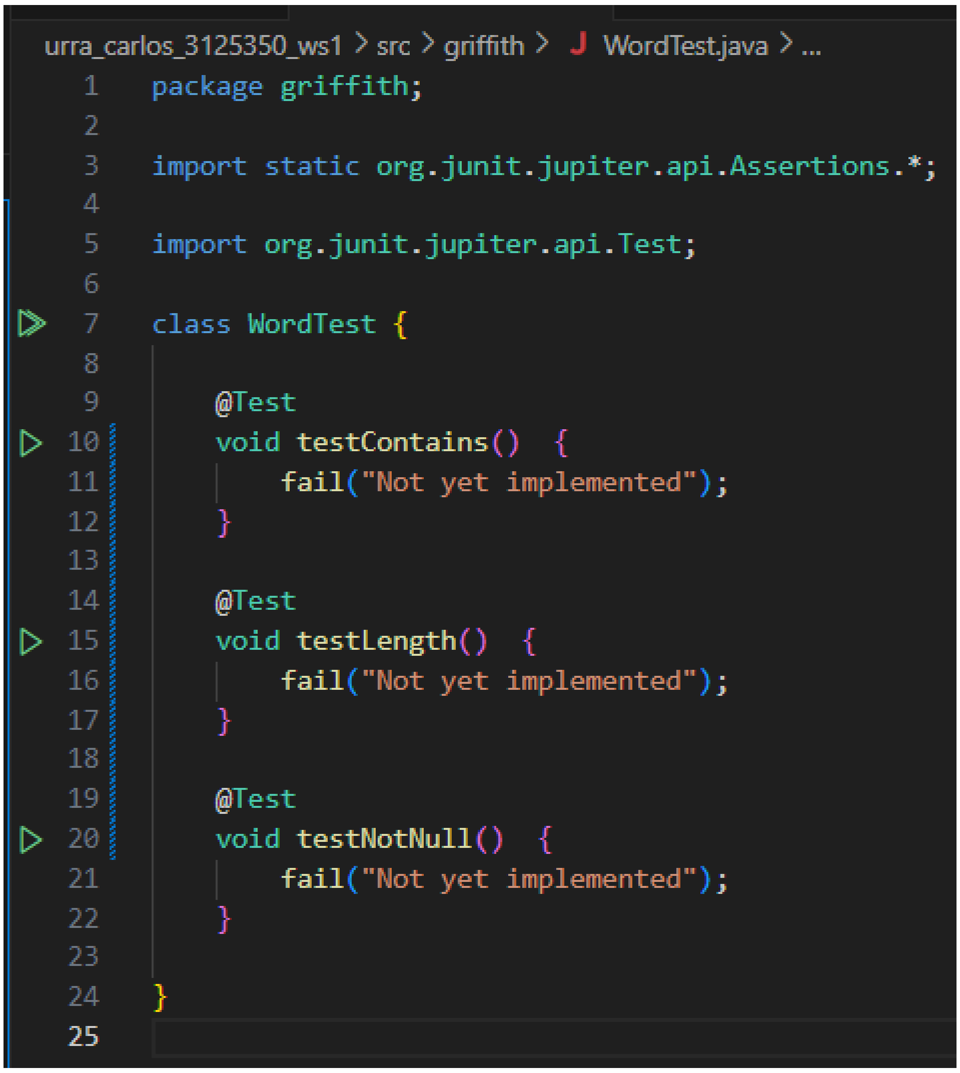
Do you want to add comments? (Configure templates and default value here)

Generate comments



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;

    }

}

```

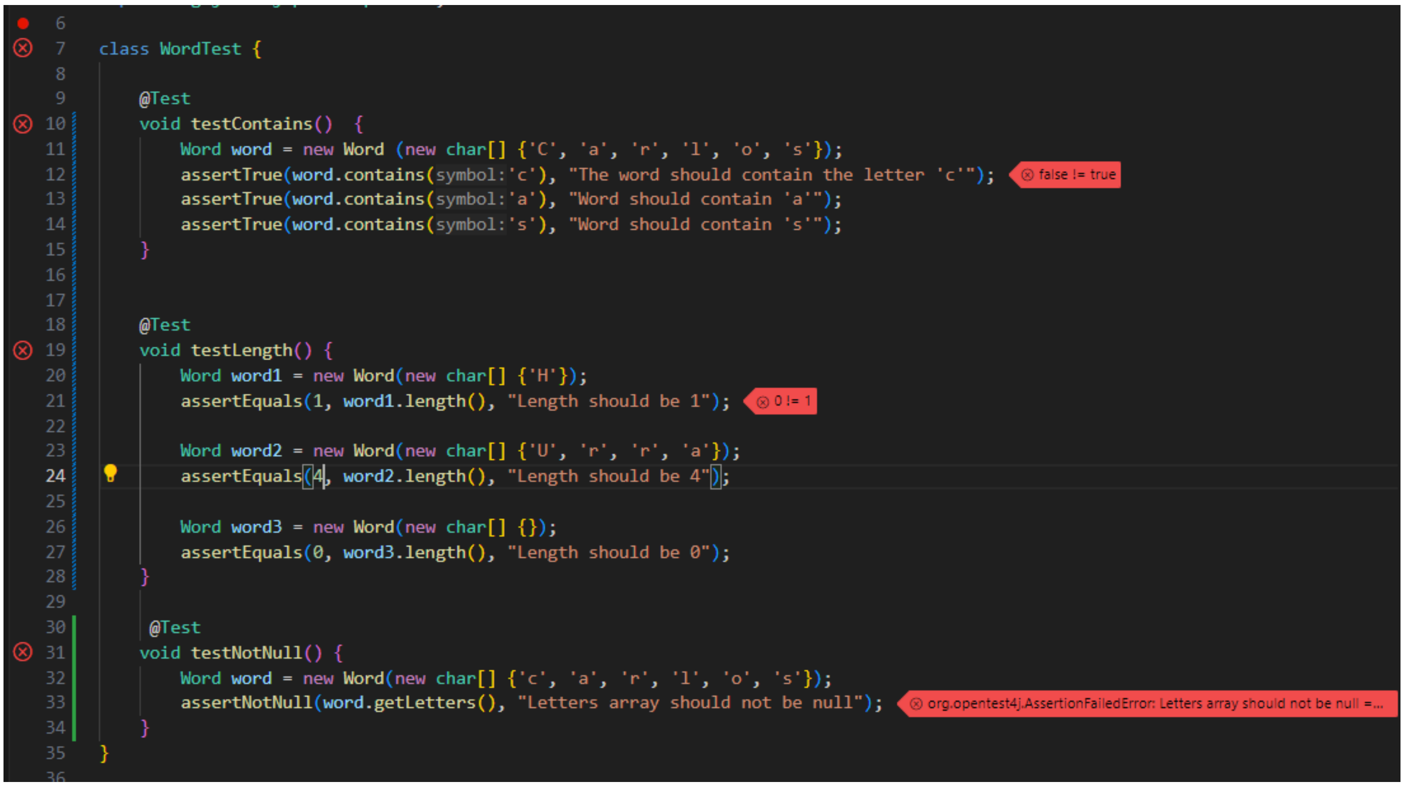


urra_carlos_3125350_ws1 > src > griffith > WordTest.java > ...

```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

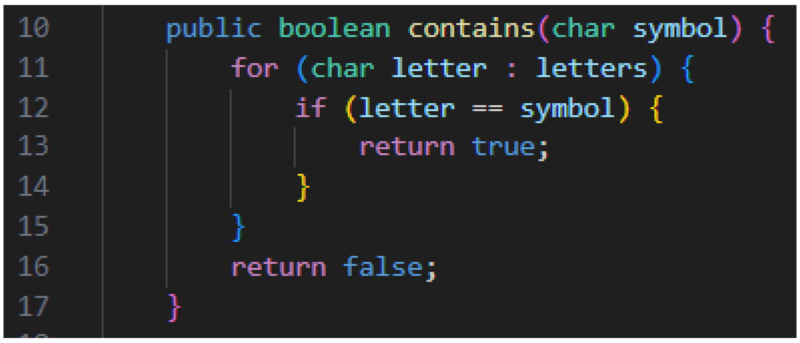
- Write test cases for each of the unit test methods that call the appropriate methods in the java file.



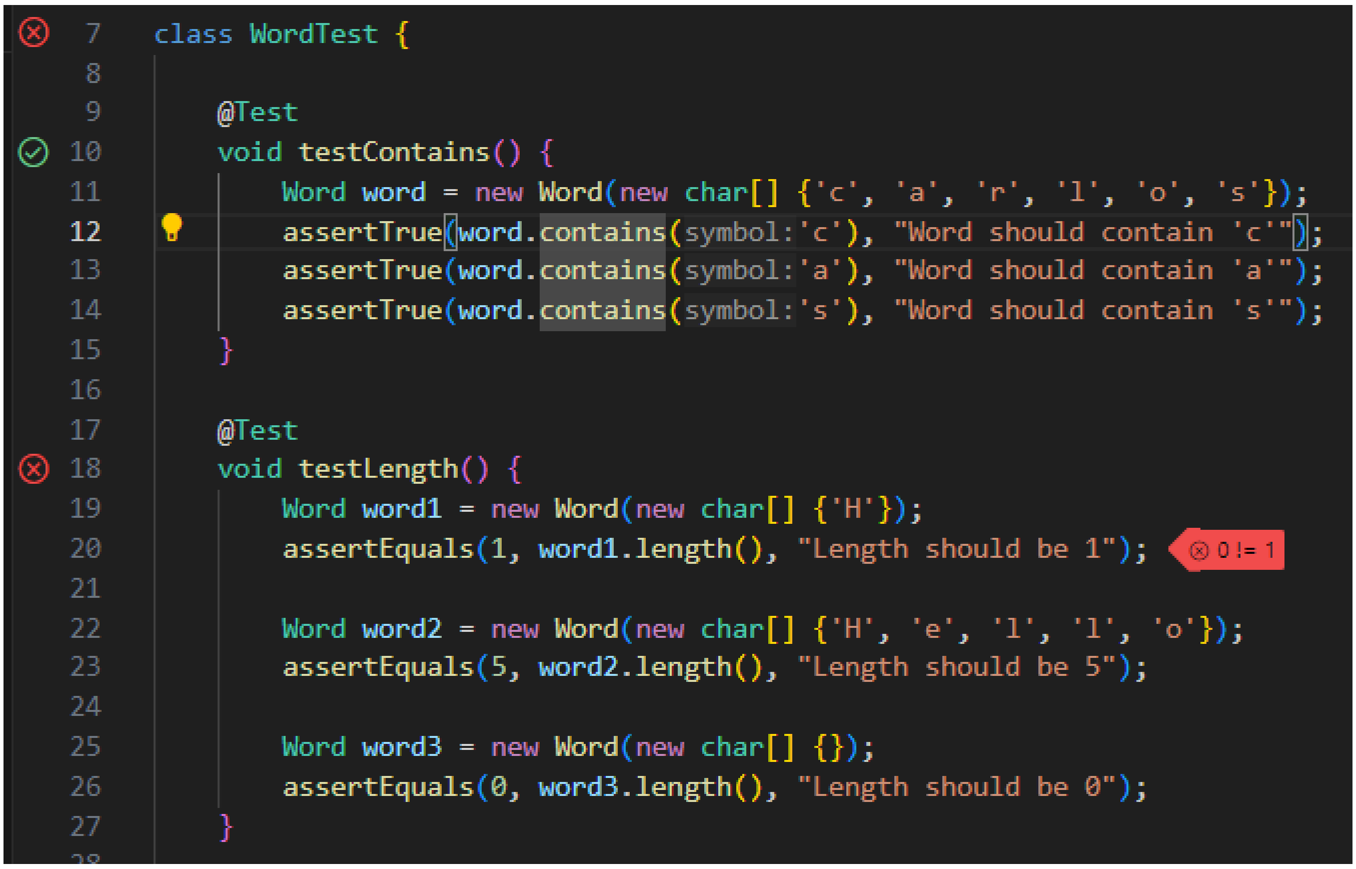
```java
class WordTest {

    @Test
    void testContains()  {
        Word word = new Word (new char[] {'C', 'a', 'r', 'l', 'o', 's'});
        assertTrue(word.contains(symbol:'c'), "The word should contain the letter 'c'");   ⊗ false != true
        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[] {'U', 'r', 'r', 'a'});
        assertEquals(4, word2.length(), "Length should be 4");

        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");   ⊗ org.opentest4j.AssertionFailedError: Letters array should not be null ==...
    }
}
```

Part 3

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

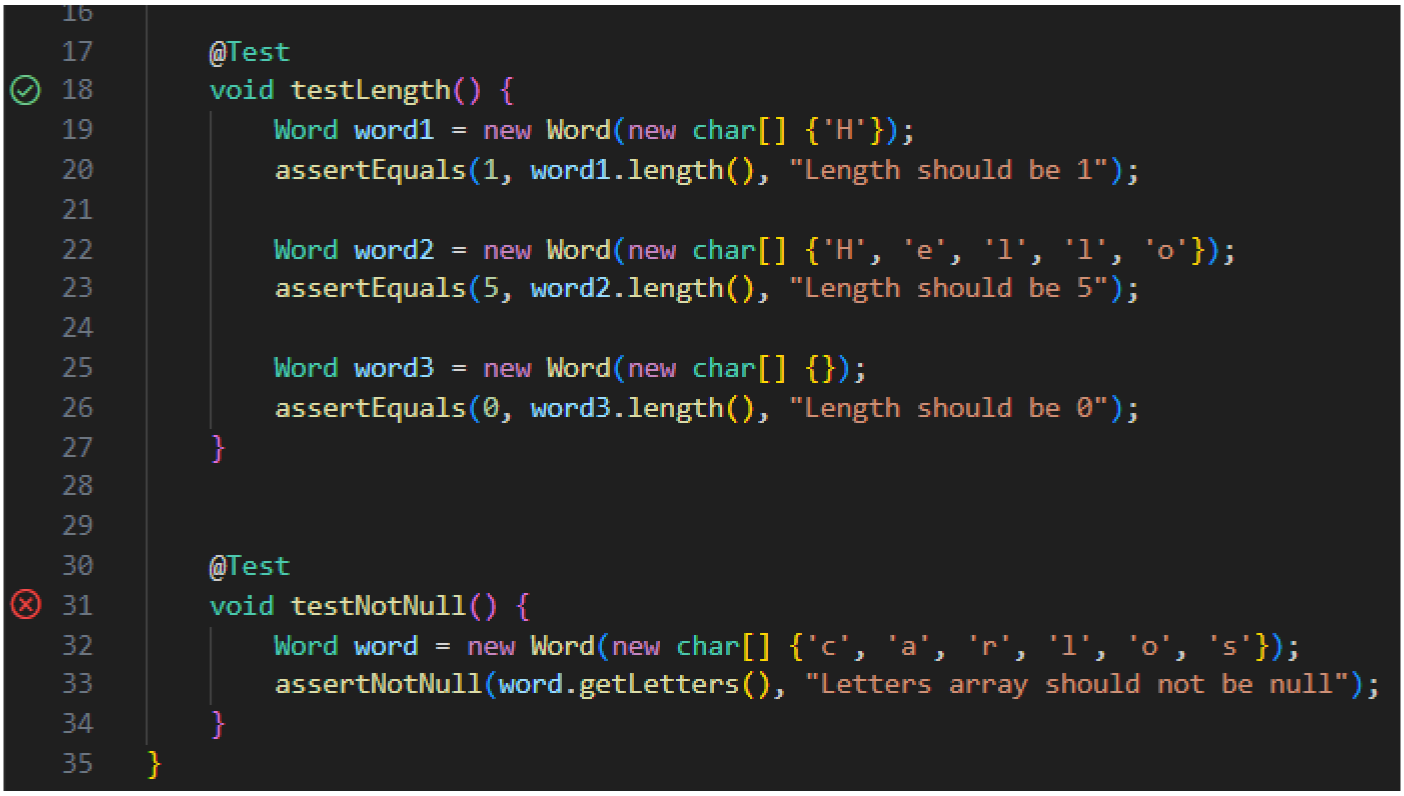


```java
    public boolean contains(char symbol) {
        for (char letter : letters) {
            if (letter == symbol) {
                return true;
            }
        }
        return false;
    }
```



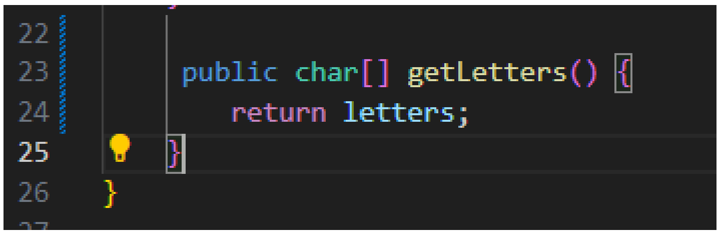
```java
class WordTest {

    @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()**

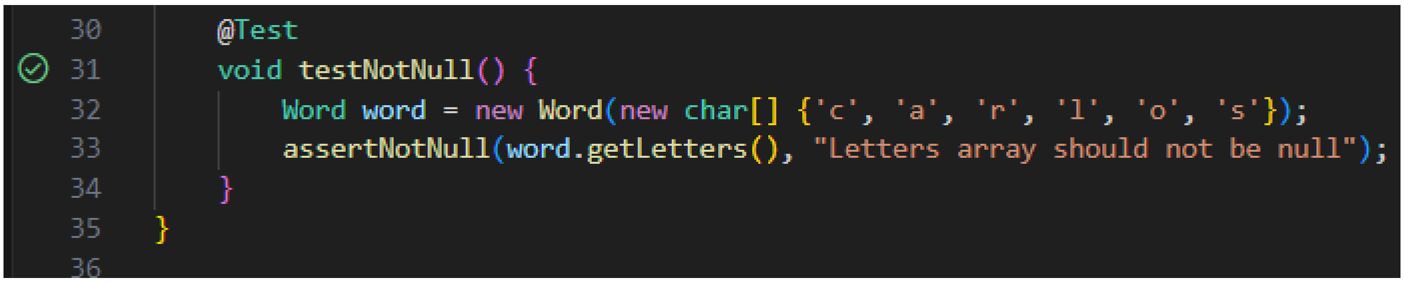


```java
    @Test
    void testLength() {
        Word word1 = new Word(new char[] {'H'});
        assertEquals(1, word1.length(), "Length should be 1");

        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()**



```
    public char[] getLetters() {
        return letters;
    }
}
```



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