CSC 212
JUnit Tutorial for Unit Testing
Due date: 11/10/2016

1 Introduction

JUnit is an open source testing framework which is used to write and run repeatable automated tests, so that we can be ensured that our code works as expected.

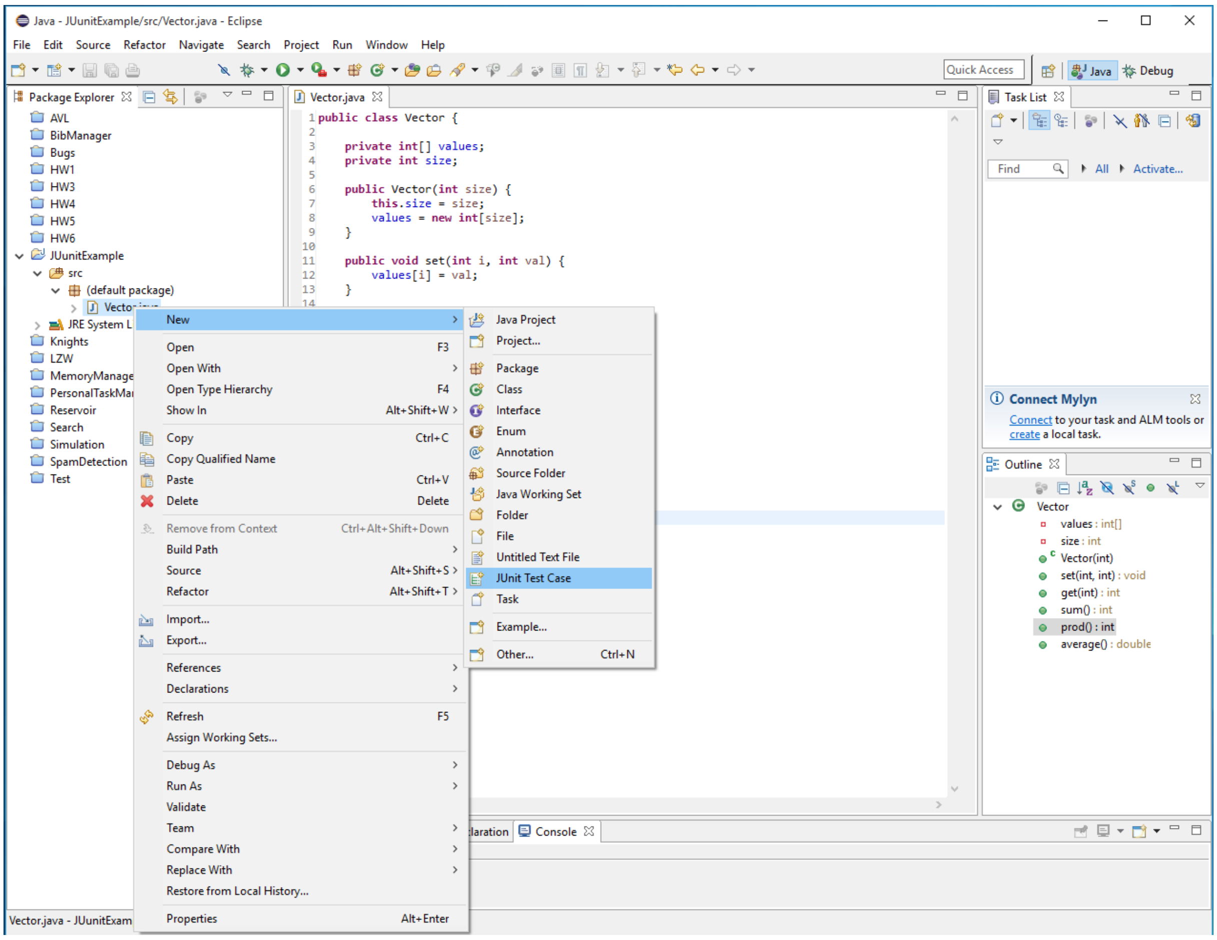
2 JUnit Simple Example using NetBeans

JUnit is a tool that allows to test units of code (for example single methods). You may use this framework to test your code as well.

Assume for example that you wrote the class Vector below and you want to check that the code is correct using JUnit.

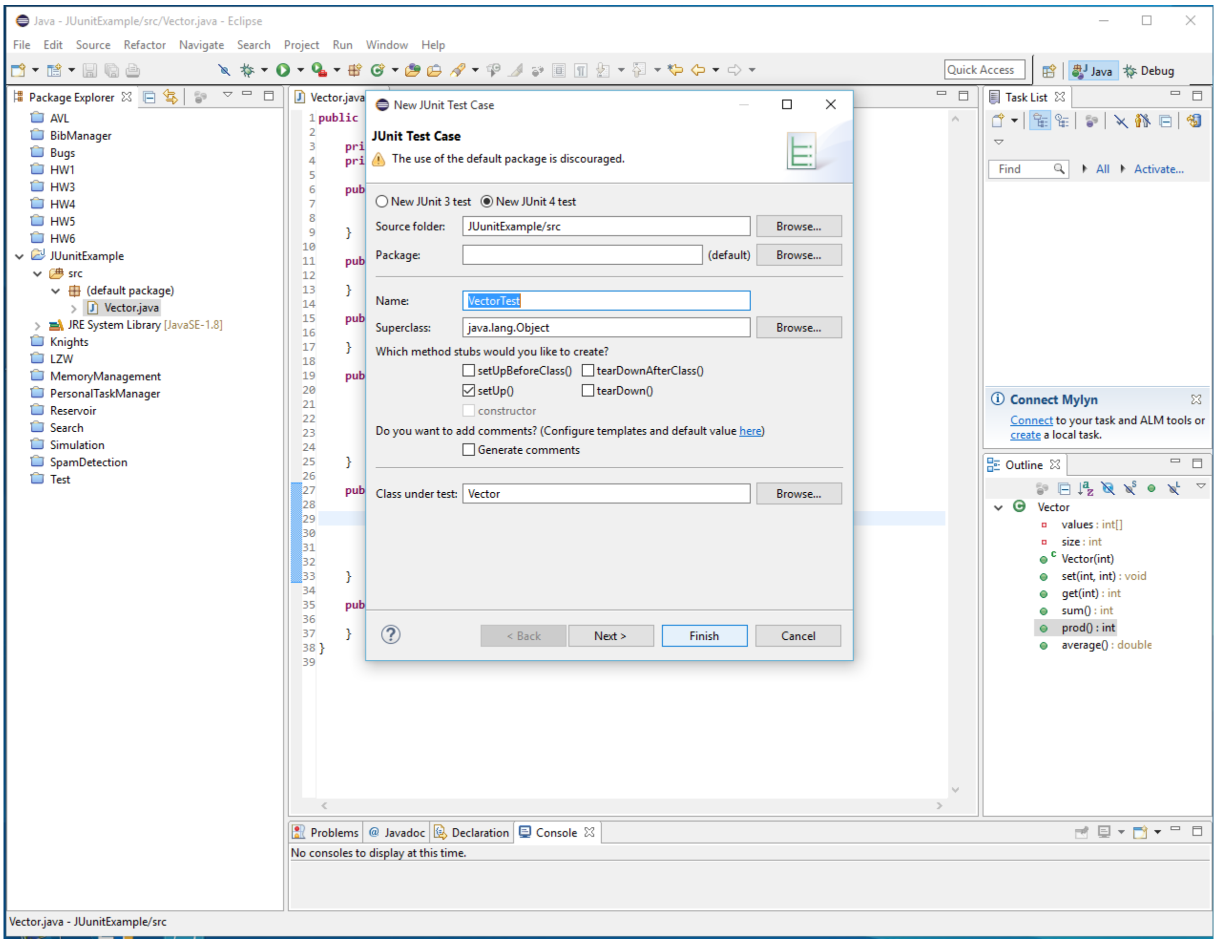
```java
public class Vector {

        private int[] values;
        private int size;

        public Vector(int size) {
                this.size = size;
                values = new int[size];
        }

        public void set(int i, int val) {
                values[i] = val;
        }

        public int get(int i) {
                return values[i];
        }

        public int sum() {
                int s = 0;
                for (int i = 1; i < size; i++) {
                        s += values[i];
                }
                return s;
        }

        public int prod() {
                int p = 1;
                for (int i = 0; i <= size; i++) {
                        p *= values[i];
                }
                return p;
        }

        public double average() {
                return (double) sum() / (size - 1);
        }
}
```

1. Right-click on the name of the class then choose New >JUnit Test Case.



```
public class Vector {

    private int[] values;
    private int size;

    public Vector(int size) {
        this.size = size;
        values = new int[size];
    }

    public void set(int i, int val) {
        values[i] = val;
    }
```

2. A window appears that allows you to set the properties of the test class. Keep the default options and click on "Finish".


New JUnit Test Case

JUnit Test Case

The use of the default package is discouraged.

New JUnit 3 test [✓ New JUnit 4 test]

Source folder: JUnitExample/src

Package: (default)

Name: VectorTest

Superclass: java.lang.Object

Which method stubs would you like to create?

- [ ] setUpBeforeClass()
- [ ] tearDownAfterClass()
- [✓] setUp()
- [ ] tearDown()
- [ ] constructor

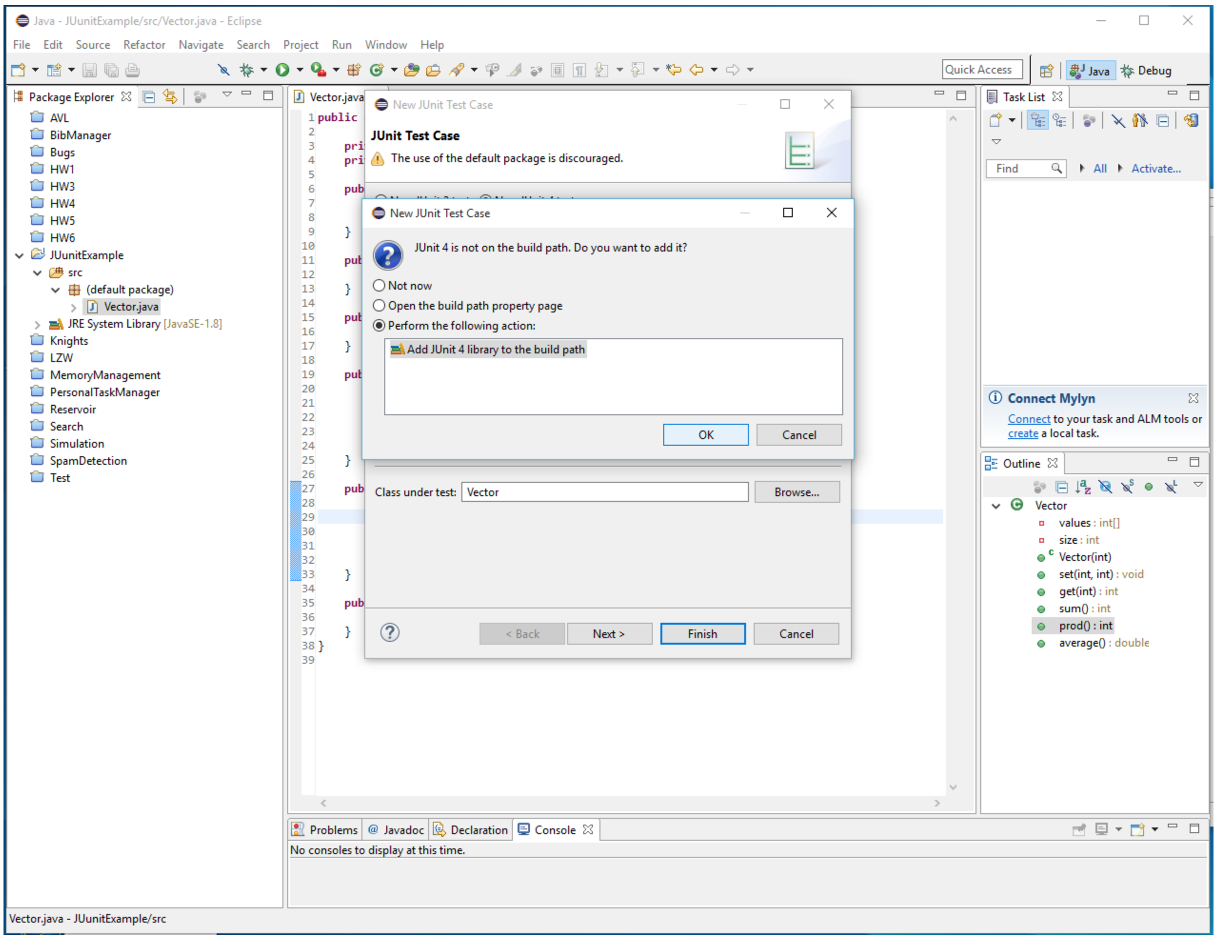
Do you want to add comments? (Configure templates and default value here)

- [ ] Generate comments

Class under test: Vector

Java - JUnitExample/src/Vector.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Vector.java
Task List
Outline
Browse...
< Back
Next >
Finish
Cancel
Problems
Javadoc
Declaration
Console

3. Eclipse will then ask you to add the JUnit library to the build path, click ”OK”.



New JUnit Test Case

JUnit Test Case

The use of the default package is discouraged.

New JUnit Test Case

JUnit 4 is not on the build path. Do you want to add it?

- Not now
- Open the build path property page
- [✓ Perform the following action:]

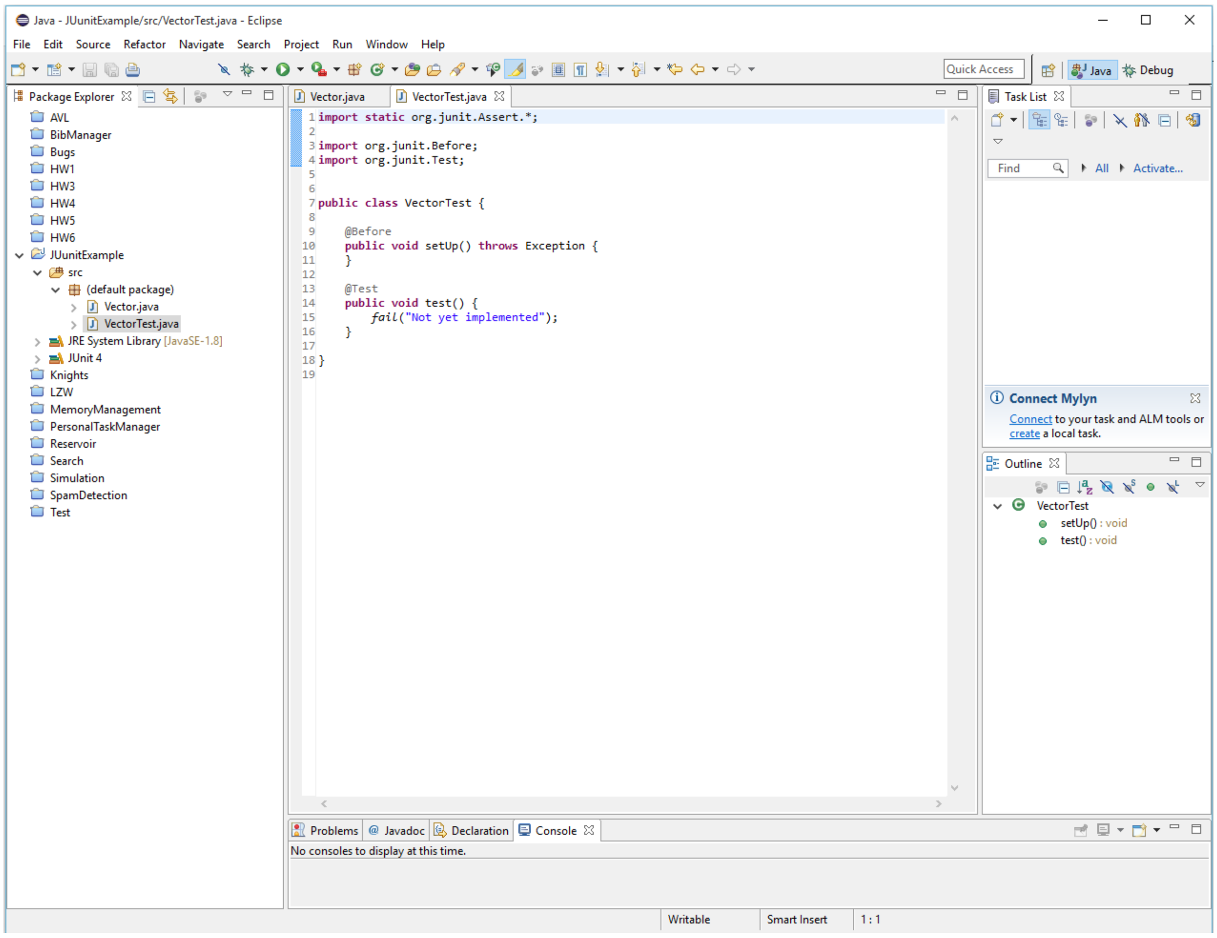
Add JUnit 4 library to the build path

Class under test: Vector

No consoles to display at this time.

Vector.java - JUnitExample/src

4. A skeleton of the test class VectorTest is then created:

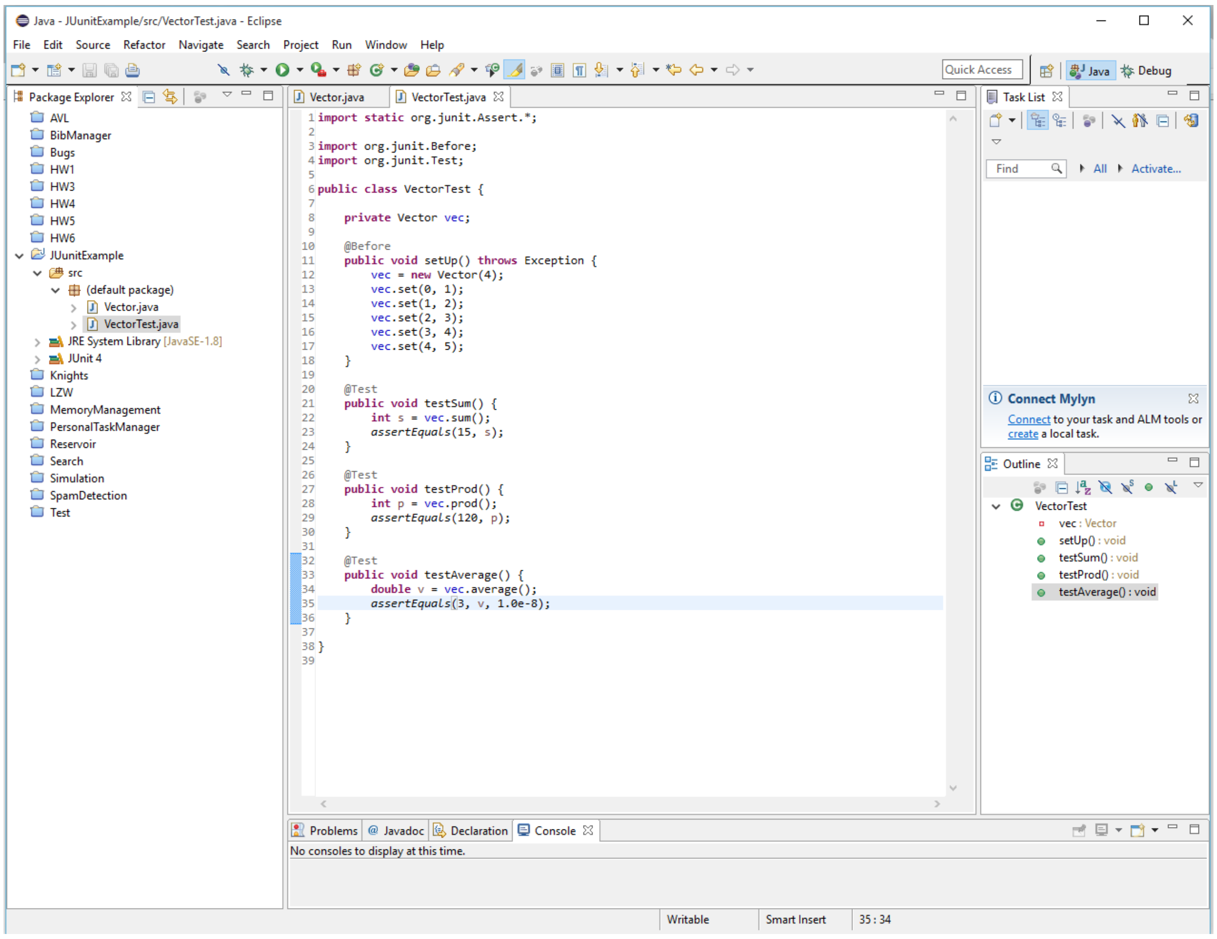


```
import static org.junit.Assert.*;

import org.junit.Before;
import org.junit.Test;


public class VectorTest {

    @Before
    public void setUp() throws Exception {
    }

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

}
```

No consoles to display at this time.

Add the following code to the class (do not forget the "@Test" before every test):

```
import static org.junit.Assert.*;

import org.junit.Before;
import org.junit.Test;

public class VectorTest {

        private Vector vec;

        @Before
        public void setUp() throws Exception {
                vec = new Vector(4);
                vec.set(0, 1);
                vec.set(1, 2);
                vec.set(2, 3);
                vec.set(3, 4);
                vec.set(4, 5);
        }

        @Test
        public void testSum() {
                int s = vec.sum();
                assertEquals(15, s);
        }

        @Test
        public void testProd() {
                int p = vec.prod();
                assertEquals(120, p);
        }

        @Test
        public void testAverage() {
                double v = vec.average();
                assertEquals(3, v, 1.0e-8);
        }

}
```



```
import static org.junit.Assert.*;

import org.junit.Before;
import org.junit.Test;

public class VectorTest {

    private Vector vec;

    @Before
    public void setUp() throws Exception {
        vec = new Vector(4);
        vec.set(0, 1);
        vec.set(1, 2);
        vec.set(2, 3);
        vec.set(3, 4);
        vec.set(4, 5);
    }

    @Test
    public void testSum() {
        int s = vec.sum();
        assertEquals(15, s);
    }

    @Test
    public void testProd() {
        int p = vec.prod();
        assertEquals(120, p);
    }

    @Test
    public void testAverage() {
        double v = vec.average();
        assertEquals(3, v, 1.0e-8);
    }

}
```

No consoles to display at this time.

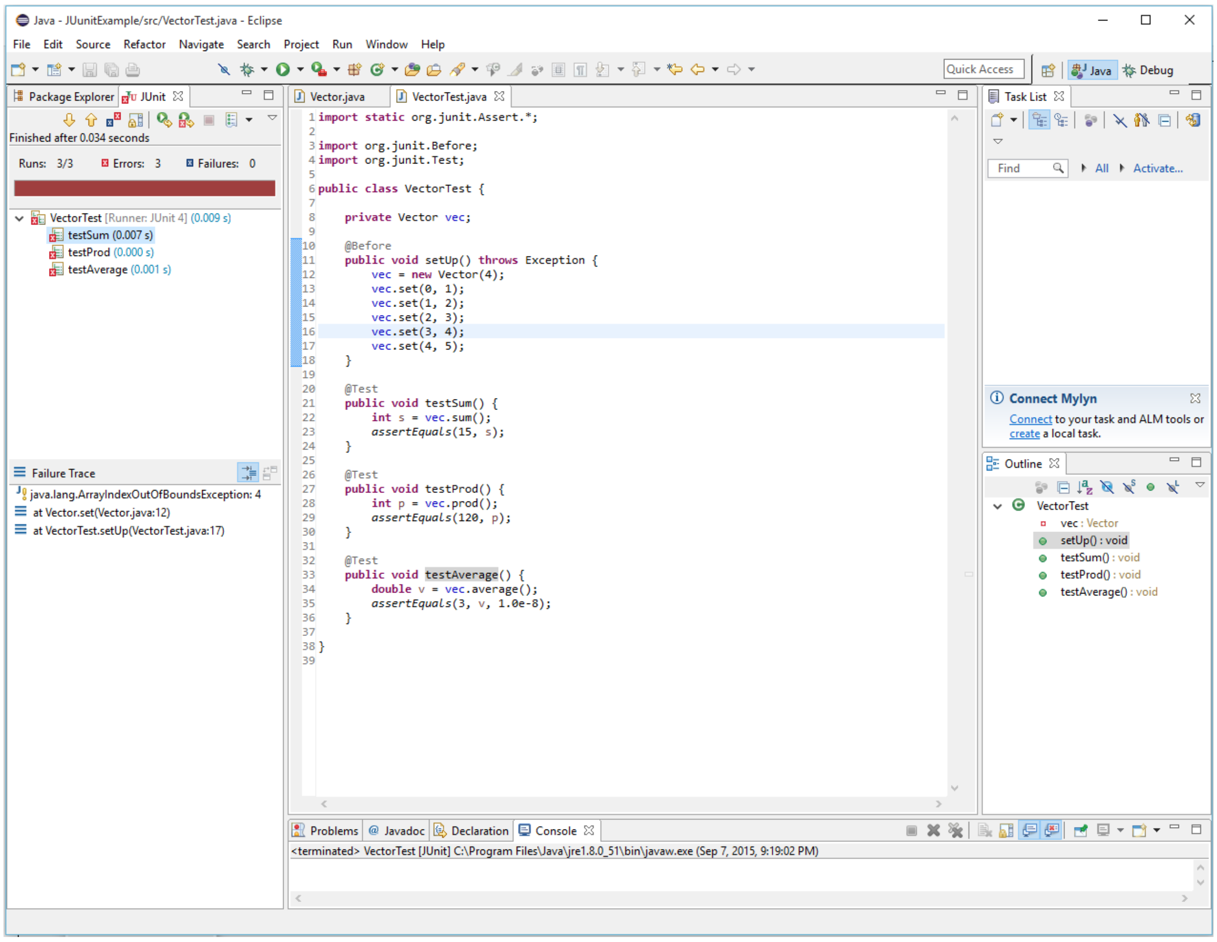
The test cases (three in this case: *testSum, testProd, testAverage*) are executed one after the other, but before running each test, the method *setUp* is executed first. The method *assertEquals* is used to compare the expected value with the one returned by a call to the method to be tested. For example, the sum of the array should be 15, so we compare 15 (the expected value) to the value returned by *Vector.sum()* (the actual value). If the two values are equal, the test is passed, otherwise the test fails.

For values of type double, a third parameter is needed, which is the tolerance ϵ. The test is passed if $|expected - actual| < \epsilon$.

5. You may run the test by clicking run (you do not need a main function to run the test cases!). The results will appear in the left panel:

   - **Runs**: gives you the number of tests that have been run.
   - **Errors**: number of tests that were not successful because of exceptions (index out of bound, null pointer exception etc.).
   - **Failures**: number of tests where the test assertion has failed (the expected value is different from the actual value).

   Notice that we have three errors in this case, which means that all tests were interrupted by an exception.

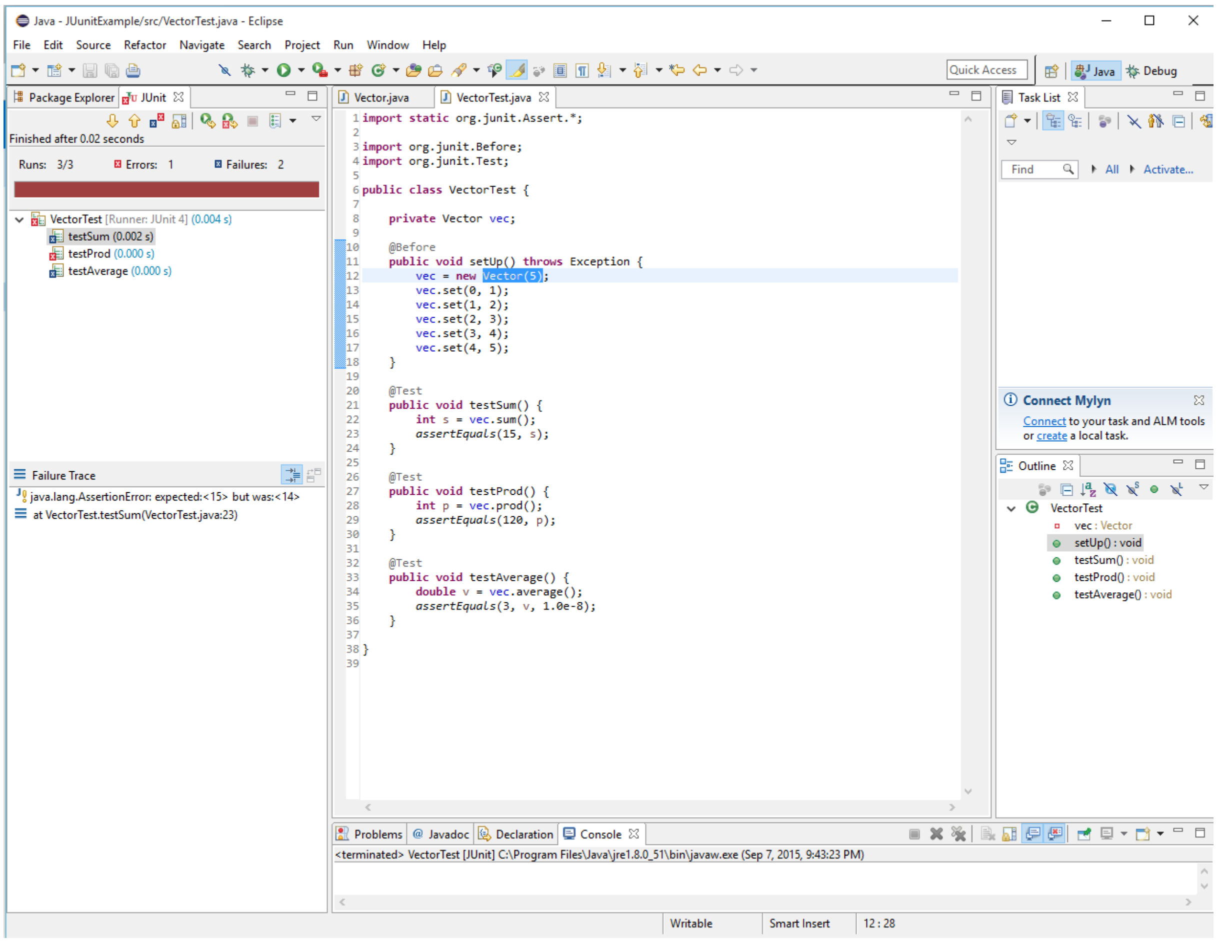


```
Finished after 0.034 seconds
Runs: 3/3     Errors: 3     Failures: 0

VectorTest [Runner: JUnit 4] (0.009 s)
    testSum (0.007 s)
    testProd (0.000 s)
    testAverage (0.001 s)

Failure Trace
java.lang.ArrayIndexOutOfBoundsException: 4
at Vector.set(Vector.java:12)
at VectorTest.setUp(VectorTest.java:17)
```

```java
import static org.junit.Assert.*;

import org.junit.Before;
import org.junit.Test;

public class VectorTest {

    private Vector vec;

    @Before
    public void setUp() throws Exception {
        vec = new Vector(4);
        vec.set(0, 1);
        vec.set(1, 2);
        vec.set(2, 3);
        vec.set(3, 4);
        vec.set(4, 5);
    }

    @Test
    public void testSum() {
        int s = vec.sum();
        assertEquals(15, s);
    }

    @Test
    public void testProd() {
        int p = vec.prod();
        assertEquals(120, p);
    }

    @Test
    public void testAverage() {
        double v = vec.average();
        assertEquals(3, v, 1.0e-8);
    }

}
```

```
<terminated> VectorTest [JUnit] C:\Program Files\Java\jre1.8.0_51\bin\javaw.exe (Sep 7, 2015, 9:19:02 PM)
```

6. The reason for this is an error in the test class and not the class Vector. The size of the vector is wrong. In the method *setUp* change the size of the vector to 5 instead of 4 and run the test again. Now we have two failures and one error.


Java - JUunitExample/src/VectorTest.java - Eclipse

```
Finished after 0.02 seconds

Runs:  3/3      Errors:  1      Failures:  2

VectorTest [Runner: JUnit 4] (0.004 s)
    testSum (0.002 s)
    testProd (0.000 s)
    testAverage (0.000 s)
```

Failure Trace

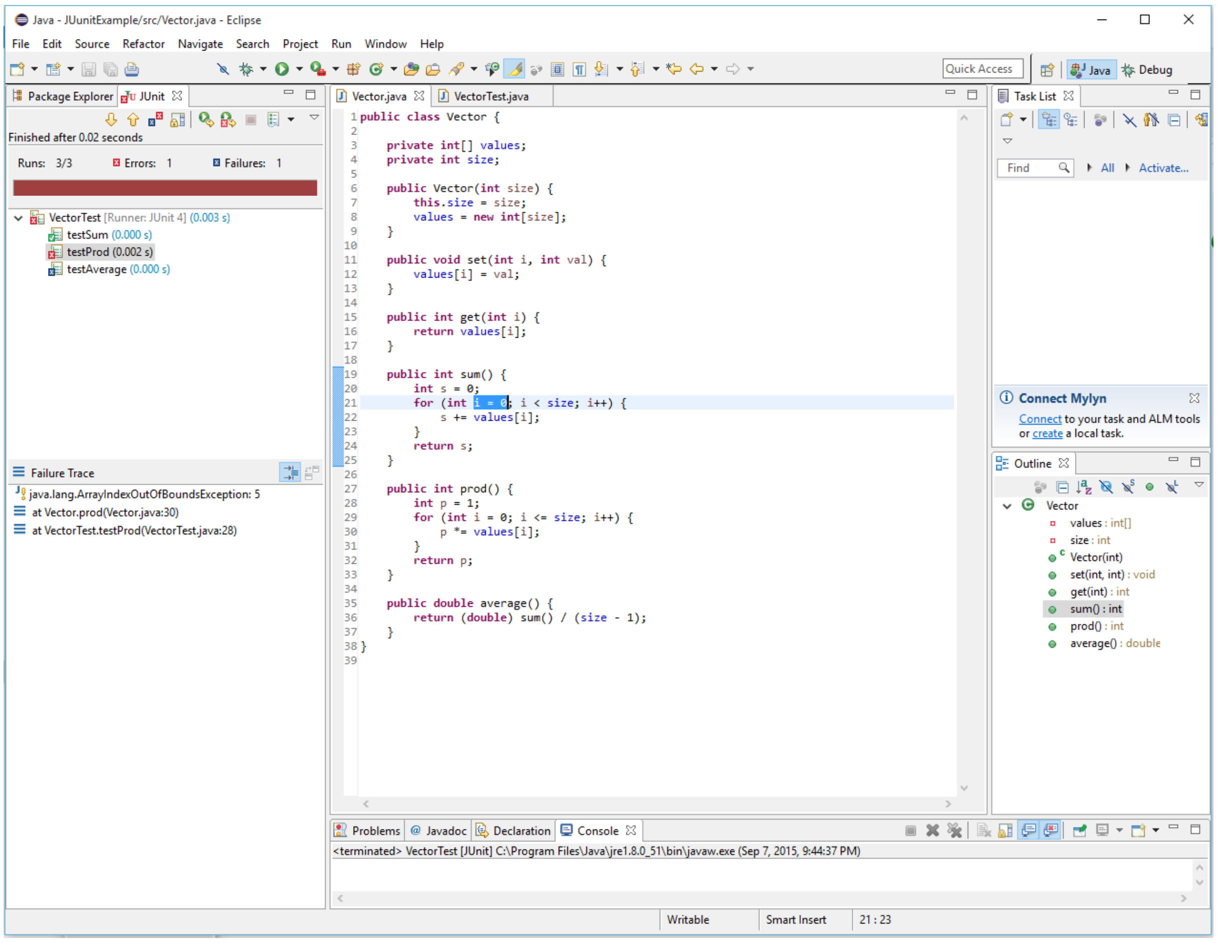
```
java.lang.AssertionError: expected:<15> but was:<14>
at VectorTest.testSum(VectorTest.java:23)
```

VectorTest.java

```java
import static org.junit.Assert.*;

import org.junit.Before;
import org.junit.Test;

public class VectorTest {

    private Vector vec;

    @Before
    public void setUp() throws Exception {
        vec = new Vector(5);
        vec.set(0, 1);
        vec.set(1, 2);
        vec.set(2, 3);
        vec.set(3, 4);
        vec.set(4, 5);
    }

    @Test
    public void testSum() {
        int s = vec.sum();
        assertEquals(15, s);
    }

    @Test
    public void testProd() {
        int p = vec.prod();
        assertEquals(120, p);
    }

    @Test
    public void testAverage() {
        double v = vec.average();
        assertEquals(3, v, 1.0e-8);
    }

}
```

```
<terminated> VectorTest [JUnit] C:\Program Files\Java\jre1.8.0_51\bin\javaw.exe (Sep 7, 2015, 9:43:23 PM)
```

7. The test *testSum* has a failure. If you click on it, you will see that the actual value is 14, whereas the expected is 15. The error in the code is that we are starting to sum from index 1 instead of 0. Correct this and run again. Now, *testSum* is successful.



Java - JUnitExample/src/Vector.java - Eclipse

Finished after 0.02 seconds

Runs: 3/3 Errors: 1 Failures: 1

```
VectorTest [Runner: JUnit 4] (0.003 s)
    testSum (0.000 s)
    testProd (0.002 s)
    testAverage (0.000 s)
```

Failure Trace

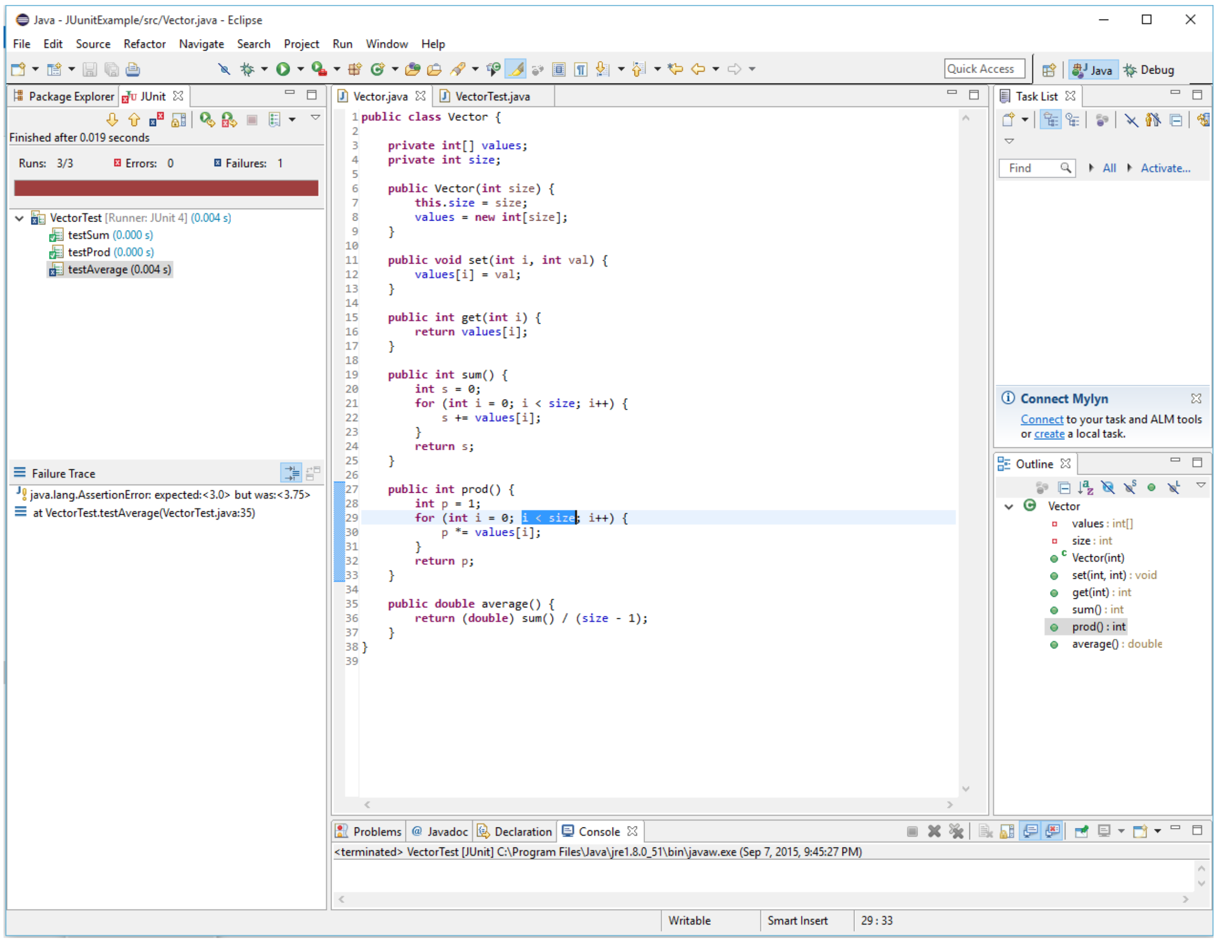
```
java.lang.ArrayIndexOutOfBoundsException: 5
at Vector.prod(Vector.java:30)
at VectorTest.testProd(VectorTest.java:28)
```

```java
public class Vector {

    private int[] values;
    private int size;

    public Vector(int size) {
        this.size = size;
        values = new int[size];
    }

    public void set(int i, int val) {
        values[i] = val;
    }

    public int get(int i) {
        return values[i];
    }

    public int sum() {
        int s = 0;
        for (int i = 0; i < size; i++) {
            s += values[i];
        }
        return s;
    }

    public int prod() {
        int p = 1;
        for (int i = 0; i <= size; i++) {
            p *= values[i];
        }
        return p;
    }

    public double average() {
        return (double) sum() / (size - 1);
    }
}
```

```
<terminated> VectorTest [JUnit] C:\Program Files\Java\jre1.8.0_51\bin\javaw.exe (Sep 7, 2015, 9:44:37 PM)
```

Writable Smart Insert 21 : 23

8. In *testProd*, we have an error due to an index out of bound exception. Changing the condition in the for loop to $i < size$ solves the problem.



Finished after 0.019 seconds

Runs: 3/3   Errors: 0   Failures: 1

```
VectorTest [Runner: JUnit 4] (0.004 s)
    testSum (0.000 s)
    testProd (0.000 s)
    testAverage (0.004 s)
```

Failure Trace

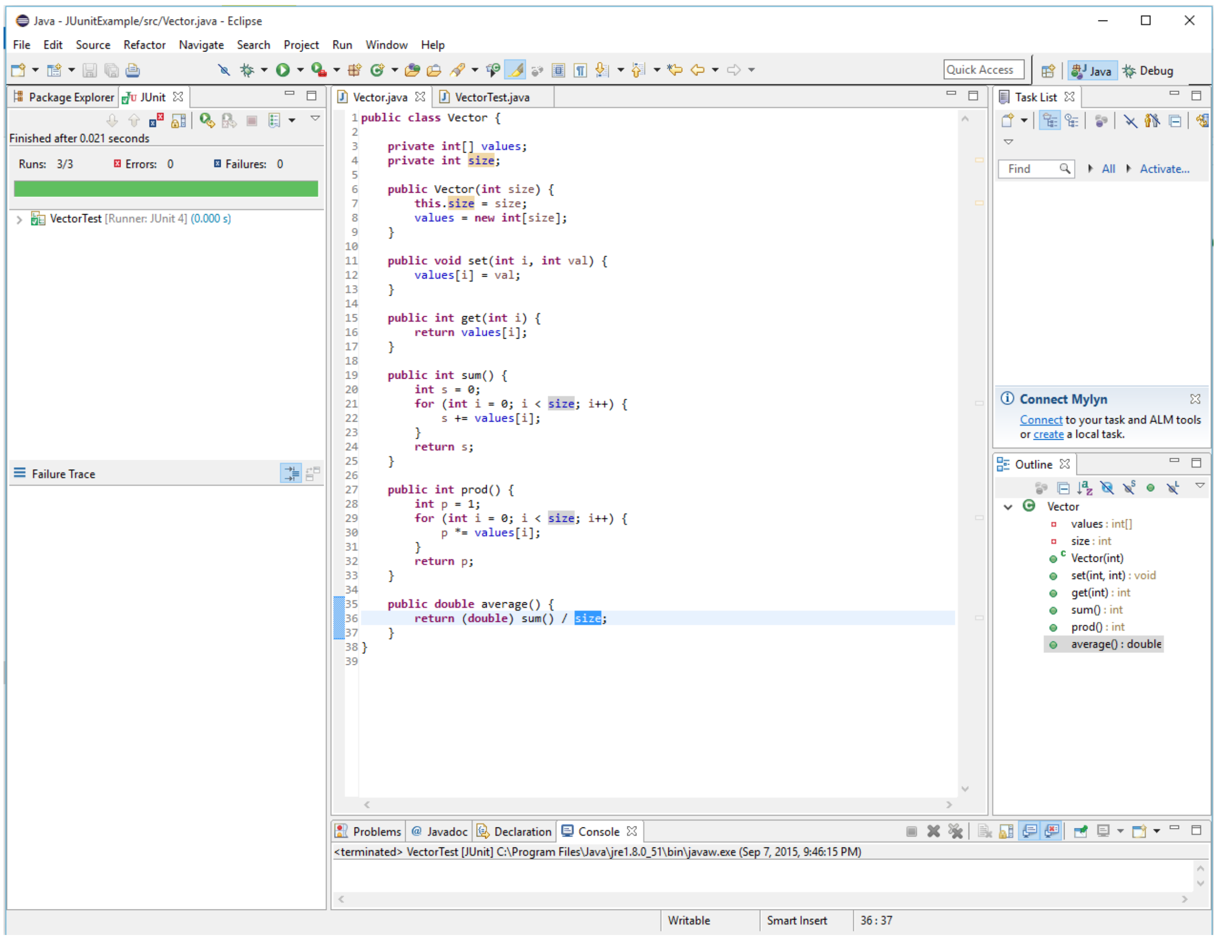
```
java.lang.AssertionError: expected:<3.0> but was:<3.75>
at VectorTest.testAverage(VectorTest.java:35)
```

```java
public class Vector {

    private int[] values;
    private int size;

    public Vector(int size) {
        this.size = size;
        values = new int[size];
    }

    public void set(int i, int val) {
        values[i] = val;
    }

    public int get(int i) {
        return values[i];
    }

    public int sum() {
        int s = 0;
        for (int i = 0; i < size; i++) {
            s += values[i];
        }
        return s;
    }

    public int prod() {
        int p = 1;
        for (int i = 0; i < size; i++) {
            p *= values[i];
        }
        return p;
    }

    public double average() {
        return (double) sum() / (size - 1);
    }
}
```

```
<terminated> VectorTest [JUnit] C:\Program Files\Java\jre1.8.0_51\bin\javaw.exe (Sep 7, 2015, 9:45:27 PM)
```

```
Vector
    values : int[]
    size : int
    Vector(int)
    set(int, int) : void
    get(int) : int
    sum() : int
    prod() : int
    average() : double
```

9. In the last test, *testAverage*, the expected value is 3, but the actual value is 3.75. The reason for this is that we are dividing by (*size-1*) instead of *size*. After correcting this, all tests are passed:


Java - JUunitExample/src/Vector.java - Eclipse

Finished after 0.021 seconds

Runs: 3/3 Errors: 0 Failures: 0

VectorTest [Runner: JUnit 4] (0.000 s)

```
public class Vector {

    private int[] values;
    private int size;

    public Vector(int size) {
        this.size = size;
        values = new int[size];
    }

    public void set(int i, int val) {
        values[i] = val;
    }

    public int get(int i) {
        return values[i];
    }

    public int sum() {
        int s = 0;
        for (int i = 0; i < size; i++) {
            s += values[i];
        }
        return s;
    }

    public int prod() {
        int p = 1;
        for (int i = 0; i < size; i++) {
            p *= values[i];
        }
        return p;
    }

    public double average() {
        return (double) sum() / size;
    }
}
```

Outline:

```
Vector
    values : int[]
    size : int
    Vector(int)
    set(int, int) : void
    get(int) : int
    sum() : int
    prod() : int
    average() : double
```

```
<terminated> VectorTest [JUnit] C:\Program Files\Java\jre1.8.0_51\bin\javaw.exe (Sep 7, 2015, 9:46:15 PM)
```

Writable | Smart Insert | 36 : 37

The final (and correct) code for the class Vector and the test class VectorTest are as follows.

```
public class Vector {

        private int[] values;
        private int size;

        public Vector(int size) {
                this.size = size;
                values = new int[size];
        }

        public void set(int i, int val) {
                values[i] = val;
        }

        public int get(int i) {
                return values[i];
        }

        public int sum() {
                int s = 0;
                for (int i = 0; i < size; i++) {
                        s += values[i];
                }
                return s;
        }

        public int prod() {
                int p = 1;
                for (int i = 0; i < size; i++) {
                        p *= values[i];
                }
                return p;
        }

        public double average() {
                return (double) sum() / size;
        }
}
```

```
import static org.junit.Assert.*;

import org.junit.Before;
import org.junit.Test;

public class VectorTest {

        private Vector vec;

        @Before
        public void setUp() throws Exception {
                vec = new Vector(5);
                vec.set(0, 1);
                vec.set(1, 2);
                vec.set(2, 3);
                vec.set(3, 4);
                vec.set(4, 5);
        }

        @Test
        public void testSum() {
                int s = vec.sum();
                assertEquals(15, s);
        }

        @Test
        public void testProd() {
                int p = vec.prod();
                assertEquals(120, p);
        }

        @Test
        public void testAverage() {
                double v = vec.average();
                assertEquals(3, v, 1.0e-8);
        }

}
```

3 Exercise

Add to class vector method *int count(int m)* that counts the number of occurrences of m in vector v. Write a JUnit test cases for this method and check that your code is correct using JUnit.
-Submit Vector.java and VectorTest-2.java and the test results as zip file to LMS