

**ECSE 428 Assignment B**  
**Amees Joshipura (260461226)**  
**Lilly Tong (260459522)**

**JUnit Test Cases & Screen shots**

## Test 1

**Test name:** cToP\_returns\_double\_array

**Call setup:** cartesianToPolar(3,4)

**Expected:** cartesianToPolar() returns a double array

## Fail:

The screenshot shows an IDE with two tabs: `Calculator.java` and `CalculatorTests.java`. The `Calculator.java` file contains the following code:

```
1 package util;
2
3 public class Calculator {
4
5     public static double[] cartesianToPolar(double x, double y)
6     {
7
8     }
9 }
10
```

The `CalculatorTests.java` file is not visible. The `JUnit` runner is shown at the bottom, indicating the test `cToP_returns_double_array` failed. The failure message is:

```
java.lang.Error: Unresolved compilation problem:
This method must return a result of type double[]

at util.Calculator.cartesianToPolar(Calculator.java:5)
at tests.CalculatorTests.cToP_returns_double_array(CalculatorTests.java:13)
```

The `Outline` pane on the right shows the project structure with `util` and `Calculator` classes.

## Succeed:

The screenshot shows the same IDE with the `Calculator.java` file updated to include a return statement:

```
1 package util;
2
3 public class Calculator {
4
5     public static double[] cartesianToPolar(double x, double y)
6     {
7         double[] result = new double[2];
8         return result;
9     }
10 }
```

The `JUnit` runner is shown at the bottom, indicating the test `cToP_returns_double_array` passed. The `Outline` pane on the right shows the project structure with `util` and `Calculator` classes.

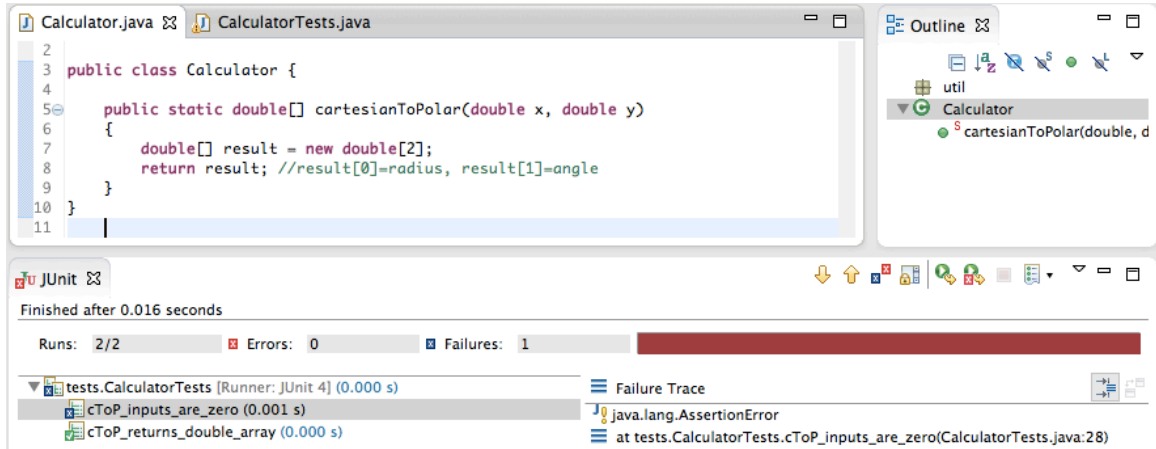
## Test 2

**Test name:** cToP\_inputs\_are\_zero

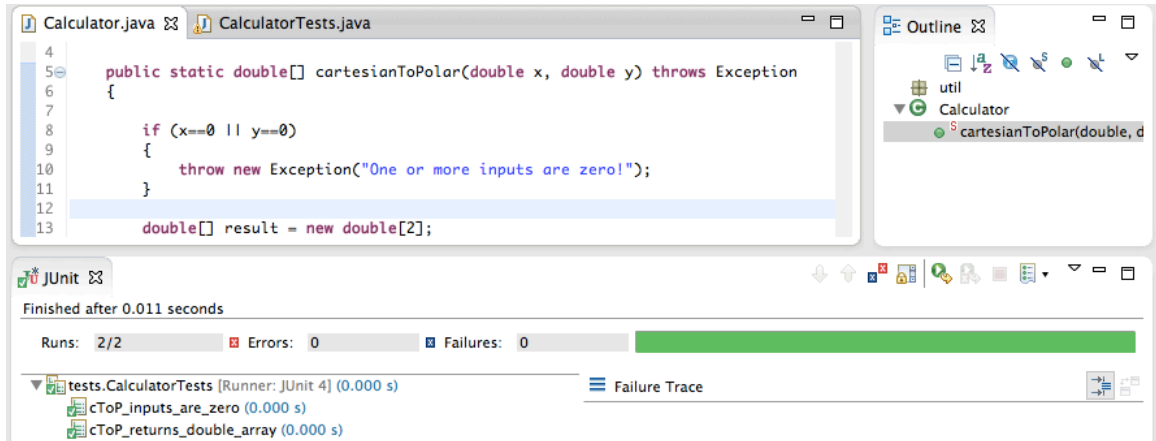
**Call setup:** cartesianToPolar(0,0)

**Expected:** cartesianToPolar() throws exception with message "One or more inputs are zero!"

## Fail:



## Succeed:



### Test 3

**Test name:** cToP\_inputs\_are\_nonzero

**Call setup:** cartesianToPolar(3,4)

**Expected:** cartesianToPolar() returns [5, arctan(4.0/3.0)]

**Fail:**

The screenshot shows an IDE with two tabs: Calculator.java and CalculatorTests.java. The CalculatorTests.java tab is active, showing the following code:

```
5 public static double[] cartesianToPolar(double x, double y) throws Exception
6 {
7
8     if (x==0 || y==0)
9     {
10         throw new Exception("One or more inputs are zero!");
11     }
12
13     double[] result = new double[2];
14     return result; //result[0]=radius, result[1]=angle
15 }
16 }
```

The Outline view on the right shows the project structure with 'util' and 'Calculator' packages. The 'Calculator' package contains the 'cartesianToPolar(double, d)' method.

The JUnit test runner at the bottom shows the results of the test run. It indicates that the test finished after 0.023 seconds. The summary shows 3 runs, 0 errors, and 1 failure. The failure trace for the 'cToP\_inputs\_are\_nonzero' test is as follows:

```
Failure Trace
java.lang.AssertionError
at tests.CalculatorTests.cToP_inputs_are_nonzero(CalculatorTests.java:43)
```

**Succeed:**

The screenshot shows the same IDE as before, but now the CalculatorTests.java tab is active, showing the following code:

```
4
5 public static double[] cartesianToPolar(double x, double y) throws Exception
6 {
7
8     if (x==0 || y==0)
9     {
10         throw new Exception("One or more inputs are zero!");
11     }
12
13     double[] result = new double[2];
14     result[0]=Math.sqrt(Math.pow(x, 2)+Math.pow(y, 2));
15     result[1]=Math.atan(y/x);
16     return result; //result[0]=radius, result[1]=angle
```

The Outline view on the right shows the project structure with 'util' and 'Calculator' packages. The 'Calculator' package contains the 'cartesianToPolar(double, d)' method.

The JUnit test runner at the bottom shows the results of the test run. It indicates that the test finished after 0.013 seconds. The summary shows 3 runs, 0 errors, and 0 failures. The test results are as follows:

```
tests.CalculatorTests [Runner: JUnit 4] (0.001 s)
  cToP_inputs_are_zero (0.001 s)
  cToP_inputs_are_nonzero (0.000 s)
  cToP_returns_double_array (0.000 s)
```

## Test 4

**Test name:** cToP\_contains\_negative\_input

**Call setup:** cartesianToPolar(3,-4)

**Expected:** cartesianToPolar() returns [5, arctan(-4.0/3.0)]

**Succeed:**

The screenshot shows an IDE with two tabs: Calculator.java and CalculatorTests.java. The Calculator.java file contains the following code:

```
1 package util;
2
3 public class Calculator {
4
5     public static double[] cartesianToPolar(double x, double y) throws Exception
6     {
7
8         if (x==0 || y==0)
9         {
10             throw new Exception("One or more inputs are zero!");
11         }
12
13         double[] result = new double[2];
14         result[0]=Math.sqrt(Math.pow(x, 2)+Math.pow(y, 2));
15         result[1]=Math.atan(y/x);
16         return result; //result[0]=radius, result[1]=angle
17     }
18 }
19
```

The CalculatorTests.java file contains the following code:

```
1 import static org.junit.Assert.*;
2
3 public class CalculatorTests {
4
5     @Test
6     public void cToP_inputs_are_zero() {
7         try {
8             Calculator.cartesianToPolar(0, 0);
9             fail("Expected an exception");
10         } catch (Exception e) {
11             // Expected
12         }
13     }
14
15     @Test
16     public void cToP_inputs_are_nonzero() {
17         double[] result = Calculator.cartesianToPolar(3, 4);
18         assertEquals("radius", 5, result[0], 0.0001);
19         assertEquals("angle", Math.atan(4/3), result[1], 0.0001);
20     }
21
22     @Test
23     public void cToP_returns_double_array() {
24         double[] result = Calculator.cartesianToPolar(3, 4);
25         assertEquals("array length", 2, result.length);
26     }
27
28     @Test
29     public void cToP_contains_negative_input() {
30         double[] result = Calculator.cartesianToPolar(3, -4);
31         assertEquals("radius", 5, result[0], 0.0001);
32         assertEquals("angle", Math.atan(-4/3), result[1], 0.0001);
33     }
34 }

```

The IDE also shows the Outline view on the right, which lists the classes and methods in the project. The JUnit test results are shown at the bottom, indicating that all tests passed.

JUnit 4

Finished after 0.016 seconds

Runs: 4/4 Errors: 0 Failures: 0

tests.CalculatorTests [Runner: JUnit 4] (0.001 s)

- cToP\_inputs\_are\_zero (0.001 s)
- cToP\_inputs\_are\_nonzero (0.000 s)
- cToP\_returns\_double\_array (0.000 s)
- cToP\_contains\_negative\_input (0.000 s)

Failure Trace

## Test 5

**Test name:** cToP\_inputs\_too\_large

**Call setup:** cartesianToPolar(10000,-10000)

**Expected:** cartesianToPolar() throws exception with message "Inputs too large!"

**Fail:**

The screenshot shows an IDE with two tabs: `Calculator.java` and `CalculatorTests.java`. The `Calculator.java` file contains the following code:

```
1 package util;
2
3 public class Calculator {
4
5     public static double[] cartesianToPolar(double x, double y) throws Exception
6     {
7
8         if (x==0 || y==0)
9         {
10             throw new Exception("One or more inputs are zero!");
11         }
12
13         double[] result = new double[2];
14         result[0]=Math.sqrt(Math.pow(x, 2)+Math.pow(y, 2));
15         result[1]=Math.atan(y/x);
16         return result; //result[0]=radius, result[1]=angle
17     }
18 }
```

The `CalculatorTests.java` file is not visible. The `JUnit` runner shows the test results:

- Finished after 0.024 seconds
- Runs: 5/5
- Errors: 0
- Failures: 1

The test results list shows the following tests:

- cToP\_inputs\_are\_zero (0.000 s)
- cToP\_inputs\_are\_nonzero (0.000 s)
- cToP\_inputs\_too\_large (0.000 s)
- cToP\_returns\_double\_array (0.000 s)
- cToP\_contains\_negative\_input (0.000 s)

The `cToP_inputs_too_large` test is highlighted in red, indicating it failed. The failure trace shows:

```
java.lang.AssertionError
    at tests.CalculatorTests.cToP_inputs_too_large(CalculatorTests.java:71)
```

**Succeed:**

The screenshot shows the same IDE with the `Calculator.java` file updated to include the new test case:

```
1 package util;
2
3 public class Calculator {
4
5     public static double[] cartesianToPolar(double x, double y) throws Exception
6     {
7
8         if (x==0 || y==0)
9         {
10             throw new Exception("One or more inputs are zero!");
11         }
12
13         else if (Math.abs(x)>=10000 || Math.abs(y)>=10000)
14         {
15             throw new Exception("Inputs too large!");
16         }
17
18         double[] result = new double[2];
19     }
```

The `CalculatorTests.java` file is not visible. The `JUnit` runner shows the test results:

- Finished after 0.016 seconds
- Runs: 5/5
- Errors: 0
- Failures: 0

The test results list shows the following tests:

- cToP\_inputs\_are\_zero (0.000 s)
- cToP\_inputs\_are\_nonzero (0.000 s)
- cToP\_inputs\_too\_large (0.000 s)
- cToP\_returns\_double\_array (0.000 s)
- cToP\_contains\_negative\_input (0.000 s)

All tests are highlighted in green, indicating they passed. The `cToP_inputs_too_large` test is now successful.

## Test 6

**Test name:** cToP\_inputs\_too\_small

**Call setup:** cartesianToPolar(0.01,-0.01)

**Expected:** cartesianToPolar() throws exception with message "Inputs too small!"

**Fail:**

The screenshot shows an IDE with two panes. The top pane displays the `Calculator.java` file with the following code:

```
public class Calculator {  
    public static double[] cartesianToPolar(double x, double y) throws Exception  
    {  
        if (x==0 || y==0)  
        {  
            throw new Exception("One or more inputs are zero!");  
        }  
        else if (Math.abs(x)>=10000 || Math.abs(y)>=10000)  
        {  
            throw new Exception("Inputs too large!");  
        }  
        double[] result = new double[2];  
        result[0]=Math.sqrt(Math.pow(x, 2)+Math.pow(y, 2));  
        result[1]=Math.atan(y/x);  
    }  
}
```

The bottom pane shows the JUnit test results. The test `cToP_inputs_too_small` failed with the following error:

```
java.lang.AssertionError  
at tests.CalculatorTests.cToP_inputs_too_small(CalculatorTests.java:85)
```

**Succeed:**

The screenshot shows an IDE with two panes. The top pane displays the `CalculatorTests.java` file with the following code:

```
@Test  
public void cToP_inputs_too_small() throws Exception  
{  
    boolean thrown = false;  
    double[] result={-1,-1};  
    try {  
        result = Calculator.cartesianToPolar(0.01,0.01);  
    } catch (Exception e) {  
        thrown = true;  
    }  
    Assert.assertTrue(thrown);  
}
```

The bottom pane shows the JUnit test results. All tests passed, including `cToP_inputs_too_small`.

## Test 7

**Test name:** pToC\_returns\_double\_array

**Call setup:** polarToCartesian(3,0.5)

**Expected:** polarToCartesian() returns a double array

**Fail:**

Calculator.java

```
24 double[] result = new double[2];
25 result[0]=Math.sqrt(Math.pow(x, 2)+Math.pow(y, 2));
26 result[1]=Math.atan(y/x);
27 return result; //result[0]=radius, result[1]=angle
28
29 public static double[] polarToCartesian(double r, double theta)
30 {
31 }
32 }
33
34
```

CalculatorTests.java

```
29 public static double[] polarToCartesian(double r, double theta)
30 {
31 }
32 }
33
34
```

JUnit

Finished after 0.034 seconds

Runs: 7/7 Errors: 1 Failures: 0

tests.CalculatorTests [Runner: JUnit 4] (0.000 s)

- cToP\_inputs\_are\_zero (0.000 s)
- cToP\_inputs\_are\_nonzero (0.000 s)
- pToC\_returns\_double\_array (0.000 s)
- cToP\_inputs\_too\_large (0.000 s)
- cToP\_inputs\_too\_small (0.000 s)
- cToP\_returns\_double\_array (0.000 s)
- cToP\_contains\_negative\_input (0.000 s)

Failure Trace

java.lang.Error: Unresolved compilation problem:  
This method must return a result of type double[]

at util.Calculator.polarToCartesian(Calculator.java:29)  
at tests.CalculatorTests.pToC\_returns\_double\_array(CalculatorTests.java:91)

**Succeed:**

Calculator.java

```
24 double[] result = new double[2];
25 result[0]=Math.sqrt(Math.pow(x, 2)+Math.pow(y, 2));
26 result[1]=Math.atan(y/x);
27 return result; //result[0]=radius, result[1]=angle
28
29 public static double[] polarToCartesian(double r, double theta)
30 {
31 double[] result = new double[2];
32 return result; //result[0]=x, result[1]=y
33 }
34 }
```

CalculatorTests.java

```
29 public static double[] polarToCartesian(double r, double theta)
30 {
31 double[] result = new double[2];
32 return result; //result[0]=x, result[1]=y
33 }
34 }
```

JUnit

Finished after 0.018 seconds

Runs: 7/7 Errors: 0 Failures: 0

tests.CalculatorTests [Runner: JUnit 4] (0.000 s)

- cToP\_inputs\_are\_zero (0.000 s)
- cToP\_inputs\_are\_nonzero (0.000 s)
- pToC\_returns\_double\_array (0.000 s)
- cToP\_inputs\_too\_large (0.000 s)
- cToP\_inputs\_too\_small (0.000 s)
- cToP\_returns\_double\_array (0.000 s)
- cToP\_contains\_negative\_input (0.000 s)

Failure Trace



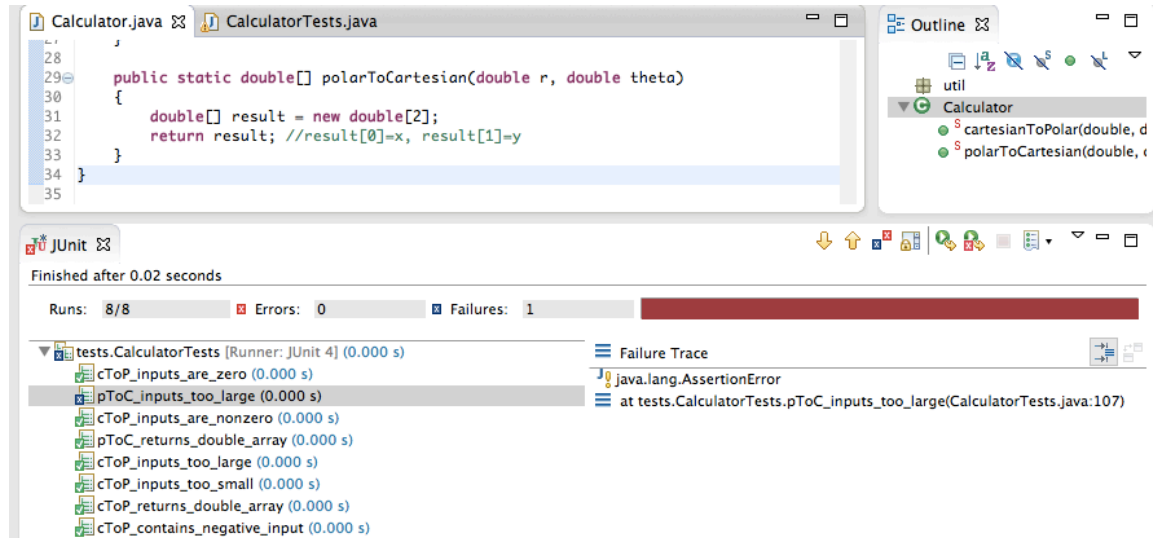
## Test 8

**Test name:** pToC\_inputs\_too\_large

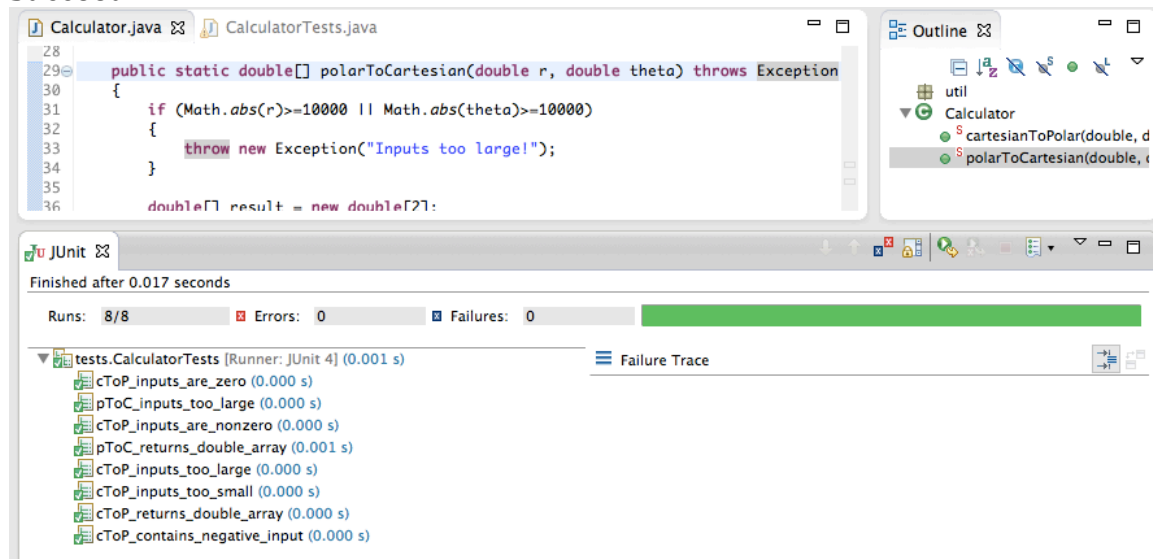
**Call setup:** polarToCartesian(10000,-10000)

**Expected:** polarToCartesian () throws exception with message "Inputs too large!"

**Fail:**



**Succeed:**



## Test 9

**Test name:** pToC\_inputs\_are\_nonzero

**Call setup:** polarToCartesian(5,arctan(4/3))

**Expected:** polarToCartesian() returns [3,4]

**Fail:**

The screenshot shows an IDE with two tabs: Calculator.java and CalculatorTests.java. The CalculatorTests.java tab is active, showing the test method pToC\_inputs\_are\_nonzero. The test is marked as failed (red icon). The failure trace indicates a java.lang.AssertionError at line 121 of CalculatorTests.java. The test results pane shows that the test pToC\_inputs\_are\_nonzero failed, while all other tests passed.

```
CalculatorTests.java
27 }
28
29 public static double[] polarToCartesian(double r, double theta) throws Exception
30 {
31     if (Math.abs(r)>=10000 || Math.abs(theta)>=10000)
32     {
33         throw new Exception("Inputs too large!");
34     }
35
36     double[] result = new double[2];
37     return result; //result[0]=x, result[1]=y
38 }
39
40
```

JUnit 4

Finished after 0.02 seconds

Runs: 9/9 Errors: 0 Failures: 1

tests.CalculatorTests [Runner: JUnit 4] (0.001 s)

- cToP\_inputs\_are\_zero (0.000 s)
- pToC\_inputs\_too\_large (0.000 s)
- cToP\_inputs\_are\_nonzero (0.000 s)
- pToC\_inputs\_are\_nonzero (0.000 s)
- pToC\_returns\_double\_array (0.001 s)
- cToP\_inputs\_too\_large (0.000 s)
- cToP\_inputs\_too\_small (0.000 s)
- cToP\_returns\_double\_array (0.000 s)
- cToP\_contains\_negative\_input (0.000 s)

Failure Trace

- java.lang.AssertionError
- at tests.CalculatorTests.pToC\_inputs\_are\_nonzero(CalculatorTests.java:121)

**Succeed:**

The screenshot shows the same IDE as before, but now the test pToC\_inputs\_are\_nonzero is marked as passed (green icon). The test results pane shows that all tests passed. The CalculatorTests.java tab is still active, showing the test method pToC\_inputs\_are\_nonzero. The test is now successful, and the failure trace is empty.

```
CalculatorTests.java
29 public static double[] polarToCartesian(double r, double theta) throws Exception
30 {
31     //theta in radians
32     if (Math.abs(r)>=10000 || Math.abs(theta)>=10000)
33     {
34         throw new Exception("Inputs too large!");
35     }
36
37     double[] result = new double[2];
38     result[0]=r*Math.cos(theta);
39     result[1]=r*Math.sin(theta);
40     return result; //result[0]=x, result[1]=y
41 }
42
```

JUnit 4

Finished after 0.02 seconds

Runs: 9/9 Errors: 0 Failures: 0

tests.CalculatorTests [Runner: JUnit 4] (0.000 s)

- cToP\_inputs\_are\_zero (0.000 s)
- pToC\_inputs\_too\_large (0.000 s)
- cToP\_inputs\_are\_nonzero (0.000 s)
- pToC\_inputs\_are\_nonzero (0.000 s)
- pToC\_returns\_double\_array (0.000 s)
- cToP\_inputs\_too\_large (0.000 s)
- cToP\_inputs\_too\_small (0.000 s)
- cToP\_returns\_double\_array (0.000 s)
- cToP\_contains\_negative\_input (0.000 s)

Failure Trace

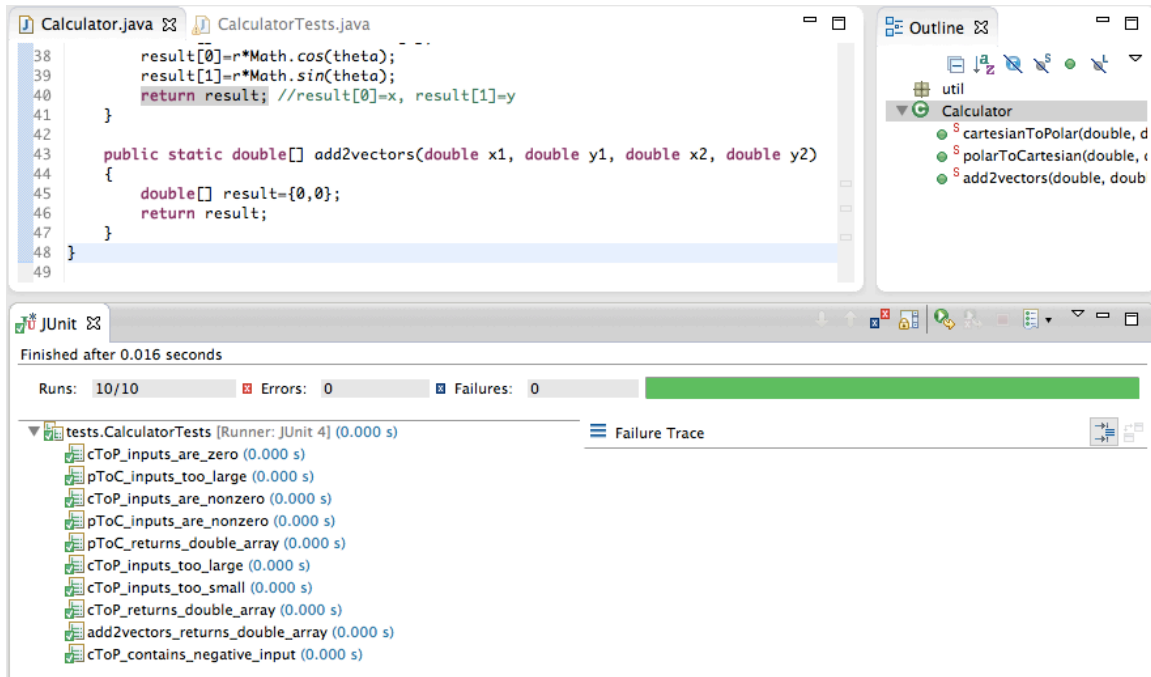
## Test 10

**Test name:** add2vectors\_returns\_double\_array

**Call setup:** add2vectors(3,4,5,6)

**Expected:** add2vectors() returns a double array

**Succeed:**



## Test 11

**Test name:** add2vectors\_regular\_input

**Call setup:** add2vectors(3,4,5,6)

**Expected:** add2vectors() returns [8,10]

**Fail:**

The screenshot shows an IDE with two tabs: Calculator.java and CalculatorTests.java. The CalculatorTests.java tab is active, showing the following code:

```
38     result[0]=r*Math.cos(theta);
39     result[1]=r*Math.sin(theta);
40     return result; //result[0]=x, result[1]=y
41 }
42
43 public static double[] add2vectors(double x1, double y1, double x2, double y2)
44 {
45     double[] result={0,0};
46     return result;
47 }
48 }
49
```

The Outline view on the right shows the following structure:

- util
- Calculator
  - cartesianToPolar(double, d)
  - polarToCartesian(double, c)
  - add2vectors(double, doub

The JUnit test runner shows the following results:

Finished after 0.023 seconds

Runs: 11/11 Errors: 0 Failures: 1

Failure Trace:

- at tests.CalculatorTests.add2vectors\_regular\_input(CalculatorTests.java:135)

The test results list shows the following tests:

- cToP\_inputs\_are\_zero (0.000 s)
- pToC\_inputs\_too\_large (0.000 s)
- add2vectors\_regular\_input (0.001 s)
- cToP\_inputs\_are\_nonzero (0.000 s)
- pToC\_inputs\_are\_nonzero (0.000 s)
- pToC\_returns\_double\_array (0.000 s)
- cToP\_inputs\_too\_large (0.000 s)
- cToP\_inputs\_too\_small (0.000 s)
- cToP\_returns\_double\_array (0.000 s)
- add2vectors\_returns\_double\_array (0.000 s)
- cToP\_contains\_negative\_input (0.000 s)

**Succeed:**

The screenshot shows the same IDE with the CalculatorTests.java tab active, showing the following code:

```
40     return result; //result[0]=x, result[1]=y
41 }
42
43 public static double[] add2vectors(double x1, double y1, double x2, double y2)
44 {
45     double[] result=new double[2];
46     result[0]=x1+x2;
47     result[1]=y1+y2;
48     return result;
49 }
50 }
51
```

The Outline view on the right shows the following structure:

- util
- Calculator
  - cartesianToPolar(double, d)
  - polarToCartesian(double, c)
  - add2vectors(double, doub

The JUnit test runner shows the following results:

Finished after 0.026 seconds

Runs: 11/11 Errors: 0 Failures: 0

Failure Trace:

The test results list shows the following tests:

- cToP\_inputs\_are\_zero (0.000 s)
- pToC\_inputs\_too\_large (0.000 s)
- add2vectors\_regular\_input (0.000 s)
- cToP\_inputs\_are\_nonzero (0.000 s)
- pToC\_inputs\_are\_nonzero (0.000 s)
- pToC\_returns\_double\_array (0.001 s)
- cToP\_inputs\_too\_large (0.000 s)
- cToP\_inputs\_too\_small (0.000 s)
- cToP\_returns\_double\_array (0.000 s)
- add2vectors\_returns\_double\_array (0.000 s)
- cToP\_contains\_negative\_input (0.000 s)

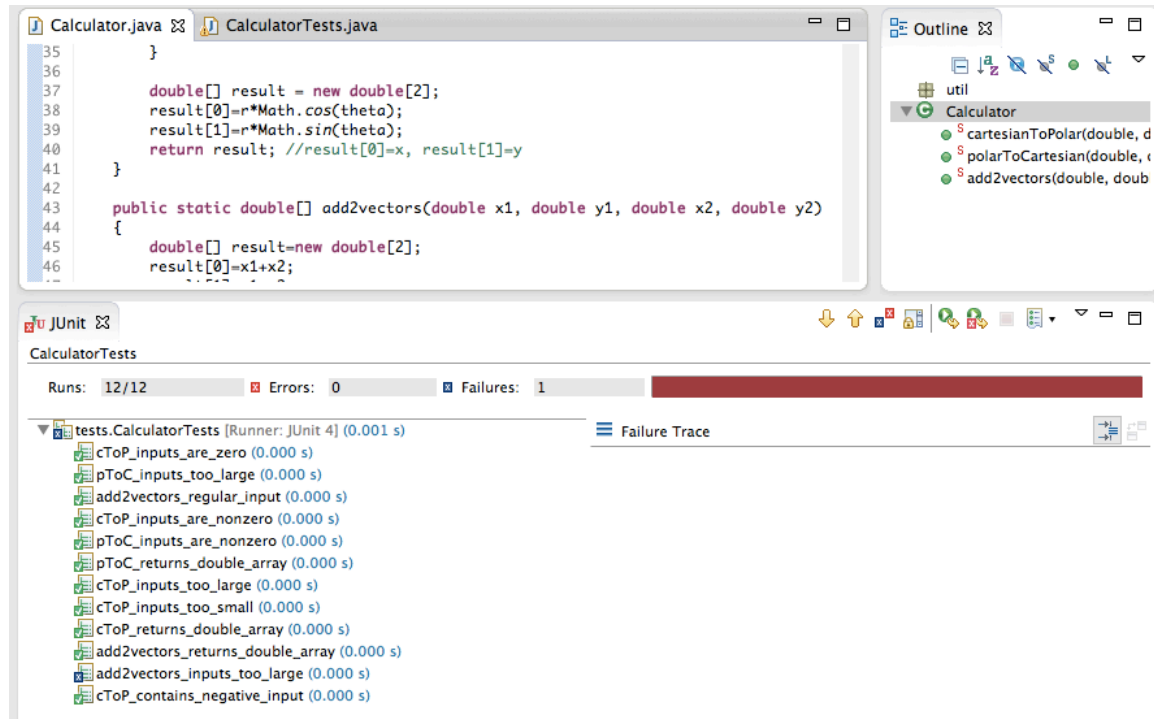
## Test 12

**Test name:** add2vectors\_inputs\_too\_large

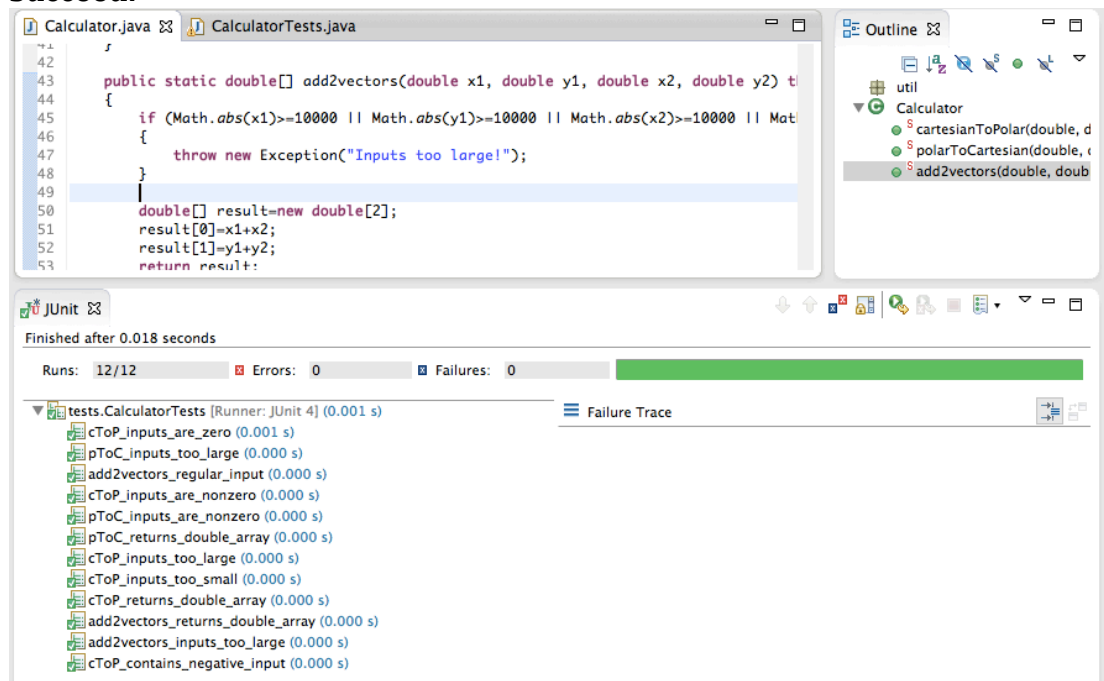
**Call setup:** add2vectors(10000,-10000,10000,-10000)

**Expected:** add2vectors() throws exception with message "Inputs too large!"

**Fail:**



**Succeed:**



## Test 13

**Test name:** add3vectors\_returns\_double\_array

**Call setup:** add3vectors(3,4,5,6,7,8)

**Expected:** add3vectors() returns a double array

**Fail:**

Calculator.java

```
54 }
55
56 public static double[] add3vectors(double x1, double y1, double x2, double y2, d
57 {
58 }
59 }
60
61 }
```

CalculatorTests.java

```
56 public static double[] add3vectors_returns_double_array() {
57 }
58 }
```

JUnit

Finished after 0.024 seconds

Runs: 13/13 Errors: 1 Failures: 0

tests.CalculatorTests [Runner: JUnit 4] (0.001 s)

- cToP\_inputs\_are\_zero (0.001 s)
- pToC\_inputs\_too\_large (0.000 s)
- add2vectors\_regular\_input (0.000 s)
- cToP\_inputs\_are\_nonzero (0.000 s)
- pToC\_inputs\_are\_nonzero (0.000 s)
- pToC\_returns\_double\_array (0.000 s)
- cToP\_inputs\_too\_large (0.000 s)
- cToP\_inputs\_too\_small (0.000 s)
- cToP\_returns\_double\_array (0.000 s)
- add2vectors\_returns\_double\_array (0.000 s)
- add2vectors\_inputs\_too\_large (0.000 s)
- cToP\_contains\_negative\_input (0.000 s)
- add3vectors\_returns\_double\_array (0.000 s)

Failure Trace

java.lang.Error: Unresolved compilation problem:  
This method must return a result of type double[]

at util.Calculator.add3vectors(Calculator.java:56)  
at tests.CalculatorTests.add3vectors\_returns\_double\_array(CalculatorTests.j

**Succeed:**

Calculator.java

```
54 }
55
56 public static double[] add3vectors(double x1, double y1, double x2, double y2, d
57 {
58     double[] result = new double[2];
59     result[0] = x1 + x2 + x3;
60     result[1] = y1 + y2 + y3;
61     return result;
62 }
```

CalculatorTests.java

```
56 public static double[] add3vectors_returns_double_array() {
57 }
58 }
```

JUnit

Finished after 0.021 seconds

Runs: 13/13 Errors: 0 Failures: 0

tests.CalculatorTests [Runner: JUnit 4] (0.000 s)

- cToP\_inputs\_are\_zero (0.000 s)
- pToC\_inputs\_too\_large (0.000 s)
- add2vectors\_regular\_input (0.000 s)
- cToP\_inputs\_are\_nonzero (0.000 s)
- pToC\_inputs\_are\_nonzero (0.000 s)
- pToC\_returns\_double\_array (0.000 s)
- cToP\_inputs\_too\_large (0.000 s)
- cToP\_inputs\_too\_small (0.000 s)
- cToP\_returns\_double\_array (0.000 s)
- add2vectors\_returns\_double\_array (0.000 s)
- add2vectors\_inputs\_too\_large (0.000 s)
- cToP\_contains\_negative\_input (0.000 s)
- add3vectors\_returns\_double\_array (0.000 s)

Failure Trace

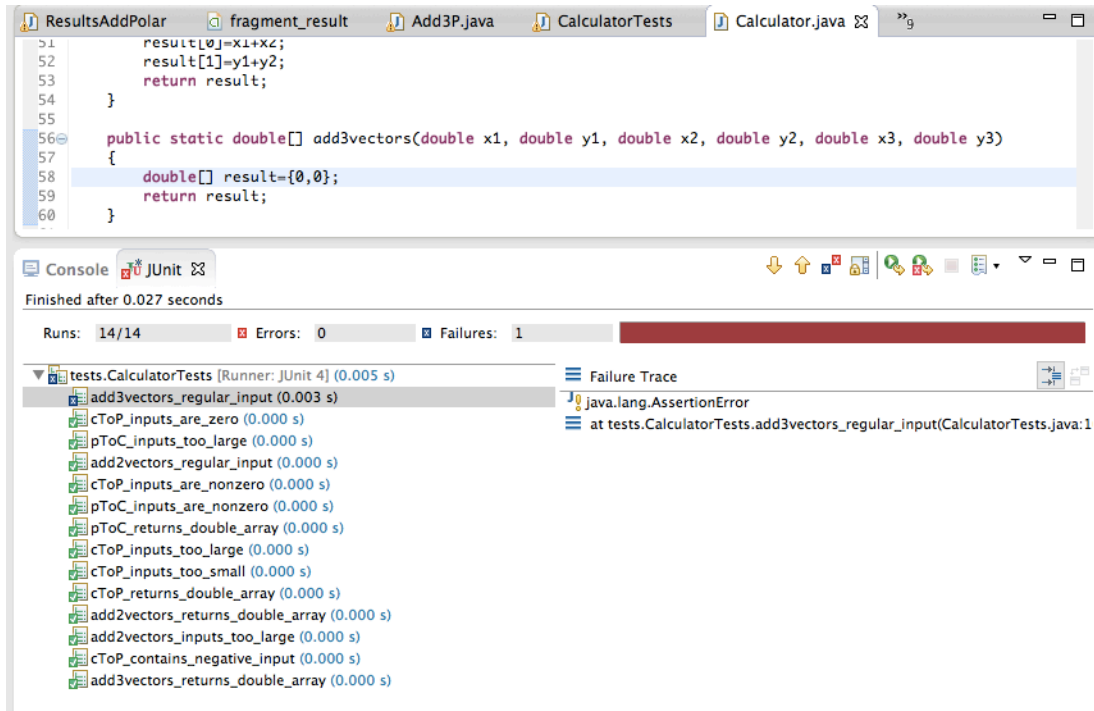
## Test 14

**Test name:** add3vectors\_regular\_input

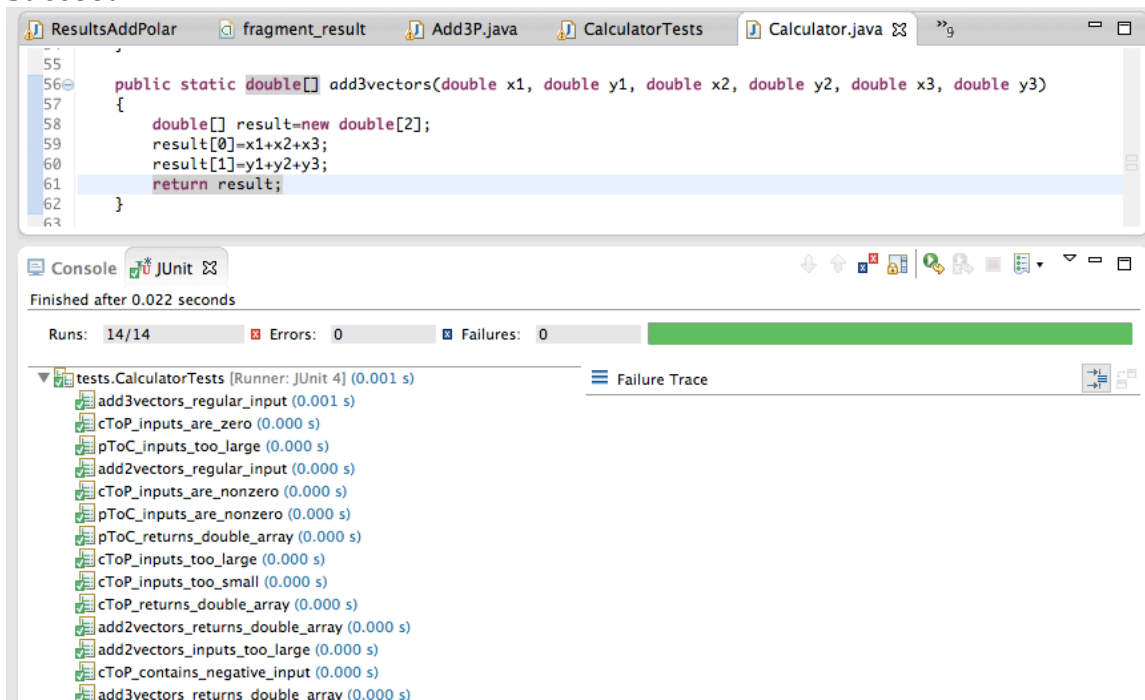
**Call setup:** add3vectors(3,4,5,6,7,8)

**Expected:** add2vectors() returns [15,18]

**Fail:**



**Succeed:**



## Test 15

**Test name:** add3vectors\_inputs\_too\_large

**Call setup:** add3vectors(10000,-10000,10000,-10000,10000,-10000)

**Expected:** add3vectors() throws exception with message "Inputs too large!"

**Fail:**

The screenshot shows an IDE with the following components:

- Calculator.java:** A method `add3vectors` that takes six `double` parameters and returns a `double[2]` array. It calculates `result[0] = x1 + x2 + x3` and `result[1] = y1 + y2 + y3`.
- JUnit Console:** Shows the test results for `tests.CalculatorTests`. It reports 15 runs, 0 errors, and 1 failure. The failure is for the test `add3vectors_inputs_too_large`.

**Succeed:**

The screenshot shows the same IDE after a code change:

- Calculator.java:** The `add3vectors` method now includes a check for large inputs. If the absolute value of any input is greater than or equal to 10000, it throws a `new Exception("Inputs too large!");` before performing the calculations.
- JUnit Console:** Shows that all 15 tests passed successfully, with 0 failures.



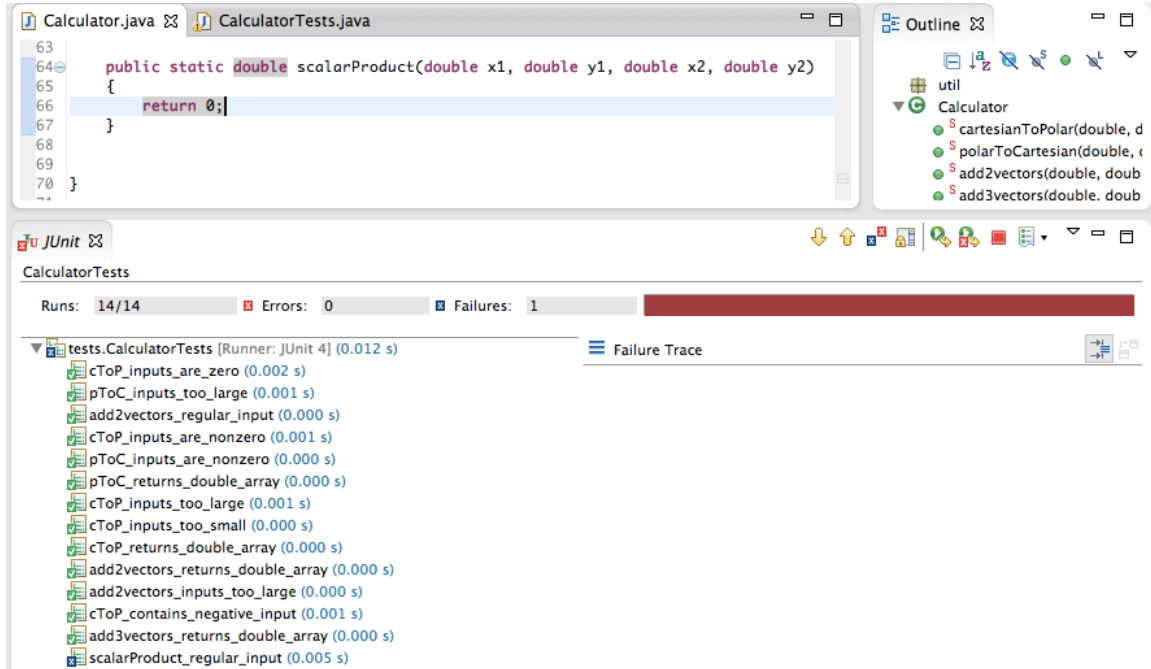
## Test 16

**Test name:** scalarProduct\_regular\_input

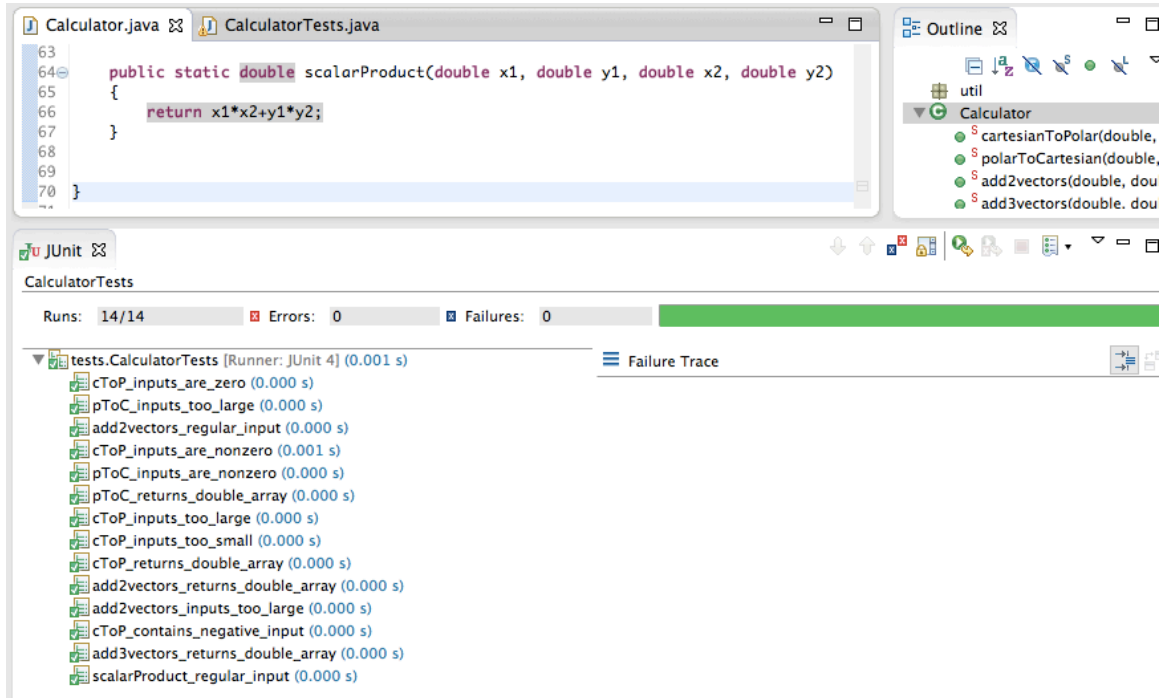
**Call setup:** scalarProduct(3,4,5,6)

**Expected:** scalarProduct() returns a 39

**Fail:**



**Succeed:**



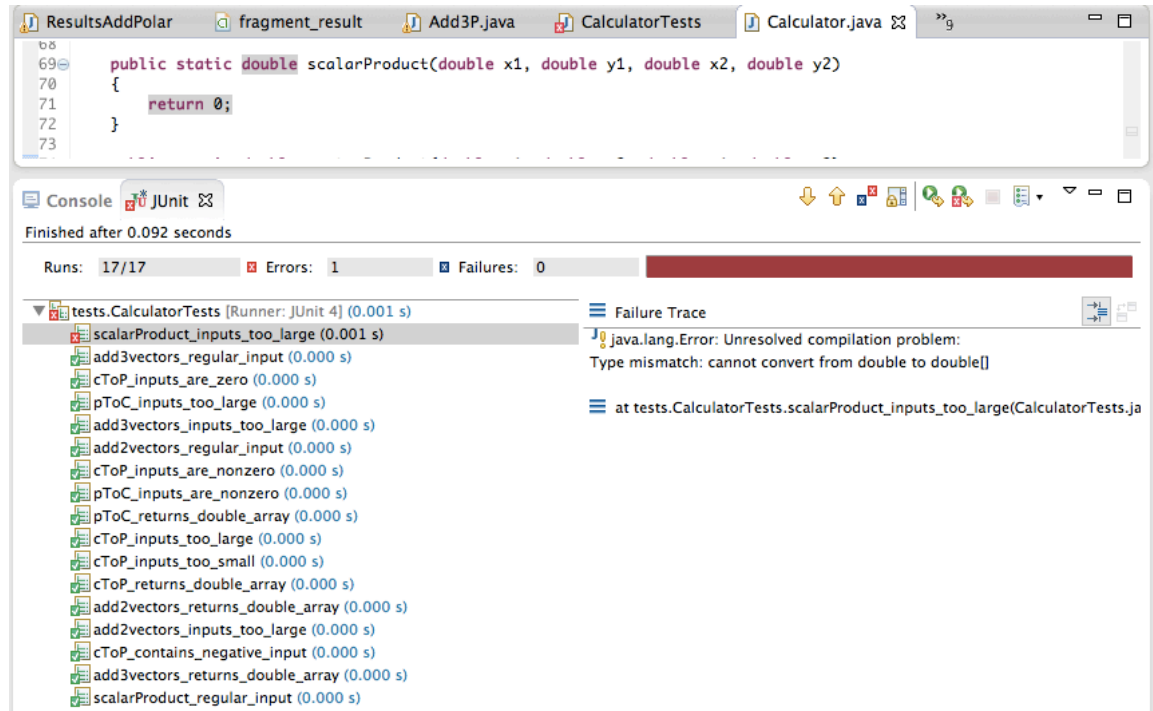
## Test 17

**Test name:** scalarProduct\_inputs\_too\_large

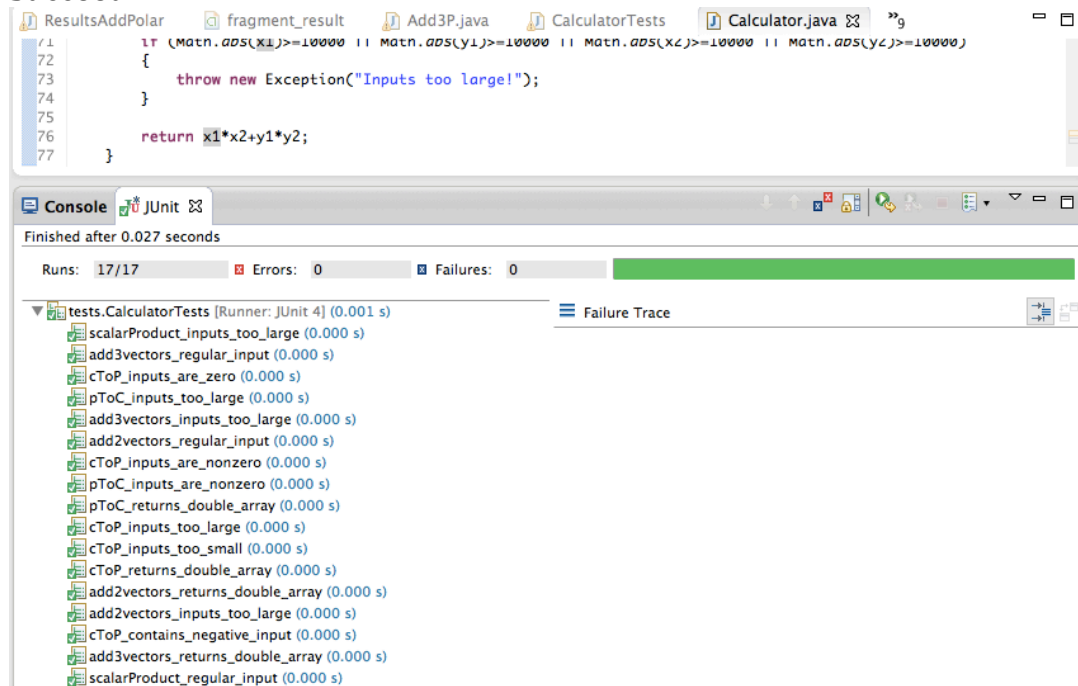
**Call setup:** scalarProduct(10000,-10000,10000,-10000)

**Expected:** scalarProduct() throws exception with message "Inputs too large!"

**Fail:**



**Succeed:**



## Test 18

**Test name:** vectorProduct\_regular\_input

**Call setup:** vectorProduct(3,4,5,6)

**Expected:** vectorProduct() returns a -2

**Fail:**

The screenshot shows an IDE with two tabs: Calculator.java and CalculatorTests.java. The CalculatorTests.java tab is active, showing the following code:

```
68
69 public static double vectorProduct(double x1, double x2, double y1, double y2)
70 {
71     return 0; //this is just the value of the z component! so answer will be (0,
72 }
73
74
75 }
```

The Outline pane on the right shows a list of methods: cartesianToPolar(double, d), polarToCartesian(double, c), add2vectors(double, doub), add3vectors(double, doub), scalarProduct(double, dou), and vectorProduct(double, dou). The JUnit test runner shows a failure for the test vectorProduct\_regular\_input (0.000 s). The failure trace indicates a java.lang.AssertionError at tests.CalculatorTests.vectorProduct\_regular\_input(CalculatorTests.java:17).

**Succeed:**

The screenshot shows the same IDE as before, but the CalculatorTests.java tab now shows the following code:

```
68
69 public static double vectorProduct(double x1, double x2, double y1, double y2)
70 {
71     return x1*y2-y1*x2; //this is just the value of the z component! so answer w
72 }
73
74
75 }
```

The Outline pane on the right shows the same list of methods. The JUnit test runner shows a successful test run for the test vectorProduct\_regular\_input (0.000 s). The failure trace is empty.

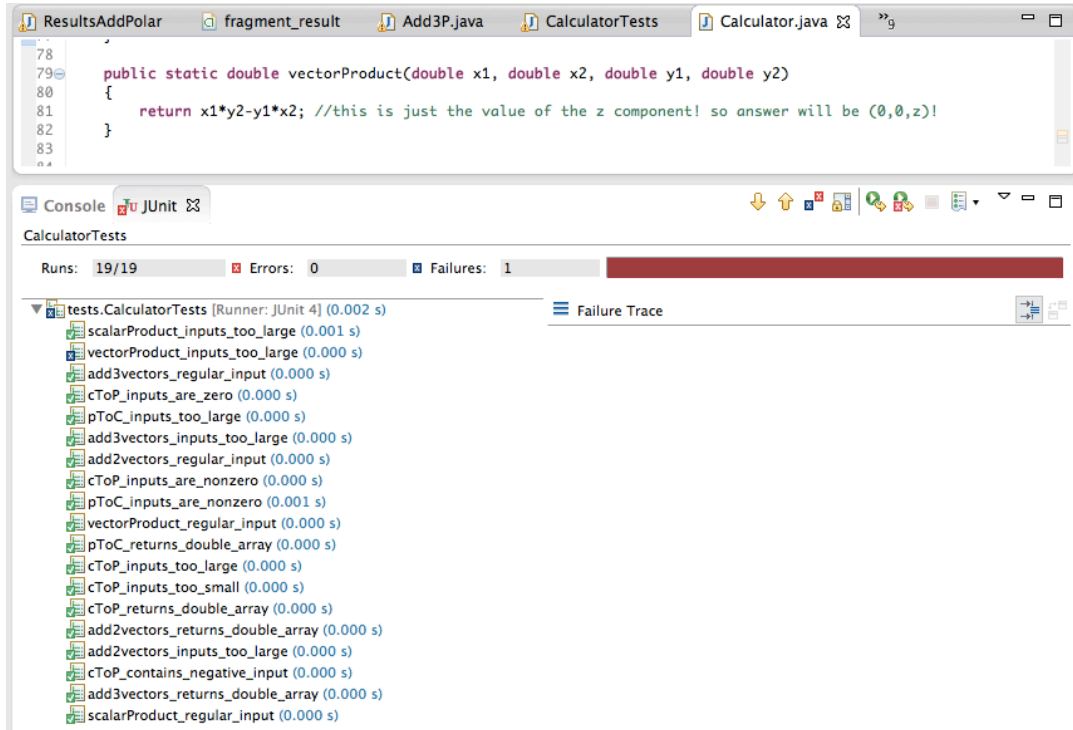
## Test 19

**Test name:** vectorProduct\_inputs\_too\_large

**Call setup:** vectorProduct(10000,-10000,10000,-10000)

**Expected:** vectorProduct() throws exception with message "Inputs too large!"

**Fail:**



**Succeed:**

