# Source code of initial SelectionSort.java class



Source code of unit tests created and executed

package selectionsort;

import static org.junit.Assert.\*;

import org.junit.Assert;

import org.junit.Test;

public class testSelectionSort {

@Test

public void test() {

testPositive();

testNegative();

testMixed();

testDuplicates();

}

public testSelectionSort() {

}

public void testPositive(){

int[] arr = new int[5];

arr[0] = 8;

arr[1] = 9;

arr[2] = 7;

arr[3] = 10;

arr[4] = 2;

int[] Sortedarr = new int[5];

Sortedarr[0] = 2;

Sortedarr[1] = 7;

Sortedarr[2] = 8;

Sortedarr[3] = 9;

Sortedarr[4] = 10;

SelectionSort arr1 = new SelectionSort();

arr = arr1.basicSelectionSort(arr);

assertArrayEquals(Sortedarr, arr);

}

public void testNegative(){

int[] arr = new int[5];

arr[0] = -8;

arr[1] = -9;

arr[2] = -7;

arr[3] = -10;

arr[4] = -2;

int[] Sortedarr = new int[5];

Sortedarr[0] = -10;

Sortedarr[1] = -9;

Sortedarr[2] = -8;

Sortedarr[3] = -7;

Sortedarr[4] = -2;

SelectionSort arr1 = new SelectionSort();

arr = arr1.basicSelectionSort(arr);

assertArrayEquals(Sortedarr, arr); }

public void testMixed(){

int[] arr = new int[5];

arr[0] = -8;

arr[1] = 9;

arr[2] = -7;

arr[3] = 10;

arr[4] = 0;

int[] Sortedarr = new int[5];

Sortedarr[0] = -8;

Sortedarr[1] = -7;

Sortedarr[2] = 0;

Sortedarr[3] = 9;

Sortedarr[4] = 10;

SelectionSort arr1 = new SelectionSort();

arr = arr1.basicSelectionSort(arr);

assertArrayEquals(Sortedarr, arr);

}

public void testDuplicates(){

int[] arr = new int[5];

arr[0] = 8;

arr[1] = 9;

arr[2] = 8;

arr[3] = 10;

arr[4] = 4;

int[] Sortedarr = new int[5];

Sortedarr[0] = 4;

Sortedarr[1] = 8;

Sortedarr[2] = 8;

Sortedarr[3] = 9;

Sortedarr[4] = 10;

SelectionSort arr1 = new SelectionSort();

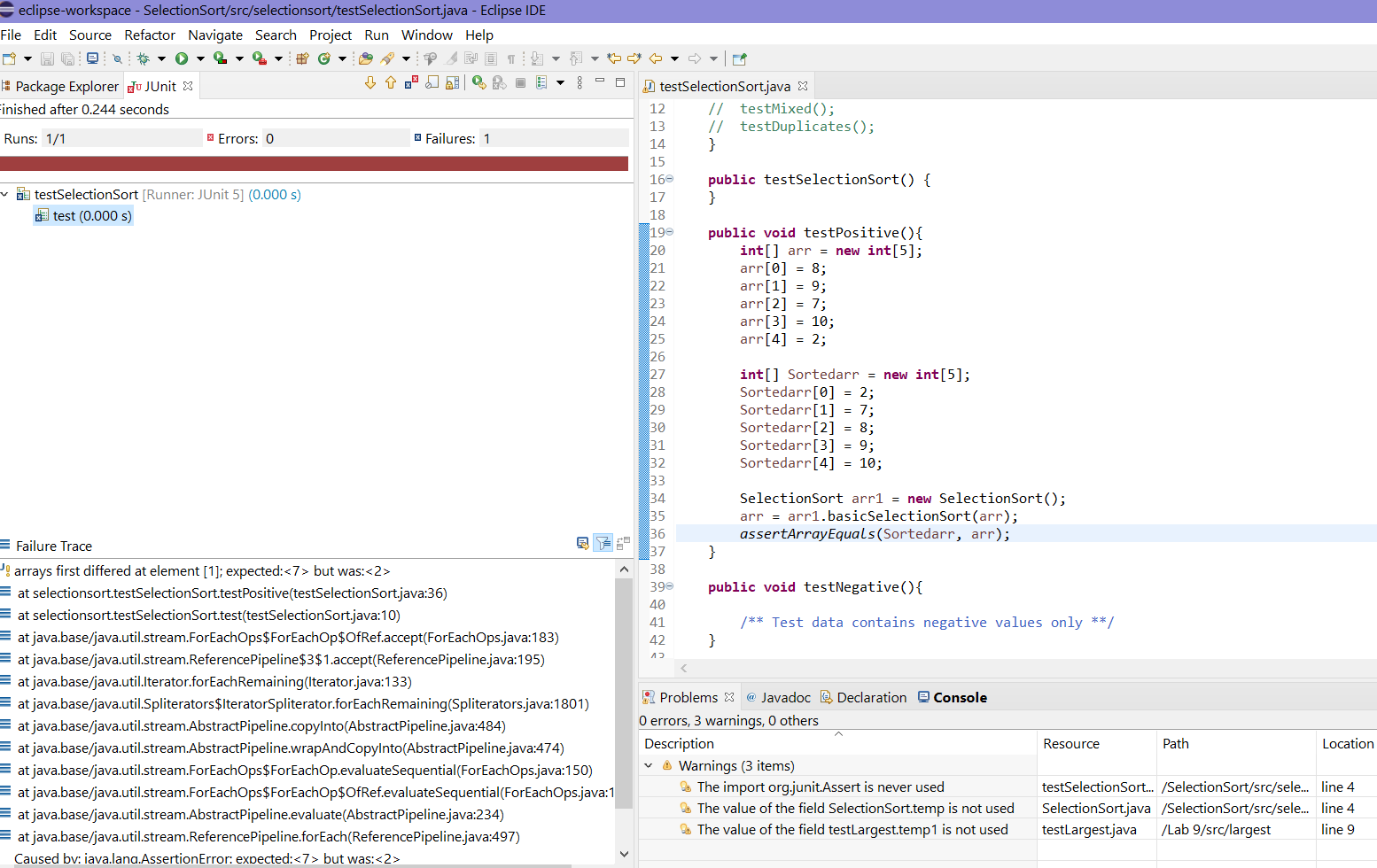
arr = arr1.basicSelectionSort(arr);

assertArrayEquals(Sortedarr, arr);

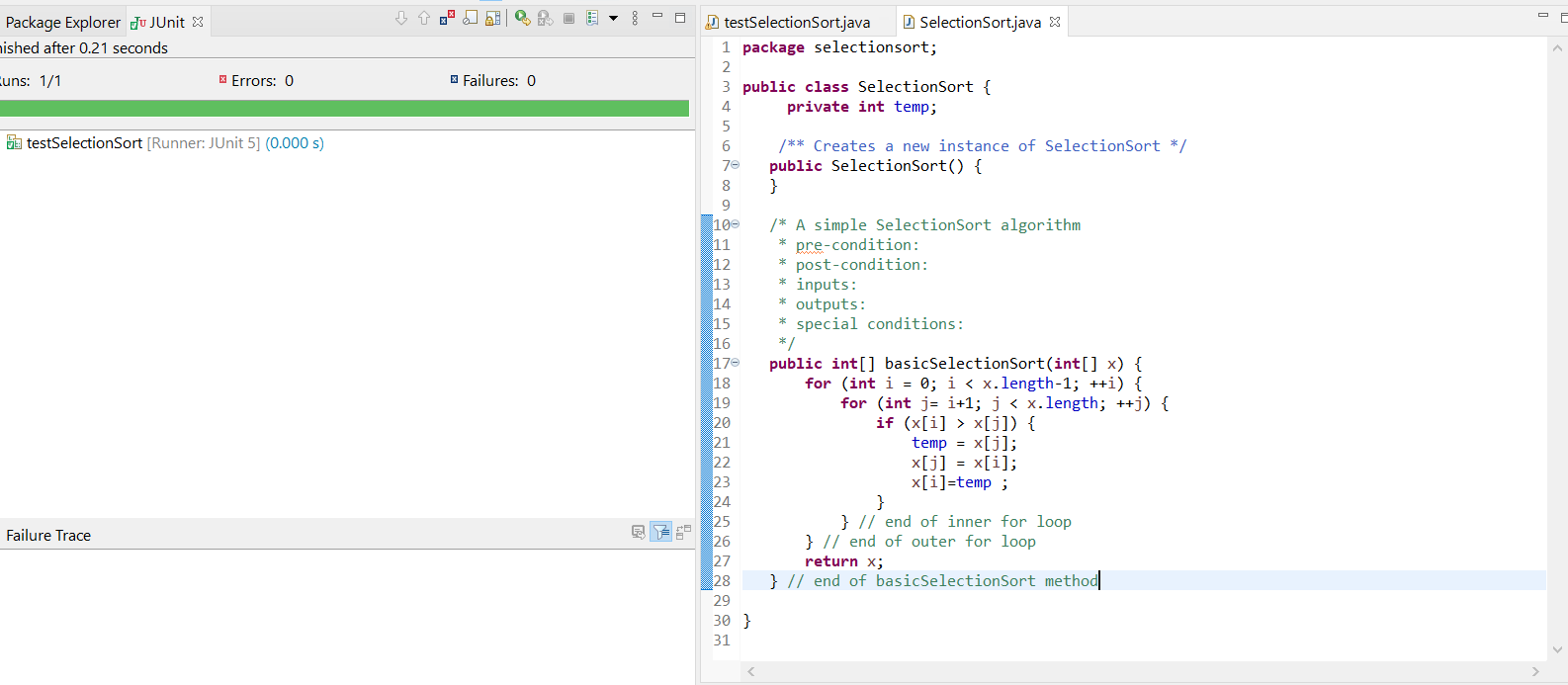
} }

# Outputs of unit tests. Include both failed and passed unit tests. In the case of failed unit tests, indicate what was wrong and what was done to fix the problem.

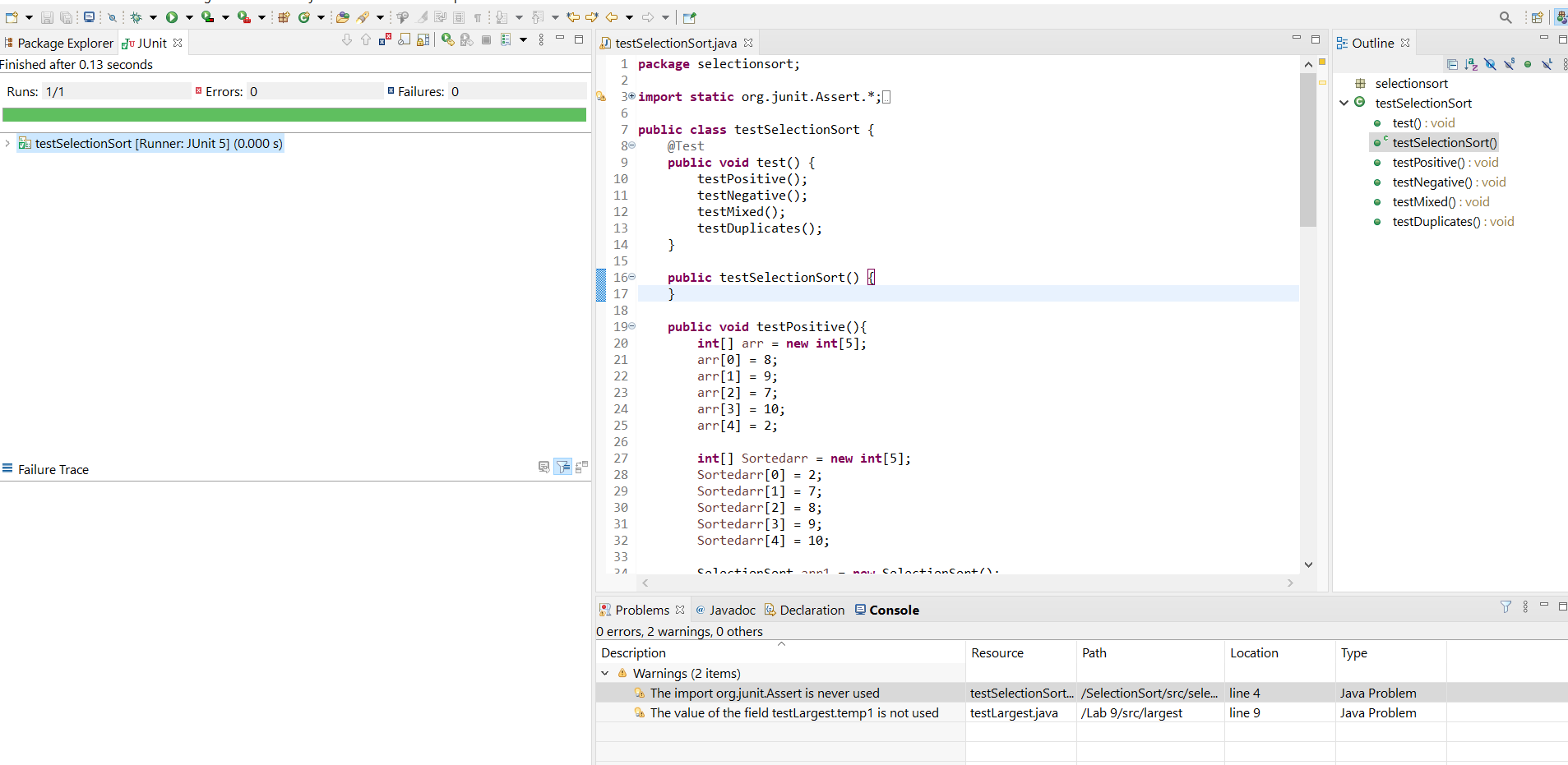
## Failed test



Fixed array code to remove failure. TestPositive code successful. Have to fix length of array



# **Final outputs of all unit tests showing successful pass for all**



# **SelectionSort.java**

**package** selectionsort;

**public** **class** SelectionSort {

**private** **int** temp;

/\*\* Creates a new instance of SelectionSort \*/

**public** SelectionSort() {

}

/\* A simple SelectionSort algorithm

\* pre-condition:

\* post-condition:

\* inputs:

\* outputs:

\* special conditions:

\*/

**public** **int**[] basicSelectionSort(**int**[] x) {

**for** (**int** i = 0; i < x.length-1; ++i) {

**for** (**int** j= i+1; j < x.length; ++j) {

**if** (x[i] > x[j]) {

temp = x[j];

x[j] = x[i];

x[i]=temp ;

}

} // end of inner for loop

} // end of outer for loop

**return** x;

} // end of basicSelectionSort method

}

# **testSelectionSort code**

package selectionsort;

import static org.junit.Assert.\*;

import org.junit.Assert;

import org.junit.Test;

public class testSelectionSort {

@Test

public void test() {

testPositive();

testNegative();

testMixed();

testDuplicates();

}

public testSelectionSort() {

}

public void testPositive(){

int[] arr = new int[5];

arr[0] = 8;

arr[1] = 9;

arr[2] = 7;

arr[3] = 10;

arr[4] = 2;

int[] Sortedarr = new int[5];

Sortedarr[0] = 2;

Sortedarr[1] = 7;

Sortedarr[2] = 8;

Sortedarr[3] = 9;

Sortedarr[4] = 10;

SelectionSort arr1 = new SelectionSort();

arr = arr1.basicSelectionSort(arr);

assertArrayEquals(Sortedarr, arr);

}

public void testNegative(){

int[] arr = new int[5];

arr[0] = -8;

arr[1] = -9;

arr[2] = -7;

arr[3] = -10;

arr[4] = -2;

int[] Sortedarr = new int[5];

Sortedarr[0] = -10;

Sortedarr[1] = -9;

Sortedarr[2] = -8;

Sortedarr[3] = -7;

Sortedarr[4] = -2;

SelectionSort arr1 = new SelectionSort();

arr = arr1.basicSelectionSort(arr);

assertArrayEquals(Sortedarr, arr); }

public void testMixed(){

int[] arr = new int[5];

arr[0] = -8;

arr[1] = 9;

arr[2] = -7;

arr[3] = 10;

arr[4] = 0;

int[] Sortedarr = new int[5];

Sortedarr[0] = -8;

Sortedarr[1] = -7;

Sortedarr[2] = 0;

Sortedarr[3] = 9;

Sortedarr[4] = 10;

SelectionSort arr1 = new SelectionSort();

arr = arr1.basicSelectionSort(arr);

assertArrayEquals(Sortedarr, arr);

}

public void testDuplicates(){

int[] arr = new int[5];

arr[0] = 8;

arr[1] = 9;

arr[2] = 8;

arr[3] = 10;

arr[4] = 4;

int[] Sortedarr = new int[5];

Sortedarr[0] = 4;

Sortedarr[1] = 8;

Sortedarr[2] = 8;

Sortedarr[3] = 9;

Sortedarr[4] = 10;

SelectionSort arr1 = new SelectionSort();

arr = arr1.basicSelectionSort(arr);

assertArrayEquals(Sortedarr, arr);

} }

