Paul Lazaros

Initial SelectionSort.java

**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; ++i) {

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

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

temp = x[i];

x[i] = x[j];

temp = x[j];

}

} // end of inner for loop

} // end of outer for loop

**return** x;

} // end of basicSelectionSort method

}

testSelectionSort.java

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

**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;

/\*\* add tests to check for this unit test \*\*/

SelectionSort sort = **new** SelectionSort();

sort.basicSelectionSort(arr);

*assertSame*("Wrong Value", Sortedarr[0], arr[0]);

*assertSame*("Wrong Value", Sortedarr[1], arr[1]);

*assertSame*("Wrong Value", Sortedarr[2], arr[2]);

*assertSame*("Wrong Value", Sortedarr[3], arr[3]);

*assertSame*("Wrong Value", Sortedarr[4], arr[4]);

}

**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;

/\*\* Test data contains negative values only \*\*/

SelectionSort sort = **new** SelectionSort();

sort.basicSelectionSort(arr);

*assertSame*("Wrong Value", Sortedarr[0], arr[0]);

*assertSame*("Wrong Value", Sortedarr[1], arr[1]);

*assertSame*("Wrong Value", Sortedarr[2], arr[2]);

*assertSame*("Wrong Value", Sortedarr[3], arr[3]);

*assertSame*("Wrong Value", Sortedarr[4], arr[4]);

}

**public** **void** testMixed(){

/\*\* Test data contains with both positive, negative and zeros \*\*/

**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] = -8;

Sortedarr[2] = -7;

Sortedarr[3] = 2;

Sortedarr[4] = 9;

SelectionSort sort = **new** SelectionSort();

sort.basicSelectionSort(arr);

*assertSame*("Wrong Value", Sortedarr[0], arr[0]);

*assertSame*("Wrong Value", Sortedarr[1], arr[1]);

*assertSame*("Wrong Value", Sortedarr[2], arr[2]);

*assertSame*("Wrong Value", Sortedarr[3], arr[3]);

*assertSame*("Wrong Value", Sortedarr[4], arr[4]);

}

**public** **void** testDuplicates(){

**int**[] arr = **new** **int**[5];

arr[0] = 8;

arr[1] = -2;

arr[2] = 8;

arr[3] = 10;

arr[4] = -2;

**int**[] Sortedarr = **new** **int**[5];

Sortedarr[0] = -2;

Sortedarr[1] = -2;

Sortedarr[2] = 8;

Sortedarr[3] = 8;

Sortedarr[4] = 10;

/\*\* Test data contains duplicates \*\*/

SelectionSort sort = **new** SelectionSort();

sort.basicSelectionSort(arr);

*assertSame*("Wrong Value", Sortedarr[0], arr[0]);

*assertSame*("Wrong Value", Sortedarr[1], arr[1]);

*assertSame*("Wrong Value", Sortedarr[2], arr[2]);

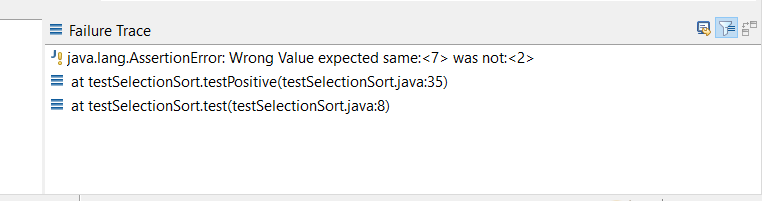
*assertSame*("Wrong Value", Sortedarr[3], arr[3]);

*assertSame*("Wrong Value", Sortedarr[4], arr[4]);

}

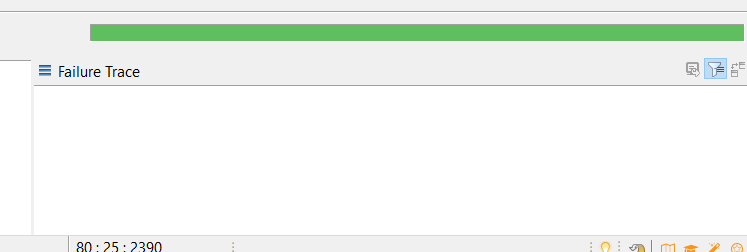
}

Test 1:



temp = x[j] should be x[j] = temp

Test 2:



Final SelectionSort.java

**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; ++i) {

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

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

temp = x[i];

x[i] = x[j];

x[j] = temp;

}

} // end of inner for loop

} // end of outer for loop

**return** x;

} // end of basicSelectionSort method

}

