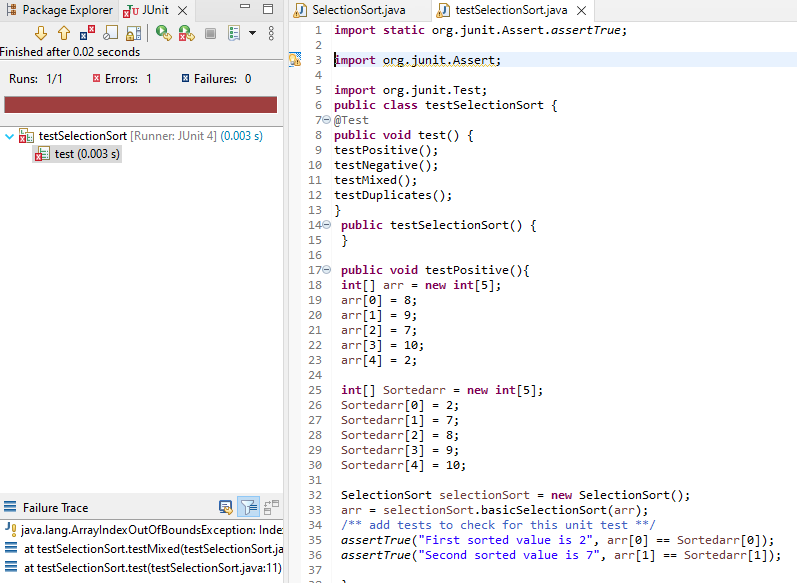
**Source code of initial SelectionSort.java class**

Source Codes:  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*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  
}  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

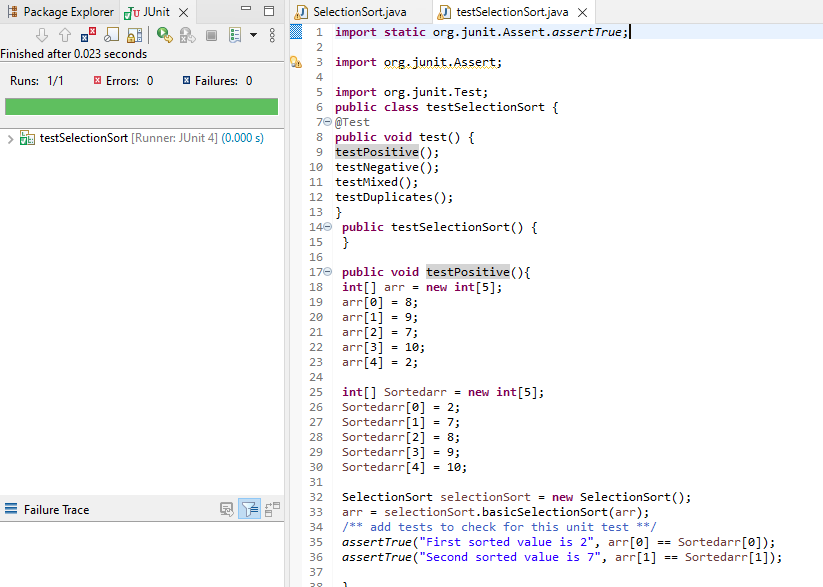
**\* testSelectionSort.java**  
 \*  
\* To change this template, choose Tools | Template Manager  
 \* and open the template in the editor.  
 \*/  
import static org.junit.Assert.\*;  
import org.junit.Assert;  
import org.junit.Test;  
public class testSelectionSort {  
@Test  
public void test() {  
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 \*\*/  
 }  
   
 public void testNegative(){  
 `  
 /\*\* Test data contains negative values only \*\*/  
 }  
   
 public void testMixed(){

/\*\* Test data contains with both positive, negative and zeros \*\*/  
 }  
   
 public void testDuplicates(){  
   
 /\*\* Test data contains duplicates \*\*/  
 }  
}  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

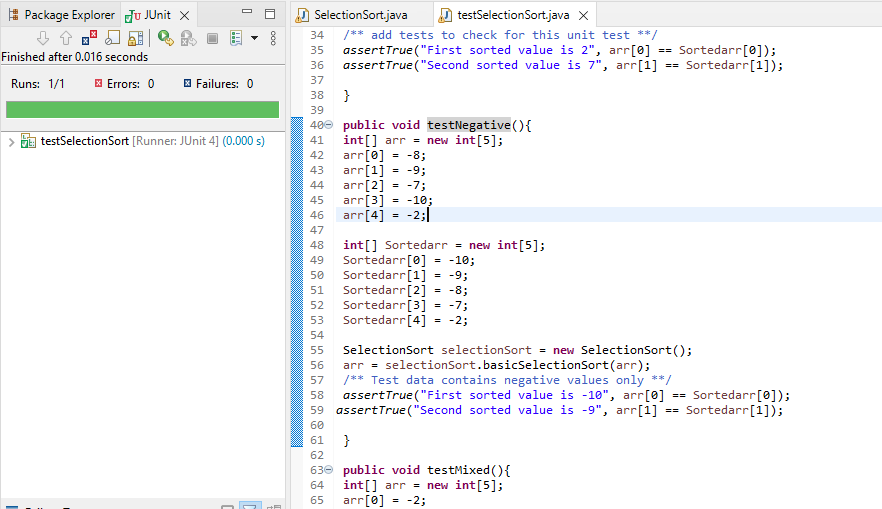
**This test failed because there was a logical error. The sort statement wasn’t correctly sorting.**



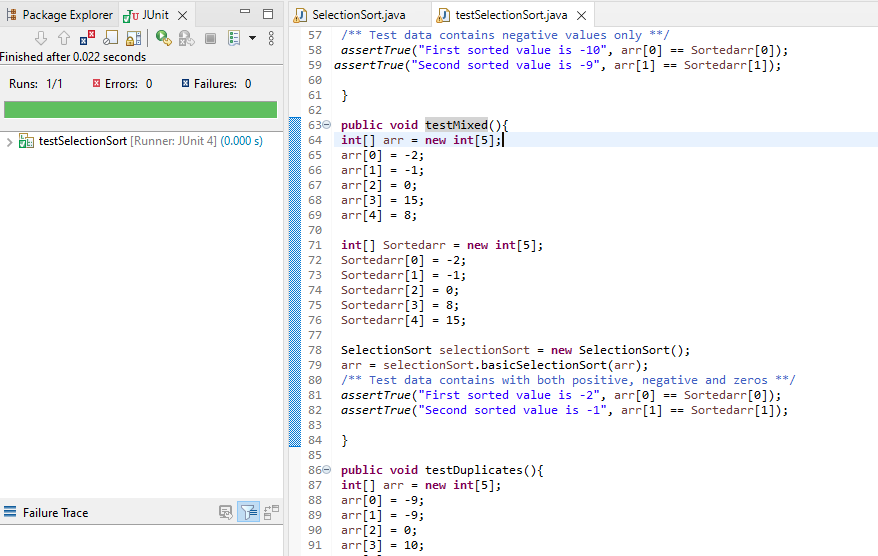
Positive Test –Pass



Test Negative – Pass

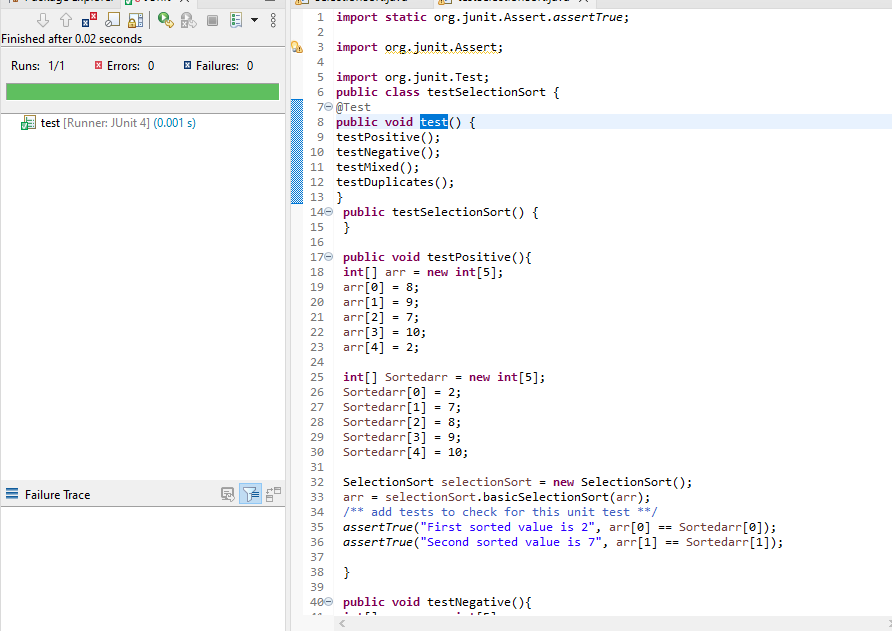


Test Mixed – Pass



Test Duplicate – Pass

All Pass



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Final Source Code:

SelectionSort.java

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

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

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

**public** SelectionSort() {

}

**int**[] basicSelectionSort(**int** arr[]) {

**int** i;

**int** j;

**int** min\_m;

**for** (i = 0; i < arr.length-1; i++)

{

min\_m = i;

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

**if** (arr[j] < arr[min\_m]) {

min\_m =j;

**int** temp;

temp = arr[min\_m];

arr[min\_m]=arr[i];

arr[i]=temp;

}

} // end of inner for loop

} // end of outer for loop

**return** arr;

} // end of basicSelectionSort method

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

testSelectionSort.java

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

**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 selectionSort = **new** SelectionSort();

arr = selectionSort.basicSelectionSort(arr);

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

*assertTrue*("First sorted value is 2", arr[0] == Sortedarr[0]);

*assertTrue*("Second sorted value is 7", arr[1] == Sortedarr[1]);

}

**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 selectionSort = **new** SelectionSort();

arr = selectionSort.basicSelectionSort(arr);

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

*assertTrue*("First sorted value is -10", arr[0] == Sortedarr[0]);

*assertTrue*("Second sorted value is -9", arr[1] == Sortedarr[1]);

}

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

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

arr[0] = -2;

arr[1] = -1;

arr[2] = 0;

arr[3] = 15;

arr[4] = 8;

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

Sortedarr[0] = -2;

Sortedarr[1] = -1;

Sortedarr[2] = 0;

Sortedarr[3] = 8;

Sortedarr[4] = 15;

SelectionSort selectionSort = **new** SelectionSort();

arr = selectionSort.basicSelectionSort(arr);

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

*assertTrue*("First sorted value is -2", arr[0] == Sortedarr[0]);

*assertTrue*("Second sorted value is -1", arr[1] == Sortedarr[1]);

}

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

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

arr[0] = -9;

arr[1] = -9;

arr[2] = 0;

arr[3] = 10;

arr[4] = 10;

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

Sortedarr[0] = -9;

Sortedarr[1] = -9;

Sortedarr[2] = 0;

Sortedarr[3] = 10;

Sortedarr[4] = 10;

SelectionSort selectionSort = **new** SelectionSort();

arr = selectionSort.basicSelectionSort(arr);

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

*assertTrue*("First sorted value is -9", arr[0] == Sortedarr[0]);

*assertTrue*("Second sorted value is -9", arr[1] == Sortedarr[1]);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*