**Programming Assignment #4**

**Buggy Search and Sort**

**Lab#8**

CS 1103 – 01 Programming 2

Fency Felix Subin

Sept 27 2022

Upon first run of the ‘Buggy Search and Sort’ code, it is apparent that the first number of lines in the main() method are working well. The code creates an Array A of ten random integers. By adding a breakpoint at line 25, the debugger proves that arrays B, C, and D, are also created and cloned properly with different ID references, but the same values stored in each array. When an array with the number 5 is created, the contains() method does not properly determine that the number 5 is part of the array.

So next, a look at the contains method is required. By placing a breakpoint at line 62, we can see that it the method is simply returning false or true after the first value is checked. The code reads:

public static boolean contains(int[] array, int val) {

for (int i = 0; i < array.length; i++) {

if (array[i] == val)

return true;

else

return false;

}

return false;

}

In the area shown in the debugger, on the first pass of the for loop, i = 0 the corresponding value in array[i] shown in the debugger is 4. 4 does not equal the given val 5. So the loop runs the else statement which returns false before checking any other value in the loop. To fix this first method we must delete lines 61 and 62 so that the for loop can iterate over each member of the array before returning a value of false if there are truly no 5’s in the array.

public static boolean contains(int[] array, int val) {

for (int i = 0; i < array.length; i++) {

if (array[i] == val)

return true;

}

return false;

}

This fixed code will now properly return true when the iteration over the array first finds the value given stored in one of the indexes of the array, which in this case is 5. If there are no 5’s, the method will return false.

The next problem to sort out is why the next method, bubbleSort() is throwing our main() method into an infinite loop. When putting breakpoints on lines 73,74,75,76,77 and using the ‘resume’ key to cycle through each pass of these lines, it becomes apparent that the if statement runs once, but then the program gets stuck cycling through the for loop of lines 73 and 74 and the debugger shows the value of i increasing infinitely, but the value of j remains as 0 for each pass. A closer look at the for loop on line 73 reveals the problem.

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

The increment statement for this for loop is set to increment i, not j as it should. This results in an infinite for loop as j never reaches the value of (array.length-1). By changing the statement i++ to j++, the problem is solved and the method works properly.

Now when running the code, contains(), array.sort(), and bubbleSort() work properly, but there must be semantic errors in the selectionSort() method and the insertionSort() methods as they are not sorting properly. So we move onto selectionSort(). Placing breakpoints on lines 90, 91, 92, and 93, using the resume button to scroll through each step of the function, specifically the for loop contained in the selectionSort() method.

The first observation on the variables table is that when the method reaches the first pass of line 91, i = 1 which disregards the first index of the array.

for (int i =1; i <= top; i++) {

So the first piece of the code to fix is to initialize i = 0 in the for loop on line 91 so we don’t miss any values. The second thing to note happens as the debugger passes line 92.

if (array[1] > array[positionOfMax])

The value checked against array[positionOfMax] is always array[1], so as the debugger passes over each iteration of the for loop, each value in the array[i] is not being checked against the value of array[positionOfMax]. We need to change array[1] to array[i] to ensure each increasing index of array[i] is checked against array[positionOfMax]. Upon fixing these two errors, the selectionSort() method is proven to work properly.

Lastly, there is still an error on the insertionSort() method as it is not sorting the array properly. Adding breakpoints on lines 107,108,109,110,112,113, and 115 so that each step of the method can be viewed. The first problem that becomes apparent is that when int pos = top – 1, pos = 0 for the first iteration of the while loop.

int pos = top - 1;

while (pos > 0 && array[pos] > temp) {

That means that even if array[0] is greater than array[1], nothing will happen because the code inside the while loop will be skipped. We can correct this by making int pos = top and then fixing the while loop statement to be compare array[pos-1] to temp.

int pos = top;

while (pos > 0 && array[pos-1] > temp) {

The next problem occurs when the debugger reaches the first code block within the while loop.

//move items that are bigger than temp up one position

array[pos+1] = array[pos];

This code only makes the array item above the current position equal to the larger number. But does not move the smaller number down a position in the array. I would suggest shifting the logic in the //comment. It should say move items that are smaller than temp down one position and make the current position equal to temp (the larger number of the two).

// move items that are smaller than temp down one position and

// make the current position equal to temp.

array[pos] = array[pos-1];

array[pos-1] = temp;

pos--;

The code line outside the while loop (array[pos] = temp) is made unnecessary and can be discarded as the larger temp number has already moved up to the current position the while loop is counting down from.

So the total method now looks like this and works as it should:

public static void insertionSort(int[] array) {

for (int top = 1; top < array.length; top++) {

int temp = array[top];

int pos = top;

while (pos > 0 && array[pos-1] > temp) {

// move items that are smaller than temp down one //position and make the current position equal to /temp.

array[pos] = array[pos-1];

array[pos-1] = temp;

pos--;

}

}

}

Running the class again proves that all methods are now working correctly and have been debugged successfully.