Learning Journal

Unit\_4

Dawid Blom

**Thursday, 29/04/2021, 1 – 3 pm: Reading Assignment:** In chapter 9.5 of the book I read about parsers but more interestingly, I gained better knowledge regarding the BNF syntax rules. This was very interesting as it helped me better understand algorithms I had read previously, as well as, one of the assignment we had in this course.

**Thursday, 29/04/2021, 4 – 5 pm: Discussion Assignment:**  The discussion assignment this week was tricky because I never really know what could be an advantage and what is a disadvantage. However, because of that I always push myself to find those advantages best I can. This week was again one of those tricky ones. Nonetheless, I am quite proud of myself for finding the ones I did.

**Friday, 30/04/2021, 8 – 12 am: Programming Assignment:** I learned quite a few things through this weeks programming assignment. Number was the fact that debugging is a very useful skill, debugging with a debugger is different that debugging by reading the code and checking what to fix. However, I like the second way more since it helps me really understand what the code does. Nonetheless, I can really see where a debugger will come in handy.

**Friday, 30/04/2021, 1 – 2 pm: Self Quiz:** This weeks self quiz was quite nice it really helped to establish the concepts learned during the week. Furthermore, I had to go through it a couple of times before I was where I wanted to be.

**Monday, 03/05/2021, 3 – 4 pm: Learning Journal:**

package unit\_4;

import java.util.Arrays;

/\*\*

\* This class looks like it's meant to provide a few public static methods

\* for searching and sorting arrays. It also has a main method that tests

\* the searching and sorting methods.

\*

\* TODO: The search and sort methods in this class contain bugs that can

\* cause incorrect output or infinite loops. Use the Eclipse debugger to

\* find the bugs and fix them

\*/

public class BuggySearchAndSort {

public static void main(String[] args) {

int[] A = new int[10]; // Create an array and fill it with small random ints.

for (int i = 0; i < 10; i++)

A[i] = 1 + (int)(10 \* Math.random());

int[] B = A.clone(); // Make copies of the array.

int[] C = A.clone();

int[] D = A.clone();

System.out.print("The array is:");

printArray(A);

if (contains(A,5))

System.out.println("This array DOES contain 5.");

else

System.out.println("This array DOES NOT contain 5.");

Arrays.sort(A); // Sort using Java's built-in sort method!

System.out.print("Sorted by Arrays.sort(): ");

printArray(A); // (Prints a correctly sorted array.)

bubbleSort(B);

System.out.print("Sorted by Bubble Sort: ");

printArray(B);

selectionSort(C);

System.out.print("Sorted by Selection Sort: ");

printArray(C);

insertionSort(D);

System.out.print("Sorted by Insertion Sort: ");

printArray(D);

}

/\*\*

\* Tests whether an array of ints contains a given value.

\* @param array a non-null array that is to be searched

\* @param val the value for which the method will search

\* @return true if val is one of the items in the array, false if not

\*/

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

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

if (array[i] == val)

return true;

/\*\*

\* BUG ONE------------------------.

\*

\* else

\* return false;

\*/

}

return false;

}

/\*\*

\* Sorts an array into non-decreasing order. This inefficient sorting

\* method simply sweeps through the array, exchanging neighboring elements

\* that are out of order. The number of times that it does this is equal

\* to the length of the array.

\*/

public static void bubbleSort(int[] array) {

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

for (int j = 0; j < array.length-1; j++) { // BUG TWO---------------------.

if (array[j] > array[j+1]) { // swap elements j and j+1

int temp = array[j];

array[j] = array[j+1];

array[j+1] = temp;

}

}

}

}

/\*\*

\* Sorts an array into non-decreasing order. This method uses a selection

\* sort algorithm, in which the largest item is found and placed at the end of

\* the list, then the second-largest in the next to last place, and so on.

\*/

public static void selectionSort(int[] array) {

for (int top = array.length - 1; top > 0; top--) {

int positionOfMax = 0;

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

if (array[i] > array[positionOfMax]) // BUG THREE----------------------.

positionOfMax = i;

}

int temp = array[top]; // swap top item with biggest item

array[top] = array[positionOfMax];

array[positionOfMax] = temp;

}

}

/\*\*

\* Sorts an array into non-decreasing order. This method uses a standard

\* insertion sort algorithm, in which each element in turn is moved downwards

\* past any elements that are greater than it.

\*/

public static void insertionSort(int[] array) {

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

int temp = array[top]; // copy item that into temp variable

int pos = top - 1;

while (pos >= 0 && array[pos] > temp) { // BUG FOUR----------------------.

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

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

pos--;

}

array[pos + 1] = temp; // place temp into last vacated position. // BUG FIVE------------------------.

}

}

/\*\*

\* Outputs the ints in an array on one line, separated by spaces,

\* with a line feed at the end.

\*/

private static void printArray(int[] array) {

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

System.out.print(" ");

System.out.print(array[i]);

}

System.out.println();

}

}