

INFO-C307 Laboratory Session #3

LAB GOALS

- Array Manipulation (adding more functionality to our HArray class)

Step 1: Review the lecture material about arrays, and add a new `insert_ascending_order()` functions to our user-defined `HArray_V2` class. You don't have to remove the current `insert()` function, you just have to recognize that the two insert methods cannot be used concurrently. The `insert()` function simply adds the item to the end of the array (without caring about the order of values), but the `insert_ascending_order` will maintain the array in a sorted ascending order. Rename this Array class with your initials and keep it in your proof of concept tools directory. (e.g., John Smith should call the class `JSArray_V1`)

```
// HArray.java
```

```
// demonstrates array class with high-level interface
```

```
////////////////////////////////////
```

```
class HArray_V2 {
```

```
    private int[] theArray;          // ref to array a
    private int nElems;              // number of data items
    //-----
```

```
    public HArray_V2(int max) // constructor
    {
        theArray = new int[max];    // create the array
        nElems = 0;                 // no items yet
    }
    //-----
```

```
    public int linearSearch(int searchKey) { // find value and return location
        int location;
        for (location = 0; location < nElems; location++)
        {
            if (theArray[location] == searchKey) // found item?
            {
                break; // exit loop before end
            }
        }
        if (location == nElems) // gone to end?
        {
            return -1; // can't find it, return -1
        } else {
            return location; // found it, return location
        }
    }
}
```

```
//-----
```

```
    public int binarySearch(int searchKey) { // displays array contents
```

```
        int Low, High, Middle;
        Low = 0;
        High = nElems;
```

```
        while (Low <= High) {
            Middle = (int) ((Low + High) / 2);
```

```
            if (searchKey == theArray[Middle]) { //Its a match!!
                return (Middle);
```

```

        } else {
            if (searchKey < theArray[Middle]) //Search the low end of the array
            {
                High = Middle - 1;
            } else {
                Low = Middle + 1;
            }
        }
    }
    return (-1); //return -1 to indicate the value was not found
}
//-----
// Bubble Sort()

public void bubbleSort() {
    int pass, index, hold;
    //System.out.println("Inside the Sort Function");
    for (pass = 1; pass < nElems; pass++) {
        for (index = 0; index < nElems - 1; index++) {
            if (theArray[index] > theArray[index + 1]) {
                //swap
                hold = theArray[index];
                theArray[index] = theArray[index + 1];
                theArray[index + 1] = hold;
            }
            //display();
        }
        //display();
    }
}

```

```

//-----
public void insert(int value) // put element into array
{
    theArray[nElems] = value; // insert it
    nElems++; // increment size
}

```

// add the new code here...

```

//-----

public boolean delete(int value) {
    int j;
    for (j = 0; j < nElems; j++) // look for it
    {
        if (value == theArray[j]) {
            break;
        }
    }
    if (j == nElems) // can't find it
    {
        return false;
    } else // found it
    {
        for (int k = j; k < nElems; k++) // move higher ones down
        {
            theArray[k] = theArray[k + 1];
        }
    }
}

```

```

        nElems--;           // decrement size
        return true;
    }
} // end delete()

//-----
// Mean()
public double mean() {      // calculate the average
    double sum = 0;
    for (int i = 0; i < nElems; i++) {
        sum = sum + theArray[i];
    }
    return (sum / nElems);
}
//-----
// StdDev()
public double stdDev() {    // Calculate the Standard Deviation

    double term = 0.0;
    double summation = 0.0;
    double array_element;
    double the_mean, the_std_dev;

    the_mean = mean();

    //System.out.println("Mean = " + the_mean);

    for (int i = 0; i < nElems; i++) {
        array_element = (double) theArray[i];
        term = Math.pow(array_element - the_mean, 2.0);
        summation = summation + term;
    }

    the_std_dev = Math.sqrt(summation / (nElems - 1));

    return (the_std_dev);
}

//-----
public int min() {          // Calculate the min

    int the_min = theArray[0]; // assume the first element is the smallest

    for (int i = 1; i < nElems; i++) {
        if (the_min > theArray[i]) {
            the_min = theArray[i];
        }
    }
    return (the_min);
}
//-----
public int max() {          // Calculate the max
    int the_max = theArray[0]; // assume the first element is the largest
    for (int i = 1; i < nElems; i++) {
        if (the_max < theArray[i]) {
            the_max = theArray[i];
        }
    }
    return (the_max);
}

```

```

//-----
public int mode() {    // Calculate the mode
    int currentValue, currentCount;
    int maxValue = 0;
    int maxCount = 0;

    int[] counts = new int[nElems];

    for (int i = 0; i < nElems; i++) {
        currentValue = theArray[i];           // get the current value
        currentCount = freqCount(currentValue); // see how many there are array

        if (currentCount > maxCount) {
            System.out.println("Candidate Mode: " + currentValue + ", Count= " +currentCount);
            maxCount = currentCount;
            maxValue = currentValue;
        }
    }
    return maxValue;
}
//-----
public int freqCount(int theValue) {    // Calculate the freqCount for a given value
    int count = 0;
    for (int i = 0; i < nElems; i++) {
        if(theArray[i] == theValue)
            count++;
    }
    return count;
}

//-----
public void display() // displays array contents
{
    for (int j = 0; j < nElems; j++) // for each element,
    {
        System.out.print(theArray[j] + " "); // display it
    }
    System.out.println("");
}
//-----
} // end class HHArray

```

Step 2: Add the following insert_ascending_order() function below the insert() function. Note that the following highlighted statement are for debugging purposes. Eventually they should be removed.

```
public void insert_ascending_order(int value) // put element into array in ascending order
{
    // Note that this function assumes that the array is empty or
    // all the elements of the array are already in order. Then it will
    // insert the new item in sorted order.
    if(nElems == 0) {
        theArray[nElems] = value;           // insert it at location 0
        System.out.println("Inserting new value " + value + " at location " + nElems);
    }
    else { // find the proper location for the value to be entered
        int index = 0;
        while (index < nElems && theArray[index] < value) {
            index++;
        }
        System.out.println("Inserting new value " + value + " at location " + index);
        if (index == nElems) // we are at the end of the array, add it here
            theArray[nElems] = value;
        else {
            // found the location to insert
            // shift the array element up by one, then insert at current index.
            for (int k = nElems+1; k > index; k--) // shift to the right
            {
                //System.out.println(k);
                theArray[k] = theArray[k - 1];
            }
            theArray[index] = value;
        }
    }
    nElems++;
    display(); // increment size
}
```

Now, add the following lines to the main program (make sure to comment off the old insert() statements. Mixing these two insert methods will not yield the desired effect.

```
System.out.println("-----");
System.out.println("-- Insert 10 numbers in ascending order --");
myArray.insert_ascending_order(77);           // insert 10 items
myArray.insert_ascending_order(99);
myArray.insert_ascending_order(44);
myArray.insert_ascending_order(55);
myArray.insert_ascending_order(-1);
myArray.insert_ascending_order(88);
myArray.insert_ascending_order(11);
myArray.insert_ascending_order(0);
myArray.insert_ascending_order(66);
myArray.insert_ascending_order(33);
myArray.insert_ascending_order(11);
myArray.insert_ascending_order(55);
```

Note that after every insert_ascending_order() function call, the array remain in sorted order. So, there is no need to call the sort function!

```

-----
-- Insert 10 numbers in ascending order --
Inserting new value 77 at location 0
77
Inserting new value 99 at location 1
77 99
Inserting new value 44 at location 0
44 77 99
Inserting new value 55 at location 1
44 55 77 99
Inserting new value -1 at location 0
-1 44 55 77 99
Inserting new value 88 at location 4
-1 44 55 77 88 99
Inserting new value 11 at location 1
-1 11 44 55 77 88 99
Inserting new value 0 at location 1
-1 0 11 44 55 77 88 99
Inserting new value 66 at location 5
-1 0 11 44 55 66 77 88 99
Inserting new value 33 at location 3
-1 0 11 33 44 55 66 77 88 99
Inserting new value 11 at location 2
-1 0 11 11 33 44 55 66 77 88 99
Inserting new value 55 at location 6
-1 0 11 11 33 44 55 55 66 77 88 99
-----
----- Print Array -----
-1 0 11 11 33 44 55 55 66 77 88 99
BUILD SUCCESSFUL (total time: 0 seconds)

```

What to Submit

- Source code
- Screen shot showing your results after every step
- A short paragraph indicating what new ideas you learned from this lab.
- Did you find any errors? Did you find any part of the lab confusing?

Hacker's Corner

- Add a new `insert_descending_order(int value)` function to your class.