Object-Oriented Programming Fundamentals

Lecture/Workshop (Week 9)

Searching and sorting arrays

Part 1: Searching

Carefully look over the class below that allows manipulation of an array of integers. Instance methods are provided to (re)populate an array with random numbers and display its contents.

Further functionality is required, namely to search the array for a given value and to sort the array in ascending order.

```
public class IntArray
{
   private int[] values;

   public IntArray(int size)
   {
      if (size > 0)
      {
         values = new int[size];
      }
      else
      {
         values = new int[10];
      }

      fillArray();
   }

   public void fillArray()
   {
      // random integers from 0 to 19 inclusive
      for (int i = 0; i < values.length; ++i)
      {
         values[i] = (int)(Math.random()*20);
      }
   }

   public void displayArray()
   {
      for (int i = 0; i < values.length; ++i)
      {
         System.out.print(values[i] + " ");
      }
      System.out.println();
   }
}
```

Searching an array

To find a value in an unsorted array, we must start at the beginning and iterate through the array looking for the search key

values	15	12	8	10	3	4	9	11
index no	0	1	2	3	4	5	6	7

- For example, to search for the value 8 in the array above, start by looking at position 0, then 1, then 2 (where 8 is found)
- To search for the value 2, start by looking at position 0, then 1, then 2, then 3, then 4, then 5, then 6, then 7
- As there are no more elements in the array, that search was unsuccessful

Task 1

Complete the method **`search()`**. It should return the index at which the **`target`** value is first found in the **`values`** array, or -1 if the **`target`** does not occur in the array.

```
public int search(int target)
{




}
```

Task 2

Now write a method **`searchLast()`** that searches the array from the end. It should return the index at which the **`target`** last occurs in the **`values`** array, or -1 if the **`target`** does not occur in the array.

```
public int searchLast(int target)
{




}
```

Part 2: Sorting an array

Sometimes we wish to sort an array so that the values in the array are organised in a specific order

- Unsorted array

values	15	12	8	10	3	4	9	11

- Sorted array

values	3	4	8	9	10	11	12	15

How do we sort it into numerical order?

Many sort algorithms have been developed, for example

- Selection sort
- Insertion sort
- Bubble sort
- Quick sort
- Merge sort

Bubble sort

The array elements are in any order and we want to rearrange them into order from minimum to maximum (ascending order)

So we repeatedly compare neighbouring elements and swap them if they are not in the correct order relative to each other. After the first pass, the largest element is in the correct position (it has bubbled up into the last position) and does not need to be checked in the second pass. After the second pass, the next largest element is also in the correct position (it has bubbled up into the second last position) and so on.

	0	1	2	3	4	5	6	7
initial array:	15	12	8	10	3	4	9	11
after first pass:	12	8	10	3	4	9	11	15
after second pass:	8	10	3	4	9	11	12	15
after third pass:	8	3	4	9	10	11	12	15
after fourth pass:	3	4	8	9	10	11	12	15
after fifth pass:	3	4	8	9	10	11	12	15
after sixth pass:	3	4	8	9	10	11	12	15
after seventh pass:	3	4	8	9	10	11	12	15

The dark shaded area is the sorted part of the array which grows by 1 after each pass.
(Note we are not looking at efficiency of sorting algorithms or optimised solutions here – you will consider that in other subjects such as Algorithms and Data Structures ☺)

Task 3

Show the passes in using bubble sort to sort the following array of strings in alphabetical order

	0	1	2	3	4
initial array:	bee	spider	wasp	tick	ant
after first pass:					
after second pass:					
after third pass:					
after fourth pass:					

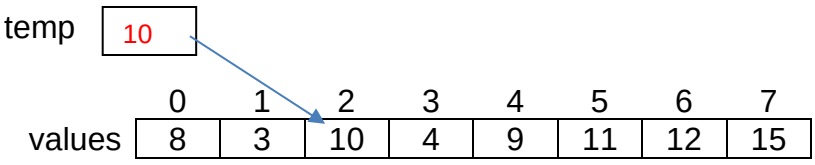
Algorithm for bubble sort

FOR pass = 1 to size of array – 1 (inclusive) DO
    Compare adjacent elements and swap if needed in the remaining unsorted array
ENDFOR

Refining the pseudo code:

FOR pass = 1 to size of array – 1 (inclusive) DO
    FOR index = 0 to size of array – 1 – pass (inclusive) DO
        IF values[index] > values[index+1] THEN
            Swap the elements
        ENDIF
    ENDFOR
ENDFOR

So sometimes we need to swap elements, for example, swap the values at index 1 and index 2



1. Copy the first value into the temporary store
2. Copy the second value into the position of the first value
3. Copy the temporary value into the position of the second value

Task 4

Complete the code for the method swap

```
private void swap(int index1, int index2)
{




}
```

Task 5

Complete the code for the method bubbleSort

```
public void bubbleSort()
{









}
```

Note:
Skeleton code relevant to this workshop (**`IntArray.java`** and **`IntArrayTest.java`**) can be copied from the csilib area into a suitable directory in your own latcs account if you wish to test your code.

```
cp  /home/1st/csilib/cse1oof/ws09/*  .
```

Optional Selection sort

The array values are in any order and we want to rearrange them into order from minimum to maximum (ascending order)

So we repeatedly select the maximum element from the remaining unsorted (front) section of the array and swap it with the last element in the unsorted part of the array

	0	1	2	3	4	5	6	7
initial array:	15	12	8	10	3	4	9	11
after first pass:	11	12	8	10	3	4	9	15
after second pass:	11	9	8	10	3	4	12	15
after third pass:	4	9	8	10	3	11	12	15
after fourth pass:	4	9	8	3	10	11	12	15
after fifth pass:	4	3	8	9	10	11	12	15
after sixth pass:	4	3	8	9	10	11	12	15
after seventh pass:	3	4	8	9	10	11	12	15

The dark shaded area is the sorted part of the array which grows by 1 after each pass.

Optional Task 6

Show the steps in using selection sort to sort the following array of strings in alphabetical order

	0	1	2	3	4
initial array:	bee	spider	wasp	tick	ant
after first pass:					
after second pass:					
after third pass:					
after fourth pass:					

Optional Algorithm for selection sort (Challenging!)

FOR endIndex = size of array – 1 to 1 (inclusive) DO
    Find index of maximum element in array from 0 to endIndex
    Swap maximum element and element at endIndex
ENDFOR

Method to find the index of the maximum element (between low and high say):

Set indexOfBiggest to low
FOR i = low + 1 to high (inclusive)
    IF the element at i is greater than the element at indexOfBiggest THEN
        Set indexOfBiggest to i
    ENDIF
ENDFOR
Return indexOfBiggest

Optional Task 7 (Challenging!)

Complete the code for the methods indexOfMaximum and selectionSort

```
private int indexOfMaximum(int low, int high)
{









}

public void selectionSort()
{









}
```