Software and Programming 1

Week 9 Lab -
Use of arrays of objects and bubble sort algorithm
Lab 9 Objectives

Array Reversal and Sorting

1. Complete the Cycle & CycleTest classes from session six.

2. Extend Cycle Test to reverse the array of objects.

3. Write a class the implements the Bubble Sort algorithm – this method sorts by comparing adjacent elements, swapping them if necessary, while moving from one end of the array to the other. The method continues by looping through the array multiple times until the element values are correctly sorted and no more swaps are made. The method keeps track of changes with a Boolean value, which is set to true if a swap is made. The method stops looping by checking the Boolean each time the loop iterates and checking to see if it is false.
Exercises 1 and 2 using the classes: Cycle and CycleTest

1. Complete the implementation of the class `CycleTest` that declares and creates an array of 10 Cycles from week 6 lab class, if you have not done so. The program then fills up the array by creating 10 instances of cycles and parks them at the docking stations specified by the user input. Fully test all the methods in class `Cycle`.

2. Extend the class CycleTest where you are now given an array of instances of Cycle, reverse the order of elements (in the same array, without creating a new copy of it).
Exercise 3

3. Implement a class **BubbleSort** to sort an array of String objects. The program fills an array of ten String objects by reading in user input at the keyboard.

The array is then sorted using the bubble sort algorithm. See ‘SP1: More on Arrays’, slides 10-12.

**Hint:** use the `compareTo()` method to compare strings in “lexicographic” order. Remember a value less than 0 is returned if the first string, s1, is less than the second string, s2, in this expression: `s1.compareTo(s2)`.

If the strings are equal, a value 0 is returned, and a value greater than 0 otherwise.
Bubble Sort: method

1. Set Boolean value to false
2. Use a loop to iterate through array multiple times until swaps are completed - start with the index of the second element within the loop

//inside loop

3. With an if condition, check this index's value against the previous index's value. If previous index value is greater than the current one - swap values as below:

1. store previous index value in a temporary variable
2. move current value into the previous element
3. move the temporary variable value back into the array at current index
4. as swap has happened change value of Boolean
Bubble Sort: Example

pass 1:  (horse  cat duck dog)  ->  (cat  horse duck dog)  swap
        (cat horse duck dog)  ->  (cat  duck horse dog)  swap
        (cat duck horse dog)  ->  (cat  duck dog horse)  swap

pass 2:  (cat duck dog horse)  ->  (cat duck dog horse)  no swap
        (cat duck dog horse)  ->  (cat dog duck horse)  swap
        (cat dog duck horse)  ->  (cat dog duck horse)  no swap

The array is already sorted but the algorithm does not know if it is completed. It needs one whole pass without any swap to know it is sorted.
import java.util.Scanner;

public class BubbleSort
{
    public static void main(String[] args)
    {
        Scanner input = new Scanner(System.in);
        /* To Do: Write more code to
           (i) declare an array data of 10 String objects.
           (ii) fill up the array data by reading in string values
                at keyboard.
        */
Anatomy of Class BubbleSort (2)

/* To Do: 
   Write the bubble sort code in this part of your program.
   Code is supplied on slide 9, but using method on slide 5 try and implement it yourself first */

System.out.println("Sorted data: ");

/* To Do: Write more code to use the enhanced for loop to visit all elements of the sorted array data and prints out each element. */

} // end of main

} // end of class BubbleSort
Bubble Sort: implementation

/* assume the array data contains the String objects */
boolean swapped = false;
do {
    swapped = false;
    for (int i = 1; i < data.length; i++)
    {
        if ( (data[i-1].compareTo(data[i])) > 0 )
        {
            /* first string is greater than second string lexicographically */
            String t = data[i-1];
            data[i-1] = data[i];
            data[i] = t;
            swapped = true;
        } // end if
    } // end for
} while (swapped);