PROGRAMMING ASSIGNMENT 3:

Read
Deitel: Ch 6, 14, pg 770 Ex 19.5, App F, (skip 6.6, 6.11-6.14)
Liang: Ch 4, 7, 23.3, App H

Programming: Name your program P3.java

DUE: Friday, July 14, 2017 @ 6am

You will write an application program to sort arrays of doubles and Strings using comparison methods, class String, and finding the averages. Write the methods as shown below.

NOTE: NO error checking of input data is required with corresponding error messages because data is type "Strings". In production programming, ALL software is error checked.

```
import java.util.Scanner; // Scanner
public class P3 {

    private static double[] scores; // Array of scores of students
    private static String[] names; // Array of names of students
    private static Scanner scan = new Scanner(System.in); // Read input from keyboard

    public static void main(String[] args) {
        int numStudents; // Number of students
        int typeOfSort; // Which type of sort

        do {
            typeOfSort = menu(); // Ascend/Descend?
            if (typeOfSort == -1) continue; // Exit on -1

            System.out.print("Enter the number of students: ");
            numStudents = scan.nextInt();
            names = new String[numStudents];
            scores = new double[numStudents];

            readInput(); // Read scores and names
            switch(typeOfSort) {
                case 1: selectionAscendScore(); // Ascend sort scores
                break;
                case 2: bubbleDescendScore(); // Descend sort scores
                break;
                case 3: bubbleAscendString(); // Ascend sort names
                break;
                case 4: bubbleDescendString(); // Descend sort names
                break;
                case 5: aboveAverage(); // Display scores above average
                break;
                case 6: belowAverage(); // Display scores below average
                break;
            }
            prt(); // Print scores and names
        } while(...);

        scan.close(); // Close this scanner
        System.exit(0); // Repeat until -1 typeOfSort
    }
}
```

(class P3 continued on next page)
CSE 11 Programming Assignments SS 1 2017

(class P3 continued from previous page)

```java
public static int menu() { /* Read type of sort. Return type of sort on exit. */
    public static void readInput() {
        /* Read names and scores into arrays with loop */
        System.out.print("Enter student names ...");
        names[i] = scan.next();
    }

    public static void selectionAscendScores() {
        /* Ascending selection sort on scores, calls swap() */
    }

    public static void bubbleDescendScores() { /* Descending bubble sort on scores */
        /* swap( i-1, j); */
    }

    public static void bubbleAscendString() { /* Descending bubble sort on names */
        /* if( names[j].compareTo(names[j]) > 0 ) */
    }

    public static void bubbleDescendString() {
        /* Descending bubble sort on names */
    }

    public static void swap(int j_1, int j) {
        /* swap elements */
    }

    public static void aboveAverage() {
        /* Find average and display ascending order names/scores above average */
    }

    public static void belowAverage() { /* Find average, display descending scores below average and names ascend */
    }

    public static void print() { /* Print elements of array */
    }
}
```

a) ALL methods are public static with a short header comment for describing purpose.
b) In `main()`, read in number of students, the size of the 2 arrays.
c) No instance variables are needed. Use code as given above:

```java
private static double [] scores; // Array of scores of students
private static String [] names; // Array of names of students
private static Scanner scan = new Scanner(System.in); // Read input from keyboard
```
d) In `selectionAscendScores()` sort each score in ascending sorted order using selection sort as given below and in Liang textbook, Listing 7.8. Modify using code in step c). Hint: No parameters.

```java
public static void selectionSort(double[] list)
```

e) In `bubbleDescendScores()` use the algorithm given in Java handout on first day. Call swap() passing the indices of the elements to swap and the 2 arrays.
f) In `bubbleAscendString()` is similar to `bubbleDescendScores()` except using `hashCode()` and `compareTo()`.
   
   e.g. `names[j-1].hashCode() > names[j].hashCode()`

f) In `bubbleDescendString()` is similar to `bubbleAscendString()` except descending order.

h) `aboveAverage()` calls `selectionAscendScores()` and `bubbleAscendString()`, finds average score, and displays ascending scores above and equal to the average with names in ascending sorted order.

i) `belowAverage()` calls `bubbleDescendScores()` and `bubbleAscendString()`, finds average score, and displays scores in descending order below and equal to the average with names in ascending sorted order.

Remember arrays have a member variable "length" and Strings have a method "length()".

**PA#3 SAMPLE OUTPUT:**  (SAMPLE INPUT is in **bold** type)

```
Choose type of sort (1 2 3 4 5 6): 1
Enter the number of students: 3
Enter student name #1: ann
Enter student score #1: 100
Enter student name #2: bob
Enter student score #2: 50
Enter student name #3: cal
Enter student score #3: 30
  cal 30.0
  bob 50.0
  ann 100.0
```

```
Choose type of sort (1 2 3 4 5 6): 3
Enter the number of students: 3
Enter student name #1: ann
Enter student score #1: 100
Enter student name #2: An
Enter student score #2: 200
Enter student name #3: ANN
Enter student score #3: 300
  ANN 300.0
  An 200.0
  ann 100.0
```

Choose type of sort (1 2 3 4 5 6): 4

**PA#3 SAMPLE OUTPUT** (continued on next page)
Enter the number of students: 4
Enter student name #1: ANN
Enter student score #1: 1
Enter student name #2: Ann
Enter student score #2: 2
Enter student name #3: ann
Enter student score #3: 3
Enter student name #4: aj
Enter student score #4: 4

ann 3.0
aj 4.0
Ann 2.0
ANN 1.0

1 Ascending Sort of Scores
2 Descending Sort of Scores
3 Ascending Sort of Names
4 Descending Sort of Names
5 Above Average Names and Scores
6 Below Average Names and Scores
-1 EXIT

Choose type of sort (1 2 3 4 5 6): 5

Enter the number of students: 3
Enter student name #1: Jen
Enter student score #1: 87
Enter student name #2: JEN
Enter student score #2: 87
Enter student name #3: jon
Enter student score #3: 2

AVERAGE score: 58.67
ABOVE AVERAGE STUDENTS
JEN 87.0
Jen 87.0

1 Ascending Sort of Scores
2 Descending Sort of Scores
3 Ascending Sort of Names
4 Descending Sort of Names
5 Above Average Names and Scores
6 Below Average Names and Scores
-1 EXIT

Choose type of sort (1 2 3 4 5 6): 6

Enter the number of students: 3
Enter student name #1: jo
Enter student score #1: 88
Enter student name #2: JOJO
Enter student score #2: 88
Enter student name #3: Jon
Enter student score #3: 100

AVERAGE score: 92.00
BELOW AVERAGE STUDENTS
JOJO 88.0
jo 88.0

1 Ascending Sort of Scores
2 Descending Sort of Scores
3 Ascending Sort of Names
4 Descending Sort of Names
5 Above Average Names and Scores
6 Below Average Names and Scores
-1 EXIT

Choose type of sort (1 2 3 4): -1

Verify you TURNED-IN your work by re-typing “bundleP3”