3. 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, 8, App H
Programming: Name your program P3.java
DUE: Saturday, August 19, 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.

```java
import java.util.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 equals -1, then 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: selectionAscendScores(); // Ascend sort scores
                        prt(); // Print scores and names
                     break;
                case 2: bubbleDescendScores(); prt(); // Descend sort scores and print
                        break;
                case 3: bubbleAscendString(); prt(); // Ascend sort names and print
                        break;
                case 4: bubbleDescendString(); prt(); // Descend sort names and print
                        break;
                case 5: aboveAverage(); // Display scores above average
                        break;
                case 6: belowAverage(); // Display scores below average
                        break;
            }
        } while (...); // Repeat until -1 typeOfSort
        scan.close(); // Close this scanner
    }
}
```

class P3 (continued on next page)
class P3 (continued from previous page)

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

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). Call bubbleAscendString() for equivalent scores. Hint: No parameters.

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

```java
int i, j, minIndex;
double min;
for( i = 0 ; i < list.length ; i++ )
{
    // Find the minimum in the list[i...list.length-1]
    min = list[i];
    minIndex = i;
    for( j = i + 1; j < list.length; j++ )
    {
        if( min > list[j] )
        {
            min = list[j];
            minIndex = j;
        }
    }
    if( minIndex != i ) // Swap list[i] with list[minIndex] if necessary
    {
        list[ minIndex ] = list[i];
        list[i] = min;
    }
```
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. Call `bubbleDescendString()` for names descending order of equivalent scores.

f) In `bubbleAscendString()` is similar to `bubbleDescendScores()`.

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

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

i) `belowAverage()` calls `bubbleDescendScores()`, finds average score, and displays scores below and equal to the average with their names in descending 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)

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

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

Enter the number of students:  4
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
Enter student name #4:  DAN
Enter student score #4:  50
  cal  30.0
  DAN  50.0
  bob  50.0
  ann  100.0

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

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

Enter the number of students:  4
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
Enter student name #4:  DAN
Enter student score #4:  50
  ann  100.0
  bob  50.0
  DAN  50.0
  cal  30.0

PA#3 SAMPLE OUTPUT (continued on next page)
PA#3 SAMPLE OUTPUT (continued):  (SAMPLE INPUT is in bold type)

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

Choose type of sort (1 2 3 4 5 6):  3
Enter the number of students:  4
Enter student name #1:  ann
Enter student score #1:  200
Enter student name #2:  bob
Enter student score #2:  50
Enter student name #3:  cal
Enter student score #3:  30
Enter student name #4:  ANN
Enter student score #4:  100

ANN  100.0
ann  200.0
bob  50.0
     cal  30.0

---

Choose type of sort (1 2 3 4 5 6):  4
Enter the number of students:  4
Enter student name #1:  ANN
Enter student score #1:  200
Enter student name #2:  ann
Enter student score #2:  200
Enter student name #3:  bob
Enter student score #3:  50
Enter student name #4:  cal
Enter student score #4:  30

cal  30.0
bob  50.0
ann  200.0
ANN  200.0

PA#3 SAMPLE OUTPUT (continued on next page)
PA#3 SAMPLE OUTPUT (continued): (SAMPLE INPUT is in **bold** type)

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

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

Enter the number of students: **4**

Enter student name #1: **cal**
Enter student score #1: **6**
Enter student name #2: **bob**
Enter student score #2: **7**
Enter student name #3: **ann**
Enter student score #3: **2**
Enter student name #4: **DAN**
Enter student score #4: **5**

AVERAGE score: **5.00**

**ABOVE AVERAGE STUDENTS**

DAN  5.0
cal  6.0
bob  7.0

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

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

Enter the number of students: **4**

Enter student name #1: **cal**
Enter student score #1: **2**
Enter student name #2: **BOB**
Enter student score #2: **1**
Enter student name #3: **ann**
Enter student score #3: **9**
Enter student name #4: **CAL**
Enter student score #4: **2**

AVERAGE score: **3.50**

**BELOW AVERAGE STUDENTS**

cal  2.0
CAL  2.0
BOB  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 Scores with Names
6  Below Average Scores with Names
-1  EXIT
```

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

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