Project 2
Examination Statistics

Distribute on September 25, 2023
Due before October 22, 2023 (Sunday) at 12:00 midnight

Learning Outcomes ((CLO) vs (SO) Mapping)

- Recognize the software and hardware components of a computer system (1)vs(6)
- Recognize and apply the software development phases (2)vs(6)
- Utilize Java syntax in fundamental programming algorithms (3)vs(1)
- Recognize and apply the various input and output devices in programming (4)vs(2)
- Recognize and apply the various control structures (5)vs(1)
- Design and implement user-defined methods (5,4,3,2)vs(1,2,6)

Objectives

In completing this project, you will gain experience with the following Java features:
- the Scanner class
- selection structures
- loops
- file input-output

Problem Statement

Processing large amounts of data to obtain statistical information is a frequent computer application in many areas. The most frequently calculated parameters are the average, the median, and the population standard deviation.
In this project, you implement a program that

- reads examination scores from a file
- determines the total number of scores in the file
- calculates the average score
- determines the maximum score and the minimum score
- calculates the population standard deviation
- determines the number of examinations in each of the letter grade groups A, B, C, D, F
- makes the output message displayed on the console, and writes the output to a file, as well

Analysis and Requirements

Input

- named constants to store the lower score limits for the letter grade groups, these are 90, 80, 65, 50, and 0 in the order of grades A, B, C, D, F
- an undetermined number of exam scores stored on an external input file; each number is an integer between 0 and 100
- the name of the input file to be solicited from the user on the console

Output

The output will contain the following data:

- The total number of examinations
- The average score
- The population standard deviation of the scores
- The minimum score
- The maximum score
- The number as well as the percentage of scores in each letter grade group

A template to be followed in displaying the output is shown below (Figure 1). Your actual output may have other numerical values.

```
Exam Statistics

Total number of (given) Scores:  77
Total number of valid scores:  70


Considering only the given valid scores:
Average score (of valid scores):  69.46
Population standard deviation of the valid scores:  16.52

# of A, 85-100:      4          5.71%
# of B, 75--84:     20         28.57%
# of C, 65--74:     25         35.71%
# of D, 50--64:     10         14.29%
# of F, 00--49:     11         15.71%

Minimum score:  16
Maximum score:  98
```

Figure 1

Note that all decimal numbers must be rounded to two decimals, and all the score listing lines must contain 10 numbers (except maybe the last lines) in the exam_scores.txt file. However, in the scores.txt file, all the scores listing lines may not contain an equal number of numbers in the scores.txt file. So is the to_test.txt file.

Formulas needed

Average: The average score is the sum of the individual scores divided by the total number of scores. Avoid integer division. If N is the total number of scores and x1, x2, …, xN are the score values then

$$average = (x1+x2+\ldots+xN)/N$$

Population standard deviation: The difference between each score and the average squared, added, divided by N, and taken square root of the result.

$$psd * psd = ((average - x1)^2+(average - x2)^2+\ldots+(average - xN)^2)/N$$

Design

For this project, you shall define a class named **`ExamStatistics.`**

Add four **static** named constant fields to store the lower limits of the grade groups, see in Input above. These variables are declared in the class but outside of the main method.

Within the main method, create commented places for codes responsible for various tasks.

//Declare variables

You will need

- six integer variables for counting; one counter variable is used to count the total number of scores on the file, and one additional variable is needed to count the scores for each of the grades A, B, C, D, and F
- three integer variables minScore, maxScore, nextScore; minScore and maxScore are used to store the lowest and highest score values, nextScore stores the score currently read from the file
- three double variables to store sum, average and psd
- two String variables to store the output message and the file name

// Declare and instantiate a Scanner object for console reading
// Declare and instantiate a File object to the file name solicited and received from the
// console
// Check if the file exists and repeat file name solicitation until the name is accepted (a
// loop will be used)
// Declare and instantiate another Scanner object **reader** to read data from the file
// Run a while loop to read, count, and sum the scores ("running total"); the loop must also
// have logic to determine maxScore, and minScore, and must count the occurrences of
scores // in the grade groups; input values are checked, wrong input is not counted and
they are // ignored in the processing (the loop continues at the next iteration if wrong input
was // found). Wrong input does not update any of the counter variables, nor the sum,
// maxScore, and minScore variables.

// Compute the average
// Re-instantiate the file reader Scanner object
// Run a for loop that makes the summation for psd, see the formula for psd
// Compute psd
// Compose the output message
// Display the message on the console

// Instantiate a PrintWriter object and write the output to a file named ExamStatFile

Implementation Requirements and Hints

- You must have a comment block preceding the header of each of your classes that has the following content:

```
/*
 * <your name>
 * CS 16000-01 – 02/03, Fall Semester 2023
 * (Specify your lab section only)
 * Project 2: Examination Statistics
 *
 * Description. <Summarize the purpose of the class here.>
 *
 */
```

- Having the name of the input file read from the console, the name shall be saved in a variable, say inputFileName
- To validate the file name
  - (i) declare a File object
  - (ii) run a while loop which solicits, reads, and saves a candidate for the file name
  - (iii) instantiate the File object to the file name
  - (iv) check if the file exists, if it does not, continue the loop, otherwise, stop the loop
- In the while loop that reads the scores from the file
  - (i) Save the currently read input in **nextScore**
  - (ii) Validate the input; wrong input is not processed, the loop turns to the next iteration
  - (iii) Update the total counter
  - (iv) Check if nextScore is greater than maxScore, if so update maxScore with nextScore
  - (v) Repeat the checking for minScore
  - (vi) Apply a nested if else if structure to determine the grade group relevant for **nextScore** and update the group counter
  - (vii) Add nextScore to sum
- After the loop, compute the average by applying the average formula; care for the integer division, the average must be exact
- Note that the score data are not stored internally by the program, and to compute the standard deviation (psd), the average must already be known. Therefore, average and psd cannot be simultaneously determined in one file reading process. To compute the value of psd, the file **inputFileName** must be read once more by a new Scanner
- As for the second reading of the file, a **for** loop is applicable since the total number of admissible items in the file is known. Input evaluation is still necessary to discard the wrong inputs. You may decide if you want to use a while loop for the second reading, or if you want to apply for a loop.

- Having the second loop completed and psd determined, the output message containing the obtained results must be built; decimal numbers in the output must be formatted and rounded to two digits after the point
- Declare and instantiate a PrintWriter object to write the output to a file named ScoreStatistics.txt
- Print the output to the console and write output to the file as required

- **Start your work on this project without delay.**

Testing

Run your code with the supplied text file to_test.txt and use the data in Figure 2 for correctness. If the program is working correctly, run the program with the input file exam_scores.txt as well.

```
Exam Statistics

Total number of (given) Scores:  25
Total number of valid scores:  17


Considering only the given valid scores:
Average score (of valid scores):  66.94
Population standard deviation of the valid scores:  27.03

# of A, 85-100:     4        23.53%
# of B, 75--84:     3        17.65%
# of C, 65--74:     3        17.65%
# of D, 50--64:     3        17.65%
# of F, 00--49:     4        23.53%

Minimum score:   0
Maximum score: 100
```

Figure 2

Your program must be tested with all three supplied files of data which are as follows:
to_test.txt,
scores.txt, and
exam_scores.txt.

The outputs of these three tests must be stored in
to_testExamStatFile.txt,
exam_scoresExamStatFile.txt, and
ExamStatFile.txt, respectively.

Evaluation

Documentation and style: 20 points.

Your program must conform to the Computer Science Department’s Java Documentation and Style Requirements. Emphasis will be placed on having the required banner and internal comments, indentation, and overall stylish appearance. Comments must be written with correct grammar and spelling. Documentation and style weigh heavily in the score, do not neglect them.

Correctness: 80 points.

These points will be allocated as follows:	points
1. Correct declarations of all variables	05
2. Correct instantiation of all objects	05
3. Correct implementation of file name solicitation	10
4. Correct implementation of the while loop reading the file first	25
5. Correct implementation of the loop of second reading	15
6. Correct construction of the output message to follow the template	10
7. Correct implementation of PrintWriter	10
Total correctness	80
Total for assignment	**100**