Computer Science 1 : Java Programming

Dr. David Cline

Program: Grade Stats

In this program you will create a utility to calculate and display various statistics about the grades of a class. In particular, you will read a CSV file (comma separated value) that stores the grades for a class, and then print out various statistics, either for the whole class, individual assignments, or individual students.

Things you will learn

- Robustly parsing simple text files
- Defining your own objects and using them in a program
- Handling multiple input commands

Specification

When your program starts, it should print the program title and your name, then load the file given as a command line argument (similar to the sorting assignment). If no file is given, or if the file does not exist, the program should print a usage message and exit.

After loading the input file, the program should go into "command" mode, which allows the user to type in various commands and get results from the system. This is similar to the calculator assignment. You should show some kind of a prompt, such as ">" to the user, and wait for commands. The commands that your system must process are as follows:

1. exit

   Causes the program to exit.

2. help

   Causes the program to print a list of commands accepted by the system. This same message should be printed if the user types an unrecognized command. The program must not crash in this case.

3. roll

   Prints out a list of students from the class (first and last name), along with total points, and final grades (A, B, C, D, F) for each student. These should be properly aligned in columns, so that the grades line up above one another, and the student first name and student last name are left aligned.

4. search [partial name]

Searches for students with a first or last name that match the partial name (partialName is a substring of either the first name or the last name), and prints out a list of these students in the same way as the students command.

5. assignments

   Prints out a list of assignments from the file, along with how many points are possible for each assignment. As with students, the assignments should be properly aligned on columns.

6. student [student name]

   Prints a report for the student. If the student does not exist, it prints a message stating that the student does not exist, and continues without crashing. The report should contain the following information:

   1. Student name
   2. A list of all assignments, and the grades the student achieved for them and points possible
   3. Overall points made in the course and total points possible.
   4. Final letter grade for the student, given the grading scale

      A >= 90%
      B >= 80%, < 90%
      C >= 70%, < 80%
      D >= 60%, < 70%
      F < 60%

   5. The report should be formatted reasonably, so that assignments and grades line up in columns, with reasonable spacing.

7. assignment [assignment name]

   Prints a report about a particular assignment, which includes the following information:

   1. The assignment name, and points possible
   2. The low, high, and average for the assignment
   3. How many students achieved different grades (A-F) for the assignment.

   If the assignment does not exist, the program should state this and continue without crashing.

8. report

   Prints a report about overall grades in the class. The report must include the following information:

   1. low, high, and average percentages in the class
   2. how many students achieved the different grades in the class (A-F)

Database file format

A database file for this assignment will be a CSV (comma separated value) text file. The first line will contain the name of the class followed by the section number, followed by the names of all of the assignments in the class, separated by commas. The second line will have the fields "firstName" and "lastName", followed by the number of points possible for each assignment. For example:

```
Philosophy 101,Section 1,essay 1,test 1,essay 2,test 2,final
firstName,lastName,20,50,20,50,100
```

Subsequent lines will all have the same format, and will contain the name of a student, followed by the number of points they achieved on each assignment. Thus, a complete input file might look like the following:

```
Philosophy 101,Section 1,essay 1,test 1,essay 2,test 2,final
firstName,lastName,5,20,5,20,50
Aristotle,Ofathens,4,18.3,3,15,40
Euclid,Elements,3,15,2,10,35
Immanuel,Kant,4.5,18,4,20,48
Benjamin,Franklin,5,20,4,19,49
```

Program defensively. Your program must be able to handle the case where there are blank lines at the end of the file. Your program must be able to handle fractional point values as well as integer values.

Example Operation

The example below shows how your ClassStats program should work. Things typed by the user are shown in bold:

```
java ClassStats philosophy101.txt
Class Stats program by <your name>
loaded 'philosophy101.txt'

> help
      Accepted commands:
      exit
      help
      students
      search [partial name]
      assignments
      report
      student [student name]
      assignment [assignment name]

> students
      Student Grades for Philosophy 101, Section 1
      Total points possible: 100

      First Name      Last Name       Points   Grade
      ----------      ---------       ------   -----
      Aristotle       Ofathens            80     B
      Euclid          Elements            65     D
```

```
      Immanuel        Kant               94     A
      Benjamin        Franklin           97     A

> assignments
      Assignments for Philosophy 101, Section 1

      Assignment     Points
      ----------     ------
      essay 1             5
      test 1             20
      essay 2             5
      test 2             20
      final              50

> search Ben
      First Name      Last Name       Points   Grade
      ----------      ---------       ------   -----
      Benjamin        Franklin            97     A

> report
      Grade breakdown for Philosophy 101, Section 1

      Low:  65%
      High: 97%
      Ave:  84%

      A:   2
      B:   1
      C:   0
      D:   1
      F:   0

> student Benjamin Franklin
      Grades for Benjamin Franklin

      Assignment     Points  Possible
      ----------     ------  --------
      essay 1             5         5
      test 1             20        20
      essay 2             4         5
      test 2             19        20
      final              49        50
      total              97       100

      Final Grade: A

> assignment essay 1
      essay 1:   5 points
      grade breakdown
      A:  1
      B:  2
      C:  1
      D:  0
      F:  0

> exit
```

Directions

1. Implement the GradeStats program as described above.

2. Make sure that your program prints your name when it begins. It should also print the name of the input file when it starts, and state that it loaded it. The input file name must be taken as a command line argument.

3. Make sure to put the proper block comment at the top of your main file, including your name and section number, etc.

4. Make sure that your program follows the class coding standards for indentation, variable naming, etc. Part of your grade will be based on this.

5. Make sure that your program can handle good and bad input files, and good and bad user inputs, without crashing.

6. Compile and test your program on the command line.

7. Pass off your program directly to the instructor or TA and turn in your source code to D2L.

Suggestions

- A good strategy for the program would be to have a main loop that gets a line of input from the user, and then figure out what kind of command you are dealing with. Note that this is as easy as seeing if the input line starts with a given word.

- The names of students and assignments can have spaces in them, and you need to be able to handle that.

- Note also that this program is quite similar in many ways to the calculator in how it handles input and output. It is also similar to the sorting program in how it deals with input files. Think about how you solved the problems related to those programs. You may be able to apply some similar solutions for this program.

- You have to read in and store several types of information, including (1) the name of the class, (2) the names of the assignments, (3) the points possible for all the assignments, (4) the names of all the students, (5) the grades for all the assignments for all the students. Note that most of the items are 1 dimensional structures, but item 5 (the grades for the students) is a 2 dimensional table. Think about what data structures you will use to store all of this information.

- Think about how you will get various quantities. Some of them must be calculated by summing rows in the table of grades, and some of them involve summing columns.

Point Breakdown (20 points total)

Compiles and runs, has name comment	Required for a non-zero score
Handles bad inputs	2 pts
The program prints your name when it starts	2 pts
exit command	2 pts
"help" command	2 pts
students command	2 pts
assignments command	2 pts
report command	2 pts
student command	2 pts
search command	2 pts
assignment command	2 pts