Team Project -- Milk Weight

Project Requirements

- (https://canvas.wisc.edu/courses/202692/pages/learning-outcomes-ateam-project) Complete in a team of 1-5 students (SUMMER 2020: STUDENTS WILL BE WORKING ALONE)
- (https://canvas.wisc.edu/courses/202692/pages/ateam-create-and-join-an-ateam) Create a JavaFX graphic user interface (GUI) program
- Submit screenshots and zip files as instructed for each milestone
- Able to build and run Java FX program (must be JDK 11+ and JavaFX 11.02 for us to run and grade)
- Complete these assignment milestones (%24CANVAS_OBJECT_REFERENCE%24/assignments/g435aec663747740c8ee39211bdae0956)
  - **a1 Milestone1: Design** (https://canvas.wisc.edu/courses/202692/assignments/835006)
  - **a2 Milestone 2: UI** (https://canvas.wisc.edu/courses/202692/assignments/835007)
  - **a3 Milestone 3: Final Project** (https://canvas.wisc.edu/courses/202692/assignments/835008)

Program Requirements

1. Internal data structure:
   1. must store data in an internal data structure
   2. may be user-defined or a Java Collection type
   3. may use multiple internal structures
2. User Interface (UI):
   1. must be a Graphic User interface coded by student using JavaFX library (no build tools allowed)
   2. must be interactive (user must be able to input data and control what happens and is displayed)
3. File I/O:
   1. program must read in data from a csv or json file to populate the internal data structure. Must do this at the start of program or as requested by the user via GUI. If you wish to use a different text-only data file format, ask first.
   2. **(NOT A HARD REQUIREMENT IN SUMMER 2020)** program must be able to save (write) something to a file (txt, csv, json) at the end of the program, or when requested by the user.
      Some file output ideas (implement any, all, or your own idea):
      - Have a way for the user to add data while program is running, and then save that data to file of the required input format.
      - Add a button that allows the user to write summary data out to a file

        - Keep a log of user events in a log and write that to a file, or append to it as the program is running
        - Create [Print Report] buttons for each different report (Annual, Monthly, by Farm, etc.) that you create, then have it print the results to a text-only (.txt) file.

4. Documentation (help):
    1. The source code must be modular, use interfaces, classes, methods, and code blocks to break the problem into small, easy to implement units.
    2. The UI must provide sufficient instructions and help tips for new users to navigate.
    3. The source code must have interface, class, and method headers and sufficient inline comments for a new developer (TA/Grader) to navigate and maintain the program source.
5. Testing
    1. Strictly speaking, testing is not required. But, if you do write test suites for any parts of your program, you can earn some credit for that work if your final project does not work as you designed and imagine.
    2. If you wish to use a data structure that you have implemented, you may do so. Be sure to document where the structure originated and include the test files for that structure.

Overview for Milk Weights Program

Goal : Create an interactive data visualizer program with a graphic user interface (GUI) to help a local farm analyze and report on the milk that has been sold to them by their farmers. The data is in one file per month-year, so a year's worth of data is in 12 files that must be read into the program and used to display results and print reports.

Must be able to construct from information read from multiple files and also write current data (that was added interactively by the user) to a file.

At Program launch

When the program is run, a GUI interface is displayed to the user.

It must be obvious to the user how to interact with your program. For example, the user must see or be instructed on how to add data, read input from a file as well as how to enter new data interactively. Use labels, context-sensitive help pop-up messages, what and where to click, select, type, etc.

Use standard JavaFX layout managers and controls like Border, HBox, VBox, GridPane, Button, Labels, TextFields, etc. (you may use other JavaFX components too, these are just reminders about the types we demonstrated).

Must not use third party packages or SceneBuilder to design the interface or provide UI controls.

The code to create and show GUI must be hard-coded by you. If you find code fragments and wish to use them, be sure to document (comment with URL) the source of any code fragments you use.

Once the program is running

The user must be able to:

- Must be able to read milk weight data that is stored in the given input data file formats (see **csv.zip (https://canvas.wisc.edu/courses/202692/files/13076895/download?wrap=1)** for samples, but do not hard-code as we must be able load our own csv data files). Do not limit the farm id to those you know from sample. Assume that some user will have their own farm ids that may be different. Accept any string as a farm id.
- [optional] Must be able to write internal data to a file with same format as input files. This will allow you to use your program to create new data for loading on a later run.
- Must handle bad user input without crashing and with reasonable error messages for the user. Note: it does not have to accept and fix bad data, and may end the program in most severe cases, but it must be clear to the user what went wrong.
- [optional] Must allow the user to add/edit/remove milk weight information for the farms for each month in the current data set
- [optional] Must be able add data via GUI for a new month and or a new farm (allow the user to enter the farm ID, and save and store as all CAPS).
- Must display statistics as described below for individual farms and any given month
- The UI components must be dynamic with only appropriate user interface controls being active if they apply.
- Must hide/show user controls as appropriate.
- The final design must be modular and support JUnit testing of independent data and computation components (this is a requirement even if you do not write the unit tests for your types).
- The internal data structure must be efficient enough for any required query options.
- Must show/hide/compute summary information about milk received and the total share of net profit.

Input File format

Copy **csv.zip (https://canvas.wisc.edu/courses/202692/files/13076895/download?wrap=1)** into your project and unzip to find the following data sets. For any given sample run chose all file in the same dataset (folder) :

- small - just a few farms worth of data for one year
- medium - a different set of more farms for one year
- large - a different set of many farms for one year

- missing - data files that are missing values - so you can see how your program handles missing data
- error - has unexpected errors - so you can see how you program handles unexpected names, data, types for the weight

Output

must be able to write data to a file (no strict requirement), and display computed results to the user via the GUI (charts would be nice too, but data tables are accepted)

the user must be able to choose which way (write to file or display) and be able to select date ranges, months, or years

the details of the output are up to you, but the value must match our expect for any date range and data set chosen

FARM REPORT [required, get at least this one working]

Prompt user for a farm id and year (or use all available data).

Then, display the total milk weight and percent of the total of all farm for each month.

Sort, the list by month number 1-12, show total weight, then that farm's percent of the total milk received for each month.

ANNUAL REPORT [next most important to get working]

Ask for year.

Then display list of total weight and percent of total weight of all farms by farm for the year.

Sort by Farm ID, or you can allow the user to select display ascending or descending by weight.

MONTHLY REPORT [next most important to get working]

Ask for year and month.

Then, display a list of totals and percent of total by farm.

The list must be sorted by Farm ID, or you can prompt for ascending or descending by weight.

DATE RANGE REPORT [least points, but still worth some if you can get this working]

Prompt user for start date (year-month-day) and end month-day.

Then, display the total milk weight per farm and the percentage of the total for each farm over that date range.

The list must be sorted by Farm ID, or you can prompt for ascending or descending order by weight or percentage.

GUI

- must allow the user to add data by reading in from a file
- must allow the user to add/edit/remove existing data for a given farm, year, month, and day
- must allow the user to display min, max, average, by month for user specified farm and year
- must allow the user to display min, max, average, for all farms for user-specified month and year
- must include each farm's percent of the total (known as their share) for any display showing all farms
- must allow user to display each farm's share (% of milk for given month or year) of net sales

Draft design doc for students

Students should keep the following areas in their mind while designing their design doc (a1)

- object-oriented design
- interfaces if possible to help divide work
- simple classes to dev and test
- design components (methods) that can call each other to create different output variations
- UML class diagrams
- find a way to allow user to select and display required operation

Getting Started

(%24CANVAS_OBJECT_REFERENCE%24/assignments/g435aec663747740c8ee39211bdae0956)

1. Read this assignment
    1. Brainstorm some ideas for what your program's user interface (UI) would or could look like.
    2. What data structures are needed for each required operation (data report, data view) UI component?

    3. What are the interactions from the user, between all UI controls?
2. Work on the back-end data structure(s) needed
    1. What data structures will provide the functionality and have reasonable time and space complexity analysis for the required operations.
    2. Implement the least dependent classes and unit test them before using instances of them in other classes.
    3. Make and test back-end data structures work from the command-line before trying to use in your JavaFX program.
3. Learn some JavaFX to get a graphic user interface (GUI) to appear
    1. Complete JavaFX assignment (p2)
    2. Create a JavaFX project (github.com would be a great way to host and work on this project)
    3. Get a GUI to pop up
    4. Get some UI controls to show up
4. **Complete the design.pdf of your program and submit it** (a1 Milestone 2: Design (https://canvas.wisc.edu/courses/202692/assignments/835006))
5. Get the basics of your JavaFX GUI to show up in your JavaFX project
    1. choose a layout for your scene
    2. add controls to main scene
    3. add scene(s) to main stage
    4. make the main (primary) stage visible
6. Get a file chooser to work or other, so user can select file (or multiple files) and have data uploaded to your program.
7. Get other GUI components added
8. Get your GUI to display and show UI controls
9. **Submit:** a2 Milestone 2: GUI (https://canvas.wisc.edu/courses/202692/assignments/835007)
10. Add functionality your GUI program so that UI controls work and get the data as needed from the back-end data structures you created.
11. Ensure that all parts work as expected.
12. Focus on correct functionality first.
13. Test efficiency for large data sets when all works for smaller data sets. (https://canvas.wisc.edu/courses/202692/pages/javafx-compile-and-run-your-java-fx-program-on-a-cs-linux-workstation)
14. **Submit:** a3 Milestone 3: Final Project (https://canvas.wisc.edu/courses/202692/assignments/835008)

Tips

Start early.

Watch for announcements regarding changes to the scope of the problem.

Use git/GitHub to help facilitate work and store backups.

Get simple parts working as quickly as possible. Don't try to do it all at once. This project should and will take time, put some time into it daily for best result. Watch for our design to be released as well as modifications to the requirements as we get feedback and questions on the assignment.

If you cannot complete all requirements, document what you are able to complete. Since you will be working on the project alone, we will adapt requirements to this new situation. But, we do want to see what everyone is able to design for a user to view these data files in various ways.

Meet with TAs and get feedback early and often on your ideas. We are here to help. Let us help you sooner rather than later.