METRO STATE UNIVERSITY

ICS 141 - 02: Programming with Objects

Spring 2023

Assignment 6: File I/O, methods, and arrays

Out: Wednesday, April, 12th, 2023

Due: Wednesday, April, 19th, 2023

Total points: 60

Objective

To receive credit, you MUST create a video recording less than 10 minutes with your face on camera describing your code step by step. Upload the java package as well. In this assignment, you will write a Java application to analyze data about users who write movie reviews on a certain movie review web site. The data is stored in a text file where each line in the file has five tab-separated fields that represent: user identifier, age, sex, occupation, and zip code. For example, the following are the first few lines of the file:

```
1    24 M  technician 85711
2    53 F  other 94043
3    23 M  writer     32067
4    24 M  technician 43537
5    33 F  other 15213
6    42 M  executive 98101
7    57 M  administrator 91344
8    36 M  administrator 05201
9    29 M  student 01002
```

In order to process this data, you will upload the data from the text file into an array of `User` objects. Then, you will write several methods to process the array of objects. Basically, these are the implementation steps that you will follow:

1- Define the `User` class that is used to represent one user.
2- Declare an array that can store up to 200 `User` objects' references (because there are 200 lines in the file).
3- Read the text file line-by-line then, for each line, use the five fields in that line to instantiate a `User` object and insert a reference to that object in the array.
4- Write four methods where each method takes the user array as input and perform some analysis on the data, for example, find the average user age or the number of 'student' users.

Implementation steps

Step 1: Implement `User` class

As explained above, you will read the data from the text file and store it in an array of `User` objects where each line in the text file is used to construct one `User` object. Hence, the first step is to define the `User` class.

Start by creating an Eclipse project, called `Assignment6`. Then, create a class, called `User`, inside `Assignment6` project. As per the data file description above, the `User` object is to be defined using the following five **`private`** instance variables:

- `userID`: of type `int`
- `age`: of type `int`
- `sex`: of type `String`
- `occupation`: of type `String`
- `zipCode`: of type `String`

Implement the following methods in the `User` class:

- getters and setters for all the instance variables.
- `toString` method that returns a `String` representation of a `User` where all the instance variables are in one line and separated by tabs.

Step 2: Upload data from the text file into an array of `User` objects

- The input data is in a file called '`Assignment-6-user-data.txt`'
- Create another class in your project, called `UserDataProcessingDriver`, that includes only the `main` method.
- In the `main` method, declare an array, called `usersArr`, that can hold up to 200 `User`s.
- In the `main` method, write a loop to read the text file. Note that, similar to Lab 9, you need to read one field from the file at a time using the appropriate `next` method. Basically, inside your '`while`' loop, you will read the five fields in each row in the following order: `nextInt`, `nextInt`, `next`, `next`, and `nextLine`.
- Inside the '`while`' loop, and after reading the five fields, instantiate a `User` object using the five fields and insert that object in `usersArr`.
- After the `while` loop completes reading the file, write a '`for`' loop to print on the screen all objects in `usersArr` to make sure that the data to uploaded correctly.

Step 3: Methods to process the users array

In this part, you are asked to implement the following four methods in the `UserDataProcessingDriver` file. Then, test the methods by calling them from the `main` method.

1- `avgAge`: a method that takes an array of `User` objects as input parameter and returns as output the average age of all users in the array.

2- `countOccupation`: a method that takes two input parameters where the first input is an array of `User` objects and the second parameter is a `String` that represents an occupation. The method then returns as output the number of users whose occupation is the same as the input occupation. For example, the call `countOccupation(usersArr,"educator")` returns how many users are there in the array `usersArr` whose occupation is 'educator'.

3- `avgOccupationAge`: a method that takes two input parameters where the first input is an array of `User` objects and the second parameter is a `String` that represents an occupation. The method then returns as output the average age of users whose occupation is the same as the input occupation. For example, the call `avgOccupationAge(usersArr,"student")` returns the average age of student users in `usersArr`.

4- `countMaleBelowAge:` a method that takes two input parameters where the first input is an array of user objects and the second input parameter is an integer that represents a person's age. The method returns as output the number of male users who are below the input age. For example, the call `countMaleBelowAge(usersArr, 50)` returns as output the number of users whose sex is 'M' and age <= 50.

Grading

Your grade in this assignment is based on the following:

- Your submission meets specifications as described above.
- **Add appropriate comments to your code.**
- Variable names should convey meaning.
- At the top of each Java file, include your name, a brief description of the program and what it does and the due date.
- All code blocks must be indented consistently and correctly. Blocks are delimited by opening and closing curly braces. Opening and closing curly braces must be aligned consistently.
- You must use the exact same name (including upper case and lower-case letter) for all methods as specified in the above description.
- The output of our program must be nicely formatted.
- You must follow the method requirements in terms of the number and data type of input parameters and the output data type.
- The program is robust with no runtime errors or problems.
- The programs should display your name.

Submission Instructions

- Follow the following steps to upload your code to D2L:

    - Create a java project and call it `<your-last-name>Assignment6` (e.g., mine using a

Student as thing will be called `DillonAssignment6`) o
Create two .java files as described above.

    - Archive your .java files into **one zip** file using Eclipse using the following steps:

- In Eclipse Project Explorer, right click on the src folder of the project and click on Export.
- Choose General->Archive File and click Next.
- Use the Browse key to choose a folder to store the archive file on your hard drive and give the file the same name as your project (e.g., DillonAssginment6.zip), then click Save, then click Finish.
- Upload the **.zip** file you created to the D2L folder called Assignment 6.
- It is important that you upload only **one** zip file. Your assignment will not be graded if you upload individual .java files to D2L.