Lab 02 Specification – Explore File Utility
Due (via your git repo) no later than 8 a.m., Wednesday, 13th Feb 2019.
50 points

Lab Goals

- Accessing external file input from a Java program.
- Implementing procedures to answer a series of questions on the data file using core programming constructs.

Suggestions for Success

- Take a look at the suggestions for successfully completing the lab assignment, which is available at: `https://www.cs.allegheny.edu/sites/amohan/resources/suggestions.pdf`

Learning Assignment

If you have not done yet, you should read the following to do well on this assignment. It is also highly important to review the code from class discussions on Plant dataset. The copy of the source code is also placed in the lab repository under the src directory.

- GT chapter 01

Assignment Details

Writing Java programs to interact with external file(s) adds a completely different dimension to the programming challenge in computer science. In real world scenario, often times the input(s) and output(s) in a program are an external body, such as a file or a database. If you recall from class discussions, a database is simply an abstraction over file system. In a database class, more details about database and its concepts will be discussed. In CS101, we are focused on files.

So why file input or output? Is it possible to write a program without the influence of an external body like files? Yes, it is possible to write a program without the use of files. But it is an inefficient method, as the programmer need to hard code the content of the input inside the code and restrict the output printed to purely console. Let us suppose if the input(s) to the program changes, then all lines in the code where the input is hard coded need to be changed and importantly recompiling the code is required every time. Assume the input to the program, changes during every 10 runs, and the code is run 1000 times a day. This is tedious and as a programmer it is important to propose a solution that is more efficient.

In Lab 01, we had designed and developed a Guess game. In order to implement this program, we had input(s) sourced from the console as user typed text and output(s) were displayed in the console. Let us suppose the game administrator would like to see the results from the game program execution for last 10 days. The current scope of Lab 01, after your program quits, all your execution results are lost. Saving it to different files, would provide an option to the game adminstrator to further analyze the results and keep a record of the results from all the game played instances.

Hence, it is extremely important to use an external body such as a file in your Java program to provide an efficient platform for the implementation. In this lab, you will write code to read the file content and store it into an array [memory based storage]. Later, you will write more code to process the array, and implement a series of questions to be asked on the data file. This process is often known as data analysis.

Also, to note that at this point after completing this lab you may be fully equipped to do any coding exercise. However, the solution produced may not be fully effective. Our rest of the semester is to focus on new techniques in programming that would allow you to provide efficient solutions. One such technique that we had recently started exploring is Object Oriented Programming. Later we will talk about Data Structures that would allow you to efficiently store and retrieve data from memory.

So how to compile the program

At this point we should all be familiar with the compilation of Java programs using **javac**. Since, there is a particular directory structure associated with the lab and to make the compilation process simple, there is a compile bash script shipped with the lab repository. In order to compile your programs, make sure to place all your Java files under the src directory, Then, open the terminal and navigate to the lab repository and type in:
"./compile.sh"
This should compile all your files and automatically place their class files in the classes sub directory located in the lab repository.

So how to execute the program

At this point we should all be familiar with executing Java programs using **java**. Since, there is a particular directory structure associated with the lab and to make the execution process simple, there is a run bash script shipped with the lab repository. In order to execute your programs, make sure to open the terminal and navigate to the lab repository and type in:
"./run.sh PlantData"
Here PlantData is the name of the class file that is to be executed. If you have a different class file that needs to be executed, then specify that class name while executing the run.sh bash file.

FileParser - Accessing File Content and storing into an Array (20 points)

Write a Java program that has a Class named "FileParser". The program should include a method called fileToArray that reads the file called foodmenu.txt and stores the content of the file into an array. The input file is provided in the lab repository under the data directory. The method signature is as provided below:

```
public static String[] fileToArray(String fileName){}
```

In fileToArray method, it is required to read the food menu details from the file input and store it into an array of type String. The textual input file contains different food menu items, with several attributes for each menu item. The attributes are as follows:

1. Name
2. Price
3. Description
4. Calories

In order to read the file input, it is highly recommended to use the FileUtility class. The FileUtility class contains a series of supporting methods that provides a platform to work with an external file.

The Java program called `FileUtility.java` that provides a platform to read, write, and replace text in a file is provided in the lab repository under the src directory. Below you can see a few examples for using the methods of `FileUtility` on a file named "dummy.txt".

```
//create a FileUtility that uses a file called "dummy.txt"
FileUtility dummy = new FileUtility("dummy.txt");

//read the first line from "dummy.txt" and store it
String first_line = dummy.read();

//read the second line from "dummy.txt" and store it
String second_line = dummy.read();

//append a line to the end of "dummy.txt"
dummy.write("Thus ends this file");

//update the first line to be "This is now the first line"
dummy.update(0, "This is now the first line");

//get how many lines there are in the file and store that
int lines = dummy.size();

//update the last line to be "This is now the last line"
dummy.update(lines-1, "This is now the last line");

//reset the reader to the beginning of the file
dummy.reset();

//read the first line again, and print it
String line = dummy.read();
System.out.println(line);
```

A more detailed example of reading textual contents and storing into an array can be accessed from the PlantData source code available in the src directory.

In your main method, call the **fileToArray** method and store the value that is returned into an array that is local to the main method.

Prompt the user through the message below:

"Do you want the data to be displayed in the console?"

If the user says "yes", then display the data in the console for the user to read. If the user says "no", then do not display the data in the console. Instead proceed to the next part.

FileParser - Setting up a series of procedures to ask questions on the data file. (30 points)

In the main method, prompt the user to provide an option for recommending food menu items.

"Hey, welcome to the restaurant! Do you want me to recommend you some food choices?"

If the user says "yes", then start the recommendation process. If the user says "no", then quit the program.

If the user says "yes" to the question above, then prompt the user to say what would they like the recommendation to be based on.

”How would you want me to recommend. I can recommend based on 1) your budget, 2) your calories if you are a calorie watcher, 3) your food item likeness. Specify a number, so I can get started!”

If the user specifies 1 as user input, then call the recommendFoodByPrice method. If the user specifies 2 as user input, then call the recommendFoodByCalorie method. If the user specifies 3 as user input, then call the recommendFoodByLikeness method.

So, let’s look at some details regarding the recommendFood methods.

1. In the “FileParser” class, create a method named recommedFoodByBudget that does the following:

   Prompt the user to say what is their budget for the breakfast?

   Based on the budget provided, list all the breakfast menu items that are lesser than equal to the budget provided by the user. For example, if the user provided a budget of 10$, then display all the menu items that are lesser than equal to 10$

2. In the “FileParser” class, create a method named recommedFoodByCalorie that does the following:

   Prompt the user to say what is their expected calories intake for the breakfast?

   Based on the expected calories intake provided, list all the breakfast menu items that are lesser than equal to the user provided calories. For example, if the user provided ”expected calories” intake of 500, then display all the menu items that are lesser than equal to 500 calories

3. In the “FileParser” class, create a method named recommedFoodByLikeness that does the following:

   Prompt the user to say what kind of breakfast, they have in mind to eat?

   Provide the following options for the user to choose from:
   <Waffles, Eggs, Bread, Hash Browns>

   Based on the option selected, list all the breakfast menu items that are connected to the option selected.

   How to find if a particular menu item is connected to the user selected option?
   In your program read the description of each menu item and check if the user selected option is part of the description or not. Here you are expcted to do String comparison with the use of Sub Strings and IndexOf in Java. If it is part of the description, then the menu item is connected to the user selected option. On the other side, if it is not part of the description, then the menu item is not connected to the user selected option.

Submission Details

For this assignment, please submit the following to your GiHub repository by using the link shared to you by the course instructor:

1. Commented source code from the “FileParser” program.
2. A readme file explaining how to compile and run your program. In addition, include a detailed self reflection of the lab exercise in the readme file. The self reflection may include answers to questions such as:
   - ”What challenges did you face in this lab?”
   - ”What did you like in this lab?”
   - ”What did you not like in this lab?”
3. It is highly important, for you to meet the honor code standards provided by the college. The honor code policy can be accessed through the course syllbus. Make sure to add the statement ”This work is mine, unless otherwise cited.” in all your deliverables such as source code and PDF files.

Grading Rubric

1. There will be full points awarded for the lab, if all the requirements in the lab specification are correctly implemented. Partial credits will be awarded if deemed appropriate.
2. Failure to upload the lab assignment code to your git repo, will lead you to receive ”0” points given for the lab submission. In this case, there is no solid base to grade the work.
3. There will be no partial credit awarded if your code doesn’t compile correctly. It is highly recommended to validate if the correct version of the code is being submitted before due date and make sure to follow the honor code policy described in the syllabus. If it is a late submission, then it is the student’s responsibility to let the professor know about it after the final submission in github. In this way, an updated version of student’s submission will be used for grading. If the student did not communicate about the late submission, then automatically, the most updated version before the submission deadline will be used for grading purposes. If the student had not submitted any code, then in this case, there is 0 points to the student automatically for the lab work.
4. If you needed any clarification on your lab grade, talk to the instructor. The lab grade may be changed if deemed appropriate.