CSCI 428 Assignment 04
File Input and Output, Exception Handling, and Classes

90 points +5 bonus points

Please read the instructions before working on the assignment:

– Assignments will be given and returned via the D2L as a convenience to the students and the instructor. Do not send assignments via email.
– It is the student's responsibility to have all assignments ready on time by the given due date. Late assignments may not be accepted.
– Discussion is allowed and encouraged, but no cheating. Assignments with copied answers or other cheating (all or in part) receive a grade of 0.
– Assignments or programs with extra features and fancy output, you may receive extra scores.
– The first lines of the source code of the program should include a comment with the following information and format:

```
/**
* A short description of the program.
*
* @author's last name, first Name
* @assignment CSCI 428 Assignment X -Qn M
* @date MM/DD/ YYYY
*/
```

– For all programming questions, copy the Java source code into the Word file named *"CSCI428-AssignX-XX.doc(x)"* and screenshots of the program's output. Please replace the first `*X*' with the Assignment number and "*XX*" with your first name and last name. For instance, the title for assignment 01 should be "*CSCI428- Assign01-JessicaWang.doc(x)*".
– List answers to all short-answer questions in the Word file.
– Submit the Word (.doc(x)) file and Java source files (.java) into the D2L.
– **NO PDF FILES.**

Questions:

Qn1. (30 points+5 bonus points, 6 points each and 5 points for the bonus question) **File Input and Output**

a. (Create a directory) Write a Java program that prompts the user to enter a directory name and creates a new directory using the File's mkdirs method under your draft project path by referring to the examples included in the "04-csci428_lecture_FileInput&Output.pdf" file and that at (https://www.tutorialspoint.com/java/io/java_io_file.htm). The program displays the message "Directory created successfully" if a directory is created or "Directory already exists" if the directory already exists. Please just use the relative path (not absolute)

b. (Measure the directory size) Define a method named *directorySize* to use the File's list or listFiles method to count the number of files under the directory you just created.

c. (Create a file to save a large dataset) Define a method named *generateDataset* that takes its parament – one string that represents the file name. This method needs to create a data file with 1,000 lines. Each line in the file consists of an employee's first name, last name, rank, and salary. The employee's first name and last name for the i-th line are FirstNamei and LastNamei. The rank is randomly generated as entry-level, middle-level, and senior software engineer. The salary is randomly generated as a number with two digits after the decimal point. The salary for an entry-level software engineer should be in the range of $40,000 to $60,000, for a middle-level software engineer from $40,000 to $150,000, and for a senior software engineer from $75,000 to $300,000.

   Here are some sample data:

```
FirstName1 LastName1 entry $49,835.95
FirstName2 LastName2 middle $85,679.35
. . .
FirstName1000 LastName1000 senior $234,567.65
```

   Save the data in "input.txt". For generating random numbers, one example is available at https://www.educative.io/answers/how-to-generate-random-numbers-in-java

d. (Data Processing) Define one or more method(s) to read the data from "input.txt", calculate and display the total salary for entry-level, middle-level, senior software engineers, and all employees, respectively, and display the average salary for each rank and all employees, respectively. Save the results in "*output.txt*"

e. (Bonus question (Optional)-5 points) Suppose you have the data files "input.txt" and "output.txt" under your current directory. Define a method named *copyfiles* to copy all the data files and create new data files. Name the new data files as "*newInput.txt*" and "*newOutput.txt*". Therefore, after the method call, there are four data files under the directory you just created.

f. (Debug and test) Test the program and all methods defined in this question. Show the results and submit .java file(s) and data files.

Qn2. (10 points, 5 points each) **Exception Handling**

Write a Java program by finishing the following tasks.

a. (*InputMismatchException*) Uses the *try/catch/finally* block and prompts the user to enter an integer that should be in the range of 50 and 1000. Your program should prompt the user to enter the number again if the input is incorrect and display the message "invalid input. Please enter again". Define and set an int variable *size* to the valid input.

b. (*ArrayIndexOutOfBoundsException*) Modifies the program to meet the following requirements:
   i. Creates an array with *size* randomly chosen integers. Please refer to Qn1.c for generating random integers.
   ii. Prompts the user to enter the index of the array, then displays the corresponding element value. If the specified index is out of bounds, display the message "Out of Bounds" by using *try/catch/finally* block instead of the *if* or *if-else* block.

c. (Debug and test) Test the program, show the results, and submit the .java file(s).

Qn3. (30 points, 4 points for each class and 5 points for the UML diagram and test program each) **Class and Class Inheritance** (The *Person*, *ParttimeEmployee*, *FulltimeEmployee*, *Junior*, and *Senior* classes)

a. Design a class named *Person*. A person has a name, address, phone number, and email address. Define constructors, necessary accessor, mutator methods and toString() method in the class. Please refer to the class named *Car* included in the "*06-csci428_lecture_Class.pdf*" file.

b. Define two subclasses named *parttimeEmployee* and *fulltimeEmployee* based on the class Person. A parttimeEmployee has a class status (regular, seasonal, on-call, or contractor) and payrate. Define the status as a constant. A fulltimeEmployee has an office, salary, and date hired.

c. Make *Junior* and *Senior* subclasses of *fulltimeEmployee*. A Junior member has a mentor and a rank. A Senior member has a team of junior members and company stock besides the salary and rank.

d. Define constructors, necessary accessor, and mutator methods in each class. And override the toString method in each class to display the class name and the person's name. One class, one java file.

e. Draw the UML diagram for the classes and implement them. One example can be found on P35 of the "*06-csci428_lecture_Class.pdf*" file. For the online tools of drawing UML diagrams, please refer to the final project file under "Content->Final Project".

f. Write a test program that creates instances of the classes defined above and invokes their toString() methods. Show the results and submit .java file(s).

Qn4. (20 points, 5 points each) **Hands-on Practice: Eclipse and IntelliJ with JavaFX**

This question is to introduce you to Eclipse for Java program development and IntelliJ with JavaFX for GUI application development and event-driven programming. It is the first move that every student must complete before working on any Java programs. The materials shared online in the D2L are your good references to complete this important move. Show the screenshots of the achievements of each step, except "Step 0". All sub-steps are negligible this time.

– Step 0: Go to the directory "D2L->CSCI 428->Contents->Final Project->Software installation”

– Step 1: Download JavaFX and IntelliJ. Show the downloaded executable files on your disk by referring to the document and videos under “Software installation/IntelliJ for JavaFX/”
– Step 2: Install the IntelliJ by referring to the video “How to install IntelliJ for JavaFX”(Recommended) or “CSCI428-JavaFX Environment”(Alternative).
– Step 3: Develop and run your first JavaFX project. The example output is given above.

Note: if possible, you can change the title of the window to "Hello Jessica! My First JavaFX" by replacing the statement.

```
stage.setTitle("Hello!");
```

with the following statement

```
stage.setTitle("Hello Jessica! My First JavaFX");
```

In your program, please replace “Jessica” with your firstname.

– Step 4: Done! We can start our journey now. Show your program and output window.