C212 Lab 9

Intro to Software Systems

Instructions:

- Review the requirements given below and complete your work. Please submit all files through Canvas (including all input and output files).
- Download the Lab9.zip file from Canvas.
- Submit a zip file that contains your finished code to Canvas by Wednesday, November 3.

Lab: Input / Output and Exception Handling

Part 1: I/O reader

Given a file of student records, completed the following tasks which include adding a student, deleting a student, computing the count of student from the student records. Run a while loop in the main method wherein the menu would consist of:

1. Add a student to the file students.txt
2. Delete a record
3. Count of seniors, juniors, sophomores, and freshmen
4. Total count of students
5. Exit

The students.txt file would look like this:

J Sebastian Rodriguez
S Tom Jordan
F Lauryn Poole
s Sarai David

Where **J – Junior, S – Senior, F – Freshmen, s – Sophomore** and are separated by whitespace

The skeleton code for the methods is given below:

```
1.  public static void addStudent(String name, String year) throws IOException {
    // Add a new student record to the file.
2.  }
3.  public static void deleteStudent(String name) throws IOException {
    // Delete an existing student record from the file
4.  }
5.  public static void countStudents() throws FileNotFoundException {
    // Print a count of seniors, juniors, sophomores, and freshman students in the file.
6.  }
```

Part 2: Exception Exercises

In the zip folder for this week's lab there is a class entitled *ExceptionExercises.java* that contains two static methods and one main method that simply calls both of the methods. Each of the methods will throw *at least* one exception and cause the program to end early. Your task is to add exception handling into these methods. **Do not edit the main() method, and do not simply fix the methods so that the exceptions are not thrown.** Your job is to handle the exceptions when they're thrown, not stop them from occurring.

Problem1()

This method iterates through an integer array and changes the values inside of the array. Your task is to handle any exceptions that are thrown such that:

1. The *for* loop does not end when an exception is thrown.
2. Any time an exception is thrown, your method will:
    a. Print the following statement: "Exception encountered: " + (the message contained in the exception).
    b. Set the value of intArray[i] to 100 (if *i* is within the bounds of the array).

    Note: You can get the message of an exception using the .getMessage() method on the exception.

3. The method returns the intArray variable no matter what. (You shouldn't add any return statements or change the current return statement).

Reminder: You should not be changing any of the code that is already in this method. Rather, you need to add code to handle any exceptions that are thrown.

Problem2()

This method simply returns an ArrayList of Integers. Your task is the handle exceptions in this method such that:

1. The for loop always runs to completion and does not exit prematurely.
2. In the case of a NullPointerException, the method will:
    a. Print the following statement:
        i. "Exception encountered: " + (the message contained in the exception).
    b. Do something to prevent this exception from occurring in future iterations of the for loop.
    c. Still add *i* into the ArrayList.
3. In the case of an ArithmeticException, the method will:
    a. Print the following statement:
        i. "Exception encountered: " + (the message contained in the exception) + " at index " + (the index that the exception occurred at).

Note: You can get the message of an exception using the .getMessage() method on the exception.

Reminder: You should not be changing any of the code that is already in this method. Rather, you need to add code to handle any exceptions that are thrown.

Keep in mind that this method will throw multiple exceptions and you will need to handle each type of exception differently. **If your code is correct, the return value of the method will be an ArrayList containing [0, 1, 2, 1, 4, 1, 6, 1, 8, 1]** (And the method will print some stuff as well).

Hints for Part 2:

1. Use try/catch statements to handle exceptions. We will link some information on try/catch statements below
    a. Exception Handling in Java | Java Exceptions - javatpoint
    b. Java try-catch - javatpoint

2. Remember that a single try statement may have multiple catch statements attached to it. This information just might come in handy on problem 2, but who's to say? (wink wink)

3. If you're having issues in which your for loops are ending whenever one exception is thrown, consider the placement of your try/catch statement and whether any other placements of the try/catch could continue the loop while still handling the thrown exceptions correctly.
    a. If you're still stuck, try googling "java exception handling inside loop" and see what the internet has to say.

Part 3: CSV Reader

Create a simple CSV reader wherein you have to read from a csv file provided in Lab9.zip. Write your code with appropriate exception classes and complete the tasks below:

Tasks:

1. Open and read the products.csv file from the zip folder
2. Compute the following:
    - Average price for each product category
    - Most expensive product amongst products listed
    - Count of products for each product category

Notes:

A Comma Separated Values (CSV) file is a plain text file that contains a list of data. These files are often used for exchanging data between different applications. CSV files have a fairly simple structure wherein data is separated by a delimiter (generally a comma)

E.g.: Name,Email,Phone Number,Address
Michael Myers,myers@example.com,123-456-7890,123 Elm Street

Hints:

- Read and review the data in the csv file. The first line of the CSV file is the header and the lines following the header is the actual data require for this task.

Useful Links:

- https://www.javatpoint.com/how-to-read-csv-file-in-java
- https://www.java67.com/2015/08/how-to-load-data-from-csv-file-in-java.html

Turn-ins:

- Develop a java program which reads a csv file and computes an average, count and the max value for the data in the csv file