CPSC 1060: Introduction to Programming in Java
School of Computing, Clemson University
Programming Project # 2
Due: Monday, April 19, 11:59pm

This project includes many of the concepts we have learned recently: object-oriented programming, inheritance, and using existing classes such as String and File. It furthermore covers basics such as selections, loops, methods, and arrays. The goal of this project is to bring all the knowledge you gained together.

As a reminder: **Programming projects are your own work and may not be copied in full or in part from any source**.

Tip: All of the concepts and classes needed in this project have example programs in the class book and in the slides that allow you to quickly find out how these classes can be used. Of course I also recommend looking at the Java documentation as we do in class.

TakeTheQuiz.java (45 points)

Write a program called **TakeTheQuiz.java** that allows you to answer quiz questions. You can add questions and answers to a database in a simple text file.

Part 1

The program asks the questions in the database in random order. The user types an answer and hits enter. If the answer is correct, the user is told so and the question counts as answered. If the answer is not correct, the user is told so and the correct answer is shown. Once all questions have been asked once, the incorrectly answered questions are repeated in random order. The process is repeated until each question has been answered correctly once. Then, the program displays how many questions have been asked and how many answers were correct.

The example database given to you called database.txt has 5 questions with their answers. Your program needs to work independently of the number of question/answer pairs in the database.

Follow these steps:

1. Create the QAndA class:
    - Create a class called "QAndA". Each question with its corresponding answer is represented as an object of type QAndA.
    - A QAndA object has a data field called q for the question, a data field called a for the answer, a constructor with two parameters of type String called q and a (for question and answer), and a method called toString that displays the question and the answer.

- You can create further data fields and methods as needed.
- Make sure to test your class.

2. Parse the text file and create QAndA objects:
   - The database where you save your questions and answers is the text file database.txt given to you. You can assume that it is formatted correctly with questions starting with *Q:* and answers starting with *A:* .
   - Read in your database.txt file Use the File and Scanner classes in a similar way than we did in lab 11 with a local pathname (independent of your machine).
   - Parse your database of questions and answers and create an array of QAndA objects using the ArrayList class (see Chapter 11 and documentation).
3. Main functionality:
   - Implement your program as described in the first paragraphs of this section.

Part 2

Wouldn't it be even better if your program could load multiple databases, say one for each chapter of the book, so you could revise each chapter on its own? Of course!! Before you start expanding the program you implemented so far, now is a great time for backing up your work.

Follow these steps:

- Create at least 3 txt files, each with at least 5 question/answer pairs in the same folder. They could be named chapter6.txt, chapter9.txt, and chapter11.txt.
- Create a subclass of QAndA, called Chapter. In addition to the data fields and methods of QAndA, it has a data field called ch to save the chapter.
- Implement your program to parse the appropriate txt-files and create a larger array of QAndA objects.
- The user can choose to either answer the questions of only a specific chapter or to answer the questions of all the chapters together where the questions are mixed in random order (not all questions of one chapter, then all questions of another chapter...).

Submission instructions

Submit your solution through canvas. It should include a single zip-file with:

- your Java program and your txt files. The programs have to compile correctly on the lab machines. Include a header as in the labs.

- **a readme file (readme.txt, 5 points)** in which you describe the functionality of your program. You might also describe problems you encountered and any known bugs or extra features of your program. At the top of your readme file, include your name, the class, the date, and a title.

The rubric for this project is

- read the files/folder: 5 points
- QAndA class: 10 points
- parse file, create QAndA array: 5 points
- part 1, main functionality: 10 points
- part 2, Chapter class and functionality: 10 points
- elegant and efficient implementation, reasonable comments, correct submission as a zip file: 5 points
- readme file: 5 points

Make sure to get them all and enjoy this project.