Given: Friday, September 18, 2015

Due: Friday, October 9, 2015 noon (just before the lecture starts)

Weight: 5%

Note: There will be an array related quiz to be held after the due date

The assignment is to be completed individually, in accordance with the policies and expectations in the course outline.

Objectives:

- Review of programming basics
- To gain experience developing a custom class with instance variables.
- To learn more about the very important concept of "encapsulation".
- To practice thinking about objects as having both *state* (via properties) and *behaviour* (via public operations).
- To gain experience working with multiple objects ("instances") of a class, each with its own distinct identity and state.
- Proper use of java arrays
- Working with arrays of objects
- Parsing strings
- Working with code that you have been provided
- Passing Command line arguments

Associated textbook sections: before starting, students should be familiar with most of **Chapter 8**

Note: You cannot use array lists at all for this assignment. You must use only arrays. There will be severe deductions if you use an ArrayList!

Short summary

You will be creating a program that implements a basic trivia game. The game is conducted like a multiple choice exam where the user is presented with a question and a list of possible choices. The user answers the questions and the final score and statistics are displayed once the user has either quit or finished the game. While playing the game the user should have the ability to quit at any time and to skip any question (which counts as an automatic fail on that specific question).

Your program will load the trivia questions from a file at the start of the program. The file name will be specified though command line arguments.

Note: You should be trying to implement a better object-oriented design for this assignment based upon the feedback you received on assignment 1.

Program Functionality

Below are the basic steps the program should go through

Note: For input you must use the provided User Interface class. If required you can added helper methods to this class as long as they are designed properly and make sense in relation to the purpose of the class. Also include comments about the methods you have included.

  a. Just copy and paste the ***UserInteraction*** file into your bluej project directory

1. Program starts up
2. The file name for the trivia question to use is retrieved through command line arguments
   a. The file name must be the first argument
   b. The file name must not include .txt **(.txt must be added to the name in your program)**
   c. If the file name has not been given on the command line an error should be printed out and the program should end gracefully.
3. All questions are loaded from the file into the program (Stored into proper objects that represent the questions). If any basic errors occur while loading the questions an error message should be printed to the console and properly shut down the program.
   a. If the file cannot be found or is empty, the program must flag the error and exit gracefully
   b. Hint: The questions should be stored into an array.
4. The user should then be allowed to take the trivia game
   a. Present one question at a time to the user (in the order they appeared in the file)
   b. Allow them to pick one of the choices (A,B,C, etc.), skip the question or quit the game
      i. If they pick one of the choices, report if they got the question correct or not. Update their score and pause (i.e. wait for them to hit enter) and then go to the next question
      ii. If they choose to skip a question award no points and go to the next question.
      iii. If they chose to quit the game, stop going through the questions and print the final score/ statistics. (If they still have more questions to answer consider these questions as being skipped and update the appropriate statistics).
   c. Once user answers the final question, the game automatically stops and the final score and statistics are printed to the console.
   d. Above each question display the following (make this easy to ready):
      i. The current question number
      ii. The total number of questions
      iii. The possible points for the question
      iv. The user's current score.
5. When the game is over (by finishing all the questions or the user chose to quit) print the following information before exiting the program
   a. The final score (The points)
   b. The total number of questions answered correctly
   c. The total number of questions skipped

Basic file format

The file has the following basic structure:

1. The first line will contain one number that indicates how many questions are in the file. All of the following lines will include information for each question in the order they are to be presented to the user.
2. Each stored question will have the following format:
    a. The first line will contain 1 number followed by 1 string followed by another number
        i. The first number is the point value of the question
        ii. The string indicates the correct answer to the question (this will be any uppercase character like: A,B,C, etc.)
        iii. The second number is the number of answers to present to the user (n)
    b. The next line will contain the full question (that will be presented to the user)
    c. The following n lines will contain the possible choices (the first line being choice A, next line choice B and so forth)

```
3
1 C 4
what is the capital of Australia?
Sydney
Melbourne
Canberra
Perth
2 A 3
Ionic, Doric and Corinthian are all orders of what important structure?
Columns
Arches
Domes
8 D 4
In what country was origami invented?
China
Canada
Russia
Japan
```

This example trivia game file contains three questions:

1. Points=1, answer = C and 4 choices

    what is the capital of Australia?

    A Sydney
    B Melbourne
    C Canberra
    D Perth

2. Points=2, answer = A and 3 choices

    Ionic, Doric and Corinthian are all orders of what important structure?

    A Columns

B Arches
C Domes

3. Points=8, answer = D and 4 choices

   In what country was origami invented?

   A China
   B Canada
   C Russia
   D Japan

Coding Style and Documentation Standards

- Try to use java doc for your headers on each file and on important methods
- Choose self-documenting identifiers (e.g. for variables, methods, etc.)
- explicit initialization of variables (where appropriate)
- consistent and correct use of white space, including:
  - proper indentation
  - other use of white space to highlight the logical structure of an algorithm
- marking all methods as either public or private, as appropriate
- marking all instance data as private
- making variables local, unless they are for genuine per-object properties
- avoiding overly long/complex methods by instead delegating key subtasks to other methods (this includes avoidance of code duplication)
- using inline comments, sparingly, to clarity especially tricky or less-obvious segments of code
- avoiding hard-coded magic literals in favour of named constants

Submission Instructions

Before attempting the steps below, ensure your solution compiles without errors or warnings, ensure all existing automated tests pass (if there are any), and ensure your program works as expected. Correct any problems before continuing. **Code that does not compile and can't be tested will be heavily penalized.**

Important reminder: this assignment is to be completed *individually*. Students are strictly cautioned against submitting work they didn't do themselves.

Please follow these submission instructions **exactly**.

1. Rename your BlueJ project folder (if you haven't already) as: <LastName>_<FirstName>_***Asg2***
2. Submit the entire BlueJ project folder (with all its contents) to the "submit" folder, which appears under the "I:" drive each time you log in to a university PC. If you are submitting from a non-university computer (e.g. a home PC), you can access the submit folder via secure.mtroyal.ca, into which you can upload a compressed (e.g. ".zip") version of your main folder.