Release Notes: Mascot-Project

1 Overview

This project will test your ability to use the following Java functionalities:

- Use Java’s Scanner object to get input from the user from the Console - specifically String input;
- Use if and if-else statements to determine the flow of control in a program.
- Use the equals and the equalsIgnoreCase methods for String objects; and,
- Use Java’s iteration statements to loop over a series of instructions.

1.1 Requirements

The program’s purpose is to test the user’s ability to correctly identify the mascots of four universities. Users will be awarded one point for each correct response, one point will be deducted for an incorrect response, and zero points will be deducted if the user responds “don’t know.” This means that your program will need to keep score as the user responds to the prompts from the quiz and print that score to the console at the end of the program’s execution.

You will complete the implementation of the main method on the MascotQuiz class. You must put all of your code between the two large comment blocks that indicate where the code should go. You can (and should) put comments in this code and elsewhere in the program.

The user will be asked for each question if he would like to be given a school and name the mascot, or if he would like to be given a mascot and name the school. At that same time, the user will be given an option to exit the program and take no further questions. The answer to this question will be an integer (1, 2, or 3). You can be certain that the user will enter a integer in response to this prompt, and you must read it as an integer. If the user selects to exit the program by entering a 3, the program must immediately terminate after printing the score.

Once the direction of that one quiz question is decided, the program will prompt the user with the name of a school or the name of a mascot followed by a question mark (as
appropriate based on the numeric choice indicated). The user will then need to type the correct school/mascot answer in the case and spelling that matches what is shown in the final variable given at the top of the program. Points for this question are added to the score based on the user’s response: 1 for a correct, −1 for an incorrect, or 0 for a “don’t know” response.

At the end of the four questions, the user will be asked if they want to play again. This must be answered with a “yes” or “no”. If the user answers “yes” using any combination of upper and lower case characters, the quiz will start again, resetting the score to 0. If the user selects to terminate at any time, the score should be printed before the program terminates.

2 Language Requirements

- You must use descriptive variable names.
- You must have comments necessary to help the reader of the source code. These comments should help the reader follow your reasoning and help the reader to find specific sections when needed.
- You must use fully blocked statements that follow the structure of the Java code convention for curly brace placement.
- You must have proper indenting that reflects the meaning and flow of the code.
- You must use final variables where appropriate.
- You may use only the following methods from the String class: equals, equalsIgnoreCase, and/or compareTo.
- Case sensitivity of input: When the user answers with a mascot or university name, the case must match exactly what is shown in the final variables to be accepted as correct, but when the user answers “don’t know” to a question, upper or lower case should be accepted as the same: “Don’t Know” or “dOn’T kNoW” etc.
- The user should be prompted for all input as shown in the sample run below.
- You must add a comment block at the beginning of your code (in the file MascotsQuiz.java) that clearly indicates
  - Your name,
  - Your section (number),
  - Your Teaching Assistant’s Name,
  - And the purpose of this program.
• The input from the user that consists of a 1, 2, or 3 for indicating what the user would like to do next needs to be read as an integer (not as a String).
• You may not use any array, ArrayList, or other collection to implement this project.

2.1 Suggestions
• Start early, and test constantly. Don’t assume that something works until you have tried it by executing the main.
• Note the spelling of the “don’t know” response. In order to pass the tests for this project, make sure that you use “don’t know” (without the double-quotes).
• You might find it easier to implement one question, complete with its options; test it, and then reproduce that general structure for each of remaining questions.

3 Sample output

Here’s a sample of your program in action. Any number that appears in this output on a line by itself was typed by the user during execution as an answer to the prompt given on the previous 3 lines. The words that appear after a question mark (on that same line) and the => symbol are also typed by the user during execution in answer to the prompt that ended with a question mark:

In this game, I ask you four questions about mascots for US collegiate sports teams. You get 1 point for each correct answer, 0 points if you type don’t know, and you lose a point for wrong answers.

Type 1 and I’ll give you the mascot and you give me the school.
Type 2 and I’ll give you the school and you give me the mascot.
Type 3 and I’ll quit.

1
Answer with one of: University of Oklahoma, University of Wisconsin, University of Michigan, University of Nebraska
Sooners ? => University of Oklahoma

Type 1 and I’ll give you the mascot and you give me the school.
Type 2 and I’ll give you the school and you give me the mascot.
Type 3 and I’ll quit.

2
Answer with one of: Sooners, Badgers, Wolverines, Cornhuskers
University of Wisconsin ? => Badgers

Type 1 and I’ll give you the mascot and you give the school.
Type 2 and I’ll give you the school and you give me the mascot.
Type 3 and I’ll quit.
2
Answer with one of : Sooners, Badgers, Wolverines, Cornhuskers

University of Michigan ? => Wolverines

Type 1 and I’ll give you the mascot and you give the school.
Type 2 and I’ll give you the school and you give me the mascot.
Type 3 and I’ll quit.
2
Answer with one of : Sooners, Badgers, Wolverines, Cornhuskers

University of Nebraska ? => Badgers
Want to play again? (type yes or no): no

Bye. Your score is 2

3.1 Overview of Public Tests

Without giving away the details, we can provide you with some summary description of the Public Tests:

Public Test 1 The player gets at least one wrong and the remaining correct.

Public Test 2 The player correctly responds to a few but doesn’t know about others.

Public Test 3 The player knows nothing, so they repeat the game, and then come up with a few correct responses. Note, in this test, the player responds with “don’t know” instead of entering wrong data.

Public Test 4 The player is clueless.

Although we cannot tell you much about Release Tests, these repeat similar question/answer scenarios, but repeatedly, where the user selects to play again.

3.2 Special considerations

You will need to pay special attention to how the Scanner object handles its input. In particular, your TAs reviewed the details of how to use next() and the nextLine() methods in Recitation. You will also need to be careful to use the correct Scanner method to retrieve the intended kind of input: nextInt() for integers, and next() or nextLine()
(depending) for Strings. When an integer is requested, you can be sure that the user will indeed type an integer.

Finally, success in this project will require that you construct nested if-else statements. You have seen many examples of these in class and in recitation. You will also be introduced to the Eclipse debugger, which you will find helpful as it will allow you to visually trace the execution of your code on a line-by-line basis.

Remember: the order of schools and their mascots if fixed. Make sure that your code presents schools/mascots in the order they appear in this document:

- University of Oklahoma, Sooners
- University of Wisconsin, Badgers
- University of Michigan, Wolverines,
- University of Nebraska, Cornhuskers.

4 Honor Code

This is a closed project, meaning that you are not to give or receive assistance from other students or from outside sources, including, but not limited to, the Internet. Please review the University policy or ask an Instructor if you have any questions or concerns about Academic Integrity.

5 Obtaining Source Files

You should follow the procedure outlined in class and reviewed in some detail in Recitation to “check out” the Project. Make sure that your CVS Repository is correctly set-up in Eclipse, and that you have downloaded and installed (if necessary) the University’s Submit Server Plug-In. You will know that you have if, after checking out your project and selecting the Java perspective, your clicking right on the Project offers a Submit Project option. If it does not, then either talk with an instructor or find the Eclipse Tutorial (available from our Elms page), read it, and make sure that you get this done before proceeding.

The actual source file(s) for this project consists of only one class MascotQuiz, plus a collection of “support files” that you should not open or examine.

6 Due date(s) & Points allocation

Due dates (and times) are as indicated on the Submit Server. The points are allocated as 15 for each Public test, 10 for each Release test, which makes \(4 \times 15 + 3 \times 10 = 90\), leaving 10 points which will be determined by examining your code for style and the use of requested operations.