**CS203 HW5 Wheel of Fortune (basic game)**

**Spring 2012**

**Due date:** Wednesday, Feb 29 at 11 AM

You will complete Java methods to create a text-version phrase guessing game similar to Wheel of Fortune. A phrase is hidden to the player, who guesses letters until the full phrase is revealed. This version is a simplified version of the TV game show “Wheel of Fortune”. However, this will be a single-player game and there is no spinning of the giant wheel for cash prizes. Instead, the player guesses letters (right or wrong) until the entire phrase is revealed.

**Specification:**

1. The hidden phrase is stored in an array of chars and declared as an instance variable called phrase. The instance variable is declared for you in the starter code. [**done for you**]

2. A boolean array show contains true for positions of the phrase are currently revealed to the player and false otherwise. This is defined as an instance variable for you in the starter code. [**done for you**]

3. The play method shows the user the hidden puzzle. Until the user has revealed the entire phrase, it prompts the user to enter a letter, check to see that it is valid, and reveals the letter if it is contained in the puzzle. Also, keeps track of the number of correct and incorrect guesses. [**done for you**]

**You need to complete the following methods in the game**:

(1 pt) 4. initPhrase – should create an array of several phrase puzzles and randomly chooses a phrase to use for the game. Store the chosen phrase into phrase. (See starter code for how to create a random number.)

(1 pt) 5. initShow – should set all locations in the show array to false, except for the corresponding positions in the phrase array that contain the character ‘ ‘ (space character).

(2 pts) 6. printPhrase – should print the current state of the phrase. If the position in show is true, then print the corresponding character in phrase. Otherwise, print a ‘\*’ for that position. Stars will serve as those letters that have not yet been revealed.

(2 pts) 7. revealLetter – takes a character as a parameter and searches for that character in phrase. If the character is found at position and the show array at position is false, then set the show array at that position to true. Do this for all positions in the phrase array. The method should return true if the letter is found somewhere in phrase (and has not previously been revealed) and false, otherwise. [Hint: when comparing characters in Java, use ==, since a character is a primitive type.]

(1 pt) 8. isWinner – if all the positions of show are true, then the method should return true since the phrase is completely revealed. If at least one position in the show is false, then return false.

**Additional Enrichment:**

If you have time, you may add extra features to your game. Be sure to document these additional features in your written summary.

1. Allow more than one player to play at once.

2. Virtually spin the giant wheel and keep track of money earned. Note that in the TV version of “Wheel of Fortune”, the players can guess consonants but must purchase vowels. Keep track of the player’s money.

3. Let the user guess the full phrase prior to revealing the entire phrase, one letter at a time.

4. Have categories for your phrases, like they do on the TV game show. For example, categories include “famous name”, “before and after”, “thing”, and “place”.

5. Keep track of the guessed letters and show the user which letters have already been guessed at each turn.

**Logistics:**

1. Download the starter code from Moodle. Remember to use good style and quality with your code. Some of the code is already finished for you.

2. Look at the TicTacToe game in the course pack for reference, if you need help with the syntax of methods and arrays.

3. When you want to submit your file, create a folder called username\_HW5. Include in the folder your PhraseGuesser.java file and your summary report file. Zip the folder (creating a file with the extension .zip) and submit it to Moodle.

**Grading Guidelines:**

Your program will be graded on a scale of 0 – 7 in two categories:

 Code Quality – design, comments, whitespace, clarity, good names, organization, indentation

 Code Operation – functionality of code and adhering to specification

Your summary report will be graded on a scale of 0 – 6 base on:

 Correct use of technical vocabulary, clarity, organization, use of headings, evidence of testing, and completeness

**Report Guidelines and Format (use the HW1 file for a template, but answer the questions below):**

**1. Introduction:** Describe the basic idea of the game for the user. Include the screen printout at the end of playing the game (include at least 3 letter guesses and the text printed prior to the program finishing).

**2. System Description:**

**2.1 Specification:** If your code adheres to the specification, write “Program adheres to specification”. If something does not work, note it here. If you added extra features, describe them here.

**2.2 Method Descriptions and Control Flow:** Paste the code for the **printPhrase** and **revealPhrase** methods here. Draw the control flow diagrams for the **printPhrase** and **revealPhrase** methods (you can label code with line numbers to create labels for the boxes and diamonds in the control flow diagrams). You may draw these by hand and scan them to include them as pictures or you may draw them using the Word drawing tool.

**2.3 Play Control Flow:** Finish the diagram showing the sequence of the method calls in the game. For example, main creates a PhraseGuesser game object upon which the play method is called. Then, play calls initPhrase. After initPhrase is called, play calls initShow. You complete the rest of the method call sequence chart. Just include the method calls for methods defined in PhraseGuesser.java.

main(String[] args)

game: PhraseGuesser

play()

 initPhrase()

 initShow()

**3. Testing and Evaluation:**

**3.1 Testing** Describe how you tested your program to ensure that it works. How many games did you play? Inputs you tried (letters, numbers, etc)? Did the program terminate when the phrase was revealed?

**3.2 User Test** Have someone who is not taking CS203 play your game. What did he/she like? What did he/she think could be improved?

**4. Conclusion:** What did you learn by completing this assignment?

1. How long did you spend on this assignment?

2. By typing your name here, you are acknowledging that the code and report you are submitting are your own.

**Appendix:** Copy and paste your code here (use **Courier** font so the characters line up correctly).

Please keep your reports concise. A high quality computer scientist can explain their code, their process, and their testing to their colleagues and bosses. When writing the report, your audience is someone else taking CS 203. I ask you to do these reports because you will be expected to explain your ideas, whether or not you become a computer scientist.