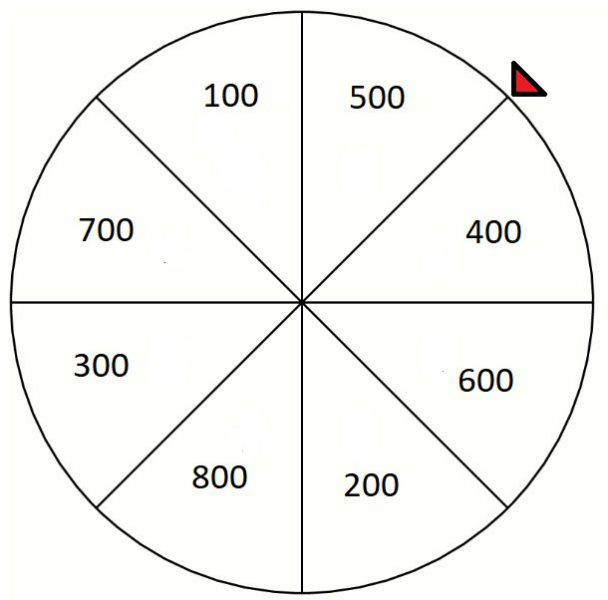
Homework 6
75 points
Due Saturday April 6th, 11:59pm

Wheel of Fortune



This programming assignment is a continuation of Homework 5, Hangman. You are welcome to reuse your code from hangman, or to rewrite it, or start over if you like.

This assignment will include the following requirements:

1. Similar to Hangman, your JavaFX application must include “Board” with a guest phrase, loaded from a file phrases.txt, provided in Homework 5. Phrases.txt includes multiple phrases, one per line. Select a random phrase, to create a board. While you are testing, you may want to use the same phrase each time, but you should implement the random phrase logic before you submit your final version.
2. Include a Wheel in your application, like the wheel above, that spins at the beginning of each turn.
3. Add a Marker that specifies to value of the wheel, when it stops. Add a button that says “Spin”, which will animate the wheel so it appears to spin. Add a Text Box (or label) that

shows the user's current score, which grows by the value of the wheel, after each correct guess.

4. Each turn will begin by the user clicking on the "Spin" button, which spins the wheel at least one full rotation, but less than two full rotations.
5. After the wheels spins to a value, the user guesses a letter. If the letter exists in the board, the guess is correct. Display a friendly "Correct" message and increase the user's Current Score by the value on the wheel. If the guess is incorrect, display a message and set the user's score back to 0.
6. Similar to Hangman, you must have a Text Box for the user to enter a letter, to guess. Also include a Label next to the Text Box, so it is clear to the user.
7. Similar to Hangman, the board should be blank to start. After each correct guess, update the board to show the letters that match the user's guess letter.
8. After the user guesses all letters on the board correctly, display a friendly winning message. Add a combobox to the GUI, with one item in the drop-down menu "Reset Board" which starts a new game. It must be a combobox, not a button or a popup window.
9. If the user selects the "reset board" option in the combobox, reset the board with a new random phrase from phrases.txt, and start the game again.

This assignment will NOT include the following requirements:

-You do NOT need to make the game for multiple players. One user playing the game by themself, is fine. Though you are welcome to add the functionality for more players, or functionality to play against a computer.
-You do NOT need to have the score labels on the wheel, (100, 200, ... , 800), rotate with the wheel. The labels can be in a fix position, or moving, either is fine.
-You do NOT need to keep any information from the previous game, but you are welcome to do so. For example you might add the user's high score somewhere on the board.

Submit to Canvas:

Submit 1 zip file containing all the files in your src folder, and your executable file. Name all your files clearly, so the grader can easily see, and run your program.

Scoring rubric:

The following rubric will be used:

Criteria	% Points
Program(s) fulfill all the requirements. All .java files are included and declare the necessary classes. Code is well organized, and easy to follow (especially for the grader). Coding stye is well utilized, including well named variables and methods. Comments are included, well written and descriptive.	100%
Program(s) fulfill almost all of the requirements. All .java files are included and declare the necessary classes. Code is fairly well organized, and somewhat easy to follow (especially for the grader). Some comments are included.	80%
Program(s) fulfill most of the requirements. All .java files are included and declare the necessary classes. Code is fairly well organized, and somewhat easy to follow (especially for the grader). Some or no comments are included.	60%
Program(s) fulfills some of the requirements, or does not run at all. Some .java files are included and declare some of the necessary classes. Some or no comments are included.	40%
Program(s) does not run at all. Some .java files are included and declare some of the necessary classes. Some or no comments are included.	20%
Either no attempt was made, or the attempt made shows no progress toward solving the problem.	0%