

# Lab 8: Wheel of Fortune

[40 pts]

This is a pair project: you may choose to work with exactly ONE partner!

**Objective:** To recreate Wheel of Fortune (simplified) using separate classes, static and non-static variables as well as methods, and algorithmic thinking to break the game up into parts.

## Understanding the Game

In the event you don't know how Wheel of Fortune works (it's okay, I didn't know for a long time), play a knock-off example here called [Wheel of Rewards](#). If this does not work, you can try downloading the app from the Apple App Store or the Google Play Store. Play for a bit and get the gist of the game.



If you really think about, Wheel of Fortune a more complex version of hangman – and we already did Olaf the Melting Snowman for Lab 6, which was our rendition of hangman. You are MORE than welcome to revisit your Lab 6 source code to assist you here in Lab 8.

## Set-Up

Due to the amount of flexibility creating multiple classes provides, there is not just one way to code our rendition of Wheel of Fortune. While I will provide a framework for you to work in should you want to try coding this my way, you are more than welcome to set up the logic in your own way.

I would suggest you have three classes in total:

- The **Player** class. This class will explain what a player is (i.e., a name and a bank account to store money) and what it can do (e.g., deposit money to or withdraw money from the bank account, tell you how much money he/she currently has).
  - This will be similar to the *Scanner* class where we have to instantiate a Scanner object to utilize its methods
- The **Game** class. This class will explain what constitutes a turn, dealing with the logic for spinning the wheel, buying a vowel, and solving the puzzle. It will check It should also explain the "Game over" logic.
  - This will be similar to the *Math* class where we don't need to instantiate a Math object to utilize its methods. Think about what this means about the types of variables and methods you'll create in the Game class...
- The **Main** class. This class will instantiate the players and create the logic that keeps the game going. This is where the main method should be so the code runs when you press "Run".

## Gameplay

You and a friend begin a game of Wheel of Fortune. When the program starts up, it should print the blank string, showing the current state of the puzzle. Both you and your friend have \$0 in the bank. It should then state the category, which player is taking their turn, and the current balance of that player's bank. After that, ask the user if they want to spin, solve, or buy a vowel. Here's what an example might look like:

```
-- _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
Category: Courses
Player: Mr. Park
You currently have: $0
Do you want to spin ('spin'), buy a vowel ('vowel'), or solve ('solve')?
```

Repeat the question if the user types in an incorrect value.

### Action 1: Spin the Wheel

If the player wants to spin, generate a random multiple of 100 from 0 to 1000, both inclusive. That will be the amount of money the player gets for each correctly guessed letter. If the randomized value is 0, the player loses a turn.

```
// Previous part of code not shown
Do you want to spin ('spin'), buy a vowel ('vowel'), or solve ('solve')? spin
The wheel shows 'Lose a Turn' :-(
```

Otherwise, the player should guess a consonant. If a vowel or a word of two letters or more are entered, ask the question again.

```
// Previous part of code not shown
Do you want to spin ('spin'), buy a vowel ('vowel'), or solve ('solve')? spin
The wheel shows $200. What consonant do you want to guess? a
"A" is not a consonant; try again: ab
"AB" is not a consonant; try again: b
```

If the consonant guessed is in the puzzle, update the puzzle and add the dollar amount shown on the wheel for each consonant that shows up in puzzle. For example, if I were to guess the letter C, I would get the random number (i.e., \$200 in the example below) times three added to my bank. You may assume that only single letters will be entered.

```
// Previous part of code not shown
The wheel shows $200. What consonant do you want to guess? c
There are 3 C's!

_ _ C _ _ _ _ _ _ _ C _ _ _ C _
Category: Courses
Player: Mr. Park
You currently have: $600
Do you want to spin ('spin'), buy a vowel ('vowel'), or solve ('solve')?
```

If the consonant does not show up in the puzzle or the consonant already appears in the puzzle, end the player's turn with the appropriate text to indicate what happened. Below is an example of the consonant already being guessed.

```
// Assume the puzzle looks like the following
_ _ C _ _ _ _ _ _ _ C _ _ _ C _
Category: Courses
Player: Mr. Park
You currently have: $600
Do you want to spin ('spin'), buy a vowel ('vowel'), or solve ('solve')? spin
The wheel shows $200. What consonant do you want to guess? c
That letter has already been guessed!
```

### Action 2: Buy a Vowel

If the player wants to purchase a vowel, vowels cost a flat \$250 fee, regardless of the number of letters this action reveals. Before a user is allowed to buy a vowel, the program should check that they have enough to do so.

```
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
Category: Courses
Player: Mr. Park
You currently have: $0
Do you want to spin ('spin'), buy a vowel ('vowel'), or solve ('solve')? vowel
You don't have enough money to do that!
```

If the player has the money to purchase the vowel, ask the player for a vowel (and repeat the question for incorrect answers). Then, update the puzzle and continue the player's turn. Note the current player's turn should continue regardless of whether or not the phrase contained the vowel they purchased!

```
// Assume the puzzle looks like the following
_ _ C _ _ _ _ _ _ _ C _ _ _ C _
Category: Courses
Player: Mr. Park
You currently have: $600
Do you want to spin ('spin'), buy a vowel ('vowel'), or solve ('solve')? vowel
What vowel would you like to purchase? v
"v" is not a vowel; try again: e
There are 3 E's!

_ P C _ _ P _ _ E _ _ C _ E _ C E
Category: Courses
Contestant: Mr. Park
You currently have: $350
// etc.
```

### Action 3: Solve the Puzzle

After a few more rounds of spinning and buying vowels, one of the players will probably be ready to guess the puzzle. If the player guesses the puzzle incorrectly, then end the player's turn.

```
_ P _ C _ _ P _ _ E _ _ _ C _ E _ C E
Category: Courses
Contestant: Mr. Park
You currently have: $350
Do you want to spin ('spin'), buy a vowel ('vowel'), or solve ('solve')?
solve
All right, go ahead: op computer science
That's not it!
```

```
_ P _ C _ _ P _ _ E _ _ _ C _ E _ C E
Category: Class
Contestant: Mr. Alcorn
// etc.
```

Otherwise, end the program.

```
// Previous part of code not shown
All right, go ahead: ap computer science
That's right, Mr. Alcorn! You won $10000!
```

### Submission

When your source code is completed, make sure to **submit** the lab on Replit; no need to worry about Schoology!

### Notes

- It may help to code this lab all in the main method first. Then, create other classes and methods that help make your code more efficient
- Be careful how you choose to pass information and data from one class to another. This will likely be the second-most confusing part (the previous bullet point hints at the most confusing part)
- While there are no challenges specifically stated for this lab, you are more than welcome to add your own flair and even additional components. If you want an algorithm that randomly picks the category and puzzle, go for it. Want to implement the "Bankrupt" option where the player loses all earnings from that round? Be my guest.

**Grading Rubric**

	Points
The game works as intended	30
At least two classes are created	4
Static variables and methods as well as non-static variables and methods are defined and utilized	4
Output is formatted neatly and in an organized fashion (it does NOT have to be like the expected output verbatim) Code is commented appropriately Variables are organized accordingly and named meaningfully	2
<b>TOTAL</b>	<b>40</b>