

Yahtzee

CS 221 - Fall 2021

Due: December 15 at 6 PM

In this project you will be creating your own Yahtzee class to model a modified Yahtzee game. The project loosely goes like this:

1. You will implement a Yahtzee class so that my YahtzeeTester class works. (Download the YahtzeeTester class from Canvas and drop it into your BlueJ project for project 4.) **You must not change any of the code in the YahtzeeTester class.**
2. The game roughly goes like this:
 - a. The game consists of only 3 dice. A die is 6-sided, between 1 and 6 inclusive.
 - b. The player starts a turn with an initial roll, and then has 3 rounds per turn (so three total individual die rolls).
 - c. The player may only roll 1 die at a time (1 die per round).
 - d. After the 3rd round of rolling, the player must choose a score to keep. The player may decide to choose a score *before* rolling 3 times. For instance, in the first turn in the example below, the player only rolled twice and ended up with 5 6 4, which is a straight. The player decided to take a straight as the score. On the second turn, the player rolled 3 times and could only choose yahtzee or chance (because the player already scored straight, and couldn't roll anymore).

The three possible scores to choose between are Yahtzee, Straight and Chance:

- i. Yahtzee: If the three dice have the same value.
 1. E.g., Dice are: 1 1 1 // **would** get Yahtzee points (50)
 2. E.g., Dice are: 5 5 6 // would **not** get Yahtzee points (0)
- ii. Straight: If the set of three dice form a consecutive run of values.
 1. E.g., Dice are: 2 3 4 // **would** get Straight points (25)
 2. E.g., Dice are: 5 3 4 // **would** get Straight points (25)
 3. E.g., Dice are: 3 4 6 // would **not** get Straight points (0)
- iii. Chance: Player scores the sum of the dice.
 1. E.g., Dice are: 3 4 1 // would score 8 for Chance
 2. E.g., Dice are: 5 5 6 // would score 16 for Chance

Once a score is chosen/used, it cannot be chosen/used again. The game is over after each score has been used.

A sample run of the program is shown below.

Dice are: 1 6 4

1 to roll a single dice.
2 for yahtzee score.
3 for straight score.
4 for chance score.
Enter a choice: 1
What die would you like to roll: 1

Dice are: 3 6 4
1 to roll a single dice.
2 for yahtzee score.
3 for straight score.
4 for chance score.
Enter a choice: 1
What die would you like to roll: 1

Dice are: 5 6 4
1 to roll a single dice.
2 for yahtzee score.
3 for straight score.
4 for chance score.
Enter a choice: 3

Your current total is: 25

Dice are: 5 5 3
1 to roll a single dice.
2 for yahtzee score.
4 for chance score.
Enter a choice: 1
What die would you like to roll: 4
Not a valid die.
What die would you like to roll: 3

Dice are: 5 5 6
1 to roll a single dice.
2 for yahtzee score.
4 for chance score.
Enter a choice: 1
What die would you like to roll: 3

Dice are: 5 5 4
1 to roll a single dice.
2 for yahtzee score.
4 for chance score.
Enter a choice: 1
What die would you like to roll: 3

Dice are: 5 5 2
2 for yahtzee score.
4 for chance score.
Enter a choice: 4

Your current total is: 37

Dice are: 1 4 4
1 to roll a single dice.
2 for yahtzee score.
Enter a choice: 1
What die would you like to roll: 1

Dice are: 5 4 4
1 to roll a single dice.
2 for yahtzee score.
Enter a choice: 1
What die would you like to roll: 1

Dice are: 6 4 4
1 to roll a single dice.
2 for yahtzee score.
Enter a choice: 1
What die would you like to roll: 1

Dice are: 1 4 4
2 for yahtzee score.
Enter a choice: 1
ERROR, try again.
2 for yahtzee score.
Enter a choice: 2

Your current total is: 37

Your final total is: 37

The YahtzeeTester implements the game play. Your job is to implement a Yahtzee class to model the Yahtzee game and make the YahtzeeTester work as it should (without changing that class **at all**). This means you have to figure out how to track the *state* of a game, and model the behavior/operations you can do in a game. Much of the behavior/operations

needed are indicated by the `YahtzeeTester` class. Pay close attention to whether the use of an operation (i.e., method call) involves passing an argument and/or receiving a return value. The `YahtzeeTester` provides many clues for how you should implement your `Yahtzee` class.

Starting Hint: Start by carefully parsing through the sample run and matching it up to the `YahtzeeTester` code. Which piece (line(s)) of code produces which output in the sample run? This will also tell you how you should implement your `Yahtzee` class.

Along with getting the `YahtzeeTester` program to work, you must meet the following requirements:

1. You must not modify my `YahtzeeTester` class ***at all***.
2. You must use the keyword `this` at least once in your program.
3. You must use composition in your `Yahtzee` class with a secondary class you create. A simple example of this might be to create a `Die` class and have the `Yahtzee` class contain 3 `Die` objects.
4. When displaying options for the user to choose, only display relevant information. For example, once a score is used, it is no longer shown as an option. After the third roll of a turn, the option to roll a die is no longer shown.
5. If you are prompting the user for values, you must ensure the user enters a correct value. In the example on the previous page, the user tried to enter a 1 when the only valid option was a 2. An appropriate error message must be displayed and the user must be prompted again. (As another example, find in the sample run where the user entered a 4 for the die choice.) You can assume the user will only type in integers.
6. You should use comments often in your code so I know what you are doing. As said before, you need to find a good medium between only using 1 comment and commenting every line of code.
7. You must document all of your classes and methods using proper javadoc comments.
8. Fill out the `README.TXT` file (except you may delete/skip `HOW TO START THIS PROJECT` and `USER INSTRUCTIONS`). Add and fill in:
 - a. `STATE OF THIS PROJECT`:
 - i. Describe the state you believe your project is in. (E.g., Is it completely finished and fully functioning and well tested? If not, what state is it in?)
 - b. `PROCESS DESCRIPTION`:
 - i. Describe how the project went for you. (E.g., Did it go smoothly? What obstacles did you face? Which parts did you find easiest/most challenging?)

Upload your zipped file to the Project 4 Canvas assignment. After you have uploaded the file, double-check to ensure your file was uploaded correctly. It is your responsibility to ensure your submission was done correctly. An improperly submitted project will cost you at least 2 points.