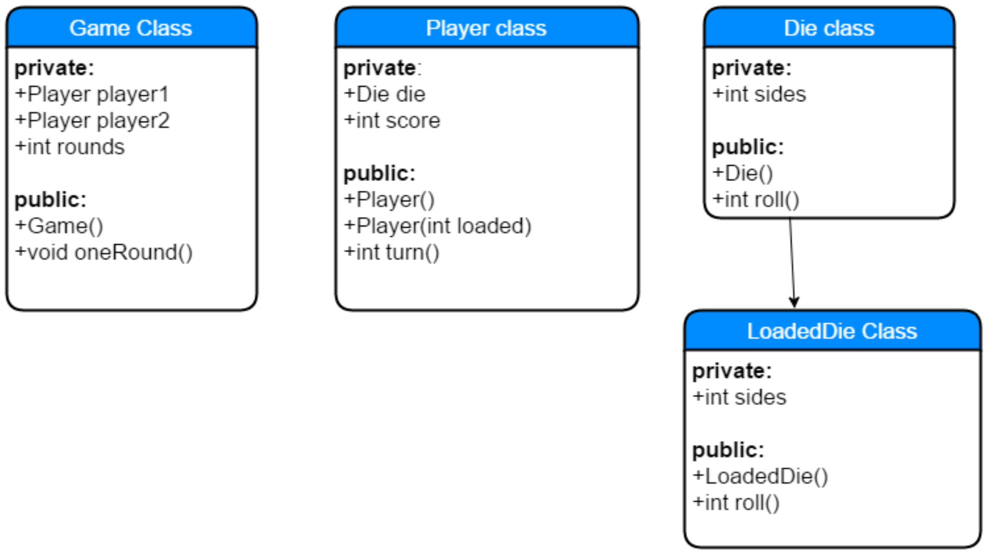
David Mednikov

CS 162 W17

onid: mednikod

Lab 2

Classes



Program Requirements

My plan for this program utilizes 4 total classes – a Game class, a Player class, a Die class, and a LoadedDie class, which is derived from the Die class. The Game glass contains 3 member variables: 2 objects of the Player class, and an integer containing the number of rounds. The Game class has two methods: the constructor and the oneRound() method, which simulates one round of the game and updates the players' scores.

The Player class contains two member variables: an object of the Die class and an integer containing that player's score. The Player class has three methods, the default constructor, an overloaded constructor (which creates a Player with a LoadedDie object), and turn(), which simulates each player's turn, returning an integer that represents the number they rolled.

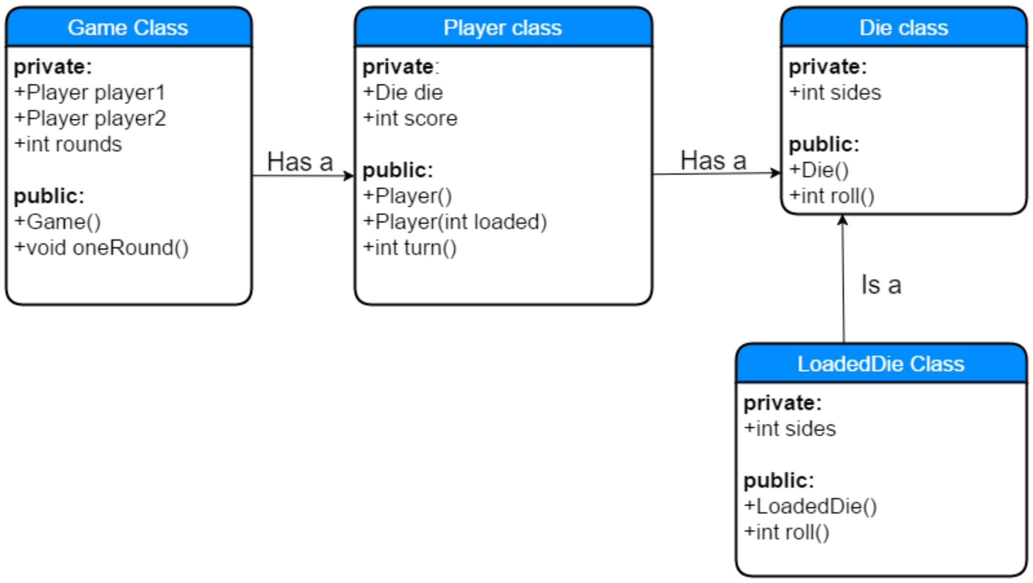
The Die class has one member variable, an integer containing the number of sides. It also contains two methods, the constructor and a method called roll(), which returns the number that the dice rolled. The LoadedDie class is derived from the Die class and has the same members. The roll() method in these two classes will be different, however.

While the Player class is not necessary for this program, I felt like it was useful. If we are truly supposed to break a program down into small parts, then creating Player classes makes sense. I could keep track of each player's score and die in the Game class, but I would rather include a Player class. Single responsibility, as they say.

Main Program Outline

1. Create object of Game class
    a. Create Game class object
    b. Use getInt() utility function to ask user for number of rounds, set rounds integer
    c. Use getInt() to ask user if dice are loaded: 1 for not loaded, 2 for loaded
        i. use overloaded Player(loaded) constructor if dice is loaded
        ii. create both Player objects with normal Die objects if dice are not loaded
    d. Use getInt() utility function to ask how many sides on each dice, set sides int variables
2. Using for loop, run oneRound() for the number of rounds specified by the user
    a. oneRound() calls turn() class, which calls roll() class
    b. oneRound() compares the integers returned by the turn() class, increment's the winner's score by 1
3. After playing every round, compare the two players' scores. Program tells user which player had the higher score.

Hierarchy Flowchart



Test Plan

Test Case	Input Values	Driver Functions	Expected Outcomes	Observed Outcomes
Negative input for # of rounds	Input < 0	main() calls recursive function to call getInt(0)	Loop back and prompt user for positive input	Loop back and prompt user for positive input
Input = 0 for # of sides	Input = 0	main() calls recursive function to call getInt(0)	Loop back and prompt user for positive input	Loop back and prompt user for positive input
Input is not 1 or 2 for dice being loaded	Input = 0 or 3	main() calls recursive function to call getInt(1,2)	Loop back and prompt user for input in range	Loop back and prompt user for input in range
Input contains letter	25a	main() calls getInt(), and cin.gcount() > 1	Loop back, prompt user to enter integer	Loop back, prompt user to enter integer
Input is a decimal	50.5	main() calls getInt(), and cin.gcount() > 1	Loop back, prompt user to enter integer	Loop back, prompt user to enter integer
Number of rounds is positive	Input > 0	main() calls recursive function to call getInt(0)	Accept user's input	Accept user's input
Number of sides is positive	Input > 0	main() calls recursive function to call getInt(0)	Accept user's input	Accept user's input