Robert Navarro
CS162

LAB 2

Design:

This program will use a base Die class for the Die class and the LoadedDie class. The Die class will include one private variable sides. The Die class will also include getter/setter methods for the number of sides, as well as a function that generates a number when the die is "rolled". A random number generator will be used to return a number that is less than or equal to the number of side on the die, but will not include 0. LoadedDie will only have one additional function, loadedRoll(), which will work similar to roll but will add to the random number if certain parameters are met.

The Game class will implement a simplified game of war using both of the die classes. In order to implement the Game class we will need several parameters. There will be a variable that stores player1 and player2's dice type, value rolled, and total wins. We will also have a variable to store the number of rounds to be played, the current round that is being played, and the number of sides that the dice will have.

With these parameters initialized we can then begin to design the program. We will first have a function chooseDie, which will prompt the user to set the number of sides on both of the die and if the die will be loaded for player one or player two. Once this the user has inputted this information a die will be generated for player 1 and player 2.

With the die parameters for each player now established we can implement the game of war. My thinking is that a simple for loop can be used to loop through the number of rounds that the user wants to play. Inside of this for loop we will have some if statements based on the type of die being used. If it is just a normal die the die function will be called, but if the die is loaded we will call the loaded roll function if the roll is even (2, 4, 6, etc..). The roll that is the game is currently on will be tracked by the curRound parameter.

We will use the p1Roll and p2Roll variables to store the value that the roll function returns. We can then compare the two values and to determine if player 1 or player 2 won the round. The wins variable will be updated accordingly depending on who won the round. However, if the values are equal the wins variable will not be updated and the game will continue on to the next round.

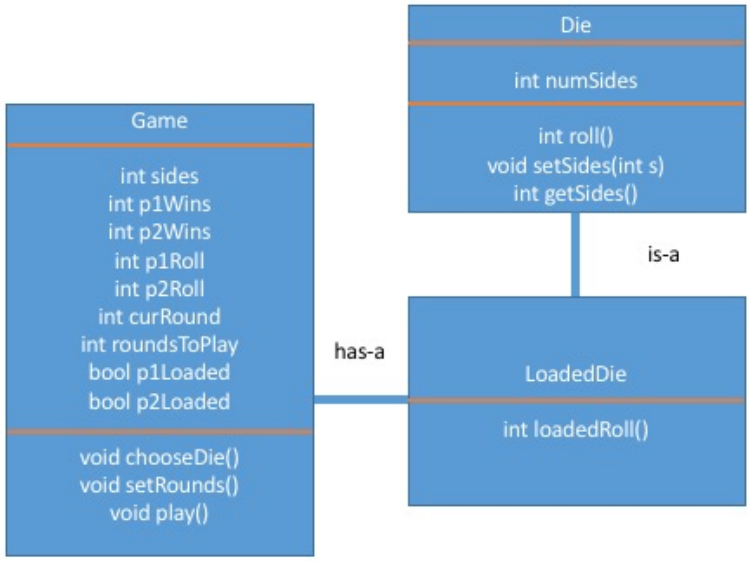
Once the for loop has processed through the number of rounds that the user specified the winner of the game will be displayed to the console.

Implementing from the Main Program

From the main program will first declare a new game object. Once we have a game object we can then call the chooseDie function, which will set the dies for both players. We can then call the setRounds function, which will set the number of rounds that we will be playing. With these

functions called we can the call the play function, which will start the game of war. The game will play through until the number of rounds the user specified have been played, and then the game will end.

Class Hierarchy



Testing

Testing for this program will ideally be down from the ground up. Initially we should test both die classes to make sure that they are generating values that we are expecting. Setting several different side values and then roll both the die and loaded die should help spot on trouble with die classes from the get go.

Once the die classes are good to go testing setting the variables inside of game should be the next objective. The best way to approach this would be to create the die for both players and then print the values that have been set for both players. This will help ensure that we are storing the correct values. Setting the number of rounds to play can also be included in this step but it is pretty straight forward and shouldn't take much testing.

At this point we can go ahead and test the actual play function. Since we have already tested the roll functions above we don't need to test the actual rolls here. What we do need to check is that the proper function is being called if the die is loaded. If you recall from above, it was decided that a loadedRoll will occur if the current roll is even. I think that using the debugger

here and following going through each step of the for loop is the best idea. This will ensure that the proper function is being called for each even roll.

The final check will be to make sure that the wins are going to the correct player. Similar to testing the loadedRoll, the best approached will be to use the debugger and make sure that the players win total is being updated accordingly.

With these checks above done we should have caught any errors in our program. However, just to be on the safe side I would run through the program several more times using different sizes for the die and playing a different number of rounds to ensure that the game is functioning properly. If all of the checks pass I would feel confident submitting the program.