Zachary Wetekamm
CS 162-400
Lab 3 Reflection

Design:

It's known that for the war game, there are two die. Each die will have a number of sides, however the die can have 2 types. User input will decide on these conditions. Using inheritance, I'll construct a base die with **n** sides and a type (basic die will be "Unloaded").

However, if the user chooses a Loaded Die, it will inherit the sides from the Die class, but type will change to "Loaded". Loaded Die has different conditions for a roll, so it will require a separate function. Therefore, the LoadedDie class will inherit the Die class, but have a unique roll member function for its class.

To create the game board, I'll use an array with a set number of rows (say, 100) and columns (2; 1 for each player). Using a loop, I'll iterate each turn using a die's Unloaded or Loaded roll function. Lastly, compare the result of each turn and display the output.

Test Case (before implementation):

Rounds	Player 1 condition	Player 2 condition	Player 1 results	Player 2 results	Winner
3	5 sides, Unloaded	9 sides, Loaded	2, 5, 4 1 win	9, 9, 3 2 wins	Player 2
5	3 sides, Loaded	8 sides, Loaded	3, 3, 1, 3, 3 0 wins, 1 draw	5, 8, 1, 8, 8 4 wins, 1 draw	Player 2

Input Validation: (**x** represents invalid)

Variable	is 0	is Int > 0	is Int < 0	is char 'y'/'n' (yes/no)	is char != 'y'/'n'	Result
rounds	x	valid	x	x	x	valid
Player 1 sides	x	valid	x	x	x	valid
Player 1 die type	x	x	x	'y' = loaded 'n' = unloaded	x	valid
Player 2 sides	x	valid	x	x	x	valid
Player 2 die type	x	x	x	'y' = loaded 'n' = unloaded	x	valid

Zachary Wetekamm
CS 162-400
Lab 3 Reflection

Changes during Implementation:

Luckily, for the input validation portion, my original idea did not deviate. I kept all variables I originally intended, although I adjusted my initial concept for the menu. Using the textbook as a resource, I recalled to Ch.6 (figure 6-14) which constructs a menu using other functions. I created a function for the first question (to play or exit) and a get() function for the user's choice. Next, I implemented while() loops for each input condition, to match whether input was a positive int or a specific char.

When creating the loaded die, I chose a roll function which gives the loaded die a 25 percent chance to roll max value or zero. I wanted to have a simplified high-risk and high-reward factor. Implementing using inheritance was challenging at first; for example, I wasn't passing an argument as reference. This caused some setbacks, but I was able to figure it out as I recalled the textbook.

Displaying the results proved challenging, and my source code did begin to look clunky as I worked around aligning the output. I did also encounter segmentation faults due to incorrectly initializing a few variables *outside* of their current class. This resulted in garbage values in output, but I was able to fine the error.

Test (after implementation):

Rounds: 12	P1: 10 sides, unloaded	P2: 5 sides, loaded	Round Winner
1	4	3	P1
2	6	5	P1
3	6	3	P1
4	10	5	P1
5	3	5	P2
6	1	5	P2
7	4	1	P1
8	7	2	P1
9	2	3	P2
10	10	1	P1
11	3	5	P2
12	8	5	P1
total:			P1 = 8 wins P2 = 4 wins
Winner:			P1 wins game