Ethan Dunham

dunhamet@oregonstate.edu

Lab 2

Dice War design

Problem:

The problem is to create a program that plays war without card, but with dice (AKA Risk without countries). The user plays against the computer. The user will choose the number of sides on each die, who, if anyone, is using the loaded die.

There must be 3 classes, Die, LoadedDie, and Game. Game will be the keeper of the number of rounds and the referee of who wins. The Die and LoadedDie classes will roll the die. I will not use a player class because the player class would keep track of the score or basically be the same as the dice class, but the score was specifically said to be kept by the Game class.

Design:

Int Main()-send user to the menu() function to choose the options of dice sides, if they are using the weighted die, and the number of turns.

Menu(&int, &int, &char, &char) returns the values by reference and validates the input. All values are validated to be more than 0 or char 'y' or 'n' if the loaded dice option. Menu asks how many sides of die for each player individually, then verifies that it is more than 0. Menu asks for number of turns and verifies that user did not enter a negative number. Menu asks if each player is using loaded die individually and verifies that 'y' or 'n' were entered. Every cin statement is on a do while loop that will use the cinFail() function I created in the verification.cpp.

Loaded die will be derived from die and will call die if the die does not have 'y' for the loaded variable.

Main() then sends all the information to the Game constructor.

Game::Game(int FirstPlayerSides, int SecondPlayerSides, char FirstPlayerWeighted, char SecondPlayerWeighted, int turns) The game constructor then keeps the value of turns and creates 2 die objects. It sends the number of sides to the corresponding dice class.

Die/LoadedDie::Die(sides) The dice constructor takes in the dice sides and saves the sides into a variable and if they are loaded or not.

Int Main()Back in main, it calls the boardPlay() function.

Void Board::BoardPlay() This function loops until the number of times played equals the number of turns requested by the user. While looping, it calls the getValue() functions from the dice objects, using the char FirstPlayerWeighted and SecondPlayerWeighted to determine which one to call. It sends a pointer to the weighted class.

Int Die::getValue() This function uses srand and rand % sides to determine the return value. If the roll=0, it set that value as the highest roll for the return, otherwise return the mod value. If the character value for the die is loaded, it will call the Loaded::getValue() function.

Int LoadedDie::getValue() loaded version will do the same as Die version, but add 1 to that dice roll before returning the value and will override the normal die class functions.

Void Board::BoardPlay() Back in boardPlay, it will output the values onto the screen along with the message 'user or computer "(The User or The Computer)rolled slightly better due to their magical dice of magic rolling magic." It will then call rollWinner().

Void Board::rollWinner(int FirstPlayerRoll, int SecondPlayerRoll) will take in the 2 values that were rolled and compare them to determine the winner. It will then increment the current score for that player.

Void Board::BoardPlay() Will then keep looping until the turns equals that which was chosen, then it will call the whoWon() function.

Void Board::whoWon() will compare the scores of the players and then output who won with a congratulations message. If it is a draw, the program will display that it is a draw. The program will then end.

Classes and what they include:

Class Die

Private

Int Sides

Char loaded

Public

Die(sides, loaded)

Int getValue()

Class LoadedDie : public Die

Private

Int Sides

Char loaded

Public

Die(sides, loaded)

Int getValue()

Game

Private

FirstPlayerScore

SecondPlayerScore

FirstPlayerWeighted

SecondPlayerWeighted

Turns

Plays

BoardPlay()

RollWinner(int, int)

WhoWon()

Public

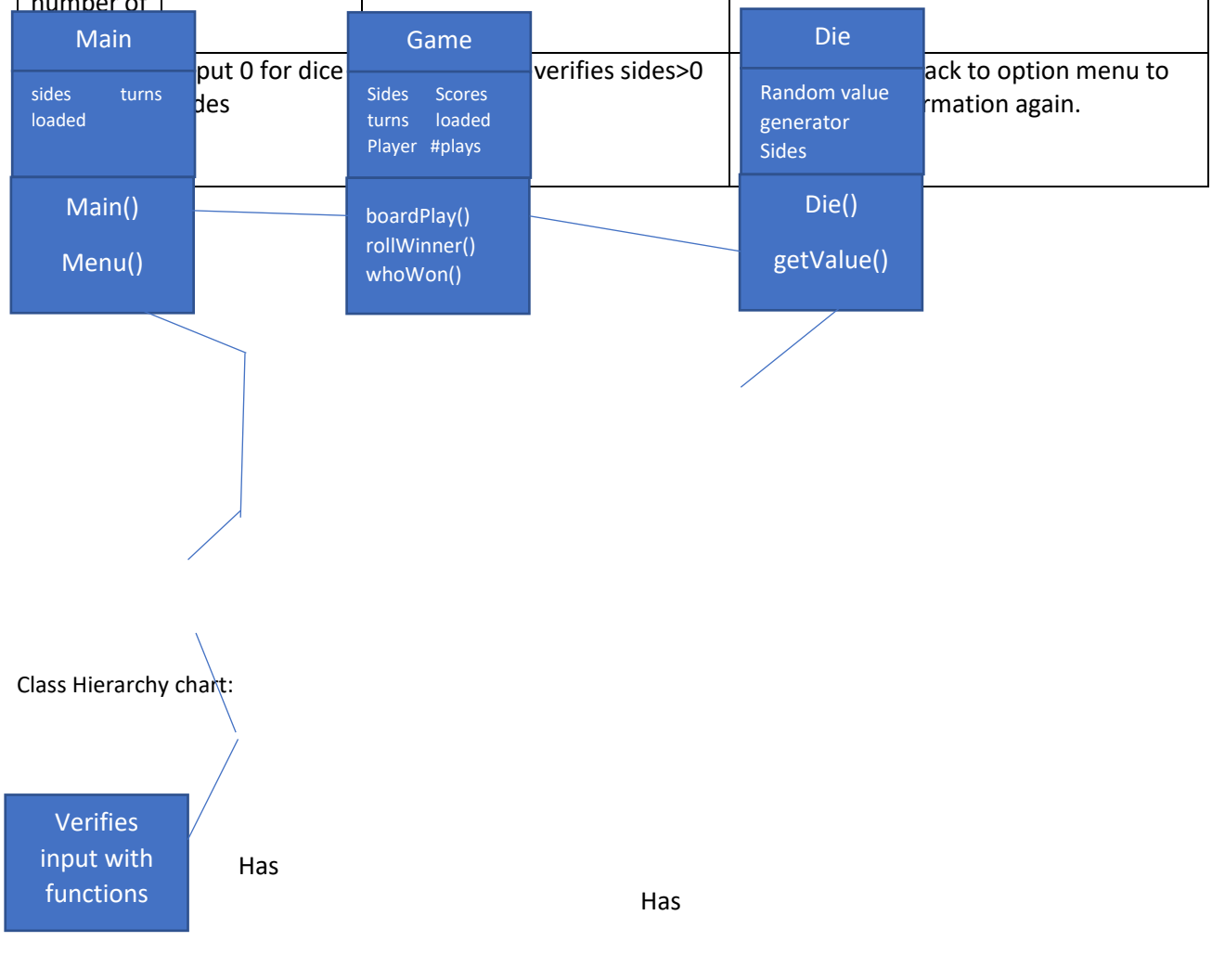
Board(int, int, char, char, int)

Also using the reusable menu functions and user validation functions created for Project 1.

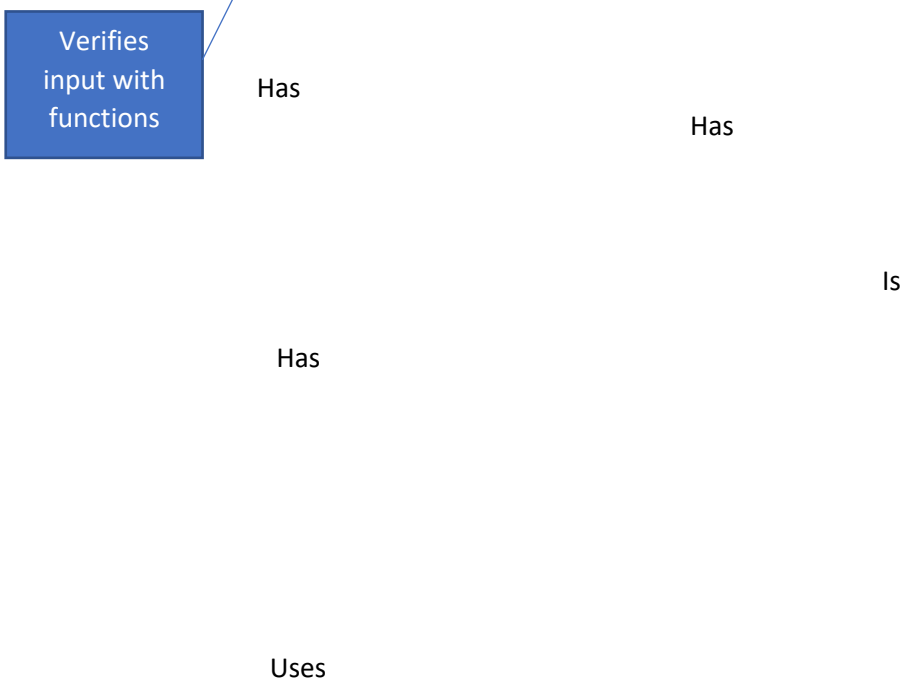
Test input:

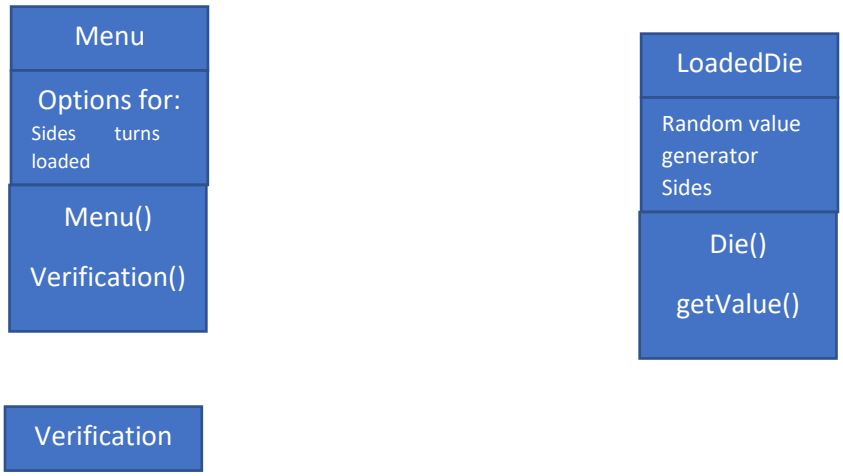
Test Case	Input Values	code	Expected outcome
Too Low	Input<0	Do while loop repeats menu option	User will be given option to repeat their input
Enter char	Input = char	Validation cinFail function called with a do while loop	cinFail function in validation file fixes the issue and lets user enter a new input
1	Input = 1	Code continues	Code runs smoothly for 1 turn
User enters something other than y or n for loaded die option	Input !=y||n	Do while loop repeats menu	Do while loop validates the input and asks the user to please try again.
User enters 0 for	Input is 0 for number of turns	Program runs for 0 turns	Program displays a draw because both players will have 0 wins.

number of			
	put 0 for dice des	verifies sides>0	ack to option menu to rmation again.



Class Hierarchy chart:




Menu
Options for:
Sides turns
loaded
Menu()
Verification()
LoadedDie
Random value
generator
Sides
Die()
getValue()
Verification