Jason Goldfine-Middleton
932675265

CS 162 – Lab 2

Requirements:

For this program, I am responsible for creating a game that uses the Die and LoadedDie classes created in Lab 1. The game's rules are simple. There are to be two players, each of whom will have a die, either balanced or loaded, as chosen by the user. The user will indicate the number of sides on a die and the number of rounds the game will have. For each round, both players will roll their respective dice and the player with the higher roll will have his/her score increased by 1. In the event of a tie, neither score will increase. At the end of the rounds, the game is over and the winner of the game will be printed to the screen.

It is essential to determine the classes needed to implement this game, so I will consider the main nouns in the program specification.

Nouns:
- war
- dice
- user
- players
- game
- number of sides
- regular die
- loaded die
- rounds of play

Nouns (after removing redundancies):
- dice
- player
- game
- number of sides
- loaded die
- rounds of play

Classes:
- dice -> Die
- game -> Game
- loaded die -> LoadedDie (subclass of Die)

Data members:
- players -> p1 and p2 (in Game class, will be Die objects)
- number of sides -> numSides (in Die class)
- rounds of play -> maxRounds (in Game class)

Jason Goldfine-Middleton
932675265

Auxiliary data members for Game:
- round (current round number)
- scoreP1 (current score for Player 1)
- scoreP2 (current score for Player 2)

Auxiliary data members for LoadedDie:
- distributionArr (to change the probability of each side being rolled)

Class Responsibilities:

Game:

Know:	*Member variables:*
The die type for each player	Die p1, p2
The number of rounds to play	int maxRounds
The current round	int round
The current scores of each player	int scoreP1, scoreP2

Do:	*Member functions:*
Set up a game of a certain number of rounds, with dice having a certain number of sides, and with either player having either a regular or loaded die	Game(int roundsToPlay, int numDiceSides, bool p1Loaded, bool p2Loaded);
Play a game	void playGame();
Play one round	void playRound(); [private]
Print winner of game or if none, print that the result was a draw	void printGameResult();

Die:

Know:	*Member variables:*
The number of sides	int numSides

Do:	*Member functions:*
Set up die with specified number of sides	Die(int sides);
Roll die	int getRoll();

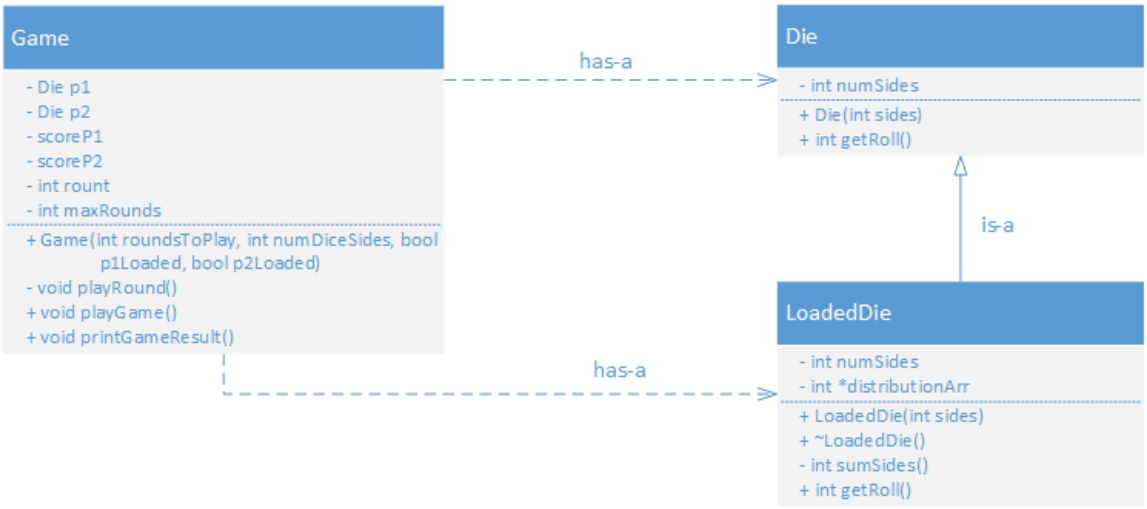
LoadedDie:

Know:	*Member variables:*
The number of sides	int numSides
How to skew the rolls	int *distributionArr

Jason Goldfine-Middleton
932675265

Do:	*Member functions:*
Set up die with specified number of sides and set up skew	LoadedDie(int sides); int sumSides()
Roll die	int getRoll();

Since there are similarities between the Die and LoadedDie classes, I will use inheritance to make the LoadedDie a subclass of Die. This will allow for me to assign p1 and p2 either a Die or a LoadedDie object, depending on user selection.



Main Program Flow:

Loop display menu to user with options:
1) play game
2) display rules
3) exit

Loop until user enters a valid choice, displaying the menu again if invalid
If user chooses 1)
- Prompt user for number of rounds in game
- Loop until user enters a positive number
- Prompt user for number of sides per die
- Loop until user enters a positive number
- Ask user whether Player 1 should have a loaded die (y/n)
- Loop until user enters 'y','Y','n', or 'N'
- Ask user whether Player 2 should have a loaded die (y/n)
- Loop until user enters 'y','Y','n', or 'N'
- Set up Game with user options
  - This will involve setting p1 and p2 equal to either a Die or LoadedDie, respectively, setting maxRounds, setting round to 1, and setting score1 and score2 to 0

Jason Goldfine-Middleton
932675265

Play the Game
Repeatedly play rounds until round = maxRounds + 1
Print each the scores each round and which round number it is
Print the result of the Game
Display the winner or state that it was a draw
If user chooses 2)
Print rules of game
If user chooses 3)
Exit program

Test Plan:

There are a few places in this program that input will need to be validated, so I will begin by testing those. If I know that these values are valid, I will be able to ensure that correct values are passed into the Game constructor.

The user is to be prompted for the number of rounds the game shall last. The user is expected to enter a positive number here. If the user enters an invalid number, then (s)he will be prompted again to enter a positive number. This will happen until a positive number is entered. I can test this by repeatedly entering a nonpositive number and making sure that I am prompted after each input for a correct value. It is also possible that the user may enter a non-numerical character. If this happens, the user should be prompted again for a positive number. I will check this by repeatedly entering non-numerical characters and expecting to be prompted again.

I will use the same approach for ensuring the user enters a valid number of sides per die. The requirement for the input here is the same: it must be a positive number.

The user will be prompted for whether (s)he wants Player 1 to use a loaded die instead of a balanced die. The valid inputs will be 'y' or 'Y' for "yes", and 'n' or 'N' for "no". If the user enters some character other than those 4, (s)he is to be prompted again for one of them. This can also be tested by repeatedly entering invalid characters and ensuring that I am prompted again until I enter a correct one. The same logic applies for testing the analogous prompt for Player 2.

As my Die and Loaded Die classes were tested in Lab 1 by printing out each roll as well as the total and average of all the rolls for each die, I will focus on testing the Game class. To test the playGame() method, I will simply use print statements to show the current value of the round member variable for each roll. This will actually be taken care of by the playRound() method, which will be called repeatedly by the playGame() method.

The playRound() method should ensure that the player whose die roll is higher wins, so I will create a test method that prints the scores for each player after each round, and have it run after each playRound() call in playGame(). If Player 1 rolls

higher, I want to see scoreP1 incremented by 1. If Player 2 rolls higher, I want to see scoreP2 incremented by 1. If the rolls are the same, I want to see that both scoreP1 and scoreP2 have not changed since the last round.

Prompt User for Number of Rounds in Game, main()			
Test Case	**Input Values**	**Driver Functions**	**Expected Outcomes**
Input too low	Input < 1, Input 6	main(), loop input	Prompt user twice, then print valid input
Input extreme low	Input = 1, Input 6	main(), loop input	Prompt user twice, then print valid input
Input extreme high	Input > 10^30, Input 6	main(), loop input	Prompt user twice, then print valid input
Input low, then high, then valid	Input < 1, Input > 10 ^ 30, Input 6	main(), loop input	Prompt user thrice, then print valid input
Many invalid inputs	Input < 1, Input > 10^30, Input < 1, Input 6	main(), loop input	Prompt user four times, then print valid input
Input a character	Input 'a', Input 6	main(), loop input	Prompt user twice, then print valid input

Prompt User for Number of Sides on Die, main()			
Test Case	**Input Values**	**Driver Functions**	**Expected Outcomes**
Input too low	Input < 1, Input 6	main(), loop input	Prompt user twice, then print valid input
Input extreme low	Input = 1, Input 6	main(), loop input	Prompt user twice, then print valid input
Input extreme high	Input > 10^30, Input 6	main(), loop input	Prompt user twice, then print valid input
Input low, then high, then valid	Input < 1, Input > 10 ^ 30, Input 6	main(), loop input	Prompt user thrice, then print valid input
Many invalid inputs	Input < 1,	main(), loop input	Prompt user four

	Input > 10^30, Input < 1, Input 6		times, then print valid input
Input a character	Input 'a', Input 6	main(), loop input	Prompt user twice, then print valid input

Ask User Whether Players Should Have a Loaded Die, main()			
Test Case	**Input Values**	**Driver Functions**	**Expected Outcomes**
Input 'y'	Input 'y'	main(), loop input	Prompt user once, then print "yes"
Input 'Y'	Input 'Y'	main(), loop input	Prompt user once, then print "yes"
Input 'n'	Input 'n'	main(), loop input	Prompt user once, then print "no"
Input 'N'	Input 'N'	main(), loop input	Prompt user once, then print "no"
Input invalid char, then input valid char	Input 't', Input 'Y'	main(), loop input	Prompt user twice, then print "yes"
Input string of chars	Input "TEST", Input 'n'	main(), loop input	Prompt user twice, then print "no"

Game::Game(int, int, bool, bool)			
Test Case	**Input Values**	**Driver Functions**	**Expected Outcomes**
Rounds too low	Game(0, 5, true, true)	Game::Game(), validation block	Print error "Number of rounds too low." and exit
Rounds high	Game(10^30, 5, true, true)	Game::Game(), validation block	Print error "Number of rounds too high." and exit
Rounds extreme low	Game(1, 5, false, false)	Game::Game(), validation block	Print success message stating both players will use regular dice
Sides too low	Game(5, 0, false, false)	Game::Game(), validation block	Print error "Number of sides too low." and exit
Sides high	Game(5, 10^30, true, false)	Game::Game(), validation block	Print error "Number of sides too high." and exit
Sides extreme low	Game(5, 1, true, false)	Game::Game(), validation block	Print success message stating Player 1 will use a

			loaded die and Player 2 will use a regular die
Rounds and sides both invalid	Game(-1, 10^30, true, true)	Game::Game(), validation block	Print error "Number of rounds too low." and exit
P1 loaded, P2 loaded	Game(5, 6, true, true)	Game::Game()	Print success message stating both players will use loaded dice
P1 loaded, P2 regular	Game(5, 6, true, false)	Game::Game()	Print success message stating Player 1 will use a loaded die and Player 2 will use a regular die
P1 regular, P2 loaded	Game(5, 6, false, true)	Game::Game()	Print success message stating Player 1 will use a regular die and Player 2 will use a loaded die
P1 regular, P2 regular	Game(5, 6, false, false)	Game::Game()	Print success message stating both players will use regular dice