Lab 3 Reflection

Byron Kooima - 932707491

CS/162

July 16, 2017

Prof. Luyao Zhang

Lab 3 Reflection

Lab Description

The main focus of this program is to play a simplified version of war using a pair of dice. The basic rules of the Dice Game scenario are:

1. In each round, roll a die of the appropriate type (loaded/normal) for each "player"
2. The higher result wins. If results are equal, it is a draw
3. The winner of the game will be the "player" who won the most rounds

The following options will be made available to the user:

1. Specify the dies sides used by each player.
2. One or both players using loaded dice.
3. Set the number of rounds in the game.
4. Play the game.

Requirements

- Make a Die Game
- The Game is made up of Die
- Die can be either normal or loaded:
    - Number of sides on the die
    - Return the number rolled by the die (random integer between 1 and N)
- Loaded Die
    - Inherits from the parent Die class
    - Return biased number rolled by the die (average output is higher than normal)
- Game has the following information:
    - Number of rounds
    - Option to use loaded die for one or both players
    - Play the game
    - Display the results to the user
        - Indicate the side and type of die used for each player

        - Rolled number for each player in each round
        - Final winner of the game
    - Keep players score

The main nouns in the program design (and whether they are classes) are as follows:

Nouns:

- Dice
- User/Players
- Game
- Rounds (to play)
- Sides (each player)
- Loaded Dice

Classes:

- Dice -> Die
- Game -> Game
- Loaded Dice -> LoadedDie (subclass of Die)

Class Design

The structure of the Die class allows the LoadedDie to inherit the number of sides. The default values for the number of sides of the dice will be 6 and they will be unloaded. The number of rounds will default to 3 to make sure there is a chance for a winning game. The user has the option to update any of these values using the menu and playing the game again.

Class Name	Game (hasA Die())
Data Members	protected member variables: (none) private member variables: **Die* player1Die** **Die* player2Die** **int numRounds** **int curRound** **int p1NumSides** **int p2NumSides** **int player1Score** **int player2Score** **bool p1Loaded** **bool p2Loaded** public member variables: (none)
Member Functions	private member functions: **void play_round()** – Play a single round of Dice War **void print() –** Print the results of the Game protected member functions: (none) public member functions: **Game(int)** – Constructor: Set the number of rounds **~Game() –** Destructor **void play_game(int, int, int)** – Passes the values for the number of sides for each player and the loaded die menu option and plays a full game.

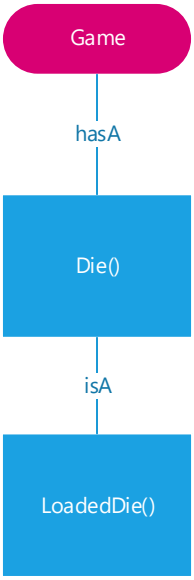
Class Name	Die
Data Members	protected member variables: **int numSides** – Number of sides of the Die private member variables: (none) public member variables: (none)
Member Functions	private member functions: (none) protected member functions: (none)

	public member functions: **Die(int numSides)** - Constructor: Set numSides for the Die **virtual int roll_die() const** - returns a random value from 1 to the numSides of the die **~Die()** - Destructor

Class Name	LoadedDie (isA Die())
Data Members	protected member variables: (none) private member variables: (none) public member variables: (none)
Member Functions	private member functions: (none) protected member functions: (none) public member functions: **LoadedDie(int numSides)** - Set numSides for the Die **int roll_die() const** - Returns a loaded value (using the rand function) from 1 to the numSides of the die

Class Name	userMenu
Data Members	private member variables: **int selectedChoice** **vector<string> choice**
Member Functions	private member functions: (none) protected member functions: (none) public member functions: **void add_choice(string) –** Add menu options to vector **void printMenu() –** Print the menu options **int makeChoice() –** Returns int for the user selection

Class Interactions



Test Case	Input Values	Driver Functions	Expected Outcome	Observed Outcome
Input too low	**Input < 0**	**inputVerification -> SafeInput()**	**Prompt user to enter correct value**	**Prompt user to enter correct value**
Input not an integer	**Input = 0d**	**inputVerification -> SafeInput()**	**Prompt user to enter correct value**	**Prompt user to enter correct value**
Input below range	**Input = -1**	**inputVerification -> SafeInput()**	**Prompt user to enter correct value**	**Prompt user to enter correct value**
Input not a double	**Input = 34d**	**inputVerification -> SafeInput()**	**Prompt user to enter correct value**	**Prompt user to enter correct value**
User chooses option #6 to quit the program	**Input = 6**	**userMenu -> makeChoice()**	**Exit the program normally**	**Program exited normally**
Get Number of sides for Player 1 Die	**Input = 10**	**Die-> Die(numSides)**	**Should obtain the integer number for number of sides between 1 and 20**	**Fail: The number of sides was always defaulting to 6. Incorrectly calling the Die() Default constructor every time. Fixed to set the numSides variable from user input**

Get Number of sides for Player 2 Die	**Input = 3**	**Die-> Die(numSides)**	**Should obtain the integer number for number of sides between 1 and 20**	**Number of sides for player 2 matched (after fix to constructor call)**
Play a normal game with unloaded die	**menuOpt = 5**	**Game -> play_game (int,int,int) Die -> Die (numSides) Game -> print()**	**The game should play for 3 rounds with 6 sides for each player's die.**	**The game runs for 3 rounds and the output shows 6 sides (unloaded) for each player**
Get loaded die for player 1 – user selections option #3 from main menu and sub menu is displayed	**menuOpt = 3 dieLoaded = 2 (player 1 loaded)**	**Game -> play_game (int,int,int) Die-> Die(numSides) LoadedDie -> LoadedDie(numSides)**	**Should load player 1 die and provide a biased game for the loaded die**	**Fail: Player rolls were still equal. Both player's die was automatically loaded by default. Fixed the if statement for the default action. Game was always selecting loaded die for both players**
Get loaded die for player 2 – user selections option #3 from main menu and sub menu is displayed	**menuOpt = 3 dieLoaded = 3 (player 2 loaded)**	**Game -> play_game (int,int,int) Die-> Die(numSides) LoadedDie -> LoadedDie(numSides)**	**Should load player 2 die and provide a biased game for the loaded die**	**Pass after default loaded die fix. Player 2 loaded die won > 75% of the time.**
Get loaded die for both players – user selections option #3 from main menu and sub menu is displayed	**menuOpt = 3 dieLoaded = 4 (player 1 & 2 loaded)**	**Game -> play_game (int,int,int) Die-> Die(numSides) LoadedDie -> LoadedDie(numSides)**	**Should load both player's die and provide equal chance if the die sides are equal**	**The rounds were close to even with 100 rounds switching evenly between player1 and player2 wins**
Get number of rounds from user.	**Input = 100**	**Game -> play_game (int,int,int) Game -> play_round()**	**The game should continue for 100 rounds**	**Game continued for 100 rounds**
Print the game results with information about the sides, type of die used, and the rolled number for each player in each round.	**menuOpt = 5**	**Game -> play_game (int,int,int) Die-> Die(numSides) Game -> print()**	**The game should run for 3 rounds and print the information for each round to include: Die sides for each player, loaded/unloaded die used, and player's score for each round.**	**Fail: The counter was not initially working for each rounds score. The value would reset b/c its scope was only maintained in the function. Fixed the variable by making it private in the Game class**

Reflection

The week 3 lab was a little more challenging than I originally thought. I had a lot of issues with inheritance and the best way to use it in the LoadedDie. The main issue was that I couldn't get the LoadedDie numSides to come out correctly.

I also realized that my menu functions would have been better served inside the Game class. If I refactored the code, I could have made the main.cpp file only contain the play_game function call. I tried to move the menu into the Game class but ended up breaking a lot of things and didn't have the time to completely fix the issues. The results I was getting from the original code were perfect (finally) and everything started going off the rails when I tried to move the menu options into the Game class. I have already started using the menu options inside of the Class objects in the Project 2 homework.