Carrie Davis

Intro Programming 162

Lab3 Project Plan

Program Requirements/Steps

War Game with Dice Design

Create a program that carries out following steps.

1. Requests from user if they want to start program or exit.
2. Starts Game with choice 1.
3. Prompts for number of rounds to play.
4. Takes user input on rounds.
5. Prompts user to define the players dice as either loaded or regular.
6. Prompts user to define number of sides for each die.
7. Creates 2 dice as either biased towards higher values “loaded” or not “regular”.
8. For each round the dice are rolled and scored based on results.
9. Totals for all rounds are tallied to get winner.
10. Results of rounds, details of type, sides and score of each die displayed.

```
Menus


main()
Prompt user if start program or quit
input user choice
1 → start program
```

2 → Do nothing and quit

#% or != 1 || 2 → while choice is not 1 or 2 prompt user to enter 1 or 2*

Game() play method

Prompt user to choose number of rounds

“rounds” int input

#% or < 1 or > 25 → while choice not met prompt user to enter 1- 25, if dec imal value the integer of value is used*

Create Die pointer array. Call createDie.

Call playRounds.

Clear dynamic memory.

Game() createDie method

Prompt user to create a die with user input on sides and type.

“N” sides int input

#% or < 4 or > 120 → while choice not met prompt user to enter 4 - 120, if decimal value the integer of value is used.*

“type” int input 1 for REGULAR or 2 for LOADED

#% or != 1 || 2 → while choice is not 1 or 2 prompt user to enter 1 or 2.*

Game() playRounds method

Arguments: Die pointer array, rounds

int scorePlayer1=0;

int scorePlayer2=0;

For each round:

	For Each Die in array:

		Dice[player#]->rollDie() method called

		scorePlayer# =+ rollDie results;
		Dice[player#]->getSides() method called
		cout return values from methods.

If/else

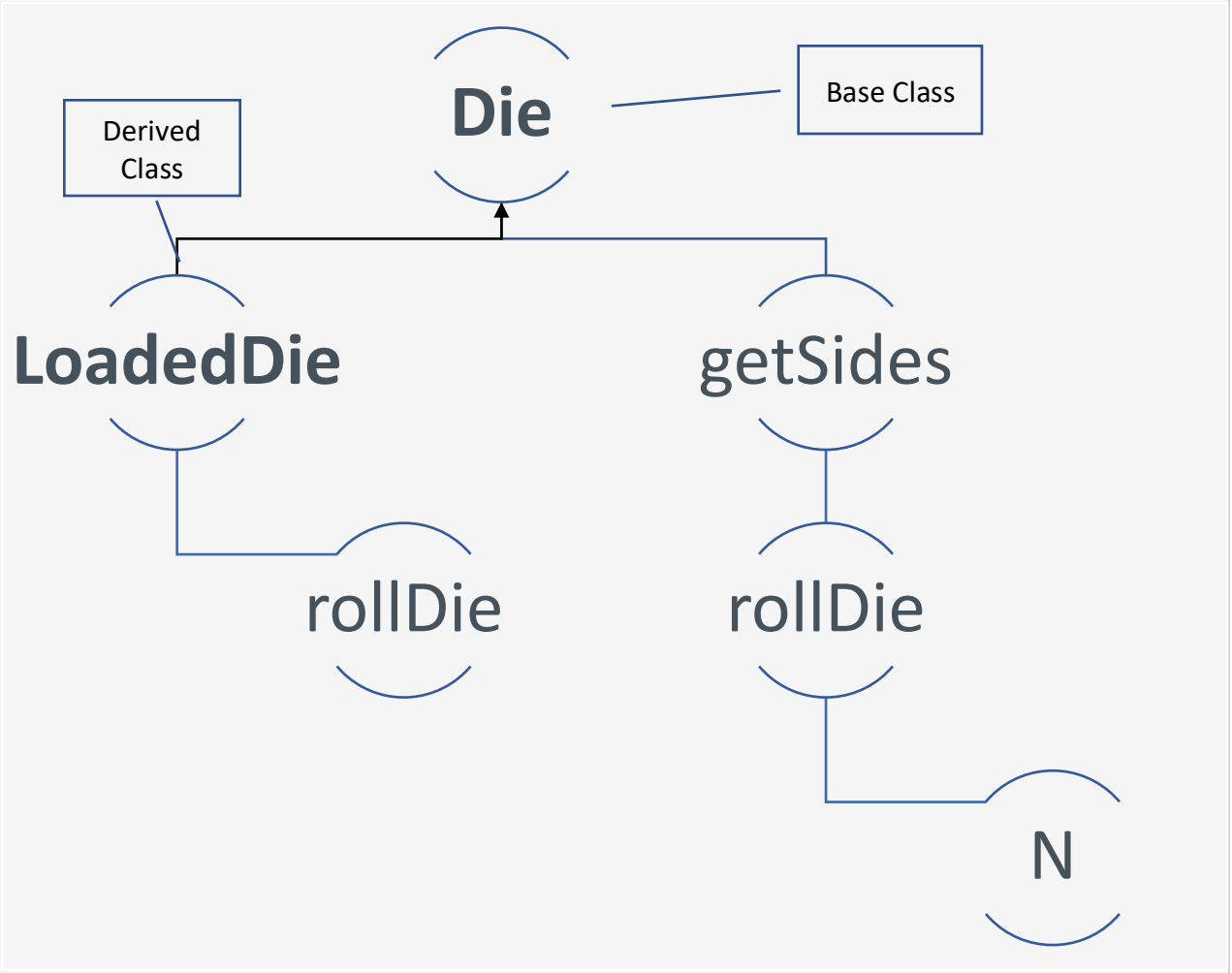
	scorePlayer1 == scorePlayer2 { Tie}

	scorePlayer1 > scorePlayer2 { Player 1 wins}

```
        scorePlayer1 < scorePlayer2 { Player 2 wins}
```

Dice Specification

```
Inheritance/Override rollDie()
```



Test Plan

Tests	*Action performed*	*Expected output*
Test1:	Rounds Entered 1	**rounds**: 1

	Player1 Die type Entered @#$%	`Enter choice: @#$%`
		`Invalid input. Enter 1 for regular or 2 for loaded.`
		`Enter choice:`

	Player1 Die type Entered 1	**type:** 1 “REGULAR”
	Sides Entered %$	`Invalid input. Please enter a number 4 - 120.` `Enter your choice:`
	Sides Entered 1	`Invalid input. Please enter a number 4 - 120.` `Enter your choice:`
	Sides Entered 4.4	**N “sides”**: 4
	Player2 Die type Entered 2.2	`Invalid input. Enter 1 for regular or 2 for loaded.` `Enter choice:`
	Player2 Die type Entered 2	**type**: 2 “LOADED”
	Sides Entered 121	`Invalid input. Please enter a number 4 - 120.` `Enter your choice:`

	Sides Entered 120	**N “sides”:** 120
		
		Print Out Result Example **note roll probability will change with each roll*
		
		`Finally Let's Play War`
		`Round 1:`
		`Player 1:`
		`4 Sides. Regular Die, Roll Result: 3`
		`Player 2:`
		`120 Sides. Loaded Die, Roll Result: 103`
		`Player 2 scores a point!`
		`Current Total Player 1: 0`
		`Current Total Player 2: 1`
		`Player 2 wins with 1`

Test 2:	Rounds Entered 25.1	**rounds**: 25
	Player1 Die type Entered 2%$#	**type:** 2 “LOADED”
	Sides Entered 120.1	**N “sides”**: 120
	Player2 Die type Entered 1.2	**type**: 1 “REGULAR”
	Sides Entered $#@4	`Invalid input. Please enter a number 4 - 120.` `Enter your choice:`
	Sides Entered 4	**N “sides”**: 4 **Print Out Result Example** **note roll probability will change with each roll*

```
Finally Let's Play War
Round 1:
Player 1:
120 Sides. Loaded Die, Roll Result: 106
Player 2:
4 Sides. Regular Die, Roll Result: 1
Player 1 scores a point!
Current Total Player 1: 1
Current Total Player 2: 0
Round 2:
Player 1:
120 Sides. Loaded Die, Roll Result: 116
Player 2:
4 Sides. Regular Die, Roll Result: 3
Player 1 scores a point!
Current Total Player 1: 2
Current Total Player 2: 0
Round 3:
Player 1:
120 Sides. Loaded Die, Roll Result: 97
Player 2:
4 Sides. Regular Die, Roll Result: 1
Player 1 scores a point!
Current Total Player 1: 3
Current Total Player 2: 0
Round 4:
Player 1:
120 Sides. Loaded Die, Roll Result: 115
Player 2:
4 Sides. Regular Die, Roll Result: 3
Player 1 scores a point!
Current Total Player 1: 4
Current Total Player 2: 0
Round 5:
Player 1:
120 Sides. Loaded Die, Roll Result: 111
Player 2:
4 Sides. Regular Die, Roll Result: 1
Player 1 scores a point!
Current Total Player 1: 5
Current Total Player 2: 0
Round 6:
Player 1:
120 Sides. Loaded Die, Roll Result: 88
Player 2:
4 Sides. Regular Die, Roll Result: 4
Player 1 scores a point!
Current Total Player 1: 6
Current Total Player 2: 0
```

```
Round 7:
Player 1:
120 Sides. Loaded Die, Roll Result: 117
Player 2:
4 Sides. Regular Die, Roll Result: 2
Player 1 scores a point!
Current Total Player 1: 7
Current Total Player 2: 0
Round 8:
Player 1:
120 Sides. Loaded Die, Roll Result: 102
Player 2:
4 Sides. Regular Die, Roll Result: 3
Player 1 scores a point!
Current Total Player 1: 8
Current Total Player 2: 0
Round 9:
Player 1:
120 Sides. Loaded Die, Roll Result: 104
Player 2:
4 Sides. Regular Die, Roll Result: 3
Player 1 scores a point!
Current Total Player 1: 9
Current Total Player 2: 0
Round 10:
Player 1:
120 Sides. Loaded Die, Roll Result: 118
Player 2:
4 Sides. Regular Die, Roll Result: 2
Player 1 scores a point!
Current Total Player 1: 10
Current Total Player 2: 0
Round 11:
Player 1:
120 Sides. Loaded Die, Roll Result: 109
Player 2:
4 Sides. Regular Die, Roll Result: 2
Player 1 scores a point!
Current Total Player 1: 11
Current Total Player 2: 0
Round 12:
Player 1:
120 Sides. Loaded Die, Roll Result: 106
Player 2:
4 Sides. Regular Die, Roll Result: 3
Player 1 scores a point!
Current Total Player 1: 12
Current Total Player 2: 0
```

```
Round 13:
Player 1:
120 Sides. Loaded Die, Roll Result: 98
Player 2:
4 Sides. Regular Die, Roll Result: 4
Player 1 scores a point!
Current Total Player 1: 13
Current Total Player 2: 0
Round 14:
Player 1:
120 Sides. Loaded Die, Roll Result: 102
Player 2:
4 Sides. Regular Die, Roll Result: 2
Player 1 scores a point!
Current Total Player 1: 14
Current Total Player 2: 0
Round 15:
Player 1:
120 Sides. Loaded Die, Roll Result: 96
Player 2:
4 Sides. Regular Die, Roll Result: 3
Player 1 scores a point!
Current Total Player 1: 15
Current Total Player 2: 0
Round 16:
Player 1:
120 Sides. Loaded Die, Roll Result: 88
Player 2:
4 Sides. Regular Die, Roll Result: 1
Player 1 scores a point!
Current Total Player 1: 16
Current Total Player 2: 0
Round 17:
Player 1:
120 Sides. Loaded Die, Roll Result: 102
Player 2:
4 Sides. Regular Die, Roll Result: 2
Player 1 scores a point!
Current Total Player 1: 17
Current Total Player 2: 0
Round 18:
Player 1:
120 Sides. Loaded Die, Roll Result: 95
Player 2:
4 Sides. Regular Die, Roll Result: 4
Player 1 scores a point!
Current Total Player 1: 18
Current Total Player 2: 0
Round 19:
Player 1:
120 Sides. Loaded Die, Roll Result: 80
Player 2:
4 Sides. Regular Die, Roll Result: 3
Player 1 scores a point!
```

```
Round 20:
Player 1:
120 Sides. Loaded Die, Roll Result: 111
Player 2:
4 Sides. Regular Die, Roll Result: 1
Player 1 scores a point!
Current Total Player 1: 20
Current Total Player 2: 0
Round 21:
Player 1:
120 Sides. Loaded Die, Roll Result: 98
Player 2:
4 Sides. Regular Die, Roll Result: 2
Player 1 scores a point!
Current Total Player 1: 21
Current Total Player 2: 0
Round 22:
Player 1:
120 Sides. Loaded Die, Roll Result: 84
Player 2:
4 Sides. Regular Die, Roll Result: 4
Player 1 scores a point!
Current Total Player 1: 22
Current Total Player 2: 0
Round 23:
Player 1:
120 Sides. Loaded Die, Roll Result: 94
Player 2:
4 Sides. Regular Die, Roll Result: 1
Player 1 scores a point!
Current Total Player 1: 23
Current Total Player 2: 0
Round 24:
Player 1:
120 Sides. Loaded Die, Roll Result: 106
Player 2:
4 Sides. Regular Die, Roll Result: 3
Player 1 scores a point!
Current Total Player 1: 24
Current Total Player 2: 0
Round 25:
Player 1:
120 Sides. Loaded Die, Roll Result: 104
Player 2:
4 Sides. Regular Die, Roll Result: 4
Player 1 scores a point!
Current Total Player 1: 25
Current Total Player 2: 0
Player 1 wins with 25
```

Reflection

I had issues with what data members should/had to be in base “Die” class versus what should be left to be to specialization in LoadedDie. It was quickly apparent that rollDie() needed to be overrided in the “LoadedDie” class and that I needed to also use all other data members from the “Die” base class except for rollDie() if LoadedDie. I was able to apply information in 11.15 in book to help to define the following in “LoadedDie.hpp” that allowed me to use data members from “Die” and also override rollDie() method.

```
LoadedDie(int N): Die(N)
   {


   }
     int rollDie();


};
```

Chapter 11 contains lots of new information. I definitely this information could be split up into multiple chapters as there was a larger learning curve for this lab. I find the book confusing because it doesn’t adhere to the same design and mult-file system taught in Computer Science 161. None of the programs in the book in this chapter show hpp files. The examples also in the lectures did not use hpp files either. I did continue to use them and did spend time to sort out which data members I needed to define in “LoadedDie.hpp”.

I continued to use incremental development and start with Die class and main then created and separated out Game class methods. Last I created LoadedDie and specifics of Game.

I created createDie() to compartmentalize creating players so that more players could be added in the future. I did have a difficult time with creating a pointer array to dynamic Die or LoadedDie instances. I found help on this with an article on http://www.cplusplus.com/forum/beginner/159581/. I would look into using shared pointers in the future, but I still fill uncertain and unpracticed on using them. Another issue I had was with overriding the rollDie() method with my array. The derived LoadedDie instance was using the base Die.rollDie(). I was able to correct by adding the following to my Die.hpp to allow my base member to be overridden in the derived class.

```
  virtual int rollDie();
```

I had initially designed the Die class with an enum to implement a loaded die. However, I understand that's this particular design of creating a separate class for the rollDie() method was a good simple introduction to inheritance and overriding a method.