Peter Moldenhauer
moldenhp@oregonstate.edu
CS 162 – 400
Lab 3

Lab 3 Reflection Doc

Initial Design:

In lab 2, I wrote a design document to prepare myself for the actual implementation and coding of the program that I was to build. In this design document I first presented the initial problem that I was to solve and the specific requirements that went along with it. I then picked apart the problem description to identify the specific nouns that would eventually become the classes I was to use in my final program. The classes that I decided on were the Die, LoadedDie and Game classes.

Along with identifying the classes, I also decided on what each class would need to know and what activities each class was responsible for doing. More specifically, this meant finding what data members and functions to use in each class. In short, the Die and LoadedDie classes were responsible for rolling a die of a specified number of sides. The Game class was responsible for setting up the game with the specified number of rounds, playing the game and then printing the results of the game.

In this design document I also presented a rough outline of how the main function will look and what activities will take place in the main function. In the main function I was to prompt the user for data (types of die, sides of die, rounds to play), validate all of the input and then of course create a Game object to set up and play the game.

The final thing I covered in the document was a testing plan to make sure that the program ran properly and would not accept bad input. I discussed the specific areas of the program in which I would have to test and the specific expected outcomes of each test.

Design changes and analysis:

Overall, my initial design is fairly similar to the finished program that I ended up building. The design gave me a nice road map to work from. However, throughout the implementation and coding process, I did ending up changing some things from my initial design.

The first change I made in the actual program compared to my initial design was how I went about creating the Die objects inside of the Game class. In my design I did not go into specifics of how exactly I was going to go about creating the Die objects, I just knew that I wanted to create the objects inside of the Game class. I mentioned in my design that I was going to use inheritance to accomplish this but at the time I did not fully understand what this meant. At the time of writing my initial design, I should have spent some extra time researching and learning about inheritance so that I could specifically state in the design how I would implement this in the program. Ultimately, used bool variables based on user input in my main function to determine if each player's die is loaded or not. Then, I passed in the bool variables in the constructor to create the Game object. Inside of the Game object constructor I then dynamically created a Die object based on each of the bool variables that I passed. When I was initially writing the design, in my mind I was imagining that I was going to end up statically creating four die objects automatically (each player would get a die or loaded die object created). However, as I was writing the code for the program, it just made more sense to dynamically create only the objects that I would need.

Another change I added in the program was to clear the screen after the user inputs data but before the game is played. I did not think about this at the time of writing my design, but during the coding and testing phase, it seemed like a good idea. Without clearing the output screen, the output seemed very cluttered and overwhelming. I wanted to make sure that the user would not have any trouble seeing the output of the scores and die rolls of each round. Clearing the screen between getting the user input and playing the game seemed like a good fix to this.

A final change that I made to the program that I did not include in my initial design was to add and format additional output information to the user in a specific manner. The dividing lines between each round and additional informative output text to the user are just a few of the things that I implemented during coding. I coded the program so the specific types of die used for each player was outputted to the user again after input so the user is reminded of the die specifics for each player. Also, the specific die rolls and player scores of each round I have outputted to the screen to eliminate any sense of doubt that the user might experience during the gameplay. I understand that a lot of the line spacing and indenting is probably too detailed to include in the initial design, but it would have still been good to start thinking about these ideas earlier on in the process.

Problems encountered and how they were solved:

There were numerous problems that I encountered throughout the process of coding which either prevented the program to not compile or caused the program to not run properly. The first of which was that when I did get my program to compile, the roll of each die was the exact same for each round. At the start, I coded the program so that each rollDie function seeded the rand function and then used the rand function to generate a number within the number of sides of the die. It turns out that I did not fully understand how the rand function actually works. After some research online, I realized that I needed to seed the rand function only once in my program and not in every time the rollDie function was called. This was a simile fix which now allowed the rollDie functions to return random numbers every single time they were called.

The other issue that I had with the program was that it was not creating a LoadedDie when it was supposed to. Instead, the Game function automatically created a Die object each time, despite whether or not the user specified one of the players to have a loaded die. I added various cout statements in my LoadedDie class and sure enough, even if the user chose to have a loaded die for each player, only the Die objects were being used. Between reading the textbook, getting some advice from fellow students and though the course of my own research online, I realized that I needed to make the rollDie function a virtual function. Seeing that this is my first time writing a program that uses inheritance, all of this was completely new to me. I need to continue to read and learn about inheritance and virtual functions so I can gain a better understanding of why this actually works and how I can use this concept in future programs that I will build.

Final test results:

The following charts give a visual representation of some of the tests I ran on the final program. Based on the charts, it is clear to see that when one player uses a loaded die, that player wins more often than the player using a regular die. Similarly, when both players are using regular die or both players are using loaded die, then each player has roughly the same chance of winning the game. Note: each one of these charts are based on testing games of 500 rounds.

Both players using LoadedDie:

Player 1 Score	Player 2 Score	Winner
188	204	Player 2
193	201	Player 2
205	196	Player 1
201	211	Player 2
218	194	Player 1

Only player 1 using LoadedDie:

Player 1 Score	Player 2 Score	Winner
243	172	Player 1
247	162	Player 1
247	165	Player 1
261	156	Player 1
242	164	Player 1

Only player 2 using LoadedDie:

Player 1 Score	Player 2 Score	Winner
159	251	Player 2
183	243	Player 2
160	264	Player 2
159	256	Player 2
163	244	Player 2

Both players using Die:

Player 1 Score	Player 2 Score	Winner
219	199	Player 1
195	203	Player 2
189	217	Player 2
205	219	Player 2
212	204	Player 1