Jason Goldfine-Middleton
932675265

CS 162 – Lab 3

In order to create an inheritance relationship between the Die class I originally wrote in Lab 1 and the LoadedDie class given by another student, there were a number of changes that had to be made.

In the other student's LoadedDie.h file, I commented out their "loaded_sides" field so that the "num_sides" field in my Die class would be used. I also changed the name of their "loaded_roll" function to "die_roll" so that it would override the "die_roll" function from the Die class. I also created a constructor that took an int as a parameter. Most importantly, I appended ": public Die" after "class LoadedDie", which made the LoadedDie class inherit from the Die class.

In the other student's LoadedDie.cpp file, I added the implementation of the LoadedDie constructor, which simply called the Die constructor taking a single int parameter. In the "set_sides" and "die_roll" functions, I renamed "loaded_sides" to "num_sides". I also had to rename the function "loaded_roll" to "die_roll".

My Die.h file also needed a minor change to make this all work. I had to prefix the "die_roll" prototype with the keyword "virtual". This was crucial for allowing polymorphism to happen in the Game class.

I began my testing regime using my original test file from Lab 1. The results were as expected. The LoadedDie produced noticeably higher rolls on average than the Die. I did have to modify my original test file to match the function name "die_roll" instead of "get_roll" as I had originally named it, but aside from that no other change was needed in my test code.

Once I was convinced the two classes worked as expected, I moved on to testing using the Game class I had designed in Lab 2. The results of games where both players used the same type of Die were about 50-50. Surprisingly, the results of games where only one player used a LoadedDie were the same. It seemed that the Die variables storing LoadedDie objects were not using polymorphism. That is, the LoadedDie overridden "die_roll" function was not being called. It was at this point that I realized I needed to make the "die_roll" function of the Die class virtual. Additionally, I realized that if I did not utilize pointers to Die objects, polymorphism could not work. This required a bit of adjustment. I made the Die fields of my Game class pointers to Die instead. In order to prevent memory leaks, I also had to implement a destructor that freed the memory from these fields. After these adjustments, the results were what I had originally expected.

The following tables contain a representative sampling of the results I found. As expected, the player with the LoadedDie won every time by a large margin. When both players used the same type of Die object, the results were about 50-50. There were some draws as well, but only in the latter case.

Jason Goldfine-Middleton
932675265

Testing With 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