McEwan Bain

Professor Crandall

CPSC 224 01

20th September 2023

Summary of HW 1

Goal/Purpose: The purpose of this homework assignment was to get an initial build of a farkle game working. This build is supposed to be able to run a single round of farkle, roll a hand of 6 dice for the player and let the player choose dice from that hand to make a meld. Once the player chooses to bank, the program checks if the meld is valid and then adds the value of the meld to the player's bank. If a "farkle" is detected, the program displays the user's hand and immediately ends the round.

Overview of Design Choices: This program ended up having 7 different classes, including the Die class provided by Professor Crandall. The other six classes used are labeled as follows: Farkle, Round, Player, Hand, Meld, and Combo. Farkle is the main class of this farkle program. Its job is to run an instance of farkle. Die is a class used to create die objects that have a side up and can be rolled to change the side up. The Round class is designed to manage a single round of farkle. To this end, it creates a Player and keeps track of whether or not the round is over or not and calls methods to create a user interface for the farkle game and to allow the user to make choices on which dice to add to a meld. The Player class keeps track of a user's Hand, Meld, Combos, and pointBank. Its methods are mostly used to manage/switch the dice between the Players Hand, Meld, and Combos as well as obtaining scoring. The Hand class contains an ArrayList of 6 randomly rolled 6-sided dice as well as an integer array that contains the number

of each value of dice in the ArrayList. Methods in this class are used to get and manipulate the dice in Hand as well as determine whether or not the user has rolled a “farkle”. Similarly to Hand, Meld is a class that also contains an ArrayList of 6 6-sided dice, however all of the dice in Meld's ArrayList are initialized to 0. Methods here are once again used to manipulate the dice in the ArrayList. Finally we have the Combos class which has an integer containing the value of the combos and an ArrayList containing dice. Its methods are used to add, manipulate, and remove dice to/from the ArrayList and perform all of the scoring operations for possible meld combos. This iteration did not use any inheritance, however I am considering where it can be used for the next couple of assignments.

Design/Programming Issues: Probably the biggest issue I faced was getting the scoring system to work properly. I probably spent 50% or more of the time I was working on this assignment on trying different ways to score and fixing the bugs I encountered. Just when I thought it was working fine I would encounter a small issue with things like the Players pointBank updating when an invalid meld is entered or some infinite looping I had accidentally created. All of this was made just a little bit worse to deal with because of how many nested if-statements and loops I had in this section of my code. It was annoying, but I eventually worked through it.

A side note about a relatively minor issue I ran into: When I tried to use some switch statements to deal with user inputs the program was doing a weird thing where it would go through the case selected and every case after it. I couldn’t figure out what was causing it, so I ended up just using an if-else tree.

Retrospective: If I had a bit more time to spend on it, I would have liked to optimize the scoring system a bit better. What I have now works, but it's long, inefficient, and can be hard to read and understand at points. Specifically I was thinking about another way to deal with triple, quad, five of a kind, and six of a kind combos. The idea I had would deal with all of them in one if tree instead of 5 separate ones and would use some simple multiplication to deal with the scoring (value of triple * (numDice-2), except in the case of ones).

UML Diagram: Note: I'm Including a png of the uml in the summaries folder in github