McEwan Bain

Professor Crandall

CPSC 224 01

8th October 2023

Summary of HW 2

Goal/Purpose: The purpose of this homework assignment is to add to our original Farkle assignment. While farkle is still currently only a single round, users can now reroll un-melded dice. The game now also detects and scores Hot Hands. Some other small niceties were added as well to make the game look better and be more user friendly.

Overview of Design Choices: This iteration of the 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, Game, Round, Player, Hand, and Combo. Most of these have the same function as the previous build, aside from one addition and deletion. 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 Game class is a new class that manages a whole game of farkle. Currently only functional for one round, but does things like get username and instantiates rounds. 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, roundPoints 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 has now taken

over the meld class as well. It 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. Also now has a DVC that creates 6 dice of a specified side up, this is how meld has been replaced. 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” or a “hot hand”. 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. Now the scoring for combos has been split up into serval different methods all called in the same method rather than having one massive method that does all the scoring. Still no inheritance being used, but maybe in future iterations.

Design/Programming Issues: I didn’t run into a bunch of programming issues this time around. There were some minor issues when I first reworked the scoring system, but with the help of new unit tests those issues were resolved fairly easily. There were also a few problems with for loops I had been using when I started shrinking the size of hand during rerolling, lots of out of range calls. But that was also fixed fairly easily by replacing some hardcoded values with arrayList.size() calls. I did run into some design issues that also involved the changing of hand and meld sizes and while they were a bit annoying, they were also fairly easy to fix.

Retrospective: If I had more time for this assignment, I would have written more unit tests. I only ended up doing tests for my Combos class and my program definitely could have benefited from having more unit tests. Would have saved me a lot of time in testing for bugs.

Unit Test: One unit test I wrote for the Combos class, testCheckForSequenceTrue(), instantiates an int array length 6 filled with 6 1's. This array is used to call Combos checkForSequenceTrue() which returns a true/false value which is given to a boolean variable "actual". Actual is then asserted to equal an expected value(true).

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