**Goal/Summary:** The overall goal of this program is a game where either one player can attempt to reach a certain score, or multiple players can compete to see who can reach a certain score first. To do this, each player gets a hand with a set of dice, and given they didn’t Farkle, they can choose which die they want to move to the meld. The player’s score is calculated. Based on which die are in the meld, the player’s score is calculated per round and added to their total score. The player can reroll any die not in the meld and add it to a new meld, accumulating more points towards their total score. Players can also bank their score, moving on to the next round where they receive a new hand. To end the game, the player can either quit, the game will end when one player reaches the certain score to win, or all players are out of the game.

**General Design:** In the design I created for my program, I created the Farkle, Player, Hand, Meld, and Combo classes. I used the Die class Professor Crandall created to create six Die instances within the Hand class, which I rolled and added to an Array List to access later. The Hand class also contained the test to see if the Hand Farkled or was a hot hand, I also used the Die class to create a new Die to reroll. I then began the interaction with the user by outputting the six dice they rolled and allowing them to move any die to their meld, bank their points, quit the game, or reroll. To take in this input from the user, I used an if statement if the user chose to move a die, tested if the die was being moved to or from the meld, and performed that action. To do this, I used the Meld class with a method that allowed integers to be added to its dice array field. These integers were the side that was up on the die being moved to the meld. Similarly, the side that is up could be removed from the array or returned to another class. When a die’s face is added to the array, it is also added to the combo array in the Combo class. The combo class has methods that test each test case that could produce points and return a score, which is used in the meld class when calculating the meld score. This meld score is returned to the main function in Farkle and output on each screen. All of these actions go between the Player class in order to access all other classes, as the Player class holds each player’s score, hand, meld, and name and calls functions to change any of those things in the Hand or Meld classes.

**Unit Test Description:** I used two unit tests for each of the combo class tests, for example, checkForOnes() has two tests for either true or false, which take in an array that is supposed to be the random dice, then puts them through the checkForOnes() function and asserts that it is either true or false depending on the test.

**Design/Programming issues:** I had some issues while creating the getRerollHand() method, as it was confusing to switch between an array that held the side up of each die and the other that held the number of each side up in the list of die. Additionally, switching between the integer side up of the dice and the Die itself was difficult. This also made the method less readable, making it more challenging to go back and edit or add to later. I addressed this by making multiple functions in different classes, as each class had private variables important to the function. The first function is in the Player class, which creates a new meld, rerolls the dice in the hand, checks if the hand is a hot hand, and returns the new dice. The hand class performs a check for a hot hand by putting the hand into a temporary meld to see if the. Meld is viable, which would make it a hot hand. The hand class also outputs a message to the console and takes in the user’s response regarding whether or not the player wants to roll a new set of dice.

**Looking Back:** If I were to have more time, I would make my Hand, Combo, and Farkle classes more readable. Specific functions within the Hand and Combo classes were complex to read, like getRerollHand(), checkForHotHand(), hotHandTrue(), or the combo tests, but the Farkle class was not very clearly written out—the order is not very easy to follow in terms of the way the gameplay works itself out. I would go back and use more comments laid out in a way that is easy to follow and read.

A diagram of a computer program

Description automatically generated with medium confidence

Class Diagram

Sequence Diagram

A diagram of a company

Description automatically generated with medium confidence

Requirement #1 game menu:

A screenshot of a computer

Description automatically generated

Full House combo:

A screenshot of a computer game

Description automatically generated

Someone winning the single player game:

A screenshot of a computer

Description automatically generated

Multiplayer:

Entering multiple players:

A screenshot of a computer screen

Description automatically generated

The total scores and the winner's screen:

A screenshot of a video game

Description automatically generated