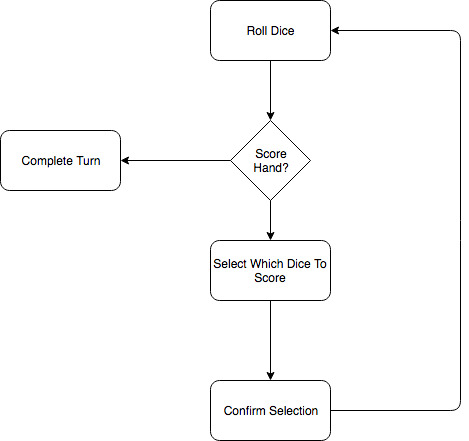
Design Alternative Analysis

**Problem 1**: Handling Scoring Dice

Unlike in Yahtzee, the hand is scored each roll, and possible outcomes are dependent on the amount of dice being rolled. For example, a player rolls are six dice, every outcome is possible, however if they only roll two dice, the only way they can score is if either of those dice are a 1 or a 5. We have to decide how to handle sending the dice the players wish to score to the scorecard.

**Alternative 1:** Send all dice player wishes to score to scorecard

Design: After their roll, the player will choose every dice they wish to score, and then confirm. This will send every dice that they wish to score to the scorecard class.

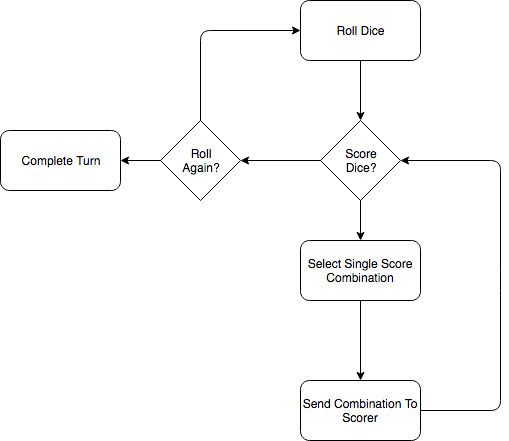


Pros: This would be the clearest way to design from a user perspective. It works the closest to it would in real life.

Cons: The main con of this method is that it makes scoring rather complicated, because a dice cannot be scored twice. For example, a hand of 554444 should be 1500 points, but we would have to find a way to ensure that the fives would be counted as part of the scoring roll 4 of a kind and a pair, rather than simply as fives.

**Alternative 2:** Send a single scoring dice combination to the scorer at a time

Design: After they roll, the player will select a single scoring combination, send it to the scorer, then repeat until they are done with their turn.



Pros: This would be the most straightforward way to score dice. For example, if a user selects 333 from a hand of 123334 would only consider the scoring options that are valid for 3 dice, instead of having to consider scoring options that are valid for 1, 2, or 3.

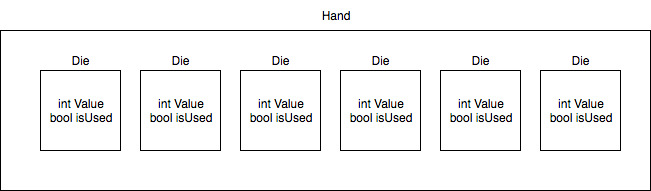
Cons: This is against how a person would play a physical game of Farkle. While it isn’t unreasonable, it does add extra steps and a layer of complexity to the game that could be avoided if we use the other strategy.

**Problem 2**: Handling Used Dice

After a player rolls their hand, they may choose to score only a portion of the dice in their hand, and then re-roll the remaining. We have to have a way to edit the size of the hand between rolls. For example, A player may roll the hand 555342. If they score the dice combination 555, we need to be able to only have the remaining three dice in play.

Alternative 1: Modify the Die class to have each die contain a Boolean representing if the dice is in play or not.

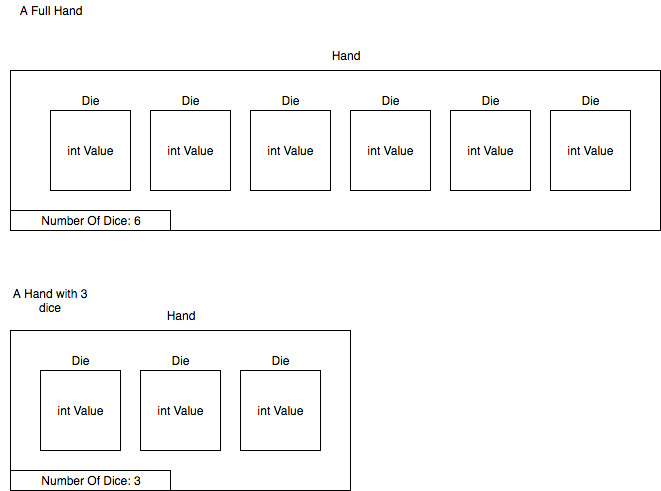
Pros: This is the simplest solution. It would be very similar to our implementation of the string in Yahtzee, except instead of the string belonging to the hand, it would belong to each individual die. Containing the Boolean as a data member of die would also allow for sorting of the hand without having to simultaneously sort the string, as we would if we used the Yahtzee implementation. This implementation would also be very easy to use in conjunction with action listeners, which may be helpful in designing our GUI.



Cons: If we make the Boolean a private data member, we will have to access it and set it via getters and setters, adding another step to the coding process.

Alternative 2: Allow a Hand to be constructed of any number of dice.

Design: Initially, a Hand will be constructed with six dice. After the player chooses the dice to score, a new Hand is constructed of the remaining dice.



Pros: Eliminates the step in rolling to see if the dice is in play. Easier to implement the option of choosing one scoring combination from problem 1.

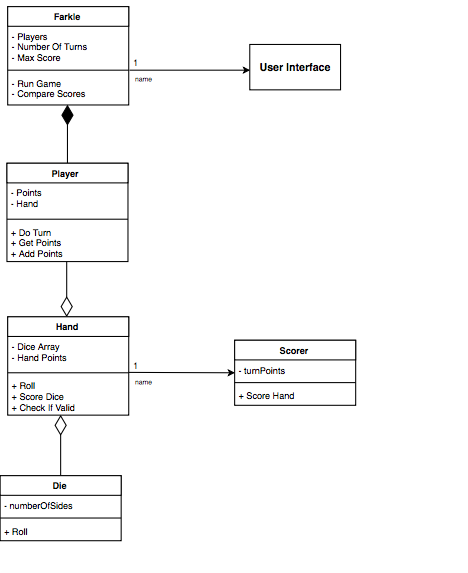
Cons: Potentially a superfluous solution, may introduce more problems than it solves.

**Problem 3**: Farkle on the First Roll

When a player rolls their hand we need to check if there are any scoring dice in that hand to see if the player has farkled or not. Our checkIfValid function is currently contained in class Scorer. Scorer requires the player to first choose their dice to keep. With this class design, we are failing to evaluate whether or not the player's turn can continue or not.

Alternative 1: Move the checkIfValid function to class Hand.

Design:

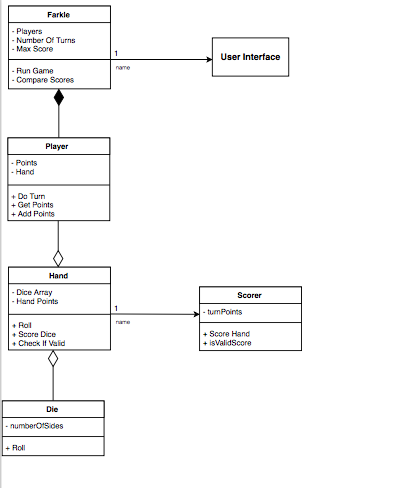


Pros: If we move checkIfValid to Hand we will solve the problem of evaluating if the player has farkled or not on the first roll. This is a simple re-design of our classes.

Cons: If checkIfValid is moved to Hand, then the dice will be evaluated for scoring before the user chooses which dice they would actually like to score.

Alternative 2: Separate checkIfValid into two separate functions. The checkIfValid function in class Hand will simply check if there are any scoring dice in any hand that the player rolls. The other function seeIfValidScore will be contained in class Scorer.

Design:



Pros: This alternative solves the problem of farkling on the first roll as well as fixing the preliminary scoring issue.

Cons: The con of this design is necessary. The function seeIfValidScore is a complicated function with a set of its own challenges similar to Problem 1 in how the function will handle various scoring options.