Summary of purpose and goal of project:

The overall goal of this project was to allow the Yahtzee user to be able to configure the number of sides on the die, the number of dice in a hand, and the number of rolls allowed per turn. In order to do this, the program opens a txt file with the initial configurations and asks the user if they want to change them, if they decide to change the rules, the new rules are applied to the current game and saved to the file. The benefits of allowing the user to change rules is that for one, it gives the user a lot more power into how they want to generate the game, but also it makes the game more dynamic with a brand-new scoring system. The upper scorecard now has to create a score for each die number on our new dice, and the new lower scorecard would have to be adjusted for the number of dice in play. Fortunately for me, in my scorecard class, the upper scorecard and lower scorecard implemented the new dynamic rules already, besides full house. Therefore, the overall goal of this project for myself was to create a new class that created these new rules and allow other classes to easily use these rules.

Overall Design:

How I went about the overall design of this project was to implement a new Rules class that basically stores the new rules that are either saved in file or set by the user. The Rules class contains three private static attributes: turns – stores the number of rolls allowed per turn as an int, handSize – stores the number of dice in a hand, sides – stores the number of sides on a die. I made these attributes static so that in order to access these new configurations in other classes, you don’t have to instantiate a new Rules object every time. These three attributes are private however, so they cannot be modified in other classes which is what we want. In order two access the new configurations, I created a public static get function for each of the attributes(i.e. getHandSize() etc.). Therefore, if the other classes need to access these new attributes, they just call one of the getters. For example, in my Die class, I originally had a private sides attribute that was set to six for the normal rules. In order for the Dice class to now implement the new configurations, all I had to change was say ‘private int sides = Rules.getSides()’, and the Die class now implements the number of sides set in Rules to its current object. The Rules class also contains the public method setRules(), which basically reads in the txt file of configurations and asks the user if they would like to change the rules, if they elect to do so, the private helper function setNewRules() is called. This function asks the user for the new rules and sets the configurations for the current game and saves them to the file. Overall, the design of this project turned out very well, and was not too difficult to implement based on the code I implemented for the previous assignment.

Graphical user interface, application, Teams

Description automatically generated

Design issues:

This biggest design issue for this project was deciding how the three new attributes in the Rules class, (sides, handSize, and turns), would be accessed in other classes. I initially had them as just private variables, although I realized this wouldn’t work as I would have to instantiate a new Rules object every time I wanted to access these rules in another class. I then decided to make them normal public static variables, which worked fine, although I realized this allowed other classes to be able to modify the new configurations, which isn’t a good idea. Eventually I decided to make the variables private and static, and create getters for each of the attributes, which as explained above seems to work well. The other issue I was having was implementing file IO in the rules class, as there are tons of interfaces and methods for using file IO which can get confusing. I was getting an error for trying to create a scanner for the file that took a while to figure out, until eventually I implemented the ‘throws’ keyword which allowed the methods to work.

What I could’ve done differently with more time:

Overall, because of the amount of time I spent in the last assignment on getting the code to be more reusable for future assignments, I didn’t spend too much time on the actual code for this assignment. Most of the time I spent on this assignment was cleaning up smaller issues from the last assignment and deciding how to implement the new Rules class in the best way. The one thing I wasn’t able to accomplish in this assignment was implementing the full house for the new configurations. I tried to analyze the graph that was put in the assignment and try out some code, although it got very confusing and I was more focused on parts of the assignment that were actually required. In future assignment, where the new full house implantation is needed, hopefully I will be more motivated to implement it and my scorecard class will be able to easily implement it.