**Yahtzee Project – Team D Report**

This is a quick take about our month of project on the Yahtzee game. Our development was staged around several parts that were mostly not linked together, allowing parallel work. These parts were the menu, the roll and reroll process, the score as well as a rudimentary AI. To all of this together, we mostly used GitKraken, a very useful interface of the famous Github, granting us swift updates to our locally stored project without having to put a mind to it.

The menu was a pretty much painless set of methods to set up, except for one big issue, unresolved to this day: a mysterious data buffering situation in our scanner that forces the user to press enter to clear said buffer before having his own input processed. We tried many solutions and did extensive research but for no avail… Except that, we just check for an integer that belongs to the range of accepted values that will chain into our game.

The heart of the game is the hand of dice: to create it we just set an array of 5 dice, assigned by default with the value -1 (keep that in mind for later, it is a crucial point). Through basic random methods, we can achieve our range of 1 through 6 for each of our die. Now for the reroll: as we said, unassigned values are set to -1. With this method we can know which die to keep and which to reroll. However, we create beforehand a failsafe that will keep in store the original hand, so that the user can backtrack if he sets the wrong die to reroll or simply changes his mind. Simply plug in a counter that ensures that the player cannot reroll more than twice a turn and we are set!

Once the current player has his hand of dice, he still needs to put his score in the score table. In order to do that, the program will call a method that calculates what his score would be in every line of the table and shows it to the user. Then, the player can choose one of the lines of the score table and the program will fill this line with the score it calculated previously. Because this part consists in filling a score table, there is not much room for improvement here, as long as it works. However, the fact that the player can see what his score would be before actually filling the score is a very strong point, because it means that the player doesn't have to think about what the best choice would be for him, the program shows it to him directly.

Finally, our program contains also an AI, which allows the user to play against an opponent even when nobody is physically with them. The AI functions in a way such that it will systematically reroll every die that is inferior to the smallest die in its hand. Then, once the AI has rerolled twice, it will calculate what the highest score it could make is, and will then put its score at this location in the score table. This is not particularly the optimal way to get a good score, but it will allow the AI to make a decent score most of the time, and should still be able to offer a good challenge to the player most of the time. We could improve this AI in many ways in the future, for example by trying to make it aim at a straight when it already has three or more dice that follow each other.

In conclusion, this project was bumpy at parts, but allowed us to attain new skills in Java, which were heavily solicitated. Some issues still remain, and many upgrades are necessary for it to be called “finished” but in essence we already have fun playing it, which entitles it to the name of “Game”.