David Giacobbi
HW-3: Lizard Spock Yahtzee Summary

Summary of Program Goals:

The main goal of this second installment of the Yahtzee project was to make the game more customizable by altering the sides, number, and rolls for the dice during each round. To do this, the dice attributes were to be stored in a text file. The program would then read in the text file, parse the strings into integers that would be stored in dice attribute variables. Likewise, methods that were only concerned with a static number of dice, sides, and rolls had to be altered for a more general format. Lastly, the modifications needed to be checked with JUnit tests.

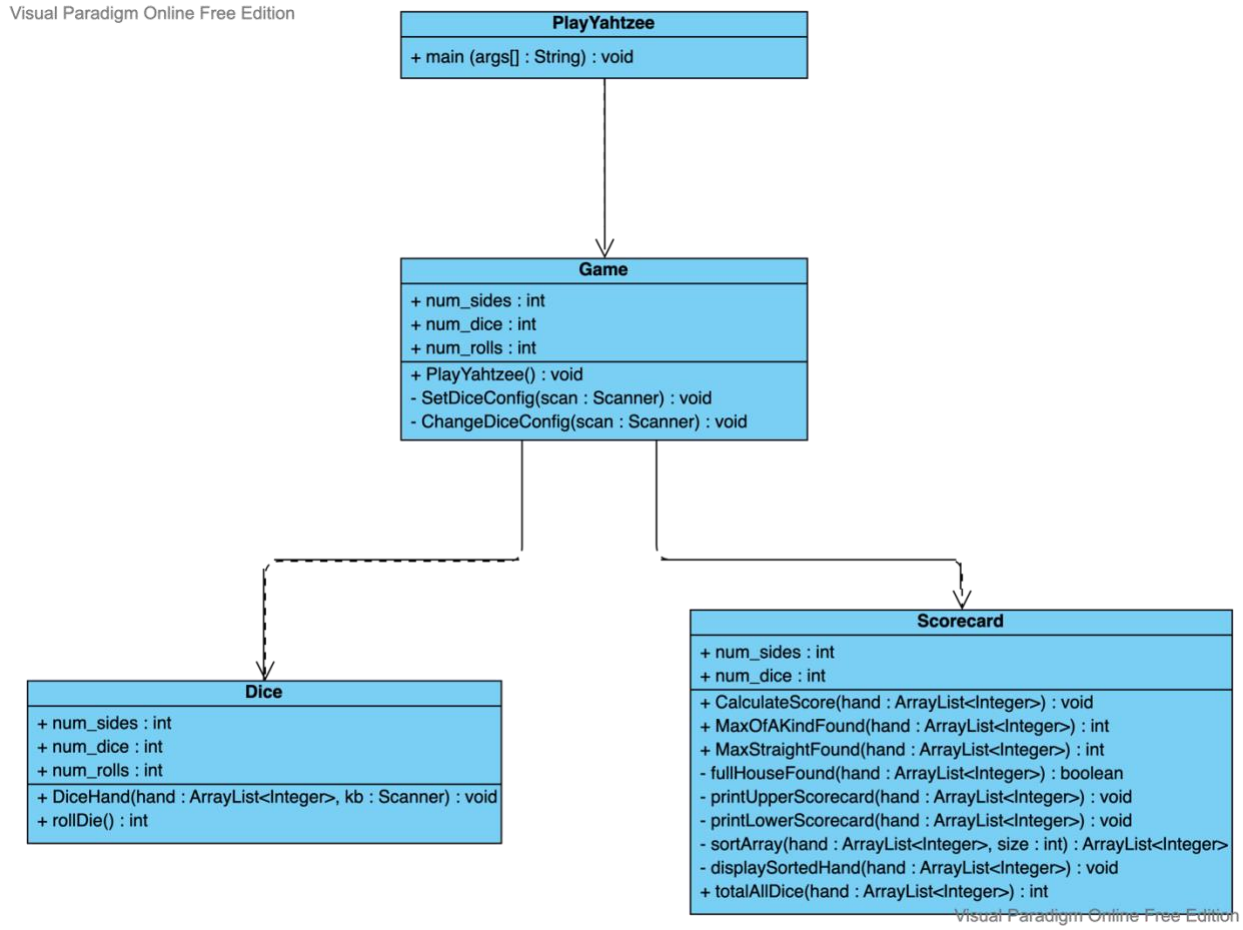
General Design Overview:

The general overview remains to be strictly object-oriented. To improve this in my program, I started to break down my Scorecard object into smaller methods. Given that the Scorecard object had the longest methods as well as the most methods, I figured I would slowly transition it into a more modularized atmosphere. I created constructors for each of my objects that now required dice attributes variables to be built. Lastly, I added methods to the Game object that would read the file and set up the game's dice configuration.

Unit Test Description:

One unit test that I checked for was found in my Dice object. The method had a simple purpose: roll a die, given certain bounds. To run this test, I created a Dice object and specified the number of sides I wanted to check for in the constructor. Provided the sides I called for, I called the rollDie() method to see if the roll would stay within the bounds of 1 to x number of sides on the dice. The assertion was done by checking that the roll was greater than 0 but less than or equal to the number of sides that the dice could have.

UML Diagram:



Design/Programming Issues:

One programming issue that I ran into was that my stopping parameters for the user's rolls was not working when the user already had their best hand and wanted to stop even though they had more rolls. I realized that I was comparing strings in an incorrect manner. Instead of using = = as my comparison between strings, I used the string method .equals(). Another issue that I found, was trying to catch all the instances where a generalized dice attribute was needed rather than just a number. The unit tests were incredibly helpful with this, and I was able to catch one of these errors in my find max straight method while writing the unit test for it.

Retrospective:

Looking back, I think that I would like to continue to tackle the issues of simplifying my objects and finding the right places for my new methods. As this game continues to grow, I think that I

need to keep breaking up the project into more digestible sections. This not only makes project overview easier to understand, but it also makes it much easier to run unit tests on simpler methods rather than ones that have many lines of code. If I had more time, I would split up my Scorecard object into the different ways to score.