**Race Class Design Document**

Ava L. Fritz

Whiting School of Engineering, Johns Hopkins University

EN.605.201.81.SU23: Introduction to Programming Using Java

Dr. Sidney I. Rubey

July 10th, 2023

**Race Class Design Document**

During the planning and execution of the Java class “Race,” I chose to separate the class into four major methods: main, tortoiseMove, hareMove, and displayRace. First, the main method initializes the variables tortoiseCurrent and hareCurrent, representing the starting positions of each animal at position 1. Throughout the logic flow of the Race class, tortoiseCurrent and hareCurrent are updated to reflect the positions of their respective animals after a randomized move takes place.

Next, the main method calls a series of other methods in a while loop. The while loop continues its flow of logic until either tortoiseCurrent or hareCurrent returns a result that is greater than or equal to 51 to represent a total of 50 spaces passed. With each iteration, the methods tortoiseMove and hareMove are called, and their results are added to the values of tortoiseCurrent and hareCurrent respectively. Each method starts with initializing its random roll variable as an integer with value 0, then sets the variable to a random value between 1 and 10. This value represents a probability of occurrence based on the animal’s potential moves. Each potential move is equal to a certain number of spaces to the right or left. Spaces to the right are represented with positive values being added to the current value of the animal’s position, while spaces to the left are returned as negative values.To protect the animals against moving backwards past 1, there are a series of if statements in place to reset the current positions to 1 if a randomized move results in the current position of an animal being less than 1.

The displayRace method takes in two integer variables, tortoiseCurrent and hareCurrent, and returns a visual of the race during that round of moves. Since displayRace is contained within the while loop, each round of moves is displayed on new lines until there is a winner.

At the conclusion of the race, the winner is announced by assessing the current positions of the animals. It is possible for the result of the race to be a tie if both animals reached or crossed position 51 in the same move.

There were no complex data structures used in this program. All variable data were of the primitive data type integer with Strings used as printed output to the user.

During the design process, I considered an alternative solution for implementing tortoiseMove and hareMove using switch statements. Using a switch statement would allow me to select specific integers from the range of 1 through 10 as probabilities to then return the result of the move. However, this was ultimately rejected in favor of if/else statements because it simplified and shortened my methods. If I chose to use switch statements, it would have required a total of 9 statements for numbers 1 through 10, whereas the same logic flow can be represented using less than 5 if/else statements.

While completing this project, I learned the value of taking time at the start of a project to define requirements and plan prior to developing code. By using the strategies described in module 1’s Introduction to Problem Solving1, I was able to define the inputs, outputs, assumptions, and algorithm using pen and paper so that I was significantly more prepared to develop. In the future, I would have created a fifth method to display the results of the race. In my current program, I call this function directly in main, and found it difficult to troubleshoot if an error occurred. With additional methods defining smaller chunks of functionality, I could more efficiently root cause problems and potentially reuse the method in future programs.

**References**

1. Rubey, Dr. Signey I. “Introduction to Problem Solving.”