Angela Chan

Design and Analysis

**General Program Design**

For this program, there are multiple components at play: the characters (Hare and Tortoise), the game play of moving the characters, the random number generator, the display of the race, the conditions of when the game end. I have thought of putting everything into one class with main and creating methods for each component. But I decided to utilize multiple classes because it would look more organized.

The Hare and Tortoise class will contain the calculation of steps that are taken i.e. big hop or fast plod. For converting the randomly generated number to an action of certain number of steps, I decided to use a switch case expression. The switch case will be contained in a loop where it adds to the steps taken by the hare and tortoise. The loop will break when either the hare or the tortoise has reached 50 steps. After breaking out of the loop, a line will be printed out exclaiming who won the race or if it was a tie.

**Alternative approaches/problems**

At the beginning of creating the outline for the program, I was confused on what to exactly do with the time and how it would play out in the program. I thought about using it as a counter, but it wouldn’t make sense since the progress is based on random number generation. The hare and tortoise can go back and forth and run longer than the intended counter. I decided to ignore the timer counter and create the program just counting the steps.

One of the difficulties I had for the program was creating the output of the race. At first, I decided to create the underscore “\_”, representing the track in the race, with the conditions of where the hare and tortoise would be and if they are on the same position. But the problem with that implementation was that when the hare and tortoise was on the same position, ouch was reiterated along with the underscore i.e. (Ouch\_Ouch\_OuchHT..). It also didn’t show the starting position of when the hare and tortoise was on square one. One way to solves this problem was to separate out the conditions. I created a base of how the game should look like at the start with Hare and Tortoise at position one. When hare and tortoise was on the same spot, it would say ouch instead of showing both hare and tortoise. This will be its’ own for loop. Another loop will be created for the general race.

It is good practice to have code be efficient and organized. The code can still be improved by separating the game class code into its’ own separate code and have main method only run the classes. Some calculations could be separated to be its’ method i.e the random number generator but it is only one line of code/calculation.

**Take away from project**

Despite creating an outline and brainstorming before creating the code, there were still a handful of errors and parts of the program that I have not considered. It was minute coding errors but that is what causes problems in a program. During the process, I forgot to put into consideration that if either the hare or tortoise had slipped, more than the steps that they have progressed, it would be calculated as negative steps. I fixed the problem by put an if-else condition. If either of them was to slip more than the steps they had, the hare and tortoise position would be back at one.

Doing this project, I’ve gained a better understanding of how to debug and take parts a code apart to see the issues within an individual code. Even when a code can compile, sometimes it isn’t what we want it to do. To figure out this issue, we can run the code step by step and see where the problem is. For instance, when I was creating the random number generator (Math.random()\*10),I was wondering why the switch would sometimes go to the default statement. I isolated that portion of code and noticed that the random number generator would generate the number 0 which resulted in the default statement. I decided to change it by adding a one in the end which changed the range from 0-10 to 1-10. Through this project, I’ve learned to look carefully at my code and learn to consider the small details of coding.