Documentation for TortoiseHareRace.java: a program simulating a tortoise and hare race

Zixuan Yu

Johns Hopkins University

[605.201] Introduction to Programming Using Java

Mini Project 1

Dr. Sidney I. Rubey.

Oct.07, 2022

Documentation for TortoiseHareRace.java: a program simulating a tortoise and hare race

General program design

This program simulates a tortoise and hare race along a horizontal course that has 60 positions (index position 1-60). It includes one TortoiseHareRace class, a main method, a checkWinner() method, a moveTortoise() method, a moveHare() method and displayRace().

I draw a beginning line and a finish line at index positions 0, and 60 respectively. I use a while loop to let the program loop until either tortoise or hare has reached the finish line. Prior to each loop, the checkWinner() is invoked to check if either of the contenders reached index position 60 and returns a Boolean value. If false, the loop will continue, the round number indicator will increase by one and the program will print a new line; if true, an if-else function in main() method will announce the race result.

The race begins with each contender at position 1. The moveTortoise(), a moveHare() method are used to determine how each contender move: They will generate a random integer from 1 to 10: For the tortoise, perform a fast plod if the number is 1-5, a slow plod if the number is 6-8, and a slip if the number is 9-10. For the hare, perform a big hop if the number is 1-2, a small hop if the number is 3-5, a big slip if the number is 6, a small slip if the number is 7-8, and fall asleep if the number is 9-10.

Then, by invoking the displayRace() method after each round, the animals' position will be displayed on the screen. ‘T’ stands for tortoise and ‘H’ stands for Hare. If the two animals appear in the same position, "OUCH!!!" will be displayed from the tortoise's/hare's position.

Possible Alternative Approaches

It is also possible to realize the loop using a do-while loop. The logit is similar, except that the do-while loop checks the condition at the end of the loop while the while loop checks the condition at the beginning of each loop. However, I think it would be more accurate to check if we already have a winner before we loop again.

A more advanced way of doing this is using multi-threading. There will be a tortoise thread and a hare thread, so their movement will be independent of each other, and they can have different lengths of movement cycles. This is closer to the real-world situation where the two animals have very different behaviors. However, due to the limited time, I did not dig into it.

Also, switch() can be used instead of if-else() when simulating how much each contender will move. If there are many cases (i.e., > 5), switch() would be more efficient.

What I learnt from this project and what I would do differently

I learned how to use Java programming constructs through this project and practiced using multiple methods to realize different functions of an object.

My intuition was to create a graphical user interface (GUI) that could display the two lanes so that the tortoise icon and the hare icon can be displayed after each movement and this program's output will not be as long as it is now. Also, it would be better if we could allow the user to hit a "start" button as well as provide a button that could allow the game to start over again. However, after further research, building an interactive GUI is quite beyond my current scope of knowledge, so I turn to a simpler method to display the results. As my programming skill progresses, I would like to realize this project in GUI.