Tortoise and the Hare Simulation written in Java

Jeffrey Wan

John's Hopkins University

Tortoise and the Hare Simulation written in Java

This program is a tortoise and the hare race simulation. It was designed with three major classes: a TortoiseHareRace class, a Tortoise class, and a Hare class. The TortoiseHareRace class is used to manage the displaying of the race, and the coordinating of moves for the tortoise and the hare. The Tortoise class contains logic of its movements and the Hare class does the same. Neither the Tortoise nor Hare class has any knowledge of how the race works; they both simply generate position values that the TortoiseHareRace class uses to move them along in the race. When the program is run, a visual representation of the course is displayed along with the positions of the tortoise, the hare, and the last previous collision which is represented by an "OUCH!". The race concludes when either the tortoise or the hare lands on position 51 or greater.

Interestingly, even though the race consists of 50 squares, I chose not to use an array of size 50. Instead, I created a string which I then concatenated a "|" to represent an empty square, a "T" to represent the tortoise, a "H" to represent the hare, and an "OUCH!" to represent either a current collision OR a past collision. The TortoiseHareRace class' "buildCourse" method was responsible for taking in the positions of the tortoise and hare and building a string representation of the course using those inputs. As positions changed, I chose to rebuild the string each time anew instead of altering a previously existing string.

I considered building the course representation using arrays instead of strings but I had trouble determining how to change the positions of the "Ouch", "T", and "H" within the array on each turn. Instead, I just rebuilt the string representation of the course for each simultaneous move. I could have rebuild the array representation of the course on each move as well, but strings had the extra benefit of not needing to be converted to string form when displaying the course. Since the displaying of the course was going to have to be a string anyway, I settled on using strings to represent the course.

Some other details of the TortoiseHareRace class:

- The initialPosition is initialized to 1. This value is then used to set the intialPosition of the tortoise and the hare. It is also used later in the program as a floor value and neither the tortoise nor hare's position can fall under this floor, i.e., the tortoise cannot slip backwards. Since this value is important and does not need to be changed during the course of the program, it is set to be a final constant.

- To handle the positions of the tortoise and the hare, there is a check in place to ensure that neither value falls below the initialPosition. The ternary in the while loop contains this logic.

I learned a lot about class organization, and separation of concerns during this project. I made the choice to confine the HareMove and TortoiseMove to the TortoiseHareRace class and keep the Tortoise and Hare classes separate from the details of the race. I think this was a good decision and it allows me to extend those two classes easily with additional integer movements in the future. Additionally, I learned a bit about design patterns as well. I think the Builder pattern is a good one for this case since I needed to “build” a course representation and return the final built version of the course at the end. Lastly, I learned a bit about using the Math class of Java and its random() method and how to coax it to return values within a minimum and maximum. That’s bound to be useful in the future when writing more quantitative programs.