The City College of New York, CUNY
Department of Compute Science
North Academic Center, Room 8/206
160 Convent Avenue, New York, NY 10031

Assignment 2 – Fall 2021

Due Date: *by Sunday October 3, 2021 11:59PM*
How to submit: *upload JAVA files to Blackboard*

Please note:
- ✓ This is an individual assignment; please do your own work. Sharing and/or copying code in part or as a whole with/from others will result in a grade of 0 and disciplinary actions for all involved parties. If you run into problems and have done your best to solve them, please talk to me during office hours or by e-mail.
- ✓ There is a 25% grade deduction for every day the assignment is late unless prior permission is granted.

The Tortoise and the Hare, a race simulation.

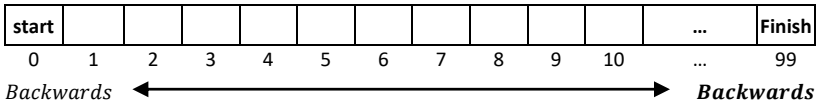
In this assignment, you'll recreate a version of the classic race of the tortoise and the hare. You'll use a random number generator to simulate the race. The contenders begin race along a track of 100 squares. Each square represents a possible position along the race course. The finish line is at square 100. The first contender to reach or pass square 100 wins. The course weaves its way up the side of a slippery mountain, so occasionally the contenders lose ground. A clock ticks once per second. With each tick of the clock, your application should adjust the position of the animals according to the rules in Figure 1. Use variables to keep track of the positions of the animals (i.e., position numbers are 1–100). Start each animal at position 1 (the "starting gate"). If an animal slips before square 1, move it back to square 1.

Racer	Move Type	% of time	Direction	Squares to move and direction
Tortoise	Sleep	10%	–	No movement
	Jump	50%	Forward	Random # of Squares between 1 and 4
	Slip	20%	Backwards	Random # of Squares between 1 and 6
	Walk	20%	Forward	Random # of Squares between 0 and 1
Hare	Sleep	20%	–	No movement
	Jump	20%	Forward	Random # of Squares between 1 and 9
	Small slip	20%	Backwards	Random # of Squares between 1 and 2
	Big Slip	10%	Backwards	Random # of Squares between 1 and 10
	Walk	30%	Forward	Random # of Squares between 0 and 1

Figure 1: Rules for adjusting the positions of the Tortoise and the Hare.

How to apply the percentages ("% of time" in Figure 1)?

- ✓ Generate the percentages in Figure 1 by producing a random integer i with the range $0 \leq i \leq 9$. For the Tortoise, perform a "$Jump$" when $0 \leq i \leq 3$, a "$slip$" when $4 \leq i \leq 6$, a "$walk$" when $7 \leq i \leq 8$, or a sleep when $i = 9$.
- ✓ Since you're using random numbers, the outputs will be different between runs.
- ✓ To avoid long-running simulations, stop the simulation if it goes over 1,000 iterations.
- ✓ A unit of time in your program (i.e., second) is one full iteration in the program. A counter shows how many iterations were made until one contestant wins. If the simulation takes 270 iterations, then this is treated as 270 seconds.
- ✓ Think of the 100-square track as shown below. A forward move means, move up the track, a backward move means move down the track. Do not use arrays; instead, use counters that track the runners' movements



The City College of New York, CUNY
Department of Compute Science
North Academic Center, Room 8/206
160 Convent Avenue, New York, NY 10031

Requirements:

1. Comment your code. Use JavaDoc and non-JavaDoc style where appropriate.
2. Program should compile and run to be graded. Check the grading table below for a breakdown of the grades.
3. One Java source file for class $TortoiseAndHare$ with the following minimum members (UML diagram in Figure 8). You may add any additional members; however, your class must at least contain the following:
   a. At least two uses of loop structures (i.e., while, do-while, for)
   b. At least one use of a SWITCH control structure
   c. Two read-only variable, MAX_MOVES and $MAX\_ITERATIONS$, with default values 100 and 1,000 respectively. MAX_MOVES indicates the finish square; $MAX\_ITERATIONS$ acts as an upper limit to the simulation loop; the loop may terminate earlier, but no bigger!
   d. $main$, starts the race by creating an instance of class $TortoiseAndHare$. Note that none of the other methods are static (Figure 8).
   e. Default constructor performs the following major steps:
      (1) Prints the start message in Figure 2
      (2) Print the racer's position using the method $printPositions$
      (3) Move both players using the methods $simulateTortoiseMove()$ and $simulateHareMove()$
      (4) Repeat step (3) until either one wins, there is a tie, or the race times out (maximum of $MAX\_ITERATIONS$).
      (5) Print the results of the simulation. The result should state if:
         ✓ The Tortoise wins (Figure 4)
         ✓ The Hare wins (Figure 5)
         ✓ There is a tie (Figure 6)
         ✓ There is a timeout (Figure 7)
   f. $randomBetween$, returns an integer random number between two limits (inclusive)
   g. $printPositions$, prints the race track and shows the position of the Tortoise and the Hare. Use the Letter 'T' to represent the Tortoise, 'H' for the Hare, and 'B' if both are on the same square. See Figure 3
   h. $simulateTortoiseMove$, a method that simulates the movements of the Tortoise according to Figure 1.
   i. $simulateHareMove$, a function which simulates the movements of the Hare according to Figure 1.

Grading:

Item	Points
Comments (JavaDoc and major steps)	10
2 Static fields	10
Two loop structures	10
Switch structure	5
Race simulation until one wins, tie, or timeout	10
Boundary checks	5
Constructor	5
randomBetween	10
printPositions	10
simulateTortoiseMove	10
simulateHareMove	10
Correct output	5
	100

Figures:

```
ON YOUR MARK, GET SET
BANG !!!!!
AND THEY'RE OFF !!!!!
```

Figure 2: initial message printed when the race starts

```
Iteration: 0
B
------------------------------------------------------------------------------------------------------------------------
Iteration: 1
HT
------------------------------------------------------------------------------------------------------------------------
Iteration: 2
HT
------------------------------------------------------------------------------------------------------------------------
Iteration: 3
TH
------------------------------------------------------------------------------------------------------------------------
Iteration: 4
 T   H
------------------------------------------------------------------------------------------------------------------------
Iteration: 5
   TH
------------------------------------------------------------------------------------------------------------------------
Iteration: 6
   HT
------------------------------------------------------------------------------------------------------------------------
Iteration: 7
     T H
------------------------------------------------------------------------------------------------------------------------
```

Figure 3: With every iteration, the racer's positions are printed. Positions differ with every iteration and each simulation

```
TORTOISE WINS!!! YAY!!!
Time Elapsed = 766 seconds
```

Figure 4: Message if the Tortoise wins. Time differs for each simulation.

```
Hare wins. Yuch!
Time Elapsed = 274 seconds
```

Figure 5: Message if the Hare wins. Time differs for each simulation

```
It's a tie
Time Elapsed = 1000 seconds
```

Figure 6: Tie message

```
Time Out!
Hare wins. Yuch!
Time Elapsed = 1000 seconds
```

Figure 7: Timeout message with winner

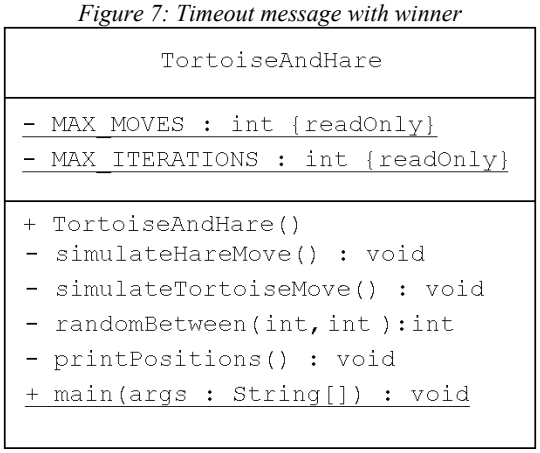


Figure 8: Class UML Diagram