Assignment 4
Turtle and Hare Race - Who Will Finish First?

Due: March 4, 2014 at 11:55PM.

You may discuss any of the assignments with your classmates (or anyone else) but all work for all assignments must be entirely your own. Any sharing or copying of assignments will be considered cheating.

In this assignment you will recreate one of the truly great moments in history, namely the classic race of the turtle and the hare. You will use your newly acquired expertise in random number generation and method development to simulate this memorable event.

Our contenders begin the race at position 1. Their destination, i.e. the finish line, is at position 70. The first contender to reach or pass position 70 is rewarded with a pail of fresh carrots and lettuce. The course weaves its way up the side of a slippery mountain, so occasionally the contenders lose ground and slip. They try to make up for those occasional slips by fast plods (the turtle) and big hops (the hare). Our hare is so sure of his advantage over the turtle that he takes short naps every now and then.

In each step of the race the position of the animals should be adjusted according to the following rules:

<table>
<thead>
<tr>
<th>Animal</th>
<th>Move type</th>
<th>Percentage of the time</th>
<th>Actual move</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turtle</td>
<td>Fast plod</td>
<td>50%</td>
<td>3 squares forward</td>
</tr>
<tr>
<td></td>
<td>Slip</td>
<td>20%</td>
<td>6 squares backward</td>
</tr>
<tr>
<td></td>
<td>Slow plod</td>
<td>30%</td>
<td>1 square forward</td>
</tr>
<tr>
<td>Hare</td>
<td>Sleep</td>
<td>20%</td>
<td>No move</td>
</tr>
<tr>
<td></td>
<td>Big hop</td>
<td>20%</td>
<td>9 squares forward</td>
</tr>
<tr>
<td></td>
<td>Big slip</td>
<td>10%</td>
<td>12 squares backward</td>
</tr>
<tr>
<td></td>
<td>Small hop</td>
<td>30%</td>
<td>1 square forward</td>
</tr>
<tr>
<td></td>
<td>Small slip</td>
<td>20%</td>
<td>2 squares backward</td>
</tr>
</tbody>
</table>

Use variables to keep track of the positions of the animals (i.e., position numbers are 1—70). Start each animal at position 1 (i.e., the "starting gate"). If an animal slips back before square 1, move the animal back to square 1. If the animal advances past the finish line, set the position to square 70.

For each step of the race (i.e., each repetition of a loop), print a line showing the letter T in the position of the turtle and a line showing the letter H in the position of the hare. All positions other than the T or the H should be blank. (This is done by printCurrentPositions() method that is given below, so all you need to do is make the correct call to the method.)

After each line is printed, test if either animal has reached or passed square 70. If so, print the winner and terminate the simulation. Otherwise, the race continues.

Your program should implement and use the following methods:

```java
public static int moveTurtle( int pos, int finishLine )
public static int moveHare( int pos, int finishLine )
public static void printCurrentPositions( int turtlePos, int harePos )
```

The first two functions take in the current position of the turtle/hare and, based on randomly generated number, compute the new position following the rules in the table above. The third function takes positions of both animals and produces the “image” on the screen reflecting the current state of the race (this method is provided at the end of this assignment document). The main method needs to repeatedly call these three methods until one of the animals reaches the finish line (at position 70).

Grading

Does the program compile? If not, you will loose all the points for that problem.

Is the program properly documented? (worth 30%)

Proper documentation includes:
• preamble with the name of the author, date of creation and brief description of the program;
• appropriately chosen variable names, i.e., descriptive names;
• comments inside the code describing steps need to be taken to accomplish the goal of the program;
• appropriate formatting, indentation and use of white space to make the code readable.

Remember that the code is read by humans and it should be easy to read for people who were not involved in its development.

Is the program well developed? (worth 40% ) Make sure you create variables of appropriate types, use control statements (conditionals and loops) that are appropriate for the task, accomplish your task in a well designed and simple way (not a convoluted algorithm that happens to produce the correct output for some unknown reason). You should also design a friendly and informative user interface.

Is the program correct? (worth 30%), Make sure that your program produces valid results that follow the specification of the problem every time it is run. If the program is not completely correct, you get credit proportional to how well it is developed and how close you got it to the completely correct code.

What and how to submit?

You should submit a single source code file compressed into a single zip file to NYU Classes (yes, your zip file will contain only one source code file in it). Do not submit all the files that Eclipse creates, just the source code files that have .java extensions. Name your classe Race - that means your file will be called Race.java.
/** *
 * display current positions of the race contenders
 * @paramturtlePos
 * current position of the turtle
 * @paramharePos
 * current position of the hare
 */
publicstaticvoidprintCurrentPositions(intturtlePos,inharePos)
{
    // clear the screen
    for(inti=0;i<50;i++)
        System.out.print("\n");

    // print status of the race
    if(harePos==turtlePos)
        System.out.printf("Tie!!!")
    elseif(harePos<turtlePos)
        System.out.printf("Turtle ahead!\n")
    else
        System.out.printf("Hare ahead!\n");

    // print positions
    System.out.printf("\n\n")
    // print turtlePos number of blanks before printing T
    inti;
    for(i=0;i<turtlePos;i++)
        System.out.print(" ");
    System.out.print("T");
    // print remaining blanks before printing the finish line marker
    for(;i<69;i++)
        System.out.print(" ");
    System.out.print("\n");
    // print harePos number of blanks before printing H
    for(i=0;i<harePos;i++)
        System.out.print(" ");
    System.out.print("H");
    // print remaining blanks before printing the finish line marker
    for(;i<69;i++)
        System.out.print(" ");
    System.out.print("\n");

    // pause the program for a fraction of a second
    // to create animation effect
    try{
        Thread.sleep(100);
    }
    catch(InterruptedExceptione){}
}