605.201 Mini-Project 1:

Please do the following to complete this assignment.

Purpose:

The purpose of this project is to provide non-trivial practice in the use of Java programming constructs discussed from the beginning of the course through Module 05 and have a bit of fun doing it.

Resources Needed:

You will need a computer system with Java 7 or greater SE edition run-time and JDK. You may optionally use a Java IDE for example NetBeans, Eclipse, etc. However application builders are not allowed.

Submitted Files:

Design and Analysis:

This word-processed document is required to be written in APA style minus the Abstract section. The file formats accepted will be announced at project assignment. The length of the document should be between 1.5 and 2 pages. The following subjects should be discussed in this order:

1. General program design. How is the program organized? What major data structures were used?
2. What alternative approaches were considered and why were they rejected?
3. What did you learn from doing this project and what would you do differently?

Source file:

All Java source will be in a single text file which can be compiled and executed in a standard Java 8 or later SE environment. Multiple methods can be used but all user-created methods must be contained in the single Java text file.

The format of the Java source must meet the general Java coding style guidelines discussed so far during the course. Please use course office hours or contact the instructor directly if there are any coding style questions.

Submit file:

The submit file is to be a Zip file containing your design and analysis document, your single Java source text file, and a screen capture of the program executing. Any appropriate file name for this Zip file is acceptable.

Collaboration:

It is encouraged to discuss technical or small design parts of this project with your fellow students. However the resulting design and implementation must be your own. When in doubt, ask during office hours or contact your instructor.

Program Specification:

This project involves writing a program to simulate a tortoise and hare race. The contenders will each race along a horizontal course that contains at least 50 positions. You may add more if you wish. The race begins with each contender at position 1. The contender that first reaches or passes the last position of the course is the winner of the race.

The following table indicates the types of moves that each contender can make.

Contender	Type of Move	Percentage of Time	Result of Move
Tortoise	Fast plod	50%	3 squares to right
	Slow plod	30%	1 square to right
	Slip	20%	6 squares to left
Hare	Big hop	20%	9 squares to right
	Small hop	30%	1 square to right
	Big slip	10%	12 squares to left
	Small slip	20%	2 squares to left
	Fall asleep	20%	

Each contender starts at position 1. When a contender slips, they can't slip any further left than position 1. You will use a random number generator to simulate the percentages of each type of move indicated in the table. To generate random numbers, you can research the built-in Java random number method that is part of the Math class.

Generate a random integer, n, in the range $1 \leq n \leq 10$. 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.

There are a number of ways to design this program. One way would be to have a looping construct be the overall controller of things. Each iteration would adjust the contender positions, and the loop would terminate when one of the contenders reaches the last square of the race course. You will decide on an approach as part of your design step.

You must keep track of each contender's position and display it each time positions change. Show the letter "T" in the position of the tortoise, and the letter "H" in the position of the Hare. It is possible for the contenders to land on the same square. When this happens, the tortoise bites the hare, and your program should display "OUCH!!" beginning at that square. All output positions other than the "T", the "H", and the "OUCH!!" should be blank.

If the tortoise wins, display "TORTOISE WINS!!". If the hare wins, display "HARE WINS!!". If the race is a tie, display "IT'S A TIE!!". At the beginning of the race, display "AND THEY'RE OFF!!".

Assessment:

	60%	70%	80%	90%	100%	Weight
Design & Analysis Document	Majority document subjects covered with accurate information. Some fluff. Document is over half the specified length with 2-4 minor APA style or other minor issues.	All but one document subject covered with accurate information. Maybe some fluff. Document is close to the specified length with 2-4 minor APA style or other minor issues.	All document subjects covered with accurate information. Maybe some fluff. Document is close to the specified length with 1-3 minor APA style or other minor issues.	All document subjects covered with insightful and accurate information. Document is of specified length with 1-3 minor APA style or other minor issues.	All document subjects covered with insightful and accurate information. Document is of specified length and properly formatted to APA style.	**20%**
Program Correctness	All but two major specified features work but a noble effort is made.	All but one major specified features work but a noble effort is made. For example the game may end prematurely or run longer than necessary.	All but two minor specified features work. For example not all the output messages are displayed or the display of the race course is a little off.	All but one minor specified features work. For example not all the output messages are displayed or the display of the race course is a little off.	All specified features work with neat and easy to understand information display.	**50%**
Code Style	Most functionality not properly segmented into methods. Some variable use proper names and types. Some commenting. Some use proper indentation line	Most functionality properly segmented into methods. Most variable use proper names and types. Some commenting. Some use proper	Most functionality properly segmented into methods. Most variable use proper names and types. Appropriate commenting. Mostly use	Functionality properly segmented into methods. Most variable use proper names and types. Appropriate commenting. Mostly use proper	Functionality properly segmented into methods. All variable use proper names and types. Appropriate commenting. Proper indentation line	**30%**

		indentation line	proper indentation line	indentation line	continuation, etc.	

If you have any questions about the specification of this project, contact your instructor *before* the project is due.