**Computer Science 1125 Programming Assignment One 50 Points**

You are to do programming problem 7.28 on page 320.

What to hand in:

1. All your .java files.
2. Copy and paste your source code into a word document. (This allows me to make comments on your code.)
3. Paste screen shots of:
   1. Successful compilation of your program.
   2. The output of your program. Should be several runs of your program.

Due Date: Friday March 8 before Midnight.

tortoiseAndHare.java

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package** tortoiseAndHare;

**public** **class** tortoiseAndHare {

**public** **static** **void** main(String[] args) **throws** InterruptedException {

Tortoise turtle = **new** Tortoise(); // class data type (obj of tortoise) for variable turtle

Hare rabbit = **new** Hare(); // class data type (obj of hare) for variable hare

**int** time = 0; // time that is needed for this project time++

**int** finish = 70; // finish line

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*(comment for me)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// calling the turtle and rabbit as a variable to be assigned from tortoise and

// hare object

// to call what we need to compare to finish variable or any variable we want to

// getter we

// would first call the variable assigned to object then the method that would

// make sense in

// comparing the getter.

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Runs a while loop that will 1). get the position of the hare from the class

// and methods

// 2). get the position of the tortoise from the class and methods

// 3). Get the information of position to output the display of HT and the lines

// 4). Thread.sleep is something that would help slow down the output to see

// the action between H and T.

// 5). while race is continuing running the time increments in seconds

// This is all managed as long a the race is not finished

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**while** (turtle.getPosition() != finish && rabbit.getPosition() != finish) {

turtle.movePosition(); // movement of turtle

rabbit.movePosition(); // movement of hare

*displayRace*(turtle, rabbit);

Thread.*sleep*(1000);

time++;

}

**if** (turtle.getPosition() > rabbit.getPosition()) // if T beats H output this

System.***out***.println("TORTOISE WINS, YAY!!!\n");

**else** **if** (turtle.getPosition() < rabbit.getPosition())// if H beats T output this

System.***out***.println("HARE WINS, YUCH!!!\n");

**else**

System.***out***.println("IT'S A TIE!!!\n");// none above then output tie

System.***out***.println("Time of race: " + time + " seconds\n"); // output time

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Name: displayRace();

//

// Pre: grabbing the class object of tortoise and hare's position in their

// classes

// to help us output our display if the position is the same output ouch. And

// output

// H for the hare position and T for the tortoise position.

// The lines help us see the race course and bounds of the race

//

// Post: displaying ouch for same position

// H for hare, and T for tortoise.

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// start method

**public** **static** **void** displayRace(Tortoise turtle, Hare hare) {

**for** (**int** i = 1; i <= 70; i++) {

**if** (i == turtle.getPosition() && i == hare.getPosition())

System.***out***.print("OUCH!!!");

**else** **if** (i == hare.getPosition())

System.***out***.print("H");

**else** **if** (i == turtle.getPosition())

System.***out***.print("T");

**else**

System.***out***.print('-');

}

System.***out***.println();

}// end display method

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Hare.java

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package** tortoiseAndHare;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//This is the object for the hare. Since in the program we focus a lot of the movement and position

// of the hare and turtle. Hence the reason why we have a hare class.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **class** Hare {

// Making the position secret for this class only to be use for this class

**private** **int** position; // car accessories/color

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Name: public Hare();

//

// Pre: This is constructor to grab the variable position to set it to 1

// since we are having the hare starting at the starting line which is 1

//

// Post: Setting position to 1 to be called and return from methods and

// dot operator.

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** Hare() { // constructor //car specific accessories/color

position = 1;

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Name: public int getPosition()

//

// Pre: This method is for grabbing our secret variable of position which is

// set to 1 by our constructor.

//

// Post: This method will return the position from another method that is being

// changed of the position of the hare

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **int** getPosition() { // method //car's functionality, car accessory function

**return** position; // returning your position

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Name: public void movePosition();

//

// Pre: This method determines the moves of the hare which adjusts the position

// of the it through the purpose of the random number generator from 1-10.

// With the output of the rand number gen we will have move determine the

// position of the hare.

//

// Post: Position is being adjusts form the move which is being return from the

// help of the method getPosition.

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **void** movePosition() { // method //car's functionality, car accessory function

**int** move;

move = (**int**) (1 + Math.*random*() \* 10); // generation move from 1-10

/\*

\* if (move==3||move==4) position+=9; //9 sq. to the right else if (move==5)

\* position-=12; //12 sq. to the left else if (move>=6 && move <=8) ++position;

\* //1 sq. to the right else if (move >8) position-=2; //2 sq. to the left

\*/

**switch** (move) {

**case** 3:

**case** 4:

position += 9; // 9 sq. to the right

**break**;

**case** 5:

position -= 12; // 12 sq. to the left

**break**;

**case** 6:

**case** 7:

**case** 8:

++position; // 1 sq. to the right

**break**;

**case** 9:

**case** 10:

position -= 2; // 2 sq. to the left

**break**;

}

**if** (position > 70) // bound check of the obj doesnt get out of the race

position = 70;

**if** (position < 1) // bound check of the obj doesnt get out of the race

position = 1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Tortoise.java

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package** tortoiseAndHare;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//This is the object for the tortoise. Since in the program we focus a lot of the movement and position

//of the hare and turtle. Hence the reason why we have a tortoise class.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **class** Tortoise {

// Making the position secret for this class only to be use for this class

**private** **int** position;

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Name: public Hare();

//

// Pre: This is constructor to grab the variable position to set it to 1

// since we are having the hare starting at the starting line which is 1

//

// Post: Setting position to 1 to be called and return from methods and

// dot operator.

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** Tortoise() { // constructor building your object (tortoise)

position = 1;

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Name: public int getPosition()

//

// Pre: This method is for grabbing our secret variable of position which is

// set to 1 by our constructor.

//

// Post: This method will return the position from another method that is being

// changed of the position of the hare

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **int** getPosition() { // method, returning a position

**return** position;

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Name: public void movePosition();

//

// Pre: This method determines the moves of the hare which adjusts the position

// of the it through the purpose of the random number generator from 1-10.

// With the output of the rand number gen we will have move determine the

// position of the hare.

//

// Post: Position is being adjusts form the move which is being return from the

// help of the method getPosition.

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **void** movePosition() {

**int** move;

move = (**int**) (1 + Math.*random*() \* 10); // generation move from 1-10

/\*

\* if (move>=1 && move <=5) position+=3; //3 sq. to the right else if (move==6

\* || move==7) position-=6; //6 sq. to the right else if (move>=8 && move<=10)

\* position++; //1 sq. to the right

\*/

**switch** (move) {

**case** 1:

**case** 2:

**case** 3:

**case** 4:

**case** 5:

position += 3; // 3 sq. to the right

**break**;

**case** 6:

**case** 7:

position -= 6; // 6 sq. to the right

**break**;

**case** 8:

**case** 9:

**case** 10:

++position; // 1 sq. to the right

**break**;

}

**if** (position > 70) // bound check of the obj doesnt get out of the race

position = 70;

**if** (position < 1) // bound check of the obj doesnt get out of the race

position = 1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

