Lab 2: Classes

CSC/EGR1054

100 points total

Part I (20 points)

Creating a TemperatureGauge class.

TemperatureGauge should have the following in it:

Members:

* double temperature //in Fahrenheit - temperature cannot be below -30 or above 180; if a value about or below these is passed in, the temp is set to -30 or 180 respectively. Your methods and constructors must take this into account. There should only be one copy of the code that prevents the temperature from going out of range.

Methods:

* constructor() //At the conclusion of the constructor, temperature should be 0.
* constructor(double) //At the conclusion of the constructor that takes in a parameter, temperature should be whatever was passed in; however, it must take into account the temperature restriction.
* getTempInF() //getTempInF returns the temperature in F. The return type should be double.
* getTempInC() //getTempInC returns the temperature in C. Your method must do some work in order to get the C temperature from the F temperature. The formula is C = (F - 32)\*5/9. The return type should be double.
* setTempInF(double) //setTempInF sets the temperature to be the new value. You must take into account the above restriction of possible temperature values.

In your client:

* Create two objects of TemperatureGauge, (1) with the constructor with no arguments and (2) with 120 are the input parameter.
* Print out (1)’s temperature in F
* Print out (1)’s temperature in C
* Print out (2)’s temperature in F
* Print out (2) ’s temperature in C
* Set (1)’s temperature to -80
* Print out (1)’s temperature in F
* Set (1)’s temperature to 180
* Print out (1)’s temperature in F
* Set (1)’s temperature to 59.5
* Print out (1)’s temperature in C
* Set (2) = (1). i.e., gauge2 = gauge1;
* Set (2)’s temperature to 33
* Print out (1)’s temperature in C

Run of Code:

0.0

-17.77777777777778

120.0

48.888888888888886

-30.0

180.0

15.277777777777779

0.5555555555555556

(20 points)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Part I signed off.

Part II

You will be creating a tic tac toe game class to work with the client I am providing. This tic tac toe class only has to work with the first 4 moves the players make (i.e., there no winners after four moves). You have to ensure no person plays in a spot, which currently has a player’s piece at it. You should examine the client code to determine the names of the methods.

Game class:

Members:

* Represent the board as a 2D array (or 2D ArrayList; the array is probably easier).
* A variable to represent whose turn it is.
* A variable to keep track of how many moves have been played.

Methods:

* A constructor to setup the tic tac toe game & board (it should take no parameters).
* A method that returns whether 4 moves have been played. (It takes no parameters and returns a Boolean).
* A method that returns whose turn it is.
* A method that returns a string representing the game board. (see the run for example output)
* A method to make a move at a spot. This method takes in the x and y of the move and places/tries to place the move on the board. It returns a Boolean (whether or not a valid move was made). The tic tac toe game class itself keeps track of who moved last. You must include “out of bounds” checking. If an index is out of bounds, the method should return false. If spot was already played in on the board, it should also return false;

Game client code:

Scanner scan = new Scanner(System.in);  
 TicTacToe tic = new TicTacToe();  
   
 while(!tic.hasFourMovesHaveBeenplayed())  
 {  
   
 System.out.println("Player "+tic.getCurrentPlayer()+ " please make your move:");  
   
 int x = scan.nextInt();  
 int y = scan.nextInt();  
   
   
 if(tic.play(x,y))  
 {  
 //if the move was successful  
 System.out.println("Current board is:");  
 System.out.println(tic.getBoard());  
 }  
 else  
 {  
 //if the move was invalid  
   
 System.out.println("Player "+tic.getCurrentPlayer()+ " please make a valid move.");  
 }  
   
 }  
 System.out.println("Four moves have been played! Game over!");

Example:

Player 1 please make your move:

-> 0 0

Current board is:

1 0 0

0 0 0

0 0 0

Player 2 please make your move:

-> 1 1

Current board is:

1 0 0

0 2 0

0 0 0

Player 1 please make your move:

-> 0 1

Current board is:

1 0 0

1 2 0

0 0 0

Player 2 please make your move:

-> 1 0

Current board is:

1 2 0

1 2 0

0 0 0

Four moves have been played! Game over!

(60 points) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Part II signed off.

Questions:

1. (10 points) Consider part I of this lab, what would the problem have been with having temperature public (instead of private) and setting the value of temperature in main?

2. (5 points) We don’t want to reinvent the wheel (have the same code twice in our program). What does that mean for Part I, given that the constructor that takes in a temperature and the mutator, setTempInF, also takes in a temperature?

3. (5 points) Why does, after Set (2) = (1).i.e., gauge2 = gauge1;, the print out (1) change when you changed (2) and not (1)?