Classes part II: toString, equals, and static

Lab 3

CSC1054

100 points total

Part I (30 points)

Create a Person class:

* Person should have:
  + string name.
  + int age.
  + int ID.
  + static int count.
* The constructor should take in an age, name, and id. Initialize the members as needed.
* Write an equals method to determine if two people are equal.
* Write a toString method to output the person (format is shown below).

Age: 19

Name: Lauren

ID: 1001

* A static method should return the number of people, which have been created.

In your client:

* Create three people: Lauren, age 19, id 1001; Matthew, age 18, id 1002; Matthew age 18, id 1002.
* Create a fourth person variable and set it to reference the third person.   
  (i.e., Person p4 = p3;)
* Print out the number of people by using your static method.
* Print out person 1 (do not explicitly call the toString() in this call or any of the calls to print a person).
* Print out person 2.
* Print out person 3.
* Check if person2.equals(person3) ; print if true. (see the output below for the output format)
* Check if person1.equals(person2) ; print if true.
* Check if person 1 == person 2; print if true.
* Check if person 2 == person 3; print if true.
* Check if person 3 == person 4; print if true.

Run of Code:

Number of people: 3

Age: 19

Name: Lauren

ID: 1001

Age: 18

Name: Matthew

ID: 1002

Age: 18

Name: Matthew

ID: 1002

p2 and p3 are .equals

p3 and p4 are ==

Part II (50 points)

You are to create a train simulation program that has two trains. The trains are the “X” train and the “Y” train. Each train can move left or right. After “X” moves then “Y” moves and after “Y” moves then “X” moves and so on (think of it as swapping between the two players in TicTacToe). If the trains collide (are occupying the same position) then you are to print out a message that they collided and conclude the program.

Example:

===============

---X----Y------

===============

Move the X train left (enter L) or right (enter R):

->R

===============

----X---Y------

===============

Move the Y train left (enter L) or right (enter R):

->L

===============

----X--Y-------

===============

Move the X train left (enter L) or right (enter R):

->R

===============

-----X-Y-------

===============

Move the Y train left (enter L) or right (enter R):

->L

===============

-----XY--------

===============

Move the X train left (enter L) or right (enter R):

->R

CRASH!!!

Moves before crash: 5

Instructions:

Create a Train class

* The Train class should have a variable for the position (int) of the train.
  + You will want to create an accessor and mutator for the Train class’s position.
  + The position can only be between 0 and 14. If it set to a value less than 0, then set it to 14. If it is set to a value greater than 14, then set it to 0. (i.e. the train is on a circular track, if it goes off the left side then it appears on the right side).
* The Train class should have a variable to count the number of times a train’s position has changed. This counter should keep track of the movements across all instances of Train. You will need to add the requisite code to count the times the position changes.
* You will need a constructor for the Train, which should initialize the position.
* You may add other methods if you would like to. Think about what you need!

Create the Train System class [this holds the trains]

* The TrainSystem has two Trains (the ‘X’ train and the ‘Y’ train) and creates them in the constructor; the trains should start at positions 3 and 8.
* The TrainSystem has a method to move a train. It should take in two parameters: one to indicate which train should move and one to indicate the direction of the move, either left (-1 to the position) or right (+1 to the position).
* The TrainSystem has a toString method, which prints out the TrainSystem in the format shown below (and above). Each of the 15 positions is either a -, the X, or the Y. The X and Y represent the current positions of the trains and the -s represent empty spaces.

===============

-----XY--------

===============

* The TrainSystem has a method to determine if the two trains are colliding (i.e., at the same position).

Create the client

* The client should create a train system.
* In a loop, ask the user to move each of the trains left or right as in the example run above. Remember to print the system before each move. If there was a crash at any point in the loop, print crash and inform the user how many times a train was moved and then conclude the program.
* Make the program error proof for all types of incorrect input and loop until valid input is entered.

Questions:

1. (5 points) Why not use == to compare objects?

2. (5 points) How do you make .equals work to compare objects?

3. (5 points) why does it not make sense for a static method to be able to use an instance class variable?

4. (5 points) how is a static variable different from a non-static variable?