Introduction to Computer Science II
Assignment 2

Due: 4:00 p.m. June 6, 2016

Programming with Classes and Objects in Java

You are to create a Car class that can represent any car in a typical transportation system.

Assumptions:

1. A typical car can be described using the car unique serial number (ID), max speed, color (e.g., "red", "black", etc.), manufacturing year, make/model (e.g., Ford, Toyota, etc.).
2. When a car object is printed, it shows at least the car make/model, its year, color, and whether the care is moving or not.
3. Two cars are similar if they have the same color, year, and make.
4. A car can be duplicated by coping all its state info except the unique serial ID.

The following (instance) methods must be supported:

1. Three constructors: one that allows you to specify the car make/mode, color, and year, and one that initializes these parameters to some default values, and the third one is a copy constructor that duplicates a car object.
   (It is recommended to initialize the max speed of all cars to 200 km/h).
2. Three *set* methods: one to set the car's manufacturing year, one to set color, and one to set the make/model of the car.
   (Setter methods should handle invalid input such as entering an invalid year).
3. Four *get* methods: one returns the manufacturing year, one returns the *current* speed of the car, one returns the make/model of the car, and one that returns its color.
4. One *move* method: updates the (current) speed of the car according to this method's *speed* parameter.
   (note that the care speed cannot exceed the max speed).
5. One *stop* method: updates the (current) speed of the car so that it is at complete stop.
6. One calculateJourney*Time* method: the time needed to travel by this car a given distance based on its current speed is computed and returned.
   (Hint: if the speed of the car is not moving then return -1 as the time needed).
7. One *carID* method: the ID number of the car is returned. As each car is created, a unique integer ID number is assigned to it. The first ID number is 1, and as new cars are created, successive numbers will be assigned.

The following class method must also be supported:

1. A method that returns the number of "active" instances. For example, if three car objects have been created, and none have been garbage collected, this method would return 3. Note that you will have to decrement the number of active instances in the finalize() method. Also note that you will need to use a separate mechanism from that which keeps track of car IDs.

Create some client code in the main() method of a class called TestCarSystem to test all of the above methods:

1. Prompt the user for the information needed (e.g., *year, color, and make/model*) to initialize a car object using the appropriate constructor. Repeat this step for another two car objects.
2. Instantiate a fourth car object and prompt the user for the car info then update the car object by the given values using the proper *setters*.
3. Prompt the user for three speed values, and ask the first three cars to start moving using the appropriate method(s).

4. Print out the cars you have, and also the number of active instances. A toString() method would be useful here.
5. Ask the last created car to move using some random speed (e.g., 80km/h) and ask the first created car to stop moving using the appropriate method(s).
6. Calculate the travel time needed to make a journey of some user-given distance for the four cars created above using the instance method, and print this info.
   (Only moving cars can be part of this calculation. If a car is not moving, then print the time needed to make the journey as N/A)
7. Identify any similar cars and print out their information in addition to their unique serial numbers.
8. Delete (reference of) the second car object, and force garbage collection. Then print out the number of active instances.
9. Instantiate another car object by copying from the first car and move it with a speed equals to the averages of the *speed* of all live cars.
10. Print the remaining cars information, and also the number of active instances.

You must use arrays to store the car objects you will create. The array may be bigger in size than the number of the actually created car objects.

Make sure your printout to the screen is clear and fully describes what is happening at each step.

Documentation

Use *javadoc* to create HTML documentation for the Car class. Be sure the class and all methods are documented. The class and methods must be declared public for this to work.

New Skills Needed for this Assignment:

- Understanding of basic object-oriented concepts
- Understanding and use of Java class fundamentals, including the class declaration, class and instance variables and methods, instantiation of objects using the *new* operator and constructors, garbage collection and the finalize() method, the main() method, and the "this" keyword
- Understanding of how to use the special methods toString() and equals().
- Understanding of how to copy objects using copy constructors.
- Use of *javadoc* to create HTML documentation

Submit the following:

1. Your Java source code via electronic submission. Use the *Assignment 2* Dropbox Folder in D2L to submit electronically. The TA will compile and run your program to test it. Your source code files must be called *Car.java* and *TestCarSystem.java*.
2. A script showing the compilation of your source code and a sample run through your program. Name this file *script.txt* and also submit it electronically to the D2L drop box.
3. Your HTML documentation produced by running *javadoc* on your source code. Also submit this to the D2L drop box. The TA will check the HTML documentation to make sure it is complete and correct.

Introduction to Computer Science II
Assignment 2 Grading

Student: ____

Variables		
Instance variables	2	______
Class variables	2	______
Methods and Constructors		
3 constructors	3	______
3 set methods	3	______
4 get methods	4	______
Move method	2	______
Stop method	1	______
Travel Time instance method	3	______
Car ID method	1	______
Active number class method	2	______
toString method	1	______
equals method	1	______
Complete testing in main()	10	______
Elimination of redundant code (e.g., constructors)	2	______
Code structure (documentation, formatting, etc.)	4	______
HTML Documentation	4	______
Total	**45**	______ _____%