CPT 237 JAVA PROGRAMMING

# Assignment 4 – Acme Taxi Co. upgrade

The Acme Company was thrilled with the program you provided last week. But, as with most customers (and all marketing types), they want to modify it a little. Part of the modifications are to the UI, part to the cab itself. For the user interface, they didn’t like all the buttons, and they are adding even more functionality. Rather than buttons, they want to use radio buttons or a dropdown menu and a single OK button. The radio buttons or menu will allow the user to choose whether to add gas, enter mileage, etc. Pushing the OK button will perform the appropriate action. The behavior will not change. They also want some more functionality.

In addition to the radio button change, the following changes need to be made to the interface:

* Add three selections marked create, service report net earnings to your interface.
* Selecting **create** will instantiate a new cab. You may want this to be an actual button that is disabled once the cab is created, or to remove the button from your radio button list. No actions can be taken on the taxi until it has been created. Assume that the input field contains a number representing the initial amount of gas the taxi has available.
* Selecting **service** indicates to the underlying class that the service has been performed.
* Selecting **report net earnings** will display the net earnings. This is the amount of gross earnings minus the cost of the fuel and maintenance.
* When the taxi has exceeded 100 miles it should report on the screen "maintenance needed" and not accept any more fares until the service button is pushed.
* An extra input field is needed when gas is being added to capture the cost per gallon.

You will need changes to the underlying class as well. Rather than modifying the original class object, we will extend it so as not to disrupt all the other programmers who are still using the original taxi class. Remember to use code from your parent class whenever possible by calling those methods.

This new derived class will have a number of enhancements.

* It will keep track of operating costs. In particular, when gas is added in addition to the amount of gas added, there will be a second parameter for the per-gallon cost of gas. These two parameters will be used to add to the maintenance cost. Calling reset will clear the cost of operation.
* It will keep a maintenance record.  It tracks the number of miles the taxi has traveled since it was last serviced.  When the taxi has exceeded 100 miles it should report on the screen "maintenance needed"; it should not accept any new fares (i.e. add additional miles) until its service method has been called.  This should also cause a $25.00 addition to operation costs
* The reset functionality should not clear the miles since last serviced.
* A new constructor is needed. This constructor will accept the initial amount of gas the taxi has available as a parameter. Note that this gas is ‘free’. It does not have to be considered in the operating cost.
* There needs to be a function, e.g. setAlertListener, that accepts an object that can respond to a maintenanceAlert message. The object is expected to implement a MaintenanceAlert interface. (See below.) The MaintenanceAlert interface should provide a method that is to be called whenever maintenance is needed. The object should call this function when a fare is added that pushes it over the 100 mile threshold. Note that the fare that pushes the number over 100 is accepted, however no further fares are accepted until maintenance is done.
* The refusal of new fares should take place at the level of the underlying object, not at the UI level.

In order to place the “maintenance needed” message on the screen, the UI needs to register some object that can respond to the maintenance alert message. This object should implement a MaintenanceAlert interface (which you will design). That interface should have a method that the cab object can call when maintenance is needed. Do not ‘poll’ the object to determine that it needs maintenance.

You may change the variables in your original taxi to ‘protected’: **make no other changes** to the original class, except to fix problems in your original design.  Remember, all data in an object should be accessed only through public methods and all new capabilities must be coded in the new child class.  Again, look for places where you can call parent class methods when they contain code that must be executed; don't rewrite code from the sub class in the super class.