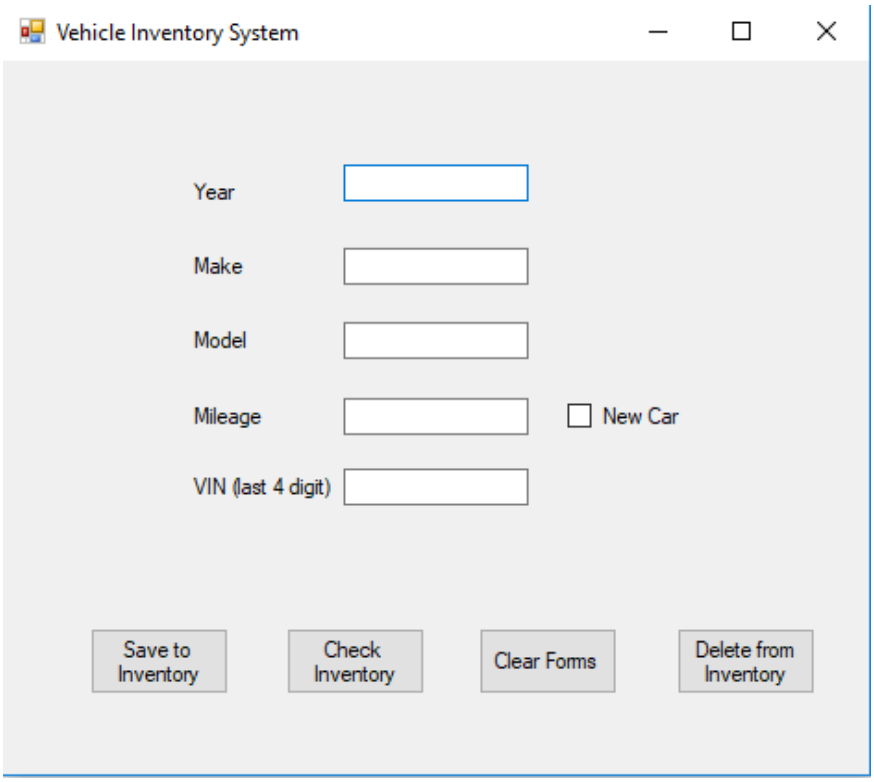
CIS 266 Introduction to .NET Development using C# (Spring 2018)

Assignment 9 (Due: May 16, 2018) Points: 80

Requirement

This assignment is to help you understand the class and objects and how to store objects in `List` In this assignment, you will use Visual Studio 2015 and write C# code for following functionalities:

1. Build a user interface that looks like this. It allows users to click on buttons to save, delete and check vehicle information from the inventory.

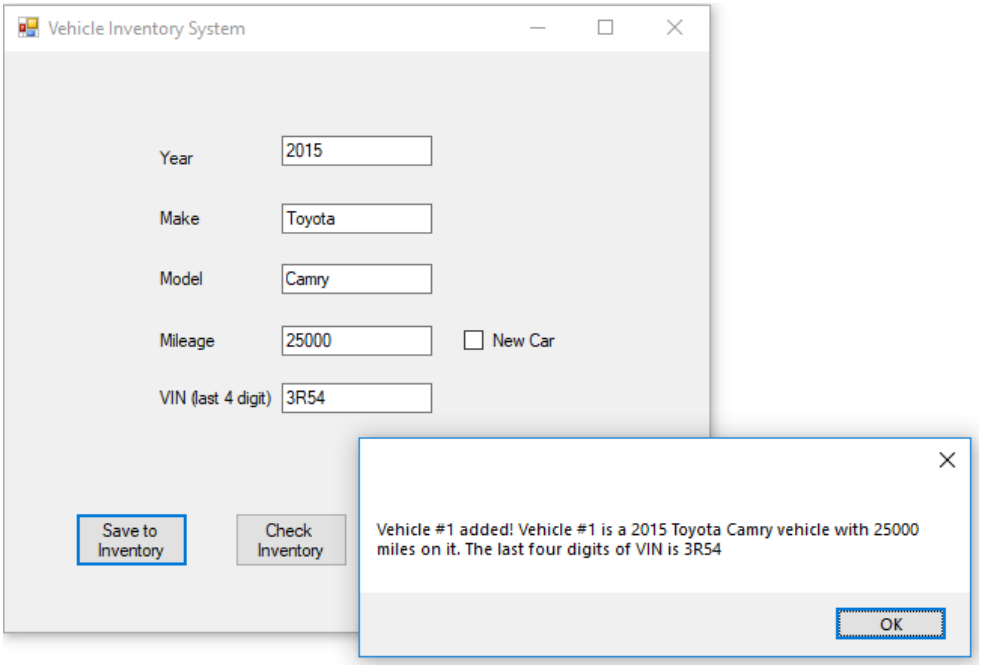


2. Vehicle information is stored in a class/object. Define the `Vehicle` class in a separate file called Vehicle.cs. `Vehicle` class should include properties of: `_id, _year, _make, _model, _mileage` and `_vin`. Design two constructor functions, the first one takes parameters for all properties, and the second one takes parameters for all properties except `_mileage`. The first constructor is used to create an object to hold a used car whose mileage must be greater than zero and thus needs to be passed in to the constructor function. The second constructor is used to create an object of a new car whose mileage is zero, and thus does not need to be passed in to the function. Also, design `get` and `set` functions for all properties. Last but not the least, design a member function called `ShowInfo()` for the

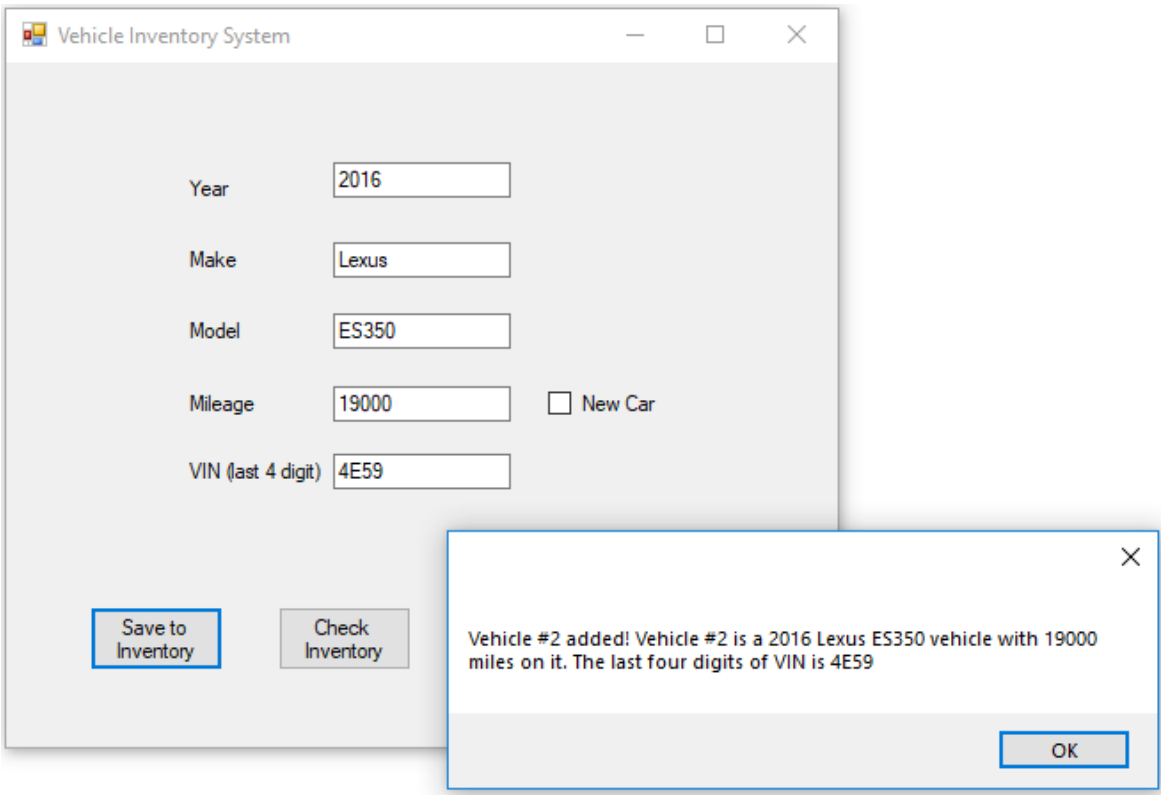
`Vehicle` class. When this function is called, return a message describing the focal vehicle. **Inside the `ShowInfo()` function, use `get` method defined in the class to access properties**.

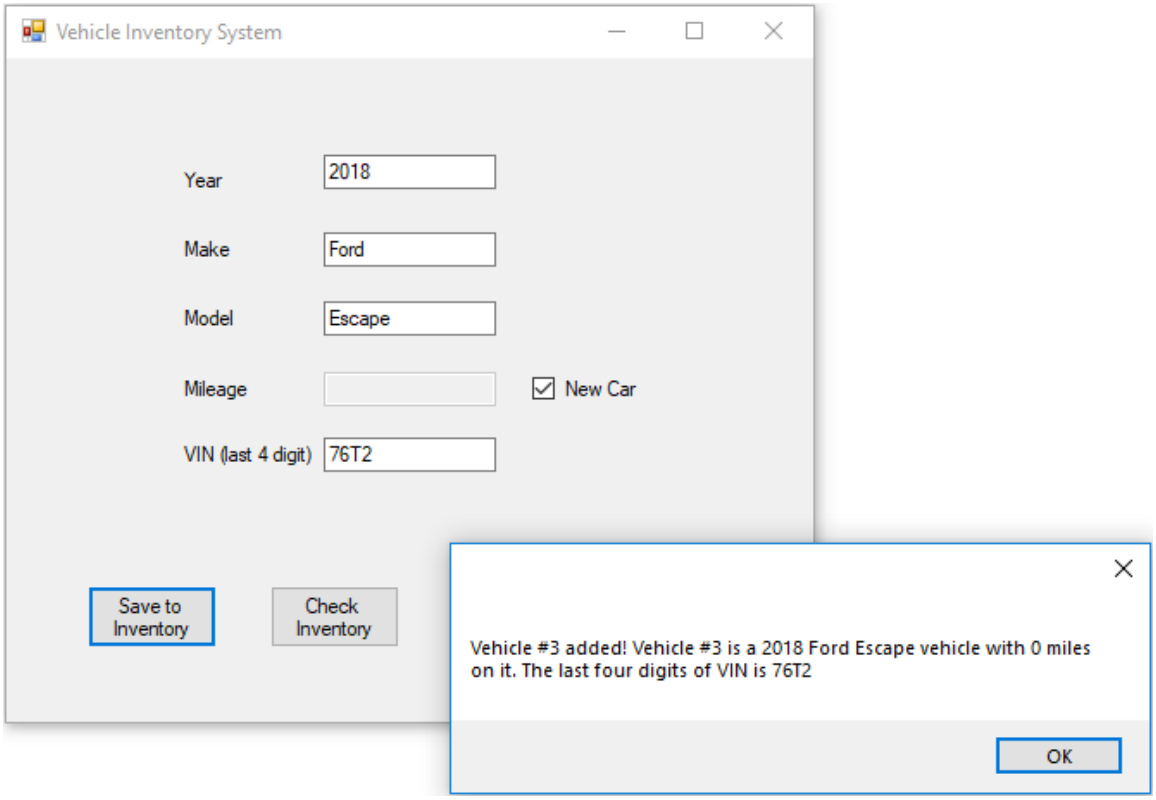
3. To keep track of all vehicles in the inventory, you need to declare a `List<Vehicle>` object called `vehicleInventory` in the fields. Program the form_loaded function to initialize the `vehicleInventory` object by using `vehicleInventory = new List<Vehicle>();`

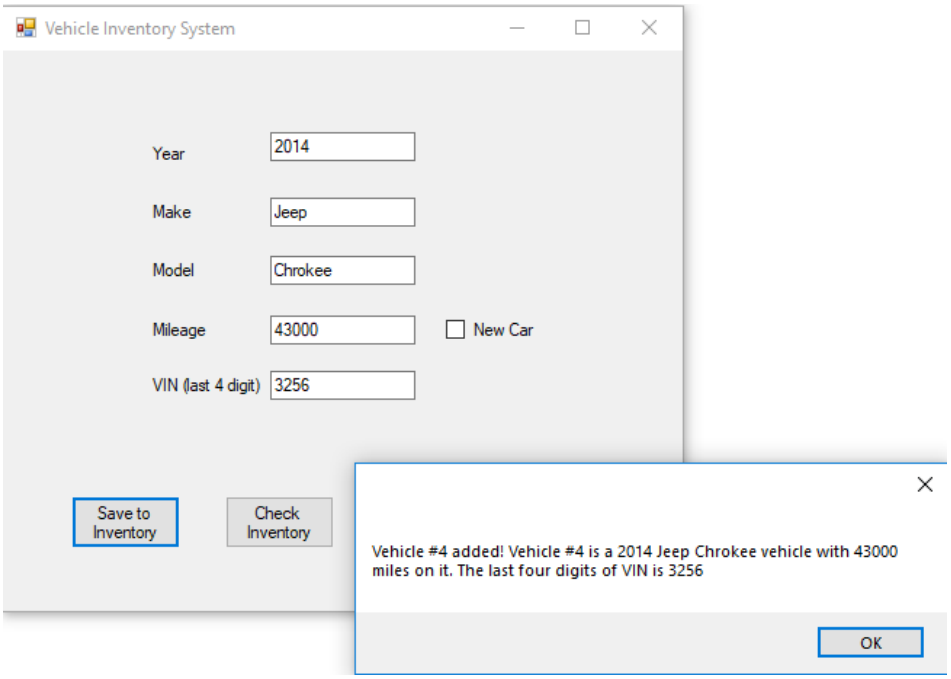
4. In the main form, program the "Save to Inventory" such that when it is clicked, a new `Vehicle` object is created. Depending on if "New Car" is checked, you use different constructor functions to initialize this object using user's inputs in the textboxes. If "New Car" is checked, clear the mileage textbox and disable it. For `_id`, you need to decide it dynamically based on the current number of vehicles in the inventory. For example, if there are 4 cars, the new car is #5. You can use `vehicleInventory.Count` to get the number of cars and increment it by 1 to get the ID for the new car. Then initialize this new Vehicle object and add it to the `vehicleInventory`. Lastly, pop up a message box to show a message like this which includes the "self-description" of the vehicle added to the inventory. **Please make sure that you use `ShowInfo()` to generate the description message.**



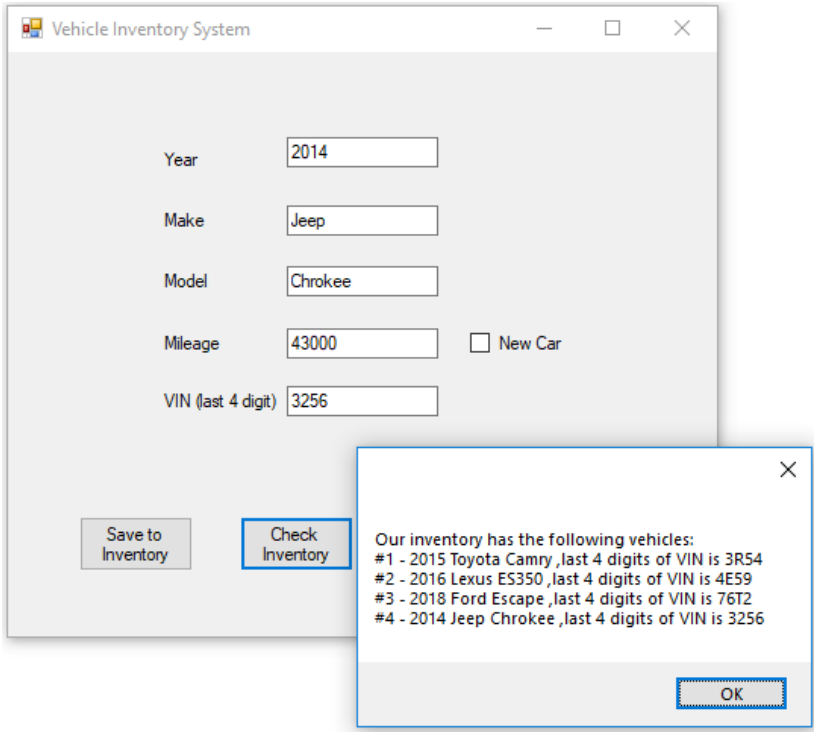
If you keep adding new vehicles to the inventory list, the ID will self-increment for each new record. If you add a new car, the mileage should be set to zero.


Vehicle Inventory System
Year
2016
Make
Lexus
Model
ES350
Mileage
19000
New Car
VIN (last 4 digit)
4E59
Save to Inventory
Check Inventory
Vehicle #2 added! Vehicle #2 is a 2016 Lexus ES350 vehicle with 19000 miles on it. The last four digits of VIN is 4E59
OK

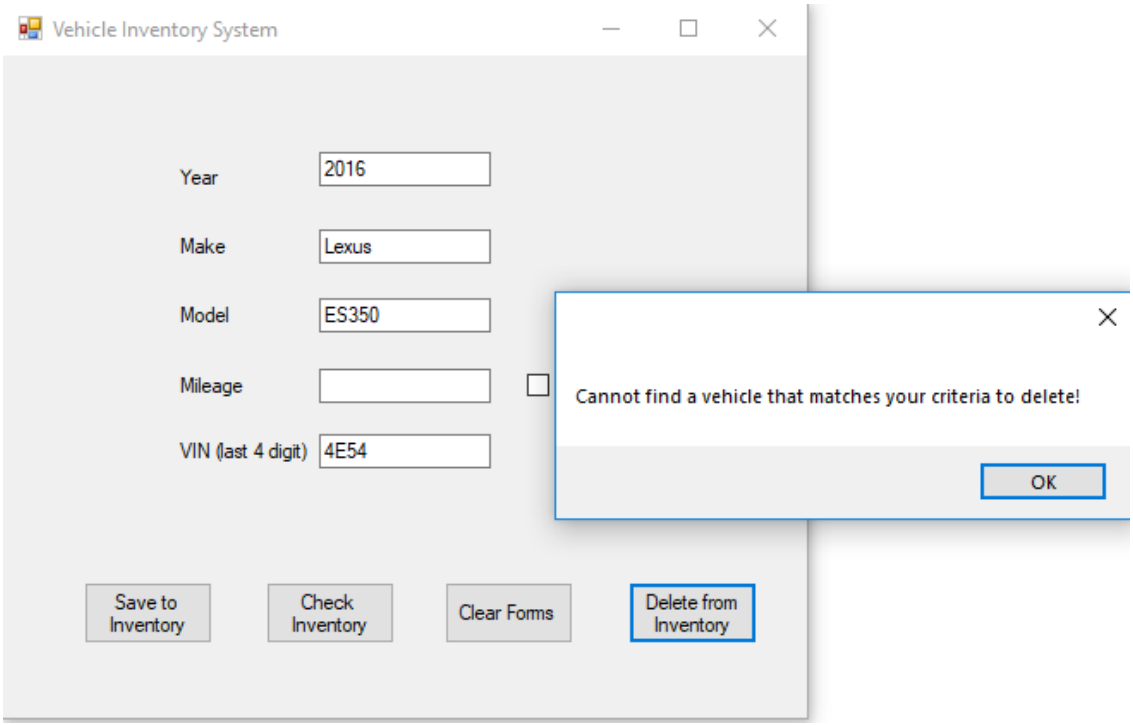

Vehicle Inventory System
Year
2018
Make
Ford
Model
Escape
Mileage
New Car
VIN (last 4 digit)
76T2
Save to Inventory
Check Inventory
Vehicle #3 added! Vehicle #3 is a 2018 Ford Escape vehicle with 0 miles on it. The last four digits of VIN is 76T2
OK

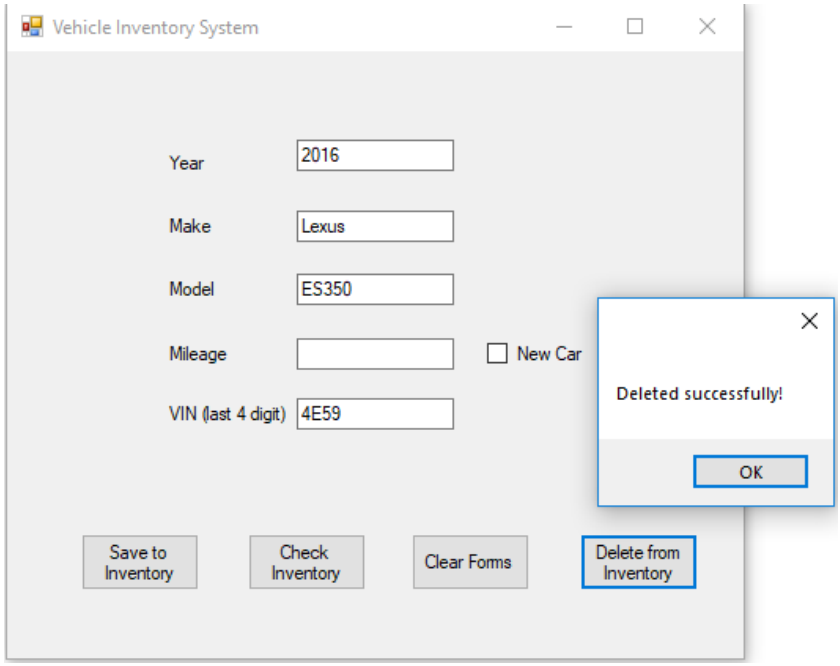


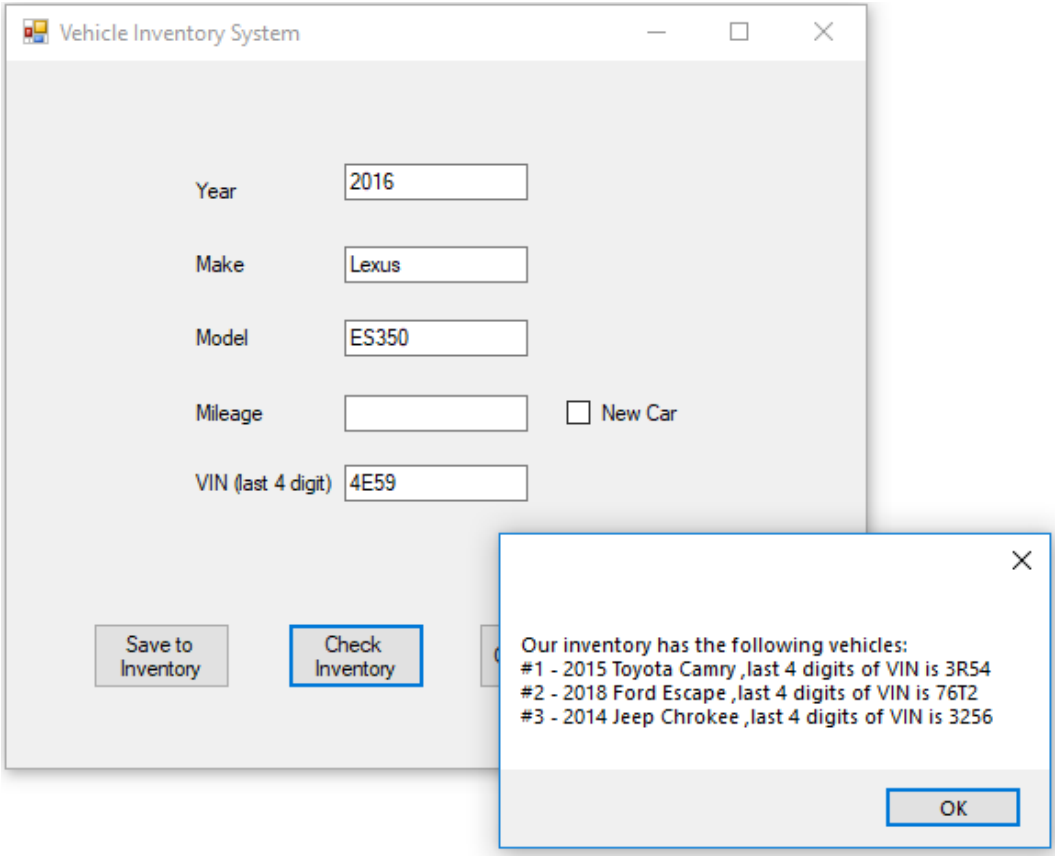
5. Program the button "Check Inventory" so that when it is clicked, a message box will be popped up with information shown below. You need to loop through the `vehicleInventory` and concatenate the message containing each vehicle's information to the final output message. **Please make sure you use `get` method defined in the class to access properties**.



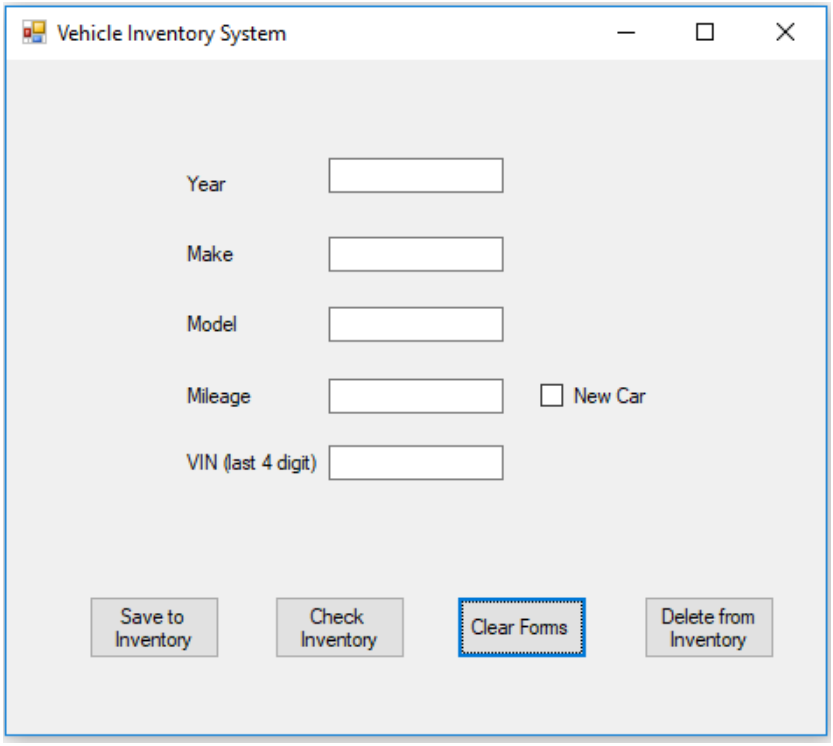
6. Program the "Delete from Inventory" button so that when it is clicked, the program will find the vehicle in the inventory based on the users' inputs of year, make, model, and VIN. If there is not a matched vehicle to delete, show an error message. Otherwise, return the index of the matched vehicle and remove it from the list. More importantly, after deleting one record, you need to re-arrange the IDs. For example, in the illustrated example below, after #2 – Lexus gets deleted, 2018 Ford becomes #2 now (it was #3 before) and 2014 Jeep becomes #3 now (it was #4). **Hint: in order to find the position where a record to be removed, you can have an index variable with an initial value of -1. Then loop through the list and if there is a match (I assume there is no more than one car with the same year, make, model, and last 4 digits of VIN), record the index, if the index is still -1 after the loop, it means we did not find a match. To re-arrange IDs, you can loop over the list after deleting a record and assign IDs to the remaining cars from 1. Please make sure you use `set` method defined in the class to reset the `_id` for each vehicle.**




Vehicle Inventory System
Year
2016
Make
Lexus
Model
ES350
Mileage
New Car
VIN (last 4 digit)
4E59
Save to Inventory
Check Inventory
Clear Forms
Delete from Inventory
Deleted successfully!
OK


Vehicle Inventory System
Year
2016
Make
Lexus
Model
ES350
Mileage
New Car
VIN (last 4 digit)
4E59
Save to Inventory
Check Inventory
Our inventory has the following vehicles:
#1 - 2015 Toyota Camry ,last 4 digits of VIN is 3R54
#2 - 2018 Ford Escape ,last 4 digits of VIN is 76T2
#3 - 2014 Jeep Chrokee ,last 4 digits of VIN is 3256
OK

7. Program the “Clear Forms” button so that if it is clicked, all inputs are cleared.



Submission

Zip your ENTIRE project folder and name your zipped file to (yourlastname)_a9. Submit your zipped file to the Blackboard dropbox as an attachment.