**PCS4 Exam – June 2016**

**Date: 22 June, 2016**

**Time: 10.45 – 13.15 h (150 minutes).**

**ADMITTED RESOURCES:**

* You are allowed to use everything on paper (books, notes, etc.) and on your laptop, but only what you bring in: you are not allowed to borrow something from someone else.
* During the exam it is not allowed to use the network. You should make the exam yourself: so no communication with MSDN or google for help and no communication with other students, like using facebook, e-mail, skype, Dropbox, gsm or whatever.

**GRADING:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Assignment** |  | **1** | **2** | **3** | **4** | **5** | **total** |
| **Max. points** |  | **15** | **30** | **25** | **20** | **10** | **100** |

The last assignment (assignment 5) can be skipped in case you have been allocated as an ‘active’ student for PCS4 this block. You will get the corresponding 10 points for this assignment as a bonus.

**THE APPLICATION**

Given is an application to be used by a garage. By means of the application, the garage owner can keep track of a list of cars that are currently for sale. The garage-owner can add cars to the list and change the price of a car.

There are 2 persons (Budget Bob and Fortune Fritz) who might be interested in newly added cars or in price-changes of cars already in the list: if a new car is added to the list, or the current price of a certain car changes (sometimes) they want to be notified.

The application consists of 3 windows: a window for the garage owner (FreakyGarageForm) and 2 windows for the clients (of type BudgetBobForm, and FortuneFritzForm respectively).

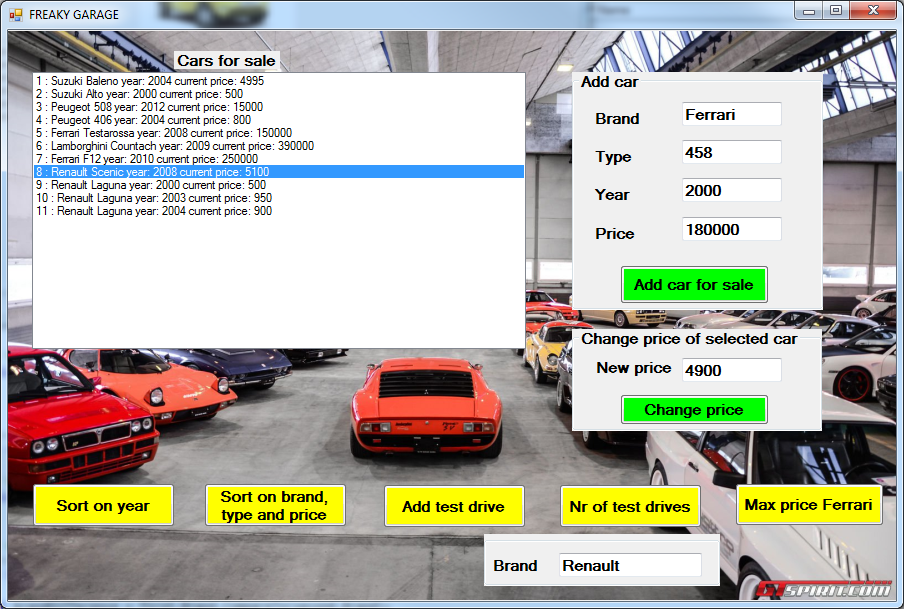
At startup all windows are created. It is possible that these windows overlap each other. Feel free to position them on the screen wherever you want.

Furthermore there are classes Car and Garage with already some code in it.

**PRELIMINARY REMARKS:**

* Whenever this exam paper suggests to use a certain name for a method, variable or anything else, you are required to indeed use that name.
* The classes Car, Garage, FreakyGarageForm, BudgetBobForm, and FortuneFritzForm already contain some code. It will be your task to add more code to these classes, if needed. You may add whatever you need. It will also be your task to add more classes, if needed.

The next picture shows the FreakyGarageForm-window:



The FreakyGarageForm-class has 3 variables:

private Garage myGarage; //the garage that has a list of cars for sale

private FortuneFritzForm frmF; // the window for Fortune Fritz

private BudgetBobForm frmB; // the window for Budget Bob

As you can see in the above picture the FreakyGarageForm window has several buttons.

The functionality of the buttons “Add car for sale” , “Change price” and “Max price Ferrari” has been partially implemented. It will be your task to complete the functionality for these buttons and to add functionality for the other buttons. For the details, see the assignments.

**Assignment 1 ( 7+8 pts).**

Consider the two buttons with text "Sort on year" and "sort on brand, type and price" on this window.

In the pcs4-lessons you learned 4 ways to sort elements in a list. First you did it yourself. Then we had 3 ways: by using IComparable, Comparison or IComparer.

By the way, to help you in testing your code, we added a method

private void addSomeTestingStuff()

which will add some cars as shown in the picture above. Using this testing-method is up to you. You may also skip it or add your own testing stuff. It will not cost you points and it will not give you points.

**Assignment 1a**:

Pushing the button with text "Sort on year" must sort the cars of myGarage from most recent (manufacturing) year to least recent (manufacturing) year.  
As you can see in the FreakyGarageForm-file this button-click is already implemented by calling:

this.myGarage.sortByYear();

The problem is solved by implementing the method

public void sortByYear()

in the Garage-class.

Give an implementation by using one of the last 3 ways (so by using IComparable, Comparison or IComparer).

**Assignment 1b**:

Pushing the button with text "Sort on brand, type and price" must sort the cars of myGarage by "brand – type - price". The cars must be sorted by brand alphabetically. If cars have the same brand, then they should be sorted alphabetically on type. If the cars have the same brand and type, they must be sorted on price from most expensive to least expensive (so on price in descending order).   
Again, as you can see in the FreakyGarageForm-file this button-click is already implemented by calling:

this.myGarage.sortByBrandTypeAndPrice();

Make it work by using one of the last 3 ways, but **not** in the same way as you did in the former assignment.

**Assignment 2 ( 30 pts).**

The buttons with text "Add car for sale" and "Change price" are partly implemented.

Clicking the button with text "Add car for sale" will create a car and add it to the list of cars.

Clicking the button with text " Change price " will check if there is a car object selected in the listbox lbCarsForSale. If so, the current price of that particular car is changed.

In both cases, the method NotifyListeners of the Garage class will be invoked. The body of this method should be implemented in such a way that all interested persons get informed.

Change the code in such a way, that:

Remember, there are 2 persons which might be interested in newly added cars or price changes to cars that are already in the list.

Budget Bob is watching window frmB, and Fortune Fritz is watching window frmF.

When a new car is added, Budget Bob wants to be informed if the price of the newly added car is below 5000 euros. Fortune Fritz wants to be informed if the brand of the newly added car is Ferrari.

If the price of a car (either higher, lower or equal to 5000 euros) is changed to a price below 5000 euros, Budget Bob would like to be informed of that. Fortune Fritz would like to be informed if the price of a Ferrari changes.

Budget Bob must be notified on frmB, and Fortune Fritz on frmF.

For Budget Bob, the class BudgetBobForm has a method

public void showCheapCar(Garage g, Car c, String reason)

to show the information of a cheap car c from garage g with reason reason in the listbox on his window. The reason may be because a new cheap car was added, or the current price of car c has changed. The method just adds the requested information to the listbox on Budget Bob's window (is already implemented).

For Fortune Fritz, the class FormF has a method

public void showFerrari(Garage g, Car c, String reason)

to show the price of car c of brand Ferrari offered by garage g with reason reason in the listbox on his window. It just adds the requested information to the listbox on Fortune Fritz's window (is already implemented).

**Assignment**:

Adjust the code in such a way, that the above mentioned functionality works correctly. You must give a solution by using events. An implementation without using events will not be rewarded.

Example as in the picture above:

Consider the situation as depicted in the screenshot above.

When the "Add car for sale"-button is clicked, Fortune Fritz should be informed, as a new Ferrari is offered for sale; Budget Bob is not interested, because the price is way too high.

When the “Change price” button is clicked, the price of the selected car, the Renault Scenic from 2008, is set to 4900 euros. As the new price of the car is now below 5000, Budget Bob should be informed. Of course, Fortune Fritz is not interested, as the selected car is not a Ferrari.

If a Ferrari is offered for a price below 5000 euros, both Budget Bob and Fortune Fritz must be notified.

**Assignment 3 ( 18+4+3 pts).**

Now about the button with text "Add test drive" and “Nr of test drives”.

We would like to register a list of cars that are used for a test drive. For this purpose, the cars are stored in a linked list. Every time a car is used for a test drive, it is being added at the front of the linked list. Note that one car may occur more than once in the linked list (if there are multiple test drives with the same car).

**Assignment 3a**:

Add a class LinkedListForTestDrives to your solution. This class must implement a linked list of cars that are used for a test drive.  
It must have a constructor, and two methods described in detail below.

First, the method

public int nrOfTestDrivesWithCarsOfBrand(string brand)

This method should return the number of cars of that brand currently in the list.

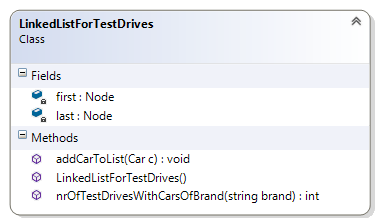
Then there is the method

public void addCarToList(Car c)

implemented in the following way:

* If the linked list is empty, then the car c must be added to the linked list.
* If the linked list is not empty, the car should be added at the front of the list.

The class diagram of the class LinkedListForTestDrives is given below:



**Assignment 3b**:

Add a variable of type LinkedListForTestDrives to your FreakyGarageForm-class.

Clicking the button with text "Add test drive " should add the car that is currently selected in the listbox lbCarsForSale to the linked list.

Give an implementation for

private void btnAddTestDrive\_Click (object sender, EventArgs e)

**Assignment 3c**:

When clicking the button with text “Nr of test drives”, the number of test drives with cars of the brand specified in tbBrandTestDrive should be displayed. Give an implementation for:

private void btnNrOfTestDrives\_Click (object sender, EventArgs e) .

**Assignment 4 ( 20 pts).**

Now about the button with text "Max price Ferrari".

First, you should give a recursive implementation of the method:

public int getMaxPriceFerrari(int n)

in the Garage class.

This method getMaxPriceFerrari has one parameter:

* an integer value n (n >= 0)

The method will check the first n cars in the list, and returns the price of the most expensive Ferrari in those first n cars in the list cars. If none of the first n cars is a Ferrari, the method will return -1.

A non-recursive solution will not be rewarded.

Next, about the button with text "Max price Ferrari".

Clicking this button should result in displaying the maximum price of all cars with brand Ferrari that are currently for sale, on the screen (how, that's up to you).

Example:

Assume the list of cars for sale is as displayed in the screenshot on page 2. Pressing the button “Max price Ferrari”, results in showing 250000.

**Assignment 5 ( 10 pts).**

This assignment can be skipped in case you have been allocated as an 'active' student for PCS4 this block. You will get the corresponding 10 points for this assignment as a bonus.

We would like to add functionality to this application to keep track of the maximum original price and the average original price of cars that come up for sale. A binary tree must be used in order to store the required information.

Every time a new car is added for sale, an object is added to the binary tree. This object will contain the brand, id and original price of the car. The binary tree should be sorted on original price.

1. Add a button to the form such that the user can request the highest original price of all the cars in the tree, and that shows the correct value according to the content of the tree (e.g. in a message box)
2. Add a button to the form such that the user can request the average original price of all cars in the tree, and that shows the correct value according to the content of the tree (e.g. in a message box)

Implement this functionality.

NOTE: An implementation without using a binary tree will NOT be rewarded.

**End of exam.**