PCS4 Exam – Feb 2018

Date: 22 Jan, 2018
Time: 12.45 – 15.15 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, mobile phone or whatever.

Introduction
RacingFed is a federation that organizes and manages races. Given is an application to be used by a racing federation (RacingFed) to **add racers** and change the **rank number**.

Besides, there are two scouts (Red and Blue), that would like to know when new **pro** racers are added to the list and if their rank is changing. They might be interested also for **rec** (recreational) racer;

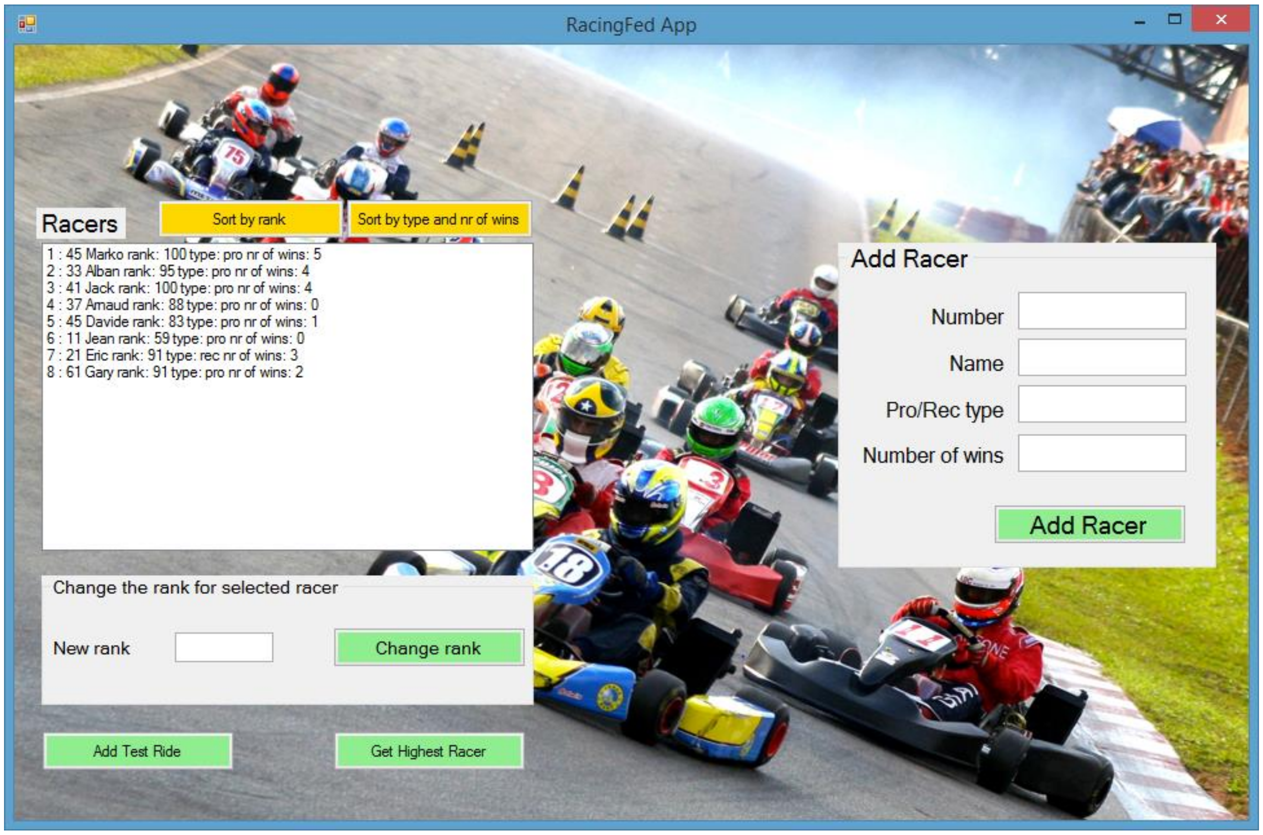
The application consists of 3 windows: a window for the RacingFed user (RacingFedForm) and 2 windows (RedScoutForm, BlueScoutForm) for the corresponding Red and Blue scouts. *At startup, all windows are created. It is possible that these windows overlap each other. Feel free to position them on the screen wherever you want.*

There are classes Racer and RacingFedForm already with 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 Player, RacingFed, RacingFedForm, RedScoutForm and BlueScoutForm 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 RacingFedForm-window:



```
1 : 45 Marko rank: 100 type: pro nr of wins: 5
2 : 33 Alban rank: 95 type: pro nr of wins: 4
3 : 41 Jack rank: 100 type: pro nr of wins: 4
4 : 37 Arnaud rank: 88 type: pro nr of wins: 0
5 : 45 Davide rank: 83 type: pro nr of wins: 1
6 : 11 Jean rank: 59 type: pro nr of wins: 0
7 : 21 Eric rank: 91 type: rec nr of wins: 3
8 : 61 Gary rank: 91 type: pro nr of wins: 2
```

The RacingFedForm-class has 2 variables:

```
private RedScoutForm redScoutFrm;  // the window for Red Team
private BlueForm blueScoutFrm;     // the window for Blue Team
```

As you can see in the above picture the RacingFed-window has several buttons.
The functionality of the buttons "Add Racer", "Change rank" 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 by rank" and "sort by type and nr of wins" 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.

To help you in testing your code, we added a method

```
	private void addSomeTestingStuff()
```

which will add some players 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 by rank" must sort the racers of from highest rank to the lowest rank.
As you can see in the RacingFedForm-file there is a button handler.
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 by type and nr of wins" must sort the racers type (rec, pro) and nr of wins.
Again, as you can see in the RacingFedForm-file this button-handler is already present.
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 (30pts).

The buttons with text "Add Racer" and "Change Rank" are partly implemented.
Clicking the button with text "Add Racer" will create a Racer and add it to the list of racers.
Clicking the button with text " Change Rank " will check if there is a racer object selected in the If so, the current rank of that particular player is changed.

Change the code in such a way, that:

Both RedScout and BlueScout forms are interested about the racers of type “pro”.
RedScout is watching window `redScoutFrm`, and BlueScout is watching window `blueScoutFrm`.

Both forms have a method showRacer

```
    public void showRedRacer(Race r)
```

to show the information of from Race `r` in the listbox on his window.

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.

When the "Add player"-button is clicked, BlueScout and RedScout should show the list of the pro players.

When the “Change rank” button is clicked, the rank of the selected player, in the list is set to a new value and BlueScout and RedScout should be informed regardless of the type (pro, rec).

Assignment 3 (18+7 pts).

Now about the button with text "Add test ride" and “Nr of test rides”.
We would like to register a list of racers that are added for a test ride. For this purpose, the players are stored in a linked list. Every time a player is added for a test ride, 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 rides of the same player).

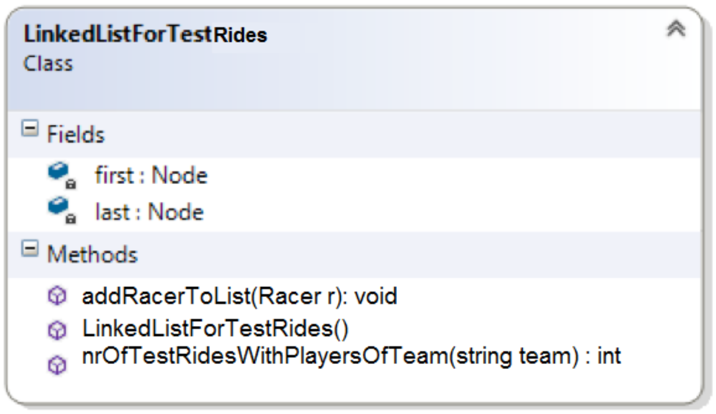
Assignment 3a:
Add a class LinkedListForTestRides to your solution. This class must implement a linked list of players that are added for a test drive.
It must have a constructor, and two methods described in detail below.

First, the method
```
public int nrOfTestRidesWithPlayersOfTeam(string team)
```

This method should return the number of players of that `team` currently in the list.

Then there is the method
```
public void addRacerToList(Player p)
```
implemented in the following way:
- If the linked list is empty, then the player p must be added to the linked list.
- If the linked list is not empty, the player should be added at the **front** of the list.

The class diagram of the class LinkedListForTestRides is given below:



LinkedListForTestRides
Class

Fields
first : Node
last : Node

Methods
addRacerToList(Racer r): void
LinkedListForTestRides()
nrOfTestRidesWithPlayersOfTeam(string team) : int

Assignment 3b:

Add a variable of type LinkedListForTestRides to your RacingFedForm-class.

Clicking the button with text "Add test ride " should add the players that is currently selected in the listbox `lbRacers` to the linked list.
Give an implementation for

```
private void btnAddTestRide_Click (object sender, EventArgs e)
```

Assignment 4 (20 pts).

Now about the button with text "Highest ranked racer".

First, you should give a recursive implementation of the `btnGetHighestRankRacer_Click()` in the RacingFedForm class.

The program should check the list, and returns the rank of the highest ranked racer from the listbox lbRacers.

Clicking this button should result in displaying on the screen, the highest rank from all racers. It is up to you how you present the value.

Example:

Assume the list of racers added to race is as displayed in the screenshot on page 2.
Pressing the button "Highest rank red racer", results in showing 100.

End of exam.