Exam PCS2 – Ticker ordering form January 2017

Time: 9:30 - 12:00 h (150 minutes).

ADMITTED RESOURCES:

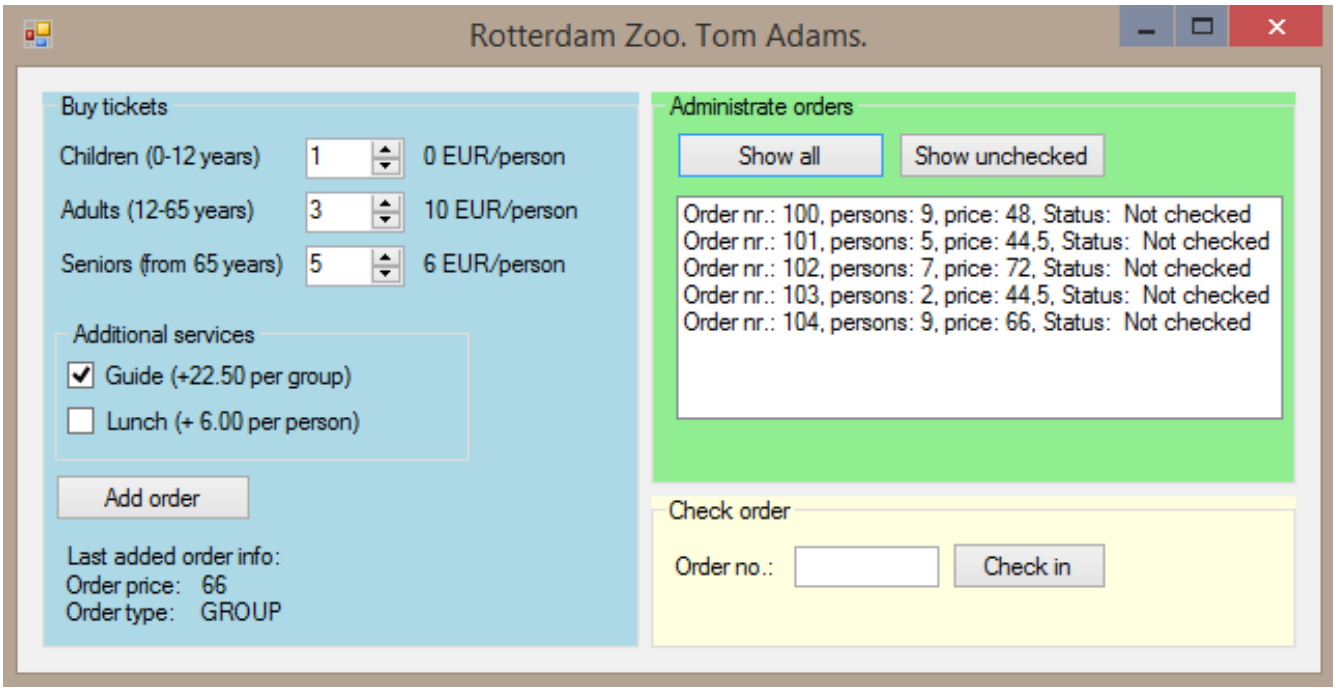
- You are allowed to use everything on paper (books, notes, etc.) and on your laptop, but only what you bring in: you are not allowed to borrow something from someone else.
- During the exam it is not allowed to use the network. You should make the exam yourself: so no communication with MSDN or google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

Exam points:

Assignment	1	2	3	4	Total
Points	3	45	26	26	100

THE APPLICATION

You are asked to develop an application to order and administrate tickets for a zoo (See image below).



The application has three parts:

- **Buy tickets (blue region)**
  User can create new orders. When ”Add order” is clicked, a new order is added to the zoo system and its “Order price” with “Order type” are calculated and shown at the bottom of this region.

- **Administrate orders (green region)**
  The zoo administrator can view orders. “Show all” button shows all the orders, while “Show unchecked” shows only orders that have not been checked in.
- **Check order (yellow region)**
  When visitors come to the zoo, their order is checked in. To do that, an employee must enter an order number in the text box and click “Check in” button.

Now open the startup-project either in Visual Studio 2013 or Visual Studio 2015. You can see that there is a form with controls on it (see the picture above). During this exam, you will implement two classes: class **Zoo** and class **Order**. Of course, you will add some code to the Form as well.

In the constructor of Form1 you can find testing-data commented out, which places the orders in the storage as seen in the image above. The use of this testing-data is up to you. You may also skip it or add your own testing code here. No points will be added or subtracted for using/not using it.

Important note about naming - Whenever you see a **bolded word**, you have to use this word as the name of a class, enum, method, field or property.

In the following assignments we ask you to implement certain things. If you feel the need to add more methods, fields or properties, then feel free to do so.

THE ASSIGNMENT

Assignment 1: OrderType enumerator (3 pts)

Since there are only three order types, we can use an enumerator. Create an enumerator called **OrderType** in the file “OrderType.cs”. It should contain the following three values: NORMAL, FAMILY, GROUP.

Assignment 2: Class Order (8 + 6 + 4 + 7 + 7 + 6 + 7 pts)

In this assignment you will be asked to implement the class “Order” in the file “Order.cs”.

a) Add the following properties (and if required, fields) inside the class:
   (Remark: When possible, use auto-implemented properties)
   - Store visitors’ number for **NoOfChildren**, **NoOfAdults** and **NoOfSeniors**. Any stored value must be 0 or bigger;
   - **OrderNumber** must be stored. It should be unique for each different order. Modifying this property value from outside of the class is forbidden;
   - A field **nextFreeOrderNumber** which increases with every order being created. The number should start from 100;
   - Boolean property **IsCheckedIn** to mark whether an order is checked in;
   - Property **includeGuide** that indicates whether guide service is included. This property should not be accessible from outside of the class;
   - Property **includeLunch.** It should indicate whether the lunch is included in the order. This property should not be accessible from outside of the class.

b) Add a constructor to the Order class.
   - Use parameters to assign a value to the following properties: “NoOfChildren”, “NoOfAdults”, “NoOfSeniors”, “includeGuide”, “includeLunch”;
   - The “OrderNumber” should be assigned with a value (unique order number) inside the constructor without the use of a parameter.

c) Add method **getTotalNrVisitors** which returns the total number of visitors in the order. This method should not be accessible from outside of the class.

d) Add method **GetOrderType** which returns “OrderType” enumeration value, based on the following rules:
   - If more than 7 people are in the order, return GROUP type;
   - If 2 adults and at least 1 child is in the order (but less than 8 people in total), return FAMILY type;
   - In other cases, return NORMAL type.

e) Add method **GetPrice** which returns a price for the order. Calculate a price according to the following rules:
   (Remark: A price should include any number of decimal places, E.g.: 10.3103, 4.1 and 5.65 are all accepted options)
   - Children visit for free;
   - Price per adult is 10 EUR;
   - Price per Senior is 6 EUR;
   - If guide service is added, 22.50 EUR to the total price should be added;
   - If lunch service is added, then add 6 EUR per person to the total price;
   - If type of the order is FAMILY, 10% discount should be applied;
   - If type of the order is GROUP, 20% discount should be applied.

f) Make a method named **AsString**. This method should return all the information about the order as shown in the ListBox in the image above.

g) Make a method named **GetExtraInfo**. The method should return extra information about an order.
   - Add string “Guide; ” to a result if a guide is added to the order;
   - Add string “Lunch; ” to a result if a lunch is added to the order;
   - Add string “Free Ice cream; ” to a result if OrderType of the order is FAMILY;
   - Add string “Free drink; ” to a result if OrderType of the order is GROUP.

   For example:
   If guide is added to an order and it’s type is FAMILY, the following string should be returned: “**Guide; Free Ice cream;** ”
   If guide and lunch are added to an order and it’s type is GROUP, the following string should be returned: “**Guide; Lunch; Free drink;** “.

Assignment 3: Class Zoo (4 + 3 + 4 + 4 + 6 + 5 pts)

In this assignment you will be asked to implement the class “Zoo” in the file “Zoo.cs”.

a) The “Zoo” class must have the functionality to store a list of “Order” objects and the possibility to store its name. Add corresponding members to this class for that functionality.

b) Make a constructor for the **Zoo** class. It should have one parameter to initialize the name of the zoo. Initialize a list of “Order” objects, too;

c) Add a method **AddOrder** with one parameter. The parameter should be an object of type "Order". The method should add this object to the "Order" list;
d) Add a method **GetAllOrders**. The method should return a list of all orders;
e) Add a method **GetUncheckedOrders**. The method should return a list of orders, which are not yet checked in;
f) Add a method **GetOrder** with one numeric parameter. The method should search for an order which "OrderNumber" corresponds to the given parameter value and return it. If no order is found, the method returns **NULL**.

Assignment 4: Main Form (4 + 4 + 4 + 4 + 4 + 6 pts)
In this assignment you will be asked to implement the "Form1" class and implement some functionalities of it.

a) Create a "Zoo" object for your favourite zoo.
b) Display the name of that zoo, followed by your name in the title bar of the window (as in the picture above).
c) Add functionalities to the "Add order" button.
   - On button click, a new "Order" object should be created and added to the list of orders in the zoo object;
   - Display a price for the last added order in the label **lbPrice**;
   - Display an order type for the last added order in the label **lbTicketType**.
d) Add functionality to the "Show all" button.
   - On button click, all the orders should be listed in the list box.
e) Add functionality to the "Show unchecked" button.
   - On button click, non-checked-in orders should be listed in the list box.
f) Add functionality to the "Check in" button.
   - On button click, it should check in the given order number;
   - If there is no order to check in, a message with text like: "**Order not found**" should be displayed;
   - If an order is already checked in, a message with text like: "**Order already checked in**" should be displayed;
   - If an order was found and that order was not checked in yet, it will be updated as checked in, and a message with its extra information should be displayed.