PCS3 Exam

Date: Thursday, 2017, October 26
Time: 12:45 – 15:15 h. (150 minutes)

Rules:

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 communication by Facebook, e-mail, Skype, gsm or whatever.

Way of working during the exam and handing in your exam: During the exam, your will use a program that block your internet-connection, takes care of downloading the exam (this description and a startup-project) and let you upload your solution. If needed, you can read a user's guide, which is present at the exam.

Grading: see assignments.

Preliminary remarks:

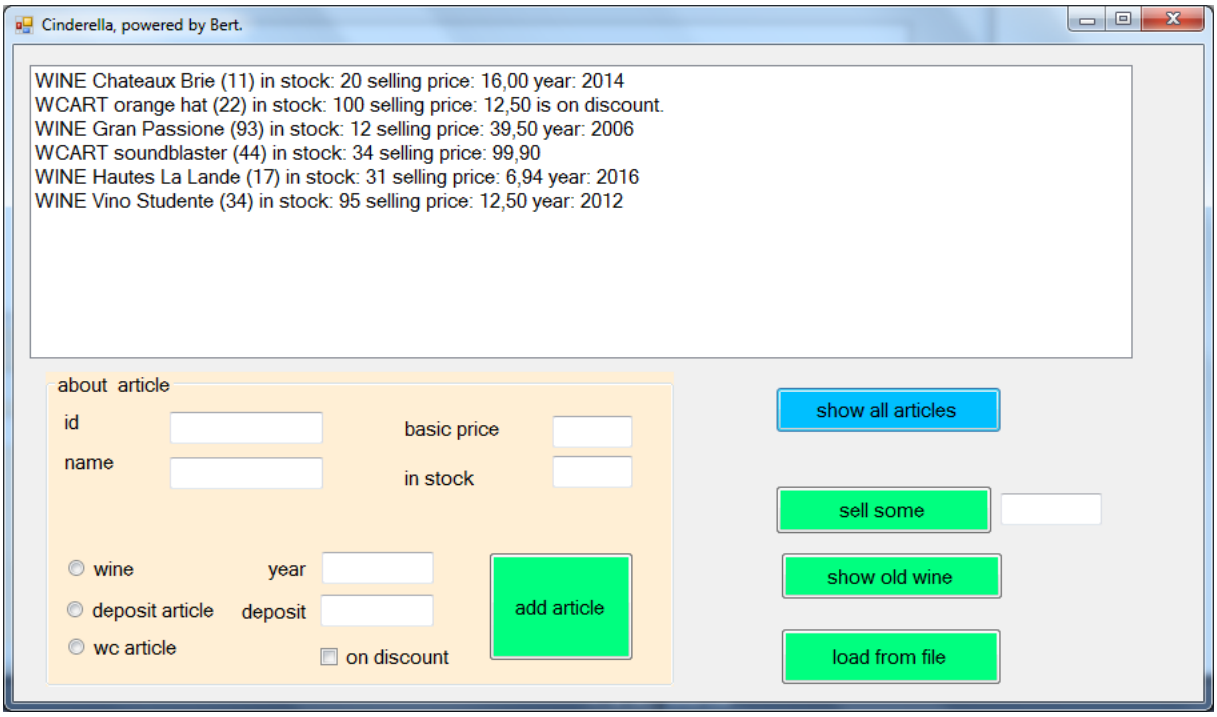
- Whenever this exam paper suggests to use a certain name for a method, variable or anything else, you are required to indeed use that name.
- If you think you need more class-members, you are allowed to add more members to the classes than specified in this exam.
- In this exam, we use the word "app", which is short for "application" or "program".

Introduction: An application for a shop with several kind of articles.

In this assignment, we ask you to write a C# application for a shop.
Another programmer, who was not familiar with inheritance, already started programming. In the current version, an article could be wine or a world-championship-article (wc-art for short; the shop hopes to sell a lot of these articles close before a world championship football, volleyball or whatever).
The owner of the shop has decided to start selling more kind of articles. For now, he/she also wants to sell deposit-articles (an article with a deposit, for instance a bottle of beer. When a person buys a bottle of beer, he/she has to pay the price for the beer and a deposit for the bottle. When the buyer returns the bottle, the buyer will get the deposit back.)
In future, the owner would like to sell more kinds of articles, so your solution should be ready for it.

Of course, in this exam we do not have the time to implement a complete application, so we will do a part of it.

The GUI of the app looks like:

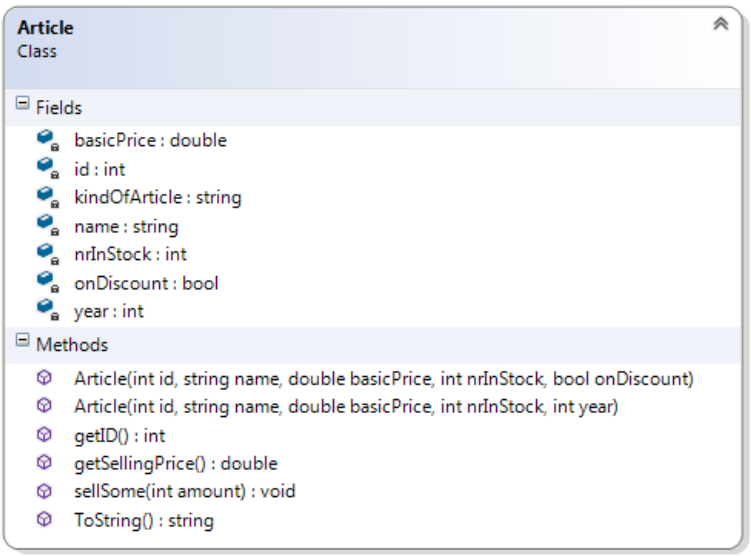


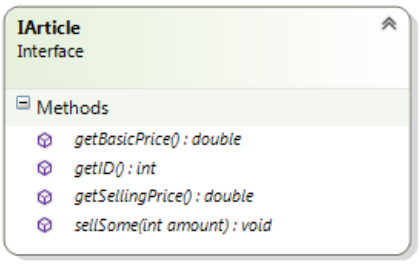
As you can see the buttons in the application have two different colours. The button handler for the blue-coloured button is already given, and you are asked to provide the implementation of the button handlers for the green-coloured buttons.

We put some testing data in it (the method `addSomeTestingStuff()`). You are free to use this testing data or your own data. Using this testing data could result in the figure above, after running the startup project and clicking the blue button.
If you insist on using the testing stuff from the method `addSomeTestingStuff()`, it could be that, due to your solution for the assignments, you have to adjust this method, but that's up to you.

ASSIGNMENT 1 (40 points): The classes for several kinds of articles, the interface IArticle and your own exception-class.

In the start-up-project there is already a class `Article` and an interface `IArticle`:





An `Article` has a field `kindOfArticle` to register if it is about wine or a world-championship-article (wc-art for short).
As also shown, in the startup-project, for every article we want to store its id, name, basic price and how many items of this article the shop has in stock.
Furthermore, for wine we want to register the year it was bottled and for a wc-art if it is on discount.
Evidently, the selling price of wine depends on the bottle-year and the selling price for a wc-art depends on being on discount or not.

As already stated before, the owner of the shop has decided to start selling deposit-articles. A deposit article also has an id, name, basic price and a number in stock, like every other kind of article, but it also needs to register its deposit. The selling price is the sum of its basic price and the deposit (an example of a deposit-article is a bottle of beer. If the basic price is € 1.25 and the deposit is € 0.10, then the selling price is € 1.25 + € 0.10 = € 1.35.)

For future, every kind of article must have certain methods. These required methods are specified in the interface `IArticle` in the startup-project.

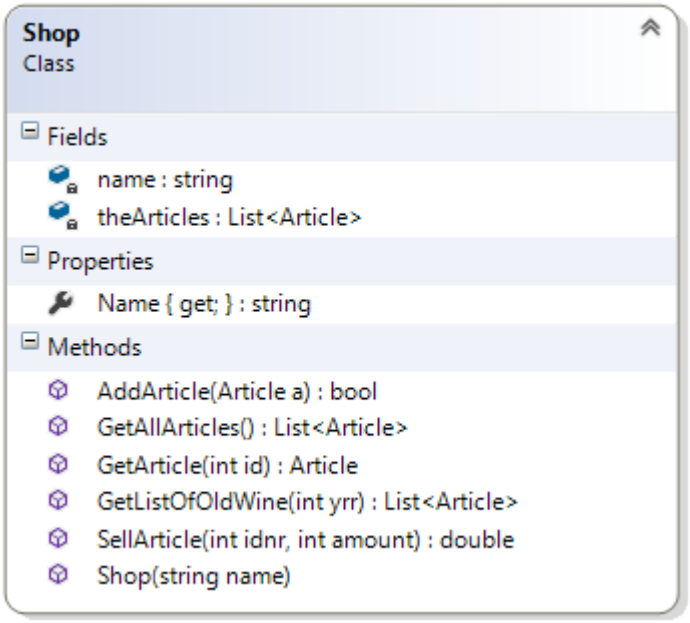
Assignment:

- Add classes for all three kind of articles. Adjust the code for the class `Article`
- The Article-class must implement the interface `IArticle`.
- You are not allowed to change the interface `IArticle`.
- The method `ToString()` for wine and for a wc-article should return a string the same as already implemented in the startup-project (so first an indication “WINE” or “WCART”, followed by information about . . .).
- The method `ToString()` for a deposit article should return a string starting with “DEP “, followed by information about its name, id, number in stock and selling price.
- The method `sellSome(int amount)` lowers the number in stock by amount. This method must throw an own made exception in case the number in stock is not sufficient or the amount is not a positive number.

Use the benefits of inheritance as much as possible.

ASSIGNMENT 2 (10 points): The class Shop

The class `Shop` has a name and a list of articles. Furthermore, it has some methods. All methods are implemented already except one method.



Assignment:
Give an implementation for the method

```
public List<Article> GetListOfOldWine(int yrr)
```

This method returns a list of all wines, bottled in year yrr or earlier.

At the bottom of the Shop-class there is a comment: *"IMPORTANT: No changes above this line."* which shows if you have changed anything in the Shop-class.
The teachers think you only need to give an implementation for the method `GetListOfOldWine(int yrr)`, but if you change more above the line, please indicate it.
Otherwise the teacher will not scroll through code above the comment.

ASSIGNMENT 3 (2 + 10 +12 + 8 + 18 = 50 points): about Form1.

The Form1 has a field:

```
private Shop myShop;
```

Assignments:

a) First of all: the field `myShop` is created with name `"Cinderella"`. Choose your own name for the shop. Change the code in such a way that the shop gets the name you have chosen. Make sure in the title bar of the window (the above part of the window) you get the text "<name of your shop> powered by <your name>".

b) About the button with text "add article".
The user wants to add a new article, the user has to input data for its id, name, basic price and how much there are in stock. Furthermore, for wine the user has to input the year it is bottled, for a deposit article how much the deposit is and for a wc-article if it is on discount or not. Clicking the button with text "add article" will result in adding an article as indicated

by the radiobuttons (if the radiobutton for wine is checked, a wine-article should be added, if the radiobutton for . . ., etc.).
Make this button work.
By the way, the id-number for every article must be unique. Show an appropriate message to inform the user if the article is added or not.
In this assignment, you may assume the user does not make any typing errors.

c) About the button with text “sell some”.
The user wants to register that a certain amount of a certain article is sold. The user also wants to see on the screen the total selling price for that amount of articles, if possible.
In the textbox tbID the user inputs an id of an article and in the textbox tbAmount an amount. Clicking the button with text “sell some” should do the job.
In this assignment you should handle all kind of possible exceptions. So, if the selling is possible, inform the user about the total selling price. If something is wrong, give the user an appropriate message about it (so not the text “Something was wrong” but a text telling what was wrong).
Make this button work.

d) When the button with text “show old wine” is clicked, you see an overview of all wine, bottled in the year as indicated in textbox tbYear or earlier.
Make it work.
In this assignment, you may assume the user does not make any typing errors.

e) When the button with text “load from file” is clicked, the user must have the possibility to choose a filename by showing a FileDialog-window. If the user chooses OK, the contents of the chosen file is used to fill the list of articles of the shop. If the user chooses to cancel, nothing should happen.
Make it work.
The contents of the file should be as follows:
- For every article there are some lines in the textfile, followed by a line with =-signs.
- For wine: first a line with the text “wine”, then per line its id, name, basic price, number in stock and bottle-year.
- For wc-article: first a line with the text “wcart”, then per line its id, name, basic price, number in stock and then a line with text “discount yes” or “discount no” telling if it is on discount.
- For a deposit article: first a line with the text “dep”, then per line its id, name, basic price, number in stock and the deposit.
For an example: see next page.

----- END OF THE EXAM

Example: contents of a textfile about 5 articles, first wine, then wc-article, deposit article, wc-article and a wine article:

```
wine
33
Merlot
12
20
2011
==========
wcart
44
orange hat
12.50
100
discount yes
==========
dep
77
Bavaria beer
2.50
300
0.25
==========
wcart
88
orange flag
25
10
discount no
==========
wine
99
Student wine
2.5
250
2016
==========
```

The name of this file is data.txt (see the root of the startup project)