PCS3 Exam

Date: Wednesday, 2017, April 5
Time: 8:45 – 11:15 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 an abbreviation for "application" or "program".

Introduction: An application for a candy store.

In this assignment we ask you to write a C# application for a candy store. The store wants to collect information about how much of each kind of candy is sold every day and to gather information about the effect of changing the prices (If the price is raised/lowered, does it influence how much candy is sold?).

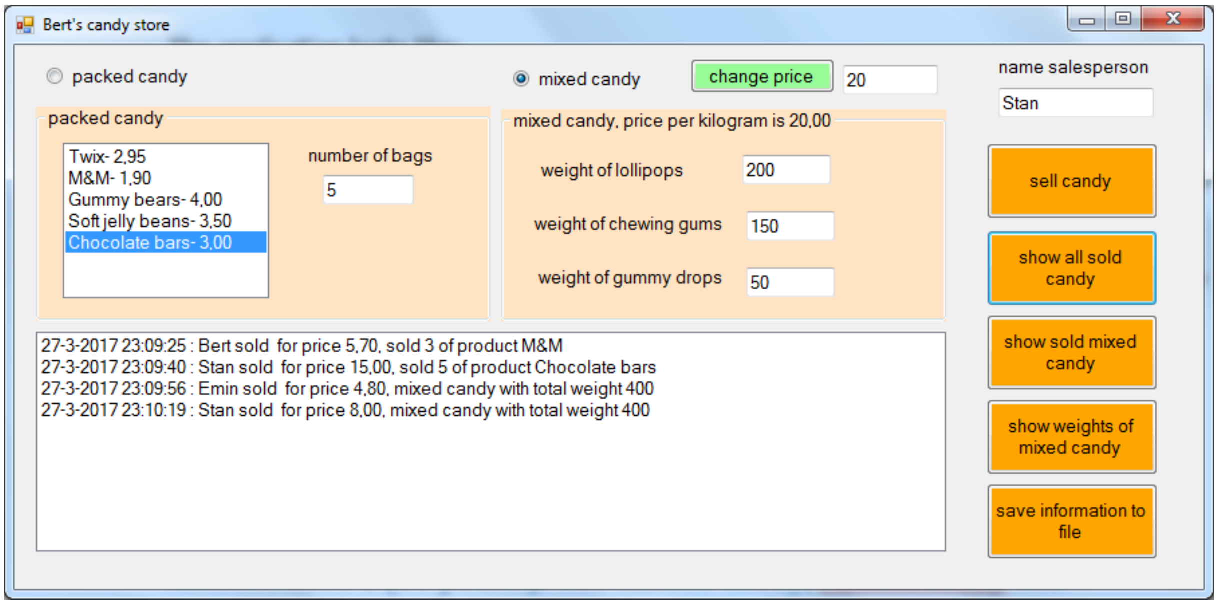
In this exam we will not implement a complete application, but we will do a part of it.

The store sells candy in two different types of packaging:

- There is candy which is already packed. A person, who wants to buy it, can buy one or more bags of this kind of candy.
  Examples: a bag with gummy bears, a bag with chocolate bars, a bag with m&m's.
  The owner of the candy store can set and change the price for this kind of candy.
- A customer can take an empty bag and choose his/her favourite candy out of several kinds of candy. The price of the bag is determined by the weight. The owner of the candy store can set and change the price for this kind of candy.

Example: suppose the price per kilogram is 12.00 euro and your bag with candy weights 400 gram. The price will be 0.4 * 12.00 = 4.80 euro.

The application looks like:



20

Stan

Twix- 2,95
M&M- 1,90
Gummy bears- 4,00
Soft jelly beans- 3,50
Chocolate bars- 3,00

5

200

150

50

```
27-3-2017 23:09:25 : Bert sold  for price 5,70, sold 3 of product M&M
27-3-2017 23:09:40 : Stan sold  for price 15,00, sold 5 of product Chocolate bars
27-3-2017 23:09:56 : Emin sold  for price 4,80, mixed candy with total weight 400
27-3-2017 23:10:19 : Stan sold  for price 8,00, mixed candy with total weight 400
```

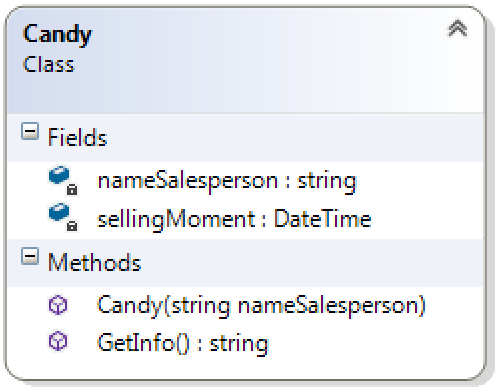
In order to help you write this application, some code has already been given in the start-up-project.

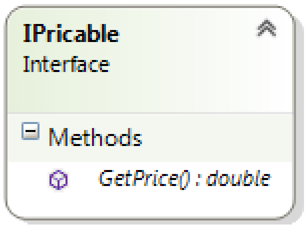
The code does NOT compile yet, because some code is missing. You will implement it in the assignments. We put some testing data in it. You are free to use this testing data or your own data. Using this testing data could result in the figure above.

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

ASSIGNMENT 1 (30 points): The classes Candy, PackedCandy and MixedCandy; and the interface IPricable.

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





You are not allowed to change the interface `IPricable`.

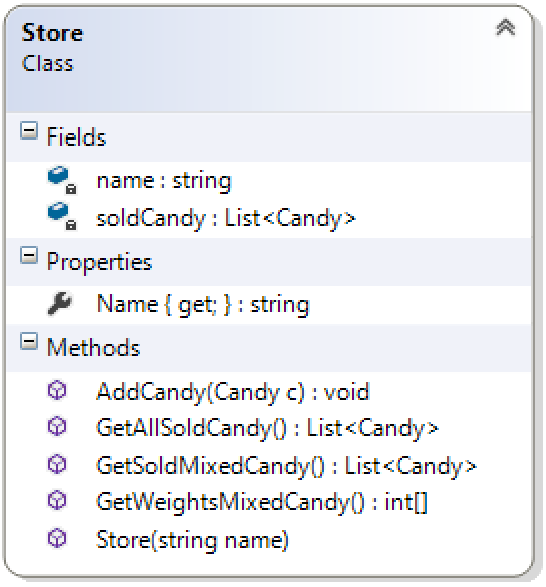
The class `Candy` has already some code. It has a field to register the name of the salesperson (the name of the employee who sold the candy) and the moment it was sold. Note that the method `GetInfo()` returns a string with information about the moment the candy was sold, its salesperson's name and its price (more about in a few lines below).

Change the code for the class `Candy` and add classes `PackedCandy` and `MixedCandy` according to the following rules (**use the benefits of inheritance as much as possible.**):

1. `Candy` must implement the interface IPricable.
2. The classes `PackedCandy` and `MixedCandy` inherit from the class `Candy`.
3. Objects of type `PackedCandy` represent selling a packed candy. For every packed candy we would like to register the name of the candy, the name of the sales person, the selling moment, the price per bag and the total number of sold bags. The selling moment is always `DateTime.Now`, as already implemented in the constructor of `Candy`.
   The heading of the constructor is:
   ```
   public PackedCandy(String nameSalesperson, String nameCandy,
                      double pricePerBag, int nrSold)
   ```
   For PackedCandy-objects, the method `GetPrice()` must return
   (the total number of sold bags) * (the price per bag).
   The method `GetInfo()` should return a string with the same information as returned by the method `GetInfo()` of the Candy-class, followed by how many are sold and the name of the candy (example: see first and second line in the lower listbox in the above screendump).
4. Objects of type `MixedCandy` represent selling a bag with mixed candy. We need to register the name of the sales person, the selling moment, the price per kilogram and 3 integer numbers. To keep it simple, the buyer can choose out of 3 candy's: lollipops, chewing gums and gummy bears. For each of them we want to register their weight in grams (so 3 integers: the weight of the lollipops, the weight of the chewing gums and the weight of the gummy drops).
   The heading of the constructor is:
   ```
   public MixedCandy(String nameSalesperson, double pricePerKilogram,
                int weightLollipops, int weightChewingGums, int weightGummyDrops)
   ```
   For MixedCandy-objects, the method `GetPrice()` must return (the weight of the lollipops + the weight of the chewing gums + the weight of the gummy drops) * (the price per kilogram) / 1000.
   The method `GetInfo()` should return a string with the same information as returned by the method `GetInfo()` of the Candy-class, followed "mixed candy" and the total weight of the bag (example: see third and fourth line in the lower listbox in the above screendump).

ASSIGNMENT 2 (10+10 = 20 points): The class Store

The class `Store` has a list `soldCandy` to store Candy-objects. Furthermore it has a name and some methods. In this assignment you need to implement two methods.



a) Give an implementation for the method
```
public List<Candy> GetSoldMixedCandy()
```
This method returns a list with all objects in the list `soldCandy` that are of type `MixedCandy`.

b) Give an implementation for the method
```
public int[] GetWeightsMixedCandy()
```
This method returns an array with 3 numbers representing the total weights of the lollipops, chewing gums and gummy drops in all `MixedCandy`-objects in the list `soldCandy`.

ASSIGNMENT 3 (2+15+ 8+4+4+4+13 = 50 points): The FormForCandyStore.

In this assignment you are asked to provide the implementation of the event handlers of the orange buttons.
The FormForCandyStore has some fields:
```
private Store myStore;              //the candy store
private String[] namesPackedCandy; //the names of the packed candy
private double[] pricesPackedCandy;//the prices of the packed candy
private double pricePerKilogramMixedCandy;
```

The arrays `namesPackedCandy` and `pricesPackedCandy` are so-called parallel-arrays: in the first array on index resp 0, 1, 2, . . . you find the name of a certain packed candy and on the same index you find in the second array its price per bag. You see these names and prices in the listbox in the left groupbox (groupbox with text “packed candy”). You see them in the listbox in parallel with the arrays (in other words: if the selectedIndex of that listbox is j, you can find the candy’s name on index j in the array and its price on the same index j in the other array).
The field `pricePerKilogramMixedCandy` stores the price for 1 kilogram of mixed candy.
The method `private void addSomeData()` initialises these fields with some data and displays it in the groupboxes, as shown in the above screendump.

The green button (is already implemented) makes it possible

- to change the price of some packed candy (by selecting the radiobutton for packed candy, selecting one of the packed candy in the listbox and specifying the new price) or
- to change the price per kilogram for the mixed candy (by selecting the radiobutton for the mixed candy and specifying the new price).

Now it is up to you. Give implementations for the event handlers of the other buttons according to the following specifications:

a) First of all: the field `myStore` is created with name `"Bert's candy store"`. Change it in such a way that it has your name. Make sure the name of the candy store is displayed in the title bar of the window (in the screendump: in the above part of the window).

b) When the button with text “sell candy” is clicked, there are 2 possibility’s:
If the radiobutton for packed candy is checked, the user should have selected one of the packed candy’s in the listbox and entered the number of bags. Of course, in the textbox on the right-top the salesperson should fill in his/her name. In this case, an object of type PackedCandy must be created and added to myStore.
If the radiobutton for mixed candy is checked, the user should have given 3 weights for resp lollipops, chewing gum and gummy drops. In this case, an object of type MixedCandy must be created and added to myStore.
In this assignment you may assume the user does not make any typing errors.

c) Approve your solution for assignment 3b) in such a way, that typing errors are handled in a user friendly way (in other words: handle all possible exceptions in a nice way).

d) When the button with text “show all sold candy” is clicked, you see an overview of all sold candy in the lower listbox. Make it work.

e) When the button with text “show sold mixed candy” is clicked, you see an overview of all sold mixed candy in the lower listbox. Make it work.

f) When the button with text “show weights of mixed candy” is clicked, you see the total weight of the lollipops, the total weight of the chewing gum and the total weight of the gummy drops on the screen. Make it work (it is up to you how you show it).

g) When the button with text “save information to file” is clicked, a save-dialog will be displayed on the screen, so the user can choose a filename. There are 2 possibility’s:
First possibility: the user choses to save. In that case, for each sold candy there is a line in the textfile with information about it. After these llines there is a line with some stars in it, followed by two lines: a line about the total profit gained by selling packed candy and the total profit gained by selling mixed candy. Below you see an example of such a textfile.
Second possibility: the user choses to cancel. In that case the application shows a proper message that nothing is saved.

Example: contents of a saved textfile for the example in the above screendump:

```
27-3-2017 23:09:25 : Bert sold  for price 5,70, sold 3 of product M&M
27-3-2017 23:09:40 : Stan sold  for price 15,00, sold 5 of product Chocolate bars
27-3-2017 23:09:56 : Emin sold  for price 4,80, mixed candy with total weight 400
27-3-2017 23:10:19 : Stan sold  for price 8,00, mixed candy with total weight 400
****************** totals *******************
profit from packed candy is: 20,7
profit from mixed candy is: 12,8
```

----- END OF THE EXAM