

IDENTIFICATION: Brushup#2 / PELE

Overall Purpose

The overall purpose for the group of 'Brush-up' assignments is to be able to quick, safe and sure in creating and using model classes, methods using collections and basic string operations.

The whole group of assignments consist of five steps:

1. [A simple model class with constraints.](#)
2. **More advanced and complex model classes. (This Assignment)**
3. Testing a model class.
4. Methods deal with collections.
5. String manipulations.

Background Material:

C# note of Per Laursen from 1st semester

(<https://github.com/perl-easj/Teaching-materials/tree/master/CSharpProgramming/Notes>)

Chapters: Prog2 og OOP2.

This Assignment: Brushup#2

Purpose

The purpose of this assignment is to refactor classes using inheritance and to add constraints to properties throwing exceptions when not fulfilled.

Mission

You are to

- a. Add a super class in the C#-library from Brushup#1
- b. Add constraints to properties
- c. Add appropriate exceptions when constraints are violated

Domain description

In overall, you are to design and implement classes for a restaurant's menu card.

You are to extract a superclass MenuItem from the two classes drink and dish and to consider the impact on the MenuCard class.

Assignment 1: Refactor classes

You are to refactor two classes the Drink and the Dish class; they both have a name and a price, which according to the DRY principle is to repeat code.

Step 1 – A new class MenuItem

You are to design and implement a new class 'MenuItem' having the properties **name** and **price**. You should make this class abstract, meaning you cannot make any object of the MenuItem class itself.

The class should have a default constructor as well as a constructor with parameter to transfer initial values.

You should also add a ToString method to the MenuItem class

Step 2 – Change Drink and Dish class

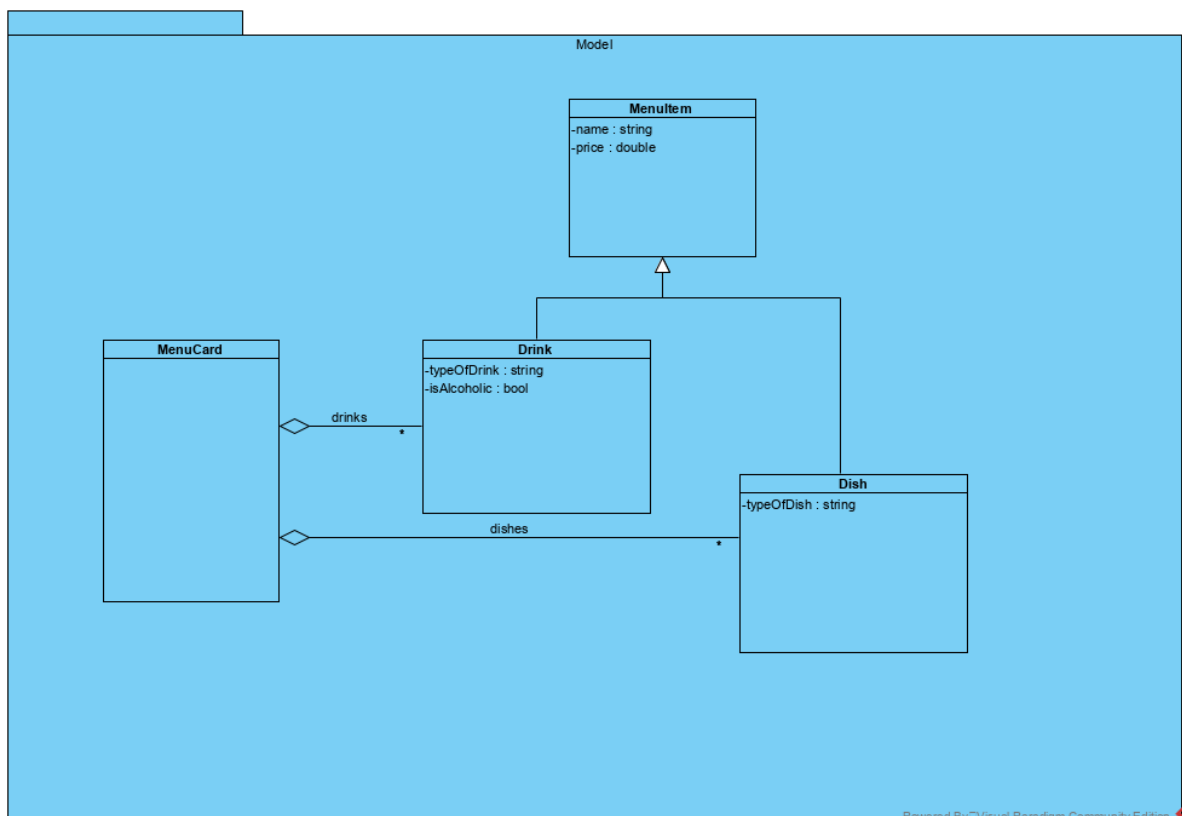
Then you have to refactor the Drink and the Dish class to inherit from this MenuItem base class.

Step 3 – Modify the Console Application

Modify your Console application, so it now uses the refactored classes.

What are the differences to the previous implementation?

The new class diagram



Assignment 2: Add constraints to properties

In general not all values are for the properties.

Here you are to add checks to your properties. The constrains are:

In MenuItem class:

- The name must not be null and it must have a length between 3-60 characters.
- The price must not be below zero

Change your MenuItem class to fulfill these requirements.

If the requirements are not fulfilled, the you should throw an ArgumentException, and it would be nice to supply some text to the exception, why it has been thrown e.g. like

```
new ArgumentException("price cannot be negative");
```

Which type of exceptions in the API exists? Find at least three different exceptions.

Assignment 3: Modify your Console Application

In your Console Application try to modify the name and the price properties, that violate the constraints e.g. `obj.Price = -35` or `obj.Name = "S"`.

Insert in your application that you catch the exception and print out the exception message.

What happened if you create a new object e.g. Drink with a negative price?

Assignment 4: Refactor the Dish class to use enum

Refactor your Dish class, whereby the TypeOfDish no longer is a string, but an enum with the values (Starter, Anti-Pasta, Main, Cheese, Dessert).

Assignment 5: Implement your own exception

Instead of using one of the ‘build in’ exceptions in the API, you could create your own exception.

Design and implement a ‘MenuException’, and use this instead of the ArgumentException.

When is it a good idea to create your own exception class?

Assignment 6: Add a class Menu

Make a class Menu, which contain exact two starters, two Main courses and two dessert, and the whole menu should have a special price.

Consider how to and implement the ToString.

