# Упражнение 3 – Капсулация
# Валидация на данните

## Problem 1. Проверка на данните

Разширяваме класа Person с подходяща валидация за всяко поле:

- Имената трябва да са поне 3 символа
- възрастта не трябва да е нула или отрицателно число
- заплатата не може да бъде по-малка от 460.0

Print proper message to end user (look at example for messages).

Use ArgumentExeption with messages from example.

Изведете подходящо съобщение за последен потребител (виж примера за съобщения).

Използвайте ArgumentExeption със съобщения от примера.

### Примери

| Вход | Изход |
|---|---|
| 5<br>Asen Ivanov -6 2200<br>B Borisov 57 3333<br>Ventsislav Ivanov 27 600<br>Asen H 44 666.66<br>Boiko Angelov 35 300<br>20 | Age cannot be zero or negative integer<br>First name cannot be less than 3 symbols<br>Last name cannot be less than 3 symbols<br>Salary cannot be less than 460 leva<br>Ventsislav Ivanov get 660.0 leva |

### Решение

Добавете проверка към всички setters на Person. Валидацията може да изглежда като това или нещо подобно:

```
public double Salary
{
    get { return this.salary; }
    set
    {
        if (value < 460)
        {
            throw new ArgumentException("Salary cannot be less than 460 leva");
        }

        this.salary = value;
    }
}
```

## Problem 2. Валидация на данните на класа Box

Всеки от ръбовете на правоъгълния паралелепипед трябва да е неотрицателно число. Разширете класа от предишната задача чрез добавяне на проверка на данните за всеки параметър, даден на конструктора. Направете частен setter, който извършва проверка на данните вътрешно.

## Примери

| Вход | Изход |
|---|---|
| 2<br>-3<br>4 | 3<br>Width cannot be zero or negative. |

# Problem 3. На пазар

Създайте два класа: клас Person и клас Product. Всеки човек трябва да има име, пари и една торба с продукти. Всеки продукт трябва да има име и стойност. Името не може да бъде празен низ. Парите не може да бъдат отрицателно число.

Създайте програма, в която всяка команда отговаря на закупуване на продукт от един обект Person (Човек). Ако човек може да си позволи продукт го добавя към чантата си. Ако човек не разполага с достатъчно пари, изведете подходящо съобщение (("[Person name] can't afford [Product name]").

На първите два реда са дадени всички хора и всички продукти. След всички покупки да се изведат за всеки човек по реда на въвеждане всички продукти, които той е купил, също в реда на въвеждане на покупките. Ако нищо не е купил, да се изведе името на човека, последвано от "**Nothing bought**".

При въвеждане на невалидни (отрицателна сума пари да се създаде изключение със съобщение: "**Money cannot be negative**") или празно име (празно име с изключение със съобщение : "**Name cannot be empty**") за край на програмата с подходящо съобщение. Вижте примерите по-долу:

## Примери

| Вход | Изход |
|---|---|
| Pesho=11;Gosho=4<br>Bread=10;Milk=2;<br>Pesho Bread<br>Gosho Milk<br>Gosho Milk<br>Pesho Milk<br>END | Pesho bought Bread<br>Gosho bought Milk<br>Gosho bought Milk<br>Pesho can't afford Milk<br>Pesho - Bread<br>Gosho - Milk, Milk |
| Mimi=0<br>Kafence=2<br>Mimi Kafence<br>END | Mimi can't afford Kafence<br>Mimi – Nothing bought |
| Jeko=-3<br>Chushki=1;<br>Jeko Chushki<br>END | Money cannot be negative |

# Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист".



- Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под **свободен лиценз CC-BY-NC-SA** (Creative Commons Attribution-Non-Commercial-Share-Alike 4.0 International).