Упражнения: Капсулация на данни

1. Сортиране на хора по име и възраст

Създайте class **Person**, който да има **private** полета:

- firstName: string
- lastName: string
- age: int

И предефиниран **public** метод:

- ToString(): string - override

Трябва да може да се ползва класа по следния начин, например:

StartUp.cs

```
public static void Main()
{
    var lines = int.Parse(Console.ReadLine());
    var persons = new List<Person>();
    for (int i = 0; i < lines; i++)
    {
        var cmdArgs = Console.ReadLine().Split();
        var person = new Person(cmdArgs[0], cmdArgs[1], int.Parse(cmdArgs[2]));
        persons.Add(person);
    }

    persons.OrderBy(p => p.FirstName)
           .ThenBy(p => p.Age)
           .ToList()
           .ForEach(p => Console.WriteLine(p.ToString()));
}
```

Примери

Input	Output
5 Asen Ivanov 65 Boiko Borisov 57 Ventsislav Ivanov 27 Asen Harizanoov 44 Boiko Angelov 35	Asen Harizanoov is a 44 years old Asen Ivanov is a 65 years old Boiko Angelov is a 35 years old Boiko Borisov is a 57 years old Ventsislav Ivanov is a 27 years old

Решение

Създаваме **нов клас** с подходящо име. Правим **private** полета

```
private string firstName;
private string lastName;
private int age;
```

Създаваме конструктор за Person, който приема 3 параметъра firstName, lastName, age.

```
public Person(string firstName, string lastName, int age)
{
    this.firstName = firstName;
    this.lastName = lastName;
    this.age = age;
}
```

Създаваме свойства (properties) за всяко поле, строго съответстващи на полетата:

```
public string FirstName
{
    get { return this.firstName; }
}

public int Age
{
    get { return this.age; }
}
```

Предефинирайте **`ToString()`** метода:

```
public override string ToString()
{
    return $"{this.firstName} {this.lastName} is a {this.age} years old";
}
```

2. Клас Box (правоъгълен паралелепипед)

Дадена е геометричната фигура box с параметри дължина, широчина и височина. Направете клас Box, от който да се създаде обект по тези параметри. Дайте на външния свят само методите за лице на повърхнина, околна повърхнина и обем (Формулите ги има на адрес: http://www.mathwords.com/r/rectangular_parallelepiped.htm).

- На първите три реда ще получите дължина, ширина и височина.
- На следващите три реда се извеждат повърхнината, околната повърхнина и обема на паралелепипеда.

Забележка

Добавете следващия код в началото на метода main.

```
static void Main(string[] args)

{

    Type boxType = typeof(Box);

    FieldInfo[] fields = boxType.GetFields(BindingFlags.NonPublic | BindingFlags.Instance);
```

```
    Console.WriteLine(fields.Count());

}
```

Ако сте дефинирали коректно класа, ще получите очаквания изход.

Примери

Вход	Изход
2 3 4	3 Surface Area – 52.00 Lateral Surface Area – 40.00 Volume – 24.00
1.3 1 6	3 Surface Area - 30.20 Lateral Surface Area - 27.60 Volume - 7.80

3. Увеличение на заплатата

Преструктурирайте (рефактурирайте) проекта с клас Person.

Въвеждаме хора (Person) с техните имена, възраст и заплата. Въвеждаме процент бонус към заплатата на всеки обект person. Обектите Persons, на възраст под 30 получават бонус наполовина.

Разширяваме **Person** от предишната задача. Нови **полета** и **методи:**

- salary: double
- IncreaseSalary(double bonus)

Използвайте класа например така:

StartUp.cs

```
var lines = int.Parse(Console.ReadLine());
var persons = new List<Person>();
for (int i = 0; i < lines; i++)
{
    var cmdArgs = Console.ReadLine().Split();
    var person = new Person(cmdArgs[0],
                            cmdArgs[1],
                            int.Parse(cmdArgs[2]),
                            double.Parse(cmdArgs[3]));

    persons.Add(person);
}
var bonus = double.Parse(Console.ReadLine());
persons.ForEach(p => Console.WriteLine(p.ToString()));
```

Примери

Input	Output
5 Asen Ivanov 65 2200 Boiko Borisov 57 3333 Ventsislav Ivanov 27 600 Asen Harizanoov 44 666.66 Boiko Angelov 35 559.4 20	Asen Ivanov get 2640.00 leva Boiko Borisov get 3999.60 leva Ventsislav Ivanov get 660.00 leva Asen Harizanoov get 799.99 leva Boiko Angelov get 671.28 leva

Решение

Добавяме ново **private** поле за заплата **salary** и променяме конструктора. Добавяме нов **метод**, който ще променя заплатата с бонус

```
public void IncreaseSalary(double percent)
{
    if (this.age > 30)
    {
        this.salary += this.salary * percent / 100;
    }
    else
    {
        this.salary += this.salary * percent / 200;
    }
}
```

Променяме метода **`toString()`** за тази задача.

4. 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

Изведете подходящо съобщение за крайния потребител (виж примера за съобщения). Използвайте ArgumentExeption и в конструктора му подавайте съответните съобщения от примера.

Примери

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

20	

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;
    }
}
```

5. 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 -3 4	3 Width cannot be zero or negative.

6. * 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 Bread=10;Milk=2; Pesho Bread Gosho Milk Gosho Milk Pesho Milk END	Pesho bought Bread Gosho bought Milk Gosho bought Milk Pesho can't afford Milk Pesho - Bread Gosho - Milk, Milk
Mimi=0 Kafence=2 Mimi Kafence END	Mimi can't afford Kafence Mimi – Nothing bought
Jeko=-3 Chushki=1; Jeko Chushki END	Money cannot be negative