# Lab: Encapsulation

Problems for exercises and homework for the ["C# OOP Basics" course @ SoftUni](https://softuni.bg/trainings/1636/c-sharp-oop-basics-june-2017)".

You can check your solutions here: <https://judge.softuni.bg/Contests/Practice/Index/678#2>

## Sort Persons by Name and Age

Create a class **Person**, which should have **private** fields for:

* firstName: string
* lastName: string
* age: int
* ToString(): string - override

You should be able to use the class like this:

|  |
| --- |
| 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()));  } |

### Examples

|  |  |
| --- | --- |
| **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 |

### Solution

Create a **new class** and ensure **proper naming**. Define the **private** fields



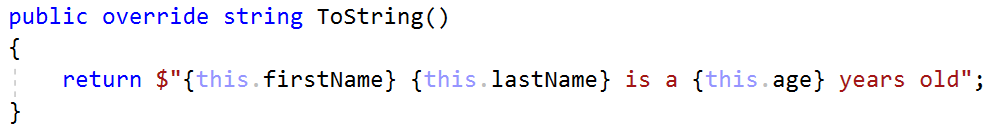
Create a constructor for Person, which takes 3 parameters firstName, lastName, age.



Create properties for these fields, which are as strictly as possible



Override **ToString()** method:



## Salary Increase

**Refactor project from last task.**

Read person with their names, age and salary. Read percent bonus to every person salary. Persons younger than 30 get half bonus. Expand **Person** from previous task. New **fields** and **methods:**

* salary: double
* IncreaseSalary(double bonus)

You should be able to use the class like this:

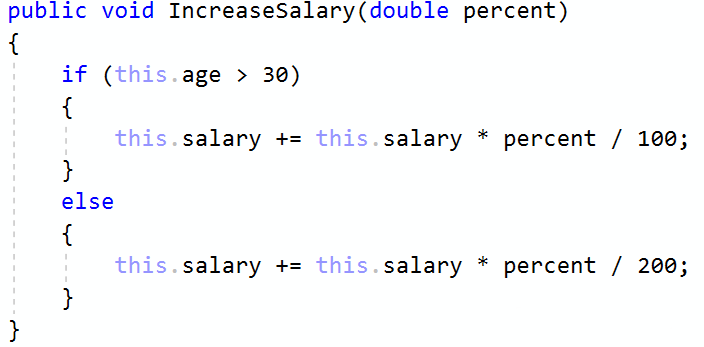
|  |
| --- |
| StartUp.cs |
| var lines = int.Parse(Console.ReadLine());  var persons = new List<Person>();  for (int i = 0; i < lines; i++)  {  var cmdArgs = Console.ReadLine().Split();  var person = new Person(cmdArgs[0],  cmdArgs[1],  int.Parse(cmdArgs[2]),  double.Parse(cmdArgs[3]));  persons.Add(person);  }  var bonus = double.Parse(Console.ReadLine());  persons.ForEach(p => Console.WriteLine(p.ToString())); |

### Examples

|  |  |
| --- | --- |
| **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 |

### Solution

Add new **private** field for **salary** and **refactor constructor**. Add new **method**, which will **update** salary with bonus



Refactor **toString()** method for this task.

## Validation Data

Expand Person with proper validation for every field:

* **Names must be at least 3 symbols**
* **Age must not be zero or negative**
* **Salary can't be less than 460.0**

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

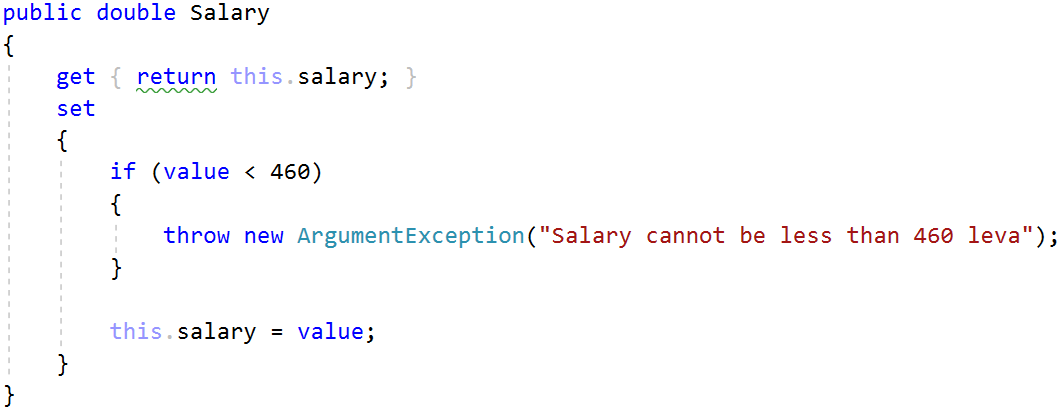
Use ArgumentExeption with messages from example.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5  Asen Ivanov -6 2200  B Borisov 57 3333  Ventsislav Ivanov 27 600  Asen H 44 666.66  Boiko Angelov 35 300  20 | 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 |

### Solution

Add validation to all setters in Person. Validation may look like this or something similar:



## First and Reserve Team

Create a Team class. Add to this team all person you read. All person younger than 40 go in first team, others go in reverse team. At the end print first and reserve team sizes.

The class should have **private fields** for:

* name: string
* firstTeam: List<Person>
* reserveTeam: List<Person>

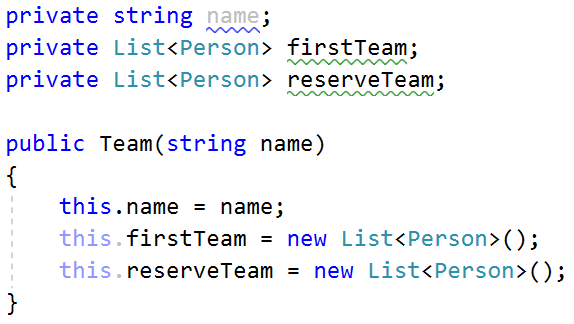
The class should have **constructors**:

* Team(string name)

The class should also have **public methods** for:

* AddPlayer(Person person): void
* FirstTeam: IReadOnlyCollection
* ReserveTeam: IReadOnlyCollection

You should be able to use the class like this:



You should **NOT** be able to use the class like this:

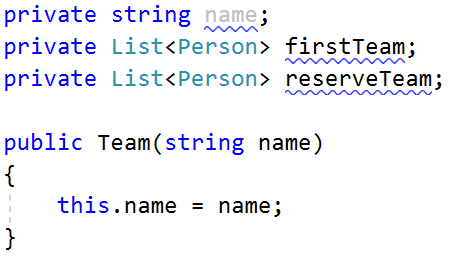


### Examples

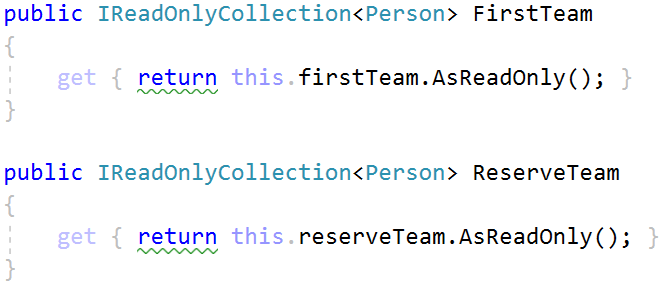
|  |  |
| --- | --- |
| **Input** | **Output** |
| 5  Asen Ivanov 20 2200  Boiko Borisov 57 3333  Ventsislav Ivanov 27 600  Grigor Dimitrov 25 666.66  Boiko Angelov 35 555 | First team have 4 players  Reserve team have 1 players |

### Solution

Add new class Team. Its fields and constructor look like:



Properties for FirstTeam and ReserveTeam have only getters:



There will be only one method, which add players to teams:

