## #1 Person

Define a class **Person** with **public** fields for **name** and **age**.

## #2 Constructors

Modify the **Person** class, to have 3 new constructors, a default one with no arguments, which **sets name to “No name” and age to 1**, and two other which accept the person’s name, and his name and age(**should not be negative**). In order to reuse code, chain the constructors. After the name and age is entered, initialize 3 Person objects, each one with different constructors.

The program should print 3 rows with the 3 objects, the first one with the first default constructor, the second one with the name and the third one with the name and age.

Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| Georgi  20 | No name 1  Georgi 1  Georgi 20 |
| Gosho  18 | No name 1  Gosho 1  Gosho 18 |
| Ivan  43 | No name 1  Ivan 1  Ivan 43 |

## #4 People

Write a program that uses the defined **Person** class and accepts people on every line, and in the end `quit` is entered, and the program **prints** every person`s **name** and age, if he’s **older than 18**, sorted by the **length** of their **name**. To sort the people you can use **LINQ** or a custom **Comparer<Person>**.  
People are entered on every line, in the format **person’s name**//**age**

|  |  |
| --- | --- |
| **Input** | **Output** |
| Georgi//20  Todor//10  Ivan//99 | Ivan 99  Georgi 20 |
| Gosho//3  Jack Ivanov//101 | Jack Ivanov 101 |
| Stamat//33  Anton//60  Nikolay//50 | Anton 60  Stamat 33  Nikolay 50 |

## #5 Course and Student objects

Write a program to manage courses and students signed up for them. Each **Course** should have the following properties, a capacity of students that it can have, all the students that are signed up for the course, a name, duration in hours per day, and an **auto-incrementing** **unique** identifier starting from **0**. The course should also be able to add or remove students, check if a student(s) exists by a given name, and when converted to a string, the course should return it`s name.

Each **student** should have a full name, age, an **auto-incrementing** **unique** identifier starting from **0**. You should **inherit** the Person object, from the first task. Every student can sign up for only **one course**! If somebody tries to sign up for 2 courses, you should throw an exception which prints “Student is already signed up”. Also every student should print his name when converted to a string.

You should also create an Academy object, which holds all the courses and students and handles signups.

The program should start with the user entering the **number** of courses, the academy has. After that all the courses are entered, on a separate line, in this format **courseName**//**duration**//**capacity**

After that the program will read a number of students that will be created. After that, on each line the user will enter a new **Student** in this format **name**//**age**

When all the courses and users are created, the program continues with users signing up for courses, until `**quit**` is entered. Students will sign up to the courses, by using this syntax **studentID** **courseID.** If a student or course does not exist, throw an exception with the message “Student does not exist”, or “Course does not exist”, and handle it so that it prints “Error: ” + message, on the console. Also if the course`s capacity is already reached, you should have an exception printing “Course #nameOfCourse is already full!”.

After you enter `**quit**`, the program should print out all courses, sorted by name, and all the students assigned to the courses, sorted by age.

Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  C#//8//30  F#//4//20  4  Ivan//10  Todor//20  Georgi//30  Nikolay//50  1 1  2 0  0 1  3 0  quit | C# - 8 hours  ##Georgi  ##Nikolay  F# - 4 hours  ##Ivan  ##Todor |