# Упражнения: Изключения

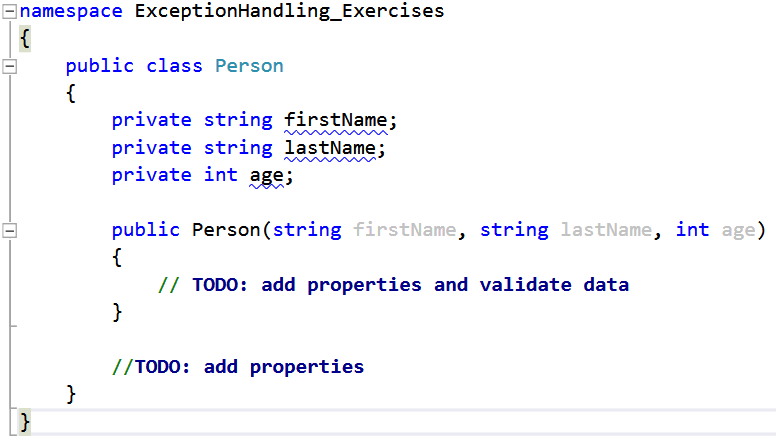
## Валидна личност

Дефинирайте прост клас Person, притежаващ следните полета: **first name**, **last name** и **age**. **Валидирайте** данните в setter-ите на свойствата и **хвърлете** подходящи **изключения** ако са въведени невалидни данни.

### Стъпка 1. Създайте клас Person

Създайте проект за това упражнение и добавете клас Person в отделен .cs файл. Класът трябва да съдържа следните полета: **first name (string), last name (string) and age (int)**.

Всички полета са **задължителни**, което означава, че ви трябва един конструктор, който приема и трите като параметри. Например:



### Стъпка 2. Добавете свойства и валидирайте данните

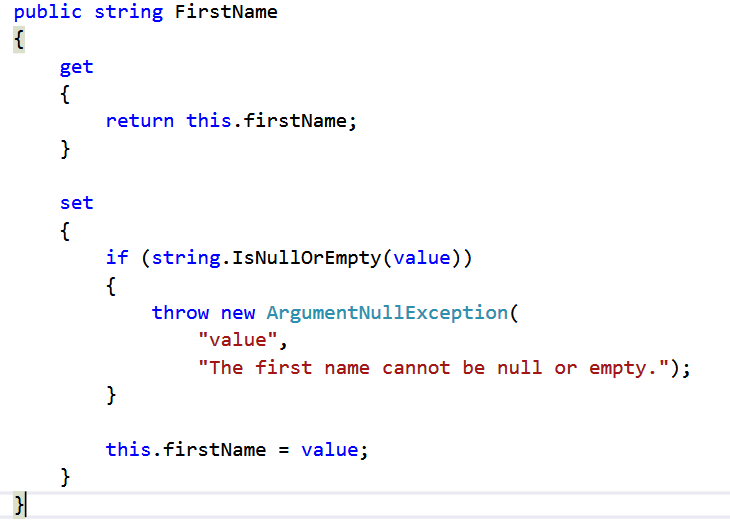
Добавете свойство за всяко от полетата. Извършете валидиране в техните **setter**-и, за да запазите състоянието на обекта Person валидно.

Свойствата **first** и **last name** не може да бъдат **null** или **празни** низове. За да проверите това, използвайте метода **string.IsNullOrEmpty()**.

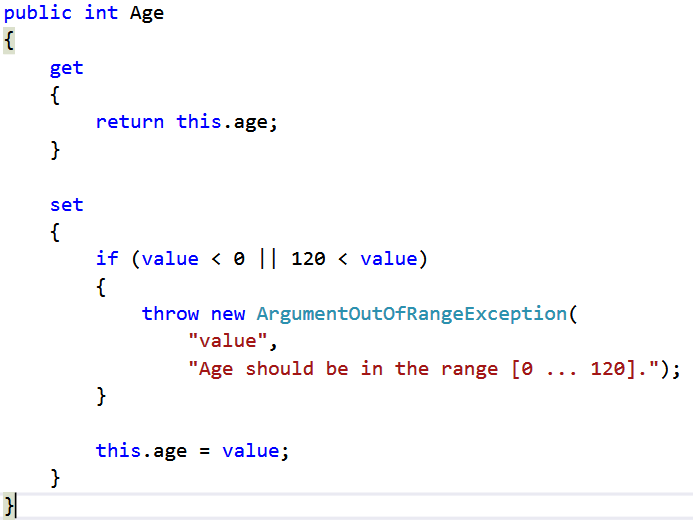
Свойството **age** трябва да е в диапазона **[0 … 120]**.

Ако са въведени невалидни данни, **хвърлете** подходящи изключения с достатъчно описателни **съобщения**. Например ако бъде въведено празно име, едно подходящо изключение би било **ArgumentNullException**. Ако възрастта е отрицателна или твърде голяма, подходящото изключение ще е **ArgumentOutOfRangeException**.

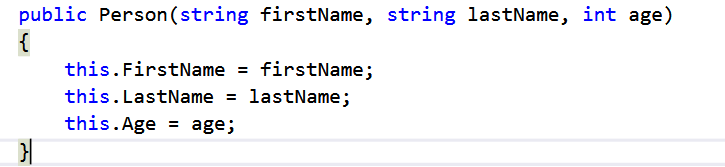
Пример за валидирането на **first name** (last name е аналогично):



Пример за валидирането на **age**:

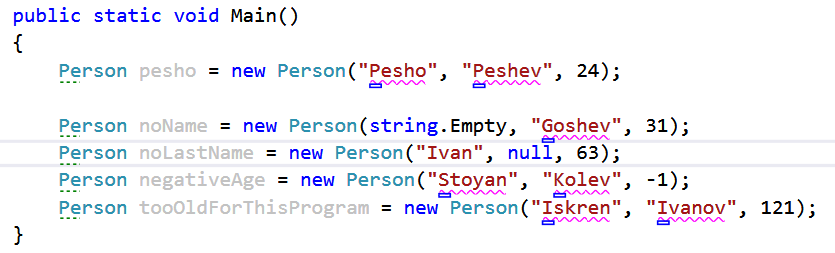


Сега вече конструкторът би трябвало да използва свойствата вместо да модифицира директно частните полета:



### Стъпка 3. Тествайте класът Person

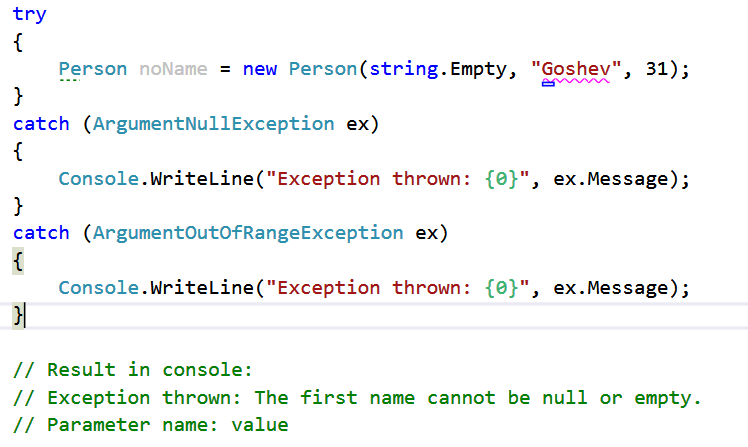
Във вашата главна програма, проверете дали вашият клас се държи както се очаква. Създайте няколко обекта от тип Person – един с **валидни данни**, един с **празно first name**, един с **null за last name**, един с **отрицателна възраст** и един с **age > 120**. Проверете дали изпълнението на кода води до грешки, когато се въведат невалидни данни. Тествайте невалидните случаи един по един чрез коментиране на редовете на другите невалидни случаи (защото вашата програма ще спре изпълнението си още при първата грешка).



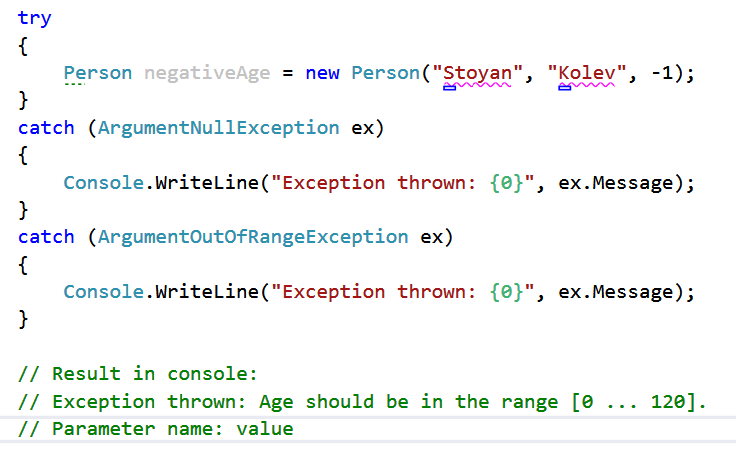
### Стъпка 4. Добавете Try-Catch блокове

За да предотвратите сриването на програмата, обградете невалидните редове в **try-catch** блокове. Добра практика е да поставите различни catch блокове за различните типове грешки, които очаквате, че командите може да предизвикат. Отпечатайте **съобщението** на изключението от catch блока.

Пример (невалидно **име**):



Пример (нeвалидна **възраст**):



## Корен квадратен

Напишете програма, която чете цяло число и отпечатва и извежда неговия корен квадратен. Ако числото е невалидно или отрицателно, отпечатайте текста "Invalid number". Във всички случаи накрая завършете с "Good bye". Използвайте конструкцията **try-catch-finally**.

## Въведете числа

Напишете метод **ReadNumber(int start, int end),** който позволява въвеждане от клавиатурата на цяло число в диапазона [start…end]. Ако се въведе неподходящо число или текст вместо число, методът трябва да хвърля изключение. Като ползвате този метод напишете програма, която въвежда 10 числа: **a1, a2, … a10, такива, че 1 < a1 < … < a10 < 100**. Ако човек въведе невалидно число, нека го въведе отново.