# Упражнения: Конструктори

## Дефиниране на класа Човек

Създайте клас **Person** (или използвайте вече създадените класове от предните уроци).

Класът трябва да има **private полета** за име, възраст и банкови сметки:

* Name: **string**
* Age: **int**
* Accounts: List<BankAccount>

Класът трябва да има и следните **конструктори**:

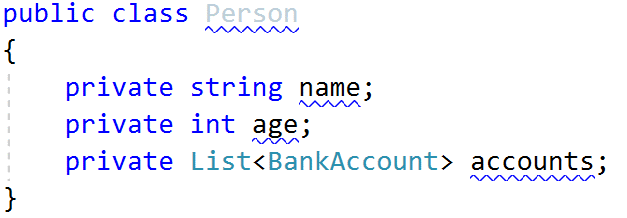
* Person(string name, int age)
* Person(string name, int age, List<BankAccount> accounts)

Класът трябва да има и **public метод** за:

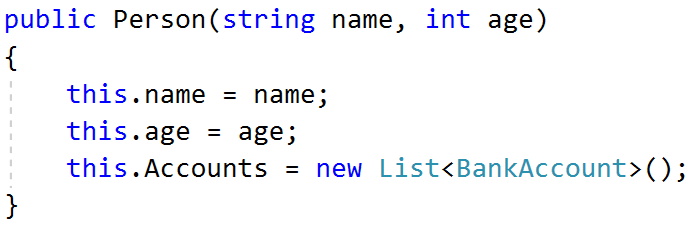
* GetBalance(): double

### Решение

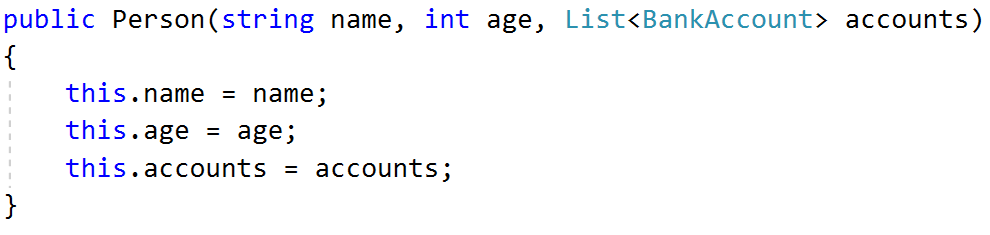
Създайте класа както обикновено:



Създайте конструктор с два параметъра:



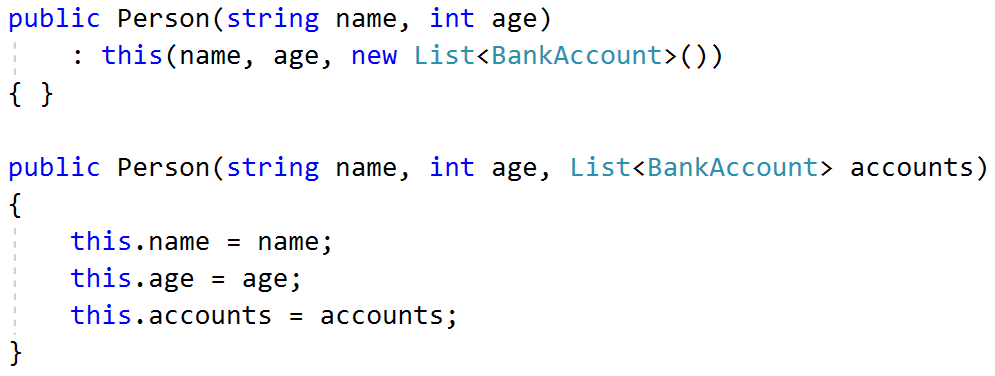
Създайте конструктор с три параметъра:



Създайте метода GetBalance()



***По желание:*** Можете да се възползвате от **верижното извикване на конструктори:**



## Конструктори за класа Човек

Добавете 2 конструктора към класа **Person** от миналата задача и с помощта на верижно извикване на кода използвайте повторно съществуващ вече програмен код:

1. Първият конструктор трябва да е без параметри и да създава човек с име “**No name**” и възраст = **1**.
2. Вторият конструктор трябва да приема само един целочислен параметър за възрастта и да създава човек с име “**No name**” и възраст равна на подадения параметър.

В класа трябва да присъства и конструктор, който приема низ за името и цяло число за възрастта и да създава личност с указаното име и възраст. Добавете следното към main метода и го качете в платформата.

|  |
| --- |
| Type personType = typeof(Person);  ConstructorInfo emptyCtor = personType.GetConstructor(new Type[] { });  ConstructorInfo ageCtor = personType.GetConstructor(new[] { typeof(int) });  ConstructorInfo nameAgeCtor = personType.GetConstructor(new[] { typeof(string), typeof(int) });  bool swapped = false;  if (nameAgeCtor == null)  {  nameAgeCtor = personType.GetConstructor(new[] { typeof(int), typeof(string) });  swapped = true;  }  string name = Console.ReadLine();  int age = int.Parse(Console.ReadLine());  Person basePerson = (Person)emptyCtor.Invoke(new object[] { });  Person personWithAge = (Person)ageCtor.Invoke(new object[] { age });  Person personWithAgeAndName = swapped ? (Person)nameAgeCtor.Invoke(new object[] { age, name }) :(Person)nameAgeCtor.Invoke(new object[] { name, age });  Console.WriteLine("{0} {1}", basePerson.name, basePerson.age);  Console.WriteLine("{0} {1}", personWithAge.name, personWithAge.age);  Console.WriteLine("{0} {1}", personWithAgeAndName.name, personWithAgeAndName.age); |

Ако сте дефинирали конструкторите коректно, тестът би трябвало да премине.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| Pesho  20 | No name 1  No name 20  Pesho 20 |
| Gosho  18 | No name 1  No name 18  Gosho 18 |
| Stamat  43 | No name 1  No name 43  Stamat 43 |

## Сурови данни

Вие сте собственик на куриерска компания и искате да направите система за проследяване на вашите коли и техния товар. Дефинирайте клас **Car** с информация за **модела, двигателя, товара** и **колекция от точно 4 гуми**. Моделът, товарът и гумите трябва да са **отделни класове**; създайте конструктор, който получава пълната информация за колата и създава и инициализира нейните вътрешни компоненти (двигател, товар и гуми).

На първия ред от входната информация ще получите число **N** - броя на колите, които имате, а на всеки от следващите **N** реда ще има информация за кола във формата “**<Модел> <СкоростНаДвигателя> <МощностнаДвигателя> <ТеглоНаТовара> <ТипНаТовара> <Гума1Налягане> <Гума1Възраст> <Гума2Налягане> <Гума2Възраст> <Гума3Налягане> <Гума3Възраст> <Гума4Налягане> <Гума4Възраст>”** където скорост, мощност, тегло на товара и възраст на гумите са **цели числа**, а налягането е дробно число, **с двойна точност.**

След тези **N** реда ще получите един-единствен ред с една от следните две команди: “**fragile**” или “**flamable**”. Ако командата е “**fragile**”, то отпечатайте всички коли с **тип на товара “fragile”** с гуми с **налягане** **< 1**; ако командата е “**flamable**”, отпечатайте всички коли с **тип на товара “flamable”** и **мощност на двигателя > 250**. Колите трябва да се изведат в реда, в който са подадени като входни данни.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 2  ChevroletAstro 200 180 1000 fragile 1.3 1 1.5 2 1.4 2 1.7 4  Citroen2CV 190 165 1200 fragile 0.9 3 0.85 2 0.95 2 1.1 1  fragile | Citroen2CV |
| 4  ChevroletExpress 215 255 1200 flamable 2.5 1 2.4 2 2.7 1 2.8 1  ChevroletAstro 210 230 1000 flamable 2 1 1.9 2 1.7 3 2.1 1  DaciaDokker 230 275 1400 flamable 2.2 1 2.3 1 2.4 1 2 1  Citroen2CV 190 165 1200 fragile 0.8 3 0.85 2 0.7 5 0.95 2  flamable | ChevroletExpress  DaciaDokker |

## 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).

