# Упражнения: Наследяване

## Единично наследяване

Създайте два класа Animal и Dog.

Animal с единствен публичен метод Eat(), който отпечатва: **"eating…"**

Dog с единствен публичен метод Bark(), който отпечатва: **"barking…"**

Dog трябва да наследи Animal.



#### Подсказка

Използвайте **: оператора**, за да построите йерархията

## Наследяване на много нива

Създайте три класа Animal, Dog и Puppy.

Animal с единствен публичен метод Eat(), който отпечатва: **"eating…"**

Dog с единствен публичен метод Bark(), който отпечатва: **"barking…"**

Puppy с единствен публичен метод Weep(), който отпечатва: **"****weeping…"**

Dog трябва да наследи Animal. Puppy трябва да наследи Dog.



## Йерархично наследяване

Създайте три класа Animal, Dog и Cat.

Animal с единствен публичен метод Eat(), който отпечатва: **"eating…"**

Dog с единствен публичен метод Bark(), който отпечатва: **"barking…"**

Cat с единствен публичен метод Meow(), който отпечатва: **"meowing…"**

Dog и Cat трябва да наследят Animal.



## \* Повторно използване на класове

Създайте клас Person с публични свойства **Name** и **Age** и единствен публичен метод Sleep(), който отпечатва името на човека и **" is sleeping … years.",** като на мястото на точките се извеждат годините на човека. Добавете подходящ конструктор, задаващ начални стойности на свойствата на обекта.

#### 4.1.Наследяване

Създайте клас **Employee**, наследник на **Person**, с публично свойство **Organization** (текст) и единствен публичен метод **Work()**, който отпечатва името на човека **" is woкing in "** и името на организацията, в която работи. Добавете подходящ конструктор за инициализиране на обекта.

**4.2.Композиция**

Добавете в класа **Employee** **private** свойство **Account** от тип **BankAccount**, което ще съдържа информация за банковата сметка с парите на служителя. Класът **BankAccount** е познат от предишни задачи. Има следните **public** свойства и методи:

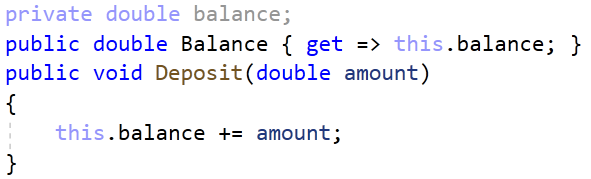
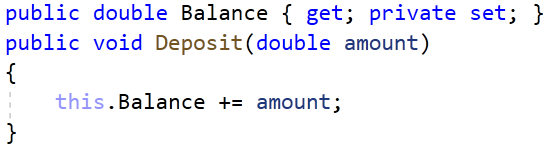
* **ID:** **int**
* **Balance:** **double** (да не може да се указва директно стойност!)
* **Deposit(Double amount):** **void**
* **Withdraw(Double amount):** **void**

Погрижете се в конструктора (или конструкторите) на класа **Employee** свойството **Account** да получи подходяща инициализация.

Добавете в класа **Employee** и публичен метод **Introduce()**, който извежда името на човека, на колко години е, къде работи и колко пари има в банковата си сметка.

##### Подсказка:

Изискването да не може някой да запише директно произволна сума по сметката, а да трябва да ползва методите **Deposit** и **Withdraw** може да бъде изпълнено по два начина в дефиницията на класа **BankAccount**:

* Като има **private** поле **balance**, което се променя от двата метода **Deposit** и **Withdraw**, а публичното свойството **Balance** да е само с getter:  
  
* Или свойството **Balance** може да има и setter, но той да е private и двата метода **Deposit** и **Withdraw** да променят директно свойството Balance (така може да си спестим изричната дефиниция на поле):  
  

**4.3.Делегиране**

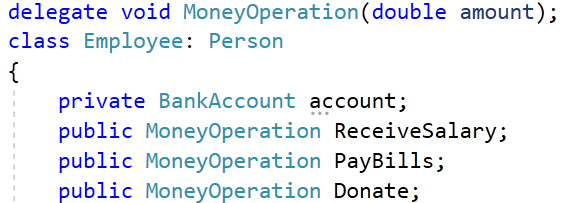
Добавете към класа **Employee** трипублични **делегирани** метода**:**

* **ReceiveSalary(Double amount):** **void**
* **PayBills(Double amount):** **void**
* **Donate(Double amount):** **void**

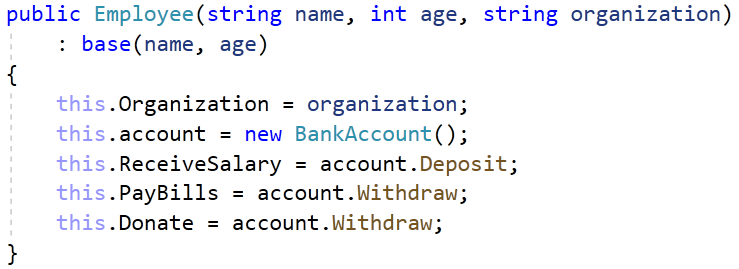
Те трябва да викат съответните методи на банковата сметка на служителя: първият - метода **Deposit** (така че да се внесат получените от заплатата пари по сметката) а другите два - метода **Withdraw** (така че да намали сумата там с указаното в параметъра).

##### Подсказка:

За да дефинирате трите делегирани метода, използвайте следния примерен код:



Не забравяйте да укажете в конструктора на **Employee** кой ще отговаря за изпълнението им:



**4.4.Използване на класовете**

С помощта на така създадения клас **Employee** демонстрирайте как служител от някаква организация работи, получава заплата, отива на ресторант с очарователна дама и ѝ се представя, накрая на вечерта плаща сметките, дарява някаква сума за благотворителност и ляга да спи:

