# Лаб: Дефиниране на класове

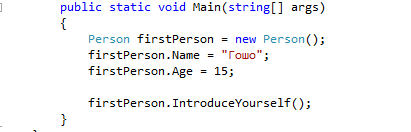
# Part I: Дефиниране на класове

## Дефиниране на клас Person

Създайте клас Person.

Класът трябва да има:

* name: String - поле
* age: int - поле
* Name: String - свойство
* Age: int - свойство

Използвайте класа в Main по аналогичен начин:  


### Решение

Разгледайте решението на задачата в презентацията към лаба.

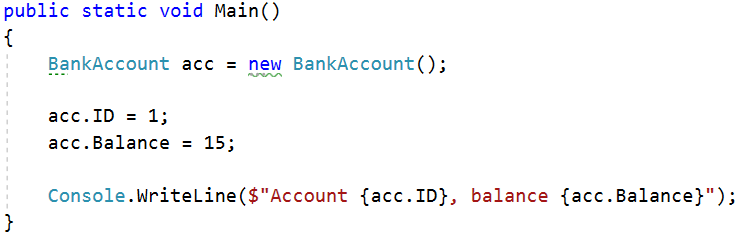
## Дефиниране на клас BankAccount (банкова сметка)

Създайте клас BankAccount.

Класът трябва да има:

* id: int
* balance: double
* ID: int
* Balance: double

Трябва да може да използвате класа по аналогичен начин:



### Решение

Решението на тази задача е аналогично на решението на предната

## Методи

Надградете класа BankAccount

Този клас трябва да има полета за:

* id: int
* balance: double

Класът трябва да има свойства за:

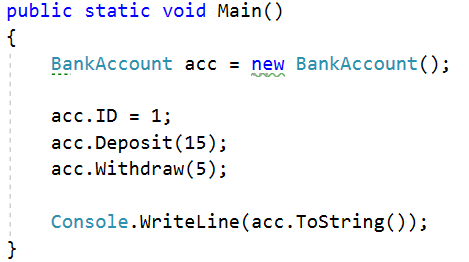
* ID: int
* Balance: double

Създайте методите:

* Deposit(Double amount): void – който да вкарва пари в сметката
* Withdraw(Double amount): void – който да изтегля пари от сметката

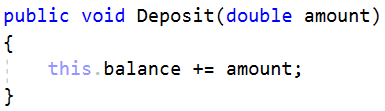
Заменете метода ToString()

Трябва да можете да използвате класа по аналогичен начин:

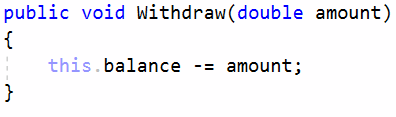


### Решение

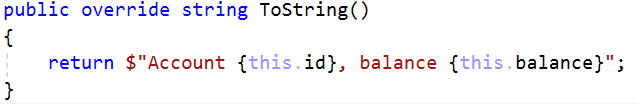
Създайте метод Deposit(double amount)



Създайте метод Withdraw(double amount)



Заменете метода ToString()



## Тестов Клиент

Създайте тестов клиент, който използва BankAccount.

Трябва да поддържате следните операции:

* **Create {Id}**
* **Deposit {Id} {Amount}**
* **Withdraw {Id} {Amount}**
* **Print {Id}**
* **End**

Ако се опитате да създадете сметка със съществуващо Id, изведете **"Account already exists".**

Ако се опитате да извършите операция върху несъществуваща сметка, изведете **"****Account does not exist"**.

Ако се опитате да изтеглите сума, която е по-голяма от баланса, изведете **"****Insufficient balance"**.

Print командата, трябва да изведе **"Account ID{id}, balance {balance}"**. Закръглете баланса до втория знак след запетаята.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| Create 1  Create 1  Deposit 1 20  Withdraw 1 30  Withdraw 1 10  Print 1  End | Account already exists  Insufficient balance  Account ID1, balance 10.00 |
| Create 1  Deposit 2 20  Withdraw 2 30  Print 2  End | Account does not exist  Account does not exist  Account does not exist |

### Решение

Използвайте Dictionary<int, BankAccount> за да пазите сметките

Направете си цикъла за приемане на команда



Създайте методи към Program.cs, за всяка от командите.

Create – проверявате дали в речника има ключ с такова id – ако няма, създавате сметката.



Имплементирайте останалите команди работейки с подобна логика

## Човекът и неговите пари

Създайте клас **Person**.

Той трябва да има полета за:

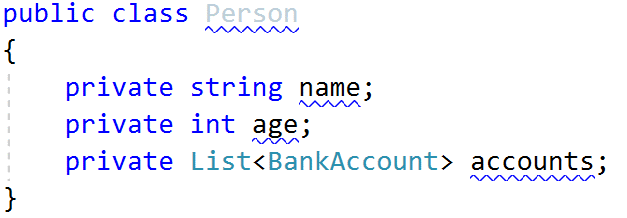
* Name: **string**
* Age: **int**
* Accounts: List<BankAccount>

Класът трябва да има метод, който изчислява всички пари, които притежава човека от сметките си:

* GetBalance(): double

### Решение

Използвайте по-горния клас и му добавете възможност за пазене на списък от банкови сметки



Създайте метод GetBalance()

