# Lab: Defining Classes

## Bank Account

**NOTE**: You need a StartUp class with the namespace BankAccount.

Create a **class** named BankAccount.

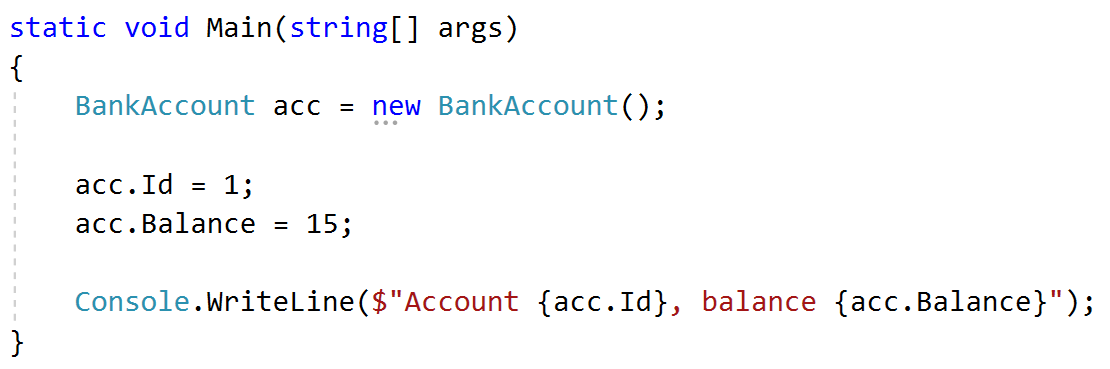
The class should have **private** **fields** for:

* id: int
* balance: decimal

The class should also have **public** **properties** for:

* Id: int
* Balance: decimal

You should be able to use the class like this:



## Bank Account Methods

**NOTE**: You need a StartUp class with the namespace BankAccount.

Create a class BankAccount (you can use the class from the previous task)

The class should have private fields for:

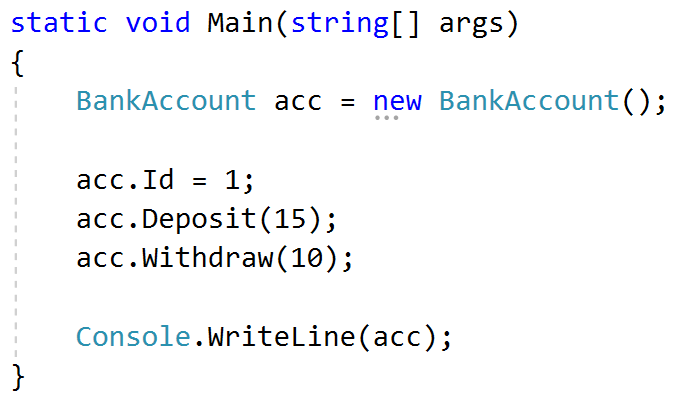
* id: int
* balance: decimal

The class should also have properties for:

* Id: int
* Balance: decimal
* Deposit(decimal amount): void
* Withdraw(decimal amount): void

Override the method ToString().

You should be able to use the class like this:



## Test Client

Create a test client that tests your BankAccount class.

Support the **following commands**:

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

If you try to create an account with an existing Id, print **"Account already exists".**

If you try to perform an operation on a **non-existing account**, print **"****Account does not exist"**.

If you try to withdraw an amount larger than the balance, print **"****Insufficient balance"**.

The Print command should print **"Account ID{id}, balance {balance}"**. Round the balance to the second digit after the decimal separator.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 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 |
| Deposit 2 20  Withdraw 2 30  Print 2  End | Account does not exist  Account does not exist  Account does not exist |

## Person Class

**NOTE**: You need a **StartUp** class with the namespace **BankAccount**.

Create a **Person** class.

The class should have **private fields** for:

* name: **string**
* age: **int**
* accounts: List<BankAccount>

The class should have **constructors**:

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

The class should also have **public methods** for:

* GetBalance(): decimal

## Problem 5. Define a Class Person

Define a class **Person** with **private** fields for **name** and **age** and **public** properties **Name** and **Age**.

### Bonus\*

Try to create a few objects of type Person:

|  |  |
| --- | --- |
| **Name** | **Age** |
| Pesho | 20 |
| Gosho | 18 |
| Stamat | 43 |

Use both the inline initialization and the default constructor.

## Problem 6. Creating Constructors

Add 3 constructors to the **Person** class from the last task, use constructor chaining to reuse code:

1. The first should take no arguments and produce a person with name “**No name**” and age = **1**.
2. The second should accept only an integer number for the age and produce a person with name “**No name**” and age equal to the passed parameter.
3. The third one should accept a string for the name and an integer for the age and should produce a person with the given name and age.

## Problem 7. Oldest Family Member

Use your **Person** **class** from the previous tasks. Create a class **Family**. The class should have **list of people**, a method for adding members (**void AddMember(Person member)**) and a method returning the oldest family member(**Person GetOldestMember())**. Write a program that reads the names and ages of **N** people and **adds them to the family**. Then **print** the **name** and **age** of the oldest member.

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 3  Pesho 3  Gosho 4  Annie 5 | Annie 5 |  | 5  Steve 10  Christopher 15  Annie 4  Ivan 35  Maria 34 | Ivan 35 |

## Problem 8. Opinion Poll

Using the **Person** class, write a program that reads from the console **N** lines of personal information and then prints all people whose **age** is **more than 30** years, **sorted in alphabetical order**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  Pesho 12  Stamat 31  Ivan 48 | Ivan - 48  Stamat - 31 |
| 5  Nikolai 33  Yordan 88  Tosho 22  Lyubo 44  Stanislav 11 | Lyubo - 44  Nikolai - 33  Yordan - 88 |

## Problem 9. Date Modifier

Create a class **DateModifier** which stores the difference of the days between two dates. It should have a method which takes **two string parameters** **representing a date** as strings and **calculates the** difference in the days between them.