

## Herança - Exercício

Escrever um programa onde devemos usar os conceitos de abstração, encapsulamento e herança para expressar o comportamento de contas bancárias considerando o seguinte cenário :

Em um banco temos clientes que possuem contas que são identificadas por um **Número e Nome do cliente**.


As operações que serão permitidas são as operações básicas de **depositar , sacar e exibir o saldo**.

**Devemos considerar 3 tipos de contas:**

- 1- **contas correntes** comuns : São *contas comuns* dos clientes e realizam todas as operações básicas
- 2- **contas de poupança** : São *contas específicas* que se destacam por possuírem juros mensais que são aplicados ao saldo da conta
- 3- **contas de investimento** : São *contas específicas* que se destacam por possuir *uma taxa de remuneração* que é aplicado ao saldo da conta e uma *taxa de imposto* que é descontado do saldo da conta

**Para simplificar o exercício considere as seguintes premissas:**

- a- O saldo inicial das contas é zero;
- b- O *juros da conta de poupança* é **0,5%** e os rendimentos devem ser calculados ao se efetuar um depósito e deve ser acrescido ao saldo da conta;
- c- A taxa da *conta de investimento* é **0,9%** e deve ser aplicada a cada depósito e *acrescida ao saldo*;
- d- O imposto da *conta de investimento* é **0,1%** e deve ser aplicado a cada saque na conta e *debitado do saldo*;
- e- As *contas de poupança e de investimento* não permitem que o saldo da conta fique **negativo** ;



```
using Investment;
using Saving;
using Checking;
```

```
CheckingAccount checkingAccount = new(clientName: "Bruna", number: 15);
```

```
checkingAccount.Deposit(15);
checkingAccount.WithDraw(19);
checkingAccount.ShowBalance();
```

```
// Account number: 15
// Name Bruna
// Your balance: -4
```

```
SavingAccount savingAccount = new(clientName: "Bruna", number: 15);
```


```
savingAccount.WithDraw(2);
savingAccount.ShowBalance();
```

```
// Cannot withdraw, Balance: 0
// Account number: 15
// Name Bruna
// Your balance: 0
```

```
InvestmentAccount investmentAccount = new(clientName: "Bruna", number: 15);
```

```
investmentAccount.Deposit(15);
investmentAccount.WithDraw(10);
investmentAccount.ShowBalance();
```

```
//Account number: 15
//Name Bruna
//Your balance: 3,55
```



```
namespace BaseAccount;

public abstract class Account
{
    private decimal balance = 0;
    public decimal Balance
    {
        get{return balance;}
        set{balance = value;}
    }


    public int Number { get; set; }
    public string? ClientName { get; set; }

    public virtual void Deposit(int value)
    {
        Balance += value;
    }
    public virtual void Withdraw(int value)
    {
        Balance -= value;
    }
    public void ShowBalance()
    {
        System.Console.WriteLine($"Account number: {Number}");
        System.Console.WriteLine($"Name {ClientName}");
        System.Console.WriteLine($"Your balance: {Balance}");
    }
}
```



```
using BaseAccount;  
namespace Checking;
```

```
public class CheckingAccount : Account  
{  
    public CheckingAccount(int number, string clientName)  
    {  
        this.Number = number;  
        this.ClientName = clientName;  
    }  
}
```




```
using BaseAccount;
namespace Saving;

public class SavingAccount : Account
{
    public SavingAccount(int number, string clientName)
    {
        this.Number = number;
        this.ClientName = clientName;
    }

    public override void Deposit(int value)
    {
        decimal interest = value * 0.05m;
        Balance += value - interest;
    }

    public override void Withdraw(int value)
    {
        if (value > Balance)
        {
            System.Console.WriteLine($"Cannot withdraw, Balance: {Balance}");
        }
        else
        {
            Balance -= value;
        }
    }
}
```



```
using BaseAccount;
namespace Investment;

public class InvestmentAccount : Account
{
    public InvestmentAccount(int number, string clientName)
    {
        this.Number = number;
        this.ClientName = clientName;
    }

    public override void Deposit(int value)
    {
        decimal interest = value * 0.09m;
        Balance += value - interest;
    }

    public override void Withdraw(int value)
    {
        if (value > Balance)
        {
            System.Console.WriteLine($"Cannot withdraw, Balance: {Balance}");
        }
        else
        {
            decimal tax = value * 0.01m;
            Balance -= value + tax;
        }
    }
}
```