# LAB 4 – Herança

Este laboratório introduz o processo de derivação de classes em C# via *herança* e sobrescrita de métodos e propriedades de uma *classe abstrata*.

## Exercício 1 – Utilizando Classes Abstratas

1. Crie um novo projeto.
2. Na área do ***Solution Explorer*** clique com o botão direito sobre o projeto e selecione o menu ***Add->New Item***. Na lista ***Templates*** selecione “***Class***”. Digite o nome “Conta.cs”.
3. Altere a definição da classe para torná-la ***pública*** e ***abstrata***:

public abstract class Conta

1. Acrescente os seguintes atributos na definição da classe. Estes atributos representam o *saldo* e o *titular* da conta.

private decimal saldo;

private string titular;

1. Acrescente um *método construtor*:

public Conta(string t)

{

titular = t;

}

1. Acrescente *propriedades* de *leitura*:

public decimal Saldo

{

get { return saldo; }

}

public string Titular

{

get { return titular; }

}

## Exercício 2 – Definindo métodos/propriedades abstratos

1. Adicionar o seguinte código à classe “Conta.cs” para definir uma *propriedade abstrata de leitura* que retorna um identificador da conta. Esta propriedade será implementada pelas *classes derivadas*.

public abstract string Id

{

get;

}

## Exercício 3 – Definindo métodos/propriedades passíveis de sobrescrita

1. Adicionar o seguinte código à classe “Conta.cs” para definir *métodos virtuais* para *depositar* e *retirar* valores.

public virtual void Depositar(decimal valor)

{

saldo += valor;

}

public virtual void Sacar(decimal valor)

{

if (valor <= saldo)

saldo -= valor;

}

## Exercício 4 – Definindo herança a partir de classes abstratas e implementando métodos e propriedades abstratas

1. Adicione uma nova classe “ContaPoupanca.cs” ao projeto. Edite a classe com o código abaixo:

public class ContaPoupanca : Conta

1. Acrescente *atributos* adicionais além daqueles *herdados* da classe “Conta.cs”, para representar a *taxa de juros* e a *data de aniversário* da conta de poupança:

private decimal taxaJuros;

private DateTime dataAniversario;

1. Acrescente um *método construtor*, fazendo referência ao construtor da classe base:

public ContaPoupanca(decimal j, DateTime d, string t)

: base(t)

{

taxaJuros = j;

dataAniversario = d;

}

1. Acrescente propriedades adicionais:

public decimal Juros

{

get { return taxaJuros; }

set { taxaJuros = value; }

}

public DateTime DataAniversario

{

get { return dataAniversario; }

}

1. Acrescente um método adicional para a aplicação do *rendimento* da conta:

public void AdicionarRendimento()

{

if (DateTime.Now.Equals(dataAniversario))

{

decimal rendimento = Saldo \* taxaJuros;

Depositar(rendimento);

}

}

1. Adicione o seguinte código para implementar a *propriedade abstrata* herdada da classe base:

public override string Id

{

get { return Titular + " (CP)"; }

}

## Exercício 5 – Definindo herança a partir de classes abstratas e sobrescrevendo métodos

1. Adicione uma nova classe “ContaCorrente.cs” ao projeto. Edite a classe com o código abaixo:

public class ContaCorrente : Conta

1. Acrescente um atributo adicional constante além daqueles herdados da classe “Conta.cs”, para representar a *taxa fixa* de operações bancárias:

public const decimal Taxa = 0.10M;

1. Acrescente um *método construtor*, fazendo referência ao construtor da classe base:

public ContaCorrente(string t)

: base(t)

{

}

1. Adicione o seguinte código para implementar a *propriedade abstrata* herdada da classe base:

public override string Id

{

get { return Titular + " (CC)"; }

}

1. Adicione o seguinte código para *sobrescrever* os métodos de *depósito* e *saque* herdados da classe base:

public override void Depositar(decimal valor)

{

decimal desconto = valor \* Taxa;

base.Depositar(valor - desconto);

}

public override void Sacar(decimal valor)

{

decimal desconto = valor \* Taxa;

base.Sacar(valor + desconto);

}

1. Compile o projeto.

## Exercícios adicionais

1. Escreva um programa que crie vários tipos de conta e teste cada um dos métodos e propriedades desenvolvidos.