Laboratório 6 – Classes Abstratas

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

1 Descrevendo o processo de criação de uma classe base abstrata

1. Crie um novo “Console Application” com nome “Laboratorio6”.

2. Adicione uma nova classe “Conta.cs”.

3. Altere a definição da classe para torná-la abstrata:

```
public abstract class Conta
```

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

```
private decimal saldo;
private string titular;
```

5. Acrescente um método construtor:

```
public Conta(string t)
{
	titular = t;
}
```

6. Acrescente propriedades de leitura:

```
public decimal Saldo
{
	get { return saldo; }
}

public string Titular
{
	get { return titular; }
}
```

2 Definindo métodos/propriedades abstratos

1. Adicionar o seguinte código à classe “Conta” 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;
}
```

3 Definindo métodos/propriedades passíveis de sobrescrita

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

```
public virtual void Depositar(decimal valor)
{
	saldo += valor;
}
```

```
public virtual void Sacar(decimal valor)
{
	saldo -= valor;
}
```

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

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

```
public class ContaPoupanca : Conta
```

2. Acrescente um atributo adicional além daqueles herdados da classe “Conta”, para representar a taxa de juros e a data de aniversário da conta de poupança:

```
private decimal taxaJuros;
private DateTime dataAniversario;
```

3. 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;
}
```

4. Acrescente propriedades adicionais:

```
public decimal Juros
{
		get { return taxaJuros; }
		set { taxaJuros = value; }
}

public DateTime DataAniversario
{
		get { return dataAniversario; }
}
```

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

```
public void AdicionarRendimento()
{
	DateTime hoje = DateTime.Now;
	if (hoje.Day == dataAniversario.Day && hoje.Month == dataAniversario.Month)
	{
			decimal rendimento = this.Saldo * taxaJuros;
			this.Depositar(rendimento);
	}
}
```

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

```
public override string Id
{
	get { return this.Titular + "(CP)"; }
}
```

5 Compilando o projeto

1. Compile o projeto. Corrija qualquer erro de compilação que possa ter sido gerado.

6 Exercícios

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

2. Crie uma coleção de objetos do tipo *Conta* e acrescente diversos objetos dos tipos das classes derivadas. Quais métodos podem ser chamados sobre os elementos da coleção?