Laboratório 5 – Classes e Herança

Este laboratório introduz a herança e sobrescrita de métodos em C#.

1 Adicionando novas classes a um projeto

1. Crie um novo “Console Application” com nome “Laboratorio5”.

2. Adicione uma nova classe. Coloque “Circulo.cs” para o nome da classe. Clique *Add*. Modique o código da classe de acordo com o seguinte:

```
public class Circulo
{
    private double coordX;
    private double coordY;
    private double raio;

    public Circulo()
        : this(0, 0, 1)
    {
    }

    public Circulo(double x, double y, double r)
    {
        coordX = x;
        coordY = y;
        raio = Math.Abs(r);
    }

    public double CentroX
    {
        get
        {
            return coordX;
        }
        set
        {
            coordX = value;
        }
    }

    public double CentroY
    {
        get
        {
            return coordY;
        }
        set
        {
            coordY = value;
        }
    }

    public double Raio
    {
        get
```

```
            {
                return raio;
            }
            set
            {
                raio = value;
            }
        }

        public override string ToString()
        {
            return "(" + string.Format("{0:F2}", CentroX) + ";"
                     + string.Format("{0:F2}", CentroY) + ")"
                     + " raio=" + string.Format("{0:F2}", Raio);
        }
    }
```

3. Adicione o código abaixo ao método Main:

```
Circulo circ1 = new Circulo();
Console.WriteLine(circ1);
Circulo circ2 = new Circulo(2.4, 5, 3);
Console.WriteLine(circ2);
```

4. Compile e execute o programa.

2 Definindo herança entre classes

1.Adicione uma nova classe “CirculoColorido” que é subclasse da classe Circulo. Você pode utilizar o mesmo arquivo “Circulo.cs” ou criar um novo arquivo para conter a nova classe.

```
public class CirculoColorido : Circulo
{
}
```

2. Adicione o seguinte atributo à classe “CirculoColorido”.

```
private string minhaCor;
```

3. Adicione a seguinte propriedade.

```
public string Cor
{
            get
            {
                return minhaCor;
            }
            set
            {
                minhaCor = value;
            }
}
```

3 Implementando construtores em subclasses

1. Adicione o seguinte código à classe CirculoColorido.

```
public CirculoColorido()
{
	minhaCor = "preto";
}
```

2. Adicione o seguinte código à classe CirculoColorido.

```
public CirculoColorido(double x, double y, double r, string c)
		: base(x, y, r)
{
	minhaCor = c;
}
```

4 Sobrescrevendo métodos herdados

1. Adicione o seguinte código à classe "CirculoColorido":

```
public override string ToString()
{
	return base.ToString() + " cor=" + Cor;
}
```

2. Substitua o código do método Main pelo código abaixo:

```
Circulo circ1 = new Circulo();
Console.WriteLine(circ1);
Circulo circ2 = new Circulo(2.4, 5, 3);
Console.WriteLine(circ2);
CirculoColorido circ3 = new CirculoColorido();
Console.WriteLine(circ3);
CirculoColorido circ4 = new CirculoColorido(1.5, 3.1, 1, "vermelho");
Console.WriteLine(circ4);
```

3. Compile e execute o programa.

5 Exercícios

1. Altere o método *Main* para mostrar o uso das propriedades do círculo e círculo colorido. É possível utilizar as propriedades *CentroX* e *CentroY* em um objeto da classe *CirculoColorido*?

2. Crie uma subclasse de *CirculoColorido* chamada *CirculoColoridoPreenchido* que possui uma cor adicional para o preenchimento do círculo. Utilize um objeto mais adequado para armazenar a cor do que uma string (procure na API do .NET por essa classe!). Teste a classe criada.

3. Crie uma array de objetos do tipo *Circulo* e a preencha com instâncias de *Circulo*, *CirculoColorido* e *CirculoColoridoPreenchido*. A seguir, percorra o array e imprima no console os dados de cada um dos objetos.