

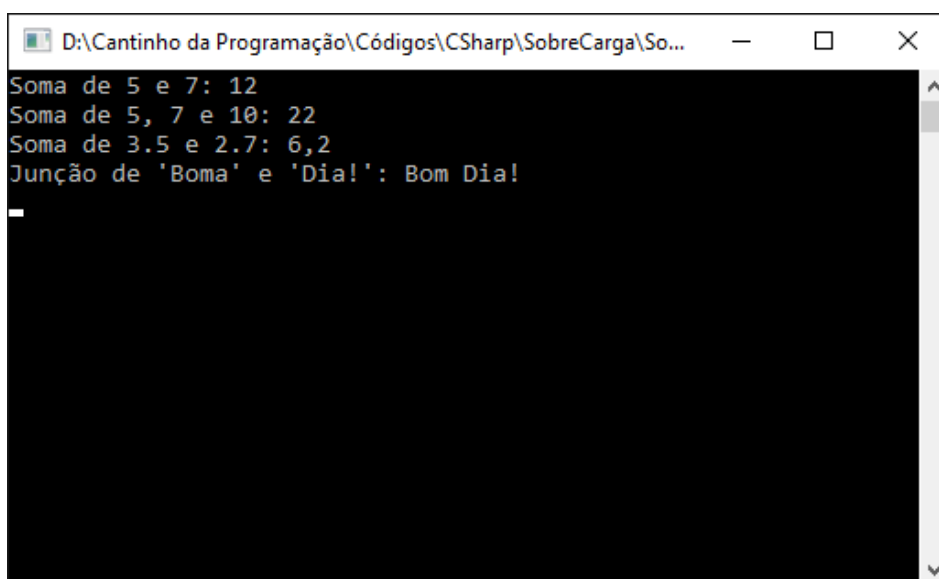
Método Sobrecarregado

Mário Leite

...

De um modo geral podemos concluir que os quatro pilares da OOP (Programação Orientada a Objetos) são: Abstração, Herança, Polimorfismo e Encapsulamento. A Abstração permite modelar características do mundo real do problema a ser resolvido, ignorando os aspectos de um assunto que não é relevante para o propósito analisado. Em outras palavras, isto quer dizer “*capacidade de agrupar objetos, pessoas e acontecimentos que possuam algo em comum, alguma relação importante*”. Na Herança ocorre, como o nome sugere, um mecanismo em que uma classe herda TUDO de outra classe, além de poder incluir características extras (atributos) e mais ações (métodos). O Polimorfismo, literalmente quer dizer “*várias formas*”; e no contexto da OOP é um princípio relacionado aos diversos comportamentos que os objetos podem assumir quando um serviço lhes é requerido. O Polimorfismo é um conceito muito importante na OOP, uma vez que permite a um objeto se expor o menos possível para outro objeto. Isto diminui em muito os acessos indesejáveis ao objeto, através dessa restrição à sua visibilidade, permitindo ao programador segregar em local seguro os aspectos mais “sensíveis” do objeto, se alguma modificação se fizer necessária. Uma aplicação muito corriqueira é a definição de variáveis locais, em vez de variáveis globais; o que protege os valores de acessos indevidos.

Embora esses quatro mecanismos são os que mais são estudados e os mais visíveis na OOP, existe um assunto não menos importante do ponto de vista operacional: **Métodos Sobrecarregados** (*overload*). Neste caso o método pode executar várias tarefas usando o mesmo nome, sem precisar nomear cada método para cada função, dependendo apenas dos parâmetros que receber. O programa “**SobreCarga**”, codificado em C#, mostra um exemplo simples de calculadora que executa várias ações com o mesmo nome. Neste exemplo a classe “Calculadora” possui quatro versões do método **Somar()**; cada uma com diferentes assinaturas de parâmetros. Ao chamar esse método com argumentos específicos, o compilador do C# determina qual versão do método deve ser invocada com base nos tipos e números de parâmetros passados.



```
D:\Cantinho da Programação\Códigos\CSharp\SobreCarga\So...
Soma de 5 e 7: 12
Soma de 5, 7 e 10: 22
Soma de 3.5 e 2.7: 6,2
Junção de 'Boma' e 'Dia!': Bom Dia!
```

Saída do programa “SobreCarga”

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SobreCarga
{
    internal class Program
    {
        public class Calculadora
        {
            // Método para somar dois inteiros
            public int Somar(int a, int b)
            {
                return a+b;
            }

            // Método sobrecarregado para somar três inteiros
            public int Somar(int a, int b, int c)
            {
                return a+b+c;
            }

            // Método sobrecarregado para somar dois doubles
            public double Somar(double a, double b)
            {
                return a+b;
            }

            // Método sobrecarregado para concatenar duas strings
            public string Somar(string a, string b)
            {
                return a+b;
            }

            static void Main(string[] args)
            {
                Calculadora calc = new Calculadora();
                Console.WriteLine("Soma de 5 e 7: " + calc.Somar(5, 7));
                Console.WriteLine("Soma de 5, 7 e 10: " + calc.Somar(5, 7, 10));
                Console.WriteLine("Soma de 3.5 e 2.7: " + calc.Somar(3.5, 2.7));
                Console.WriteLine("Junção de 'Bom' e 'Dia!': " + calc.Somar("Bom", "Dia!"));
                Console.ReadLine();
            }
        }
    }
}
```