Construtores

Com o que vimos nos capítulos anteriores, nós precisamos lembrar de colocar o nome após criarmos um novo cliente em nosso sistema. Isso pode ser visto no código a seguir:

```
Cliente guilherme = new Cliente();
guilherme.Nome = "Guilherme";
```

E se esquecermos de chamar a segunda linha desse código, teremos um cliente sem nome. Mas, será que faz sentido existir um cliente sem nome?

Para evitar isso, ao construir nosso objeto temos que obrigar o desenvolvedor a falar qual o nome do `Cliente`. Isto é, queremos ser capazes de alterar o comportamento da construção do objeto.

Queremos definir um novo comportamento que dirá como será construído o objeto. Algo como:

```
Cliente guilherme = new Cliente("Guilherme Silveira");
```

Note que esse comportamento que desejamos lembra um comportamento normal, passando argumentos, mas com a característica especial de ser quem constrói um objeto. Esse comportamento recebe o nome de construtor. E como defini-lo? Similarmente a um comportamento qualquer:

```
class Cliente
{
    // Outros atributos da classe Cliente
    public string Nome { get; set; }

    public Cliente (string nome)
    {
        this.Nome = nome;
    }
}
```

Vimos que quando criamos um construtor na classe, o C# usa o construtor criado para inicializar o objeto, porém o que acontece quando não temos nenhum construtor na classe? Quando uma classe não tem nenhum construtor, o C# coloca um **construtor padrão** dentro da classe. Esse construtor não recebe argumentos e não executa nenhuma ação, ou seja, um construtor que não recebe nenhum argumento e tem o corpo vazio.

8.1 Múltiplos construtores dentro da classe

Na seção anterior definimos um construtor dentro da classe cliente que inicializa a propriedade nome, mas e se quiséssemos inicializar também a idade do `Cliente` durante a construção do objeto? Nesse caso, precisaríamos de um construtor adicional na classe `Cliente`:

```
class Cliente
{
    public string Nome { get; set; }

    public int Idade { get; set; }

    // construtor que só recebe o nome
    public Cliente (string nome)
    {
```

```
        this.Nome = nome;
    }
    // construtor que recebe o nome e a idade
    public Cliente (string nome, int idade)
    {
        this.Nome = nome;
        this.Idade = idade;
    }
}
```

Veja que definimos duas versões diferentes do construtor da classe, uma que recebe apenas a `string nome` e outra que recebe `string nome` e `int idade`. Quando colocamos diversas versões do construtor dentro de uma classe, estamos fazendo uma **sobrecarga** de construtores.

> **Valor padrão para os parâmetros**
>
> No C#, ao invés de fazermos sobrecarga de construtores para podermos passar informações adicionais na criação do objeto, podemos utilizar os parâmetros opcionais com valores padrão.

8.2 Para saber mais — Initializer

Vimos que podemos utilizar um construtor para pedir informações obrigatórias para a classe. Mas, por exemplo, temos a classe `Cliente` e apenas seu nome é obrigatório, então podemos pedir essa informação no construtor da classe.

```
Cliente cliente = new Cliente ("Victor Harada");
```

Mas o cliente também possui CPF, RG e idade. Para colocarmos essas informações no cliente que criamos precisamos do código:

```
Cliente cliente = new Cliente ("Victor Harada");
cliente.Cpf = "123.456.789-01";
cliente.Rg = "21.345.987-x";
cliente.Idade = 25;
```

Veja que em todas as linhas estamos repetindo o nome da variável que guarda a referência para o cliente. Para evitar essa repetição, podemos utilizar os initializers do C#. O Initializer é um bloco de código que serve para inicializar as propriedades públicas do objeto.

```
Cliente cliente = new Cliente ("Victor Harada")
{
    // bloco de inicialização
    Cpf = "123.456.789-01",
    Rg = "21.345.987-x",
    Idade = 25
};
```