

PROGRAMAÇÃO DE COMPUTADORES



Técnico Integrado em
INFORMÁTICA

Professor Jefferson Chaves
jefferson.chaves@ifc-araquari.edu.br

- Controlar o acesso aos métodos, atributos e construtores, por meio dos modificadores **private** e **public**;
- Escrever métodos de acesso a atributos do tipo **getters** e **setters**;
- Escrever **construtores**;
- Utilizar variáveis e métodos **estáticos**;



**PROGRAMAÇÃO DE
COMPUTADORES**

ENCAPSULAMENTO

- Encapsular significa restringir ou liberar o acesso às propriedades e métodos das classes;
- Aplica-se este conceito através das palavras chaves: **public**, **private**, **protected**;
- Essas palavras chaves são chamadas de **modificadores de acesso** ou **modificadores de visibilidade**;

- **Escopo**, nome que se dá aos limites de uma variável;
- Uma variável só é válida dentro das chaves onde é declarada;
 - Se declarada dentro do método só vale dentro do método;
 - Se declarada dentro da classe (como atributo) ela vale na classe inteira incluindo no método.

- **Public:** ao definir atributos ou método como public, estamos dizendo que suas informações podem ser acessadas diretamente, “por todos”, a qualquer momento.
- Até este momento, todas as propriedades e métodos das classes que vimos foram definidas dessa forma.


- **Private:** É o nível de acesso mais restritivo;
- Indica que essa variável somente vai poder ser acessada dentro da própria classe;
- **Será proibido acesso atributos ou métodos privados de fora do escopo da classe.**
- Tentar acessar um método ou atributo declarado **private** de fora da Classe, resultará em uma mensagem de erro indicando que não é possível acessar este elemento.

- **Protected:** ao definimos em uma classe uma propriedade ou método do tipo **protected**, estamos definindo que ambos só poderão ser acessados pela própria classe **ou por seus herdeiros;**
- **Será proibido acesso atributos ou métodos protegidos de fora do escopo da classe.**

- Quando se usa Orientação a Objetos, é prática comum (*quase obrigatória*) proteger seus atributos como **private**;
- Encapsular é fundamental para que um programa seja suscetível a mudanças;

Programando voltado para a interface e não para a implementação

- É sempre bom programar pensando na interface da sua classe, como seus usuários a estarão utilizando, e não somente em como ela vai funcionar;
- A implementação em si, o conteúdo dos métodos, não tem tanta importância para o usuário dessa classe, uma vez que ele só precisa saber o que cada método pretende fazer, e não como ele faz, pois isto pode mudar com o tempo;
- Essa frase vem do livro Design Patterns, de Eric Gamma et al. Um livro cultuado no meio da orientação a objetos.

A faint, green-tinted background image of an elephant and its calf in a savanna landscape. The elephant is on the left, facing right, with its trunk slightly curved. A smaller calf is behind it, also facing right. The background is a hazy, open plain under a light sky.


**PROGRAMAÇÃO DE
COMPUTADORES**

GETTERS E SETTERS

- O modificador **private** faz com que o atributo não possa ser lido e nem modificado de fora do escopo da classe;
- Sempre que precisamos de uma maneira para fazer algo com um objeto, utilizamos **métodos**;

- Para permitir seu acesso (já que são *privates*) a prática é criar dois métodos:
 - um que retorne o valor (**get**);
 - outro que informe ou modifique o valor (**set**);
- A **convenção** é colocar **get** ou **set** antes do nome do atributo;
- **Não devemos criar **getters** e **setters** sem um motivo explícito.**

- O método **get_x** não necessariamente retornam o valor de um atributo **x** do objeto;
- Para atributos booleanos, pode-se usar no lugar do get o sufixo is;
 - Sendo assim, um atributo booleano ligado, em vez de **getLigado** poderíamos ter **isLigado**.

A faint, green-tinted background image of an elephant and its calf in a savanna setting. The elephant is on the right, and the calf is on the left, both facing right. The text is overlaid on this image.

**PROGRAMAÇÃO DE
COMPUTADORES**

CONSTRUTORES

- Quando usamos a palavra chave **new** estamos criando um objeto;
- Sempre que o new é chamado ele executa o **construtor da classe**;
- Construtor é um “método mágico” que é executado toda vez que o objeto é **instanciado**.
- Caso o construtor não estiver definido, um construtor padrão é utilizado

- Ao se declarar um construtor, o construtor padrão não será mais fornecido




- Um construtor tem por objetivo:
 - Realizar operações de inicialização;
 - Inicializar os atributos do objeto;
- A ideia é simples: **obrigar (ou dar a possibilidade)** o usuário de uma classe a passar argumentos para o objeto durante seu processo de criação.

- A **assinatura** de um método é composta por:
 - **nome**: um identificador para o método.
 - **tipo**: quando o método tem um valor de retorno, o tipo desse valor.
 - **argumentos**: o tipo e um identificador para cada parâmetro, se existir;
 - **visibilidade**: como para atributos, define o quão visível é um método a partir de objetos de outras classes.
 - tipo de retorno;

Qual a necessidade de um construtor ?

Obrigar ou possibilitar o usuário de uma classe a passar argumentos para o objeto durante o processo de criação do mesmo

A faint, semi-transparent image of an elephant and its calf in a savanna landscape, serving as a background for the text.

**PROGRAMAÇÃO DE
COMPUTADORES**

**ATRIBUTOS DE
CLASSE**

- Atributos e métodos estáticos pertencem à classe e não a uma instância;
- Podem ser utilizados sem ter que instanciar a classe;
- Variáveis estáticas são compartilhadas por todos os objetos;
- Nesse caso, utiliza-se a palavra chave **static**;

- Métodos estáticos acessam **apenas variáveis estáticas e métodos estáticos**;
- Isso porquê não há objeto criado no momento em que o método estático é chamado.

- Para se acessar valores estáticos, deve se usar:
 - Dentro de uma classe: palavra reservada *self* e seguido do operador de escopo ::
 - Fora da classe: o nome da Classe, seguido do operador de escopo ::
- O operador de Resolução de Escopo (também chamado de *Paamayim Nekudotayim*), ou em termos mais simples, dois pontos duplo.

Métodos e atributos estáticos

Métodos e atributos estáticos só podem acessar outros métodos e atributos estáticos da mesma classe.

Isso faz todo sentido já que dentro de um método estático não temos acesso à referência `$this`, pois um método estático é chamado através da classe, e não de um

- Nosso banco também quer controlar a quantidade de contas existentes no sistema.
- Como poderíamos fazer isto?

- A ideia mais simples:

```
Conta c = new Conta();  
totalDeContas = totalDeContas + 1;
```

- Qual o problema em se fazer assim?
- Qual seria a maneira orientada a objetos de se fazer isso?

1. Adicione o modificador de visibilidade (private, se necessário) para cada atributo e método da classe Funcionario. Tente criar um Funcionario no main e modificar ou ler um de seus atributos privados. O que acontece?
2. Crie os getters e setters necessários da sua classe Funcionario.
3. Modifique suas classes que acessam e modificam atributos de um Funcionario para utilizar os getters e setters recém criados.

4. Faça com que sua classe Funcionario receba obrigatoriamente o nome do Funcionario durante a criação do objeto. Utilize construtores para obter esse resultado.
5. Adicione um atributo na classe Funcionario de tipo int que se chama identificador. Esse identificador deve ter um valor único para cada instância do tipo Funcionario. O primeiro Funcionario instanciado tem identificador 1, o segundo 2, e assim por diante.