Programação Orientada a Objetos

Prof.ª Ma. Jessica Oliveira

Católica
Universidade Católica de Brasília

Aula 03 – 10/03/2025

Abstração e Encapsulamento.

Universidade Católica de Brasília

Abstração.

Definição.

- Significa **destacar os aspectos mais importantes** de um objeto do mundo real e ignorar os detalhes desnecessários para o contexto da aplicação.
- Isso ajuda a reduzir a complexidade e melhorar a organização do código.

Encapsulamento

Definição.

- É o conceito de **esconder a implementação interna de um objeto**, expondo apenas o que é necessário para a interação com o objeto.
- Isso é realizado por meio de modificadores de acesso, que controlam a visibilidade dos atributos e métodos.
- **Vantagens:**
  - **Segurança:** protege os dados de um objeto contra acessos não autorizados ou alterações indesejadas.
  - **Modularidade:** isola a implementação interna, permitindo que o código seja alterado sem afetar o restante do sistema.

Modificadores de acesso.

- ***public*:** é o mais **permissivo** de todos. Uma classe, atributo ou método declarado como **`public`** pode ser acessado por qualquer classe em qualquer pacote. Ou seja, ele possui **visibilidade pública** e **pode ser utilizado livremente**.
- ***private*:** é o mais **restritivo** de todos. Uma classe, atributo ou método declarado como **`private`** só pode ser acessado dentro da própria classe. Ou seja, ele possui **visibilidade restrita** e **não pode ser utilizado por outras classes**.
- ***protected*:** permite o acesso a um atributo ou método dentro da classe e em suas subclasses.

Getters e *Setters*.

- Os métodos *get* e *set* são usados para acessar e modificar atributos privados de uma classe. Eles seguem a convenção:

  - ***Getter*** (`getNomeDoAtributo`): retorna o valor de um atributo.
  - ***Setter*** (`setNomeDoAtributo`): modifica o valor de um atributo.

- Isso permite **controlar o acesso aos atributos** e aplicar regras de validação antes de alterar valores.

Microprojeto 03

🎯 **Objetivo:** aplicar encapsulamento para garantir modularidade e segurança no código.

Para o pós-aula...

- Revisar as classes criadas e aplicar abstração onde necessário.
- Definir atributos privados e criar *getters* e *setters*.
- Melhorar a estrutura do código com boas práticas de encapsulamento.
- Resolver conflitos de merge no GitHub e documentar como foram resolvidos.

Na próxima aula...

Aula 04 (17/03/2025) - Relações entre Classes e Composição.

Dúvidas?

jessica.oliveira@p.ucb.br