

# Programação Orientada a Objetos

Prof.<sup>a</sup> Ma. Jessica Oliveira

**Aula 06 – 31/03/2025**

# **Polimorfismo: sobrecarga e sobrescrita de métodos.**

# Definição.

- O termo polimorfismo vem do grego e significa "muitas formas".
- Na POO, o polimorfismo permite que objetos de diferentes classes sejam tratados de maneira uniforme por meio de uma interface comum.
- Isso significa que um mesmo método ou operação pode ser aplicado a objetos de diferentes tipos, produzindo resultados distintos conforme a implementação.
- Em outras palavras, o polimorfismo garante que métodos com a mesma assinatura em classes diferentes se comportem de maneiras variadas.
- Isso aumenta a flexibilidade do código, permitindo que sistemas extensíveis e modulares sejam desenvolvidos de maneira mais natural e eficiente.

# Polimorfismo em Tempo de Compilação (Estático).

- Ocorre quando o método a ser executado é determinado durante a compilação do programa.
- A principal forma de implementar o polimorfismo estático é por meio da **sobrecarga de métodos**.
- Exemplos clássicos são os métodos com o mesmo nome, mas assinaturas diferentes na mesma classe.

# Polimorfismo em Tempo de Execução (Dinâmico).

- Ocorre quando o método a ser executado é determinado durante a execução do programa.
- É obtido principalmente por meio da **sobrescrita de métodos**.
- Envolve o conceito de ligação tardia (*late binding*), em que a decisão de qual método será executado só ocorre em tempo de execução.

# Benefícios do Polimorfismo.

- **Flexibilidade e Extensibilidade:** permite a criação de sistemas modulares, que podem ser facilmente ampliados com novos tipos de objetos sem modificar o código existente.
- **Reutilização de Código:** a mesma interface pode ser usada para várias implementações.
- **Manutenibilidade:** a estrutura modular do polimorfismo facilita a manutenção e atualização do sistema.

# Sobrecarga de Métodos.

# Definição.

- A sobrecarga de métodos ocorre quando uma classe possui mais de um método com o mesmo nome, mas com assinaturas diferentes.
- A assinatura de um método inclui seu nome, o número de parâmetros e os tipos desses parâmetros.
- A sobrecarga permite que métodos realizem ações semelhantes com tipos de dados diferentes ou com um número variável de argumentos.



# Regras.

- Para que a sobrecarga aconteça de forma válida em Java, é necessário respeitar as seguintes regras:
  - **Nome do método igual:** todos os métodos devem ter o mesmo nome.
  - **Lista de parâmetros diferente:** a diferença pode estar no número ou no tipo dos parâmetros.
  - **Não considera o tipo de retorno:** o compilador não distingue os métodos apenas pelo tipo de retorno.
  - **Pode ter modificadores diferentes:** o nível de visibilidade ou modificador (como `static`) pode variar.

# Vantagens.

- **Facilidade de uso:** um único nome de método pode lidar com diferentes tipos de dados.
- **Organização:** centraliza funcionalidades similares.
- **Clareza de código:** Melhora a leitura e a manutenção do código.

# Sobrescrita de Métodos.

# Definição.

- A sobrescrita de métodos ocorre quando uma classe filha redefine um método herdado da classe pai com a mesma assinatura (nome e parâmetros).
- O método sobrescrito substitui o comportamento da classe pai na classe derivada.

# Regras.

- Para que a sobrescrita aconteça corretamente, é preciso seguir algumas regras:
  - **Assinatura idêntica:** o nome e os parâmetros do método devem ser exatamente os mesmos.
  - **Mesmo tipo de retorno:** o tipo de retorno deve ser igual ou compatível.
  - **Visibilidade igual ou mais acessível:** o método na classe filha deve ter a mesma ou maior visibilidade que o método da classe pai.
  - **Uso da anotação `@Override`:** embora opcional, o uso dessa anotação é uma boa prática para garantir que o método está realmente sobrescrevendo um método da superclasse.

# Classes Abstratas e Interfaces.

# Classes Abstratas.


- São classes que não podem ser instanciadas diretamente.
- Servem como modelos para subclasses, garantindo que métodos abstratos sejam implementados.
- Podem ter métodos concretos e abstratos.

# Interfaces.

- Definem um conjunto de métodos abstratos que as classes que implementam devem definir.
- Não possuem implementação de métodos (até o Java 8, quando métodos default foram introduzidos).
- Permitem herança múltipla.



# Microprojeto 06

 **Objetivo:** implementar polimorfismo, permitindo flexibilidade na chamada de métodos.

# Para o pós-aula...

- Aplicar sobrecarga e sobrescrita de métodos em algumas classes.
- Introduzir classes abstratas e interfaces para melhorar a estrutura do código.
- Criar testes unitários simples para validar o comportamento dos métodos.
- Revisar o histórico de commits para garantir boas práticas de versionamento.

# Dúvidas?

[jessica.oliveira@p.ucb.br](mailto:jessica.oliveira@p.ucb.br)