

Programação Avançada - AULA 03

Matheus Moresco

Engenharia de Software - 5º Período

2025/01

Última aula

- Revisão dos conceitos de POO
 - Encapsulamento
 - Herança
 - Polimorfismo
 - Abstração

Introdução

- Introdução a sobrecarga de métodos.
- Revisão do conceito de construtores.

O que é Sobrecarga?

- A sobrecarga de métodos é um conceito do polimorfismo que consiste basicamente em criar variações de um mesmo método, ou seja, a criação de dois ou mais métodos com nomes totalmente iguais em uma classe.
- Permite que utilizemos o mesmo nome em mais de um método contanto que suas listas de argumentos sejam diferentes para que seja feita a separação dos mesmos.
- Melhora a legibilidade e a organização do código, permitindo que uma mesma operação seja realizada de formas diferentes, dependendo dos argumentos fornecidos.

Benefícios da Sobrecarga

- ✓ **Reutilização de Código:** Permite evitar a duplicação de métodos semelhantes, tornando o código mais limpo e organizado.
- ✓ **Facilidade de Uso:** Oferece múltiplas maneiras de chamar um método, melhorando a experiência do desenvolvedor.
- ✓ **Melhor Legibilidade:** Mantém nomes coerentes para funcionalidades semelhantes, tornando o código mais intuitivo.
- ✓ **Flexibilidade:** Permite tratar diferentes cenários sem a necessidade de criar múltiplos métodos com nomes diferentes.

Regras da Sobrecarga de Métodos

- Os métodos devem ter o mesmo nome.
- Devem possuir assinaturas diferentes (quantidade ou tipo de parâmetros).
- Não podem diferir apenas pelo tipo de retorno.

```
2
3 public class Calculadora {
4     public int soma(int a, int b, int c) {
5         return a + b + c;
6     }
7     public int soma(int a, int b) {
8         return a + b;
9     }
10    public double soma(int a, int b) {
11        return a + b;
12    }
13 }
```

✘ Errado

```
2
3 public class Calculadora {
4     public int soma(int a, int b) {
5         return a + b;
6     }
7     public int soma(int a, int b, int c) {
8         return a + b + c;
9     }
10    public double soma(double a, double b) {
11        return a + b;
12    }
13 }
```

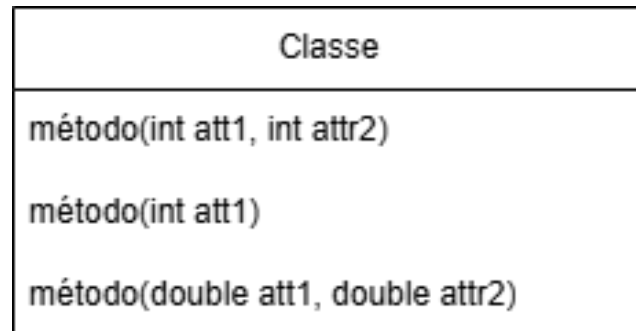
✔ Correto

Sobrecarga vs Sobreposição

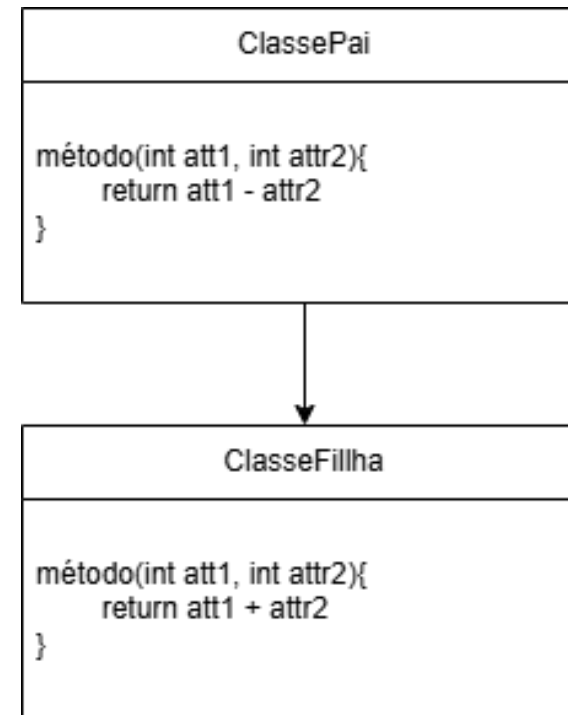
- A **sobrecarga** está ligada a variância de estados de um método, podemos entendê-la como um conjunto de opções que o programa principal tem para escolher quando recebe os parâmetros passados pelo usuário.
- A **sobreposição** funciona por meio do sistema de herança, e para a mesma funcionar o nome e lista de argumentos dos métodos devem ser totalmente iguais aos da classe herdada.

Sobrecarga vs Sobreposição

Sobrecarga



Sobreposição



Sobrecarga vs Sobreposição

Sobrecarga

```
2
3 public class Calculadora {
4     public int soma(int a, int b) {
5         return a + b;
6     }
7     public int soma(int a, int b, int c) {
8         return a + b + c;
9     }
10    public double soma(double a, double b) {
11        return a + b;
12    }
13 }
14
15
```

Sobreposição

```
2
3 public class Pessoa {
4     String nome;
5     int idade;
6
7     public void exibirInformacoes() {
8         System.out.println("Nome: " + nome);
9         System.out.println("Idade: " + idade);
10    }
11 }
```

```
2
3 public class Aluno extends Pessoa {
4     String disciplina;
5
6     @Override
7     public void exibirInformacoes() {
8         System.out.println("Nome: " + nome);
9         System.out.println("Idade: " + idade);
10        System.out.println("Disciplina: " + disciplina);
11    }
12 }
13
```

Métodos Construtores

- Os métodos construtores são métodos especiais em Java utilizados para inicializar objetos de uma classe. Eles têm o mesmo nome da classe e são chamados automaticamente quando um objeto é instanciado usando new.

Objeto objeto1 = new Objeto(attr1, attr2)

Métodos Construtores

- Principais Características dos Construtores:
 - **Mesmo nome da classe** – O nome do construtor deve ser idêntico ao da classe.
 - **Não possuem um tipo de retorno** – Nem mesmo void deve ser declarado.
 - **Executados automaticamente** – Quando um objeto é criado, o construtor é chamado sem necessidade de invocação explícita.
 - **Podem ser sobrecarregados** – Permite múltiplos construtores na mesma classe com assinaturas diferentes.

Métodos Construtores

```
1
2 public class Pessoa {
3     String nome;
4     int idade;
5
6     public Pessoa(String nome, int idade) {
7         this.nome = nome;
8         this.idade = idade;
9     }
10
11
12     public void exibirInformacoes() {
13         System.out.println("Nome: " + nome);
14         System.out.println("Idade: " + idade);
15     }
16 }
17
```

```
1
2 public class App {
3     Run | Debug | Run main | Debug main
4     public static void main(String[] args) throws Exception {
5         Pessoa pessoa = new Pessoa(nome:"João", idade:20);
6
7         pessoa.exibirInformacoes();
8     }
9
10
```

Tipos de Construtores em Java

1. Construtor Padrão (Sem Parâmetros)
2. Construtor Parametrizado
3. Sobrecarga de Construtores
4. Construtor Copiador

1. Construtor Padrão (Sem Parâmetros)

- Caso nenhum construtor seja definido, Java cria um construtor padrão automaticamente.

```
1
2  public class Pessoa {
3      String nome;
4      int idade;
5
6      public Pessoa() {
7          this.nome = "Sem Nome";
8      }
9
10     public void exibirInformacoes() {
11         System.out.println("Nome: " + nome);
12         System.out.println("Idade: " + idade);
13     }
14 }
15
```

2. Construtor Parametrizado

- Construtores podem aceitar parâmetros para inicializar atributos.

```
1
2  public class Pessoa {
3      String nome;
4      int idade;
5
6      public Pessoa(String nome, int idade) {
7          this.nome = nome;
8          this.idade = idade;
9      }
10
11
12      public void exibirInformacoes() {
13          System.out.println("Nome: " + nome);
14          System.out.println("Idade: " + idade);
15      }
16  }
17
```

3. Sobrecarga de Construtores

- Podemos criar vários construtores com diferentes parâmetros.

```
2 public class Pessoa {
3     String nome;
4     int idade;
5
6     public Pessoa(String nome, int idade) {
7         this.nome = nome;
8         this.idade = idade;
9     }
10
11     public Pessoa(String nome) {
12         this.nome = nome;
13     }
14
15     public Pessoa() {
16         this.nome = "Sem Nome";
17     }
18
19     public void exibirInformacoes() {
20         System.out.println("Nome: " + nome);
21         System.out.println("Idade: " + idade);
22     }
23 }
24
```


4. Construtor Copiador

- Permite criar um novo objeto a partir de outro já existente.

```
1
2  public class Pessoa {
3      String nome;
4      int idade;
5
6      public Pessoa(Pessoa outraPessoa) {
7          this.nome = outraPessoa.nome;
8          this.idade = outraPessoa.idade;
9      }
10
11     public void exibirInformacoes() {
12         System.out.println("Nome: " + nome);
13         System.out.println("Idade: " + idade);
14     }
15 }
16
```

Vantagens do Uso de Construtores

- ✓ Facilitam a inicialização dos objetos
- ✓ Garantem que um objeto tenha valores válidos
- ✓ Melhoram a legibilidade e manutenção do código
- ✓ Permitem flexibilidade na criação de instâncias (com sobrecarga)

Exemplo prático

- Implementar uma classe **Produto** que tenha sobrecarga de métodos para calcular o preço final com diferentes taxas de desconto e sobrecarga de construtores para inicializar produtos com diferentes parâmetros.

Trabalhos da disciplina

- Serão vários trabalhos durante o bimestre.
- Alguns trabalhos serão interligados.
- Entregues todos juntos e um único projeto.
- Entrega no final do Bimestre.
- Grupos de até 4 pessoas.

Exercício Prático

1. Crie uma classe **Pessoa** com sobrecarga de construtores, utilize diferentes de informações (nome, idade, endereço, etc)
2. Criar uma classe **Funcionario** que herda as informações da classe Pessoa, use sobrecarga de construtores para inicializar com diferentes conjuntos de informações (cargo, salário, etc.).
3. Faça um método **getSalario()** que indica o salário do funcionário, e depois crie uma sobrecarga do método **getSalário()** que calcula o salário a partir de um valor de bônus, que será informado como parâmetro do método.
4. Faça a sobreposição do método de exibir informações, para exibir as informações do funcionário.