Programação avançada – AULA 02

Matheus Moresco
Engenharia de Software - 5º Período
2025/01

Revisão da ultima aula

- **Programação Avançada** aprofunda os conceitos essenciais para o desenvolvimento de sistemas robustos, escaláveis e eficientes.
- **Ementa** da disciplina
- Vantagens do **Java.**
- Instalação do **JDK** e **Eclipse.**

Introdução à Programação Orientada a Objetos (POO)

O que é POO?

- Paradigma baseado em objetos que representam entidades do mundo real.
- Facilita reutilização de código, organização e manutenção.
- Amplamente usado em aplicações escaláveis e complexas.

Introdução à Programação Orientada a Objetos (POO)

Por que usar POO?

- Modularidade: Código organizado em pequenas partes reutilizáveis.
- Reutilização: Uso de classes já existentes.
- Facilidade de manutenção: Melhor organização e menos repetição de código.

Estrutura Básica de uma Classe em Java

- Uma **classe** é a estrutura fundamental na Programação Orientada a Objetos. Ela define um modelo para a criação de **objetos**, especificando seus atributos (dados) e métodos (comportamentos).

```
public class NomeDaClasse {
    // Atributos (dados da classe)
    tipo nomeAtributo;

    // Métodos (comportamentos da classe)
    public void nomeMetodo() {
        // Implementação do método
    }
}
```

Métodos e Atributos

- **Atributos:** São variáveis que armazenam o estado do objeto.
- **Métodos:** São funções dentro da classe que definem seu comportamento.

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

Instanciação de Objetos

- A instanciação é o processo de criar um objeto a partir de uma classe usando a palavra-chave ***new***.

```
2
3 public class main {
4     static void main(String[] args) {
5         //Criando um objeto
6         Pessoa pessoa = new Pessoa();
7
8         //Definindo valores para os Atributos
9         pessoa.nome = "João da Silva";
10        pessoa.idade = 22;
11        //Chamando Método para exibir informações
12        pessoa.exibirInformacoes();
13    }
14 }
15
```

Princípios da POO - Encapsulamento

- O **encapsulamento** é um dos princípios da Programação Orientada a Objetos (POO) que restringe o acesso direto aos dados internos de um objeto e expõe apenas métodos seguros para manipular esses dados.

```
 2
 3 public class Pessoa {
 4     private String nome;
 5
 6     public void setNome(String nome) {
 7         this.nome = nome;
 8     }
 9
10     public String getNome() {
11         return nome;
12     }
13 }
```

Princípios da POO - Herança

- A herança é um dos pilares da Programação Orientada a Objetos (POO) e permite que uma classe (filha) reutilize atributos e métodos de outra classe (pai).

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

Princípios da POO - Herança

```
 2
 3 public class Aluno extends Pessoa {
 4  private String curso;
 5
 6  public Aluno(String nome, int idade, String curso) {
 7      this.nome = nome;
 8      this.idade = idade;
 9      this.curso = curso;
10  }
11 }
12
```

```
 2
 3 public class Professor extends Pessoa {
 4  private String disciplina;
 5
 6  public Professor(String nome, int idade, String disciplina) {
 7      this.nome = nome;
 8      this.idade = idade;
 9      this.disciplina = disciplina;
10  }
11 }
12
```

Princípios da POO - Polimorfismo

- O polimorfismo permite que um mesmo método tenha comportamentos diferentes dependendo do objeto que o utiliza. Isso é possível por meio de **métodos sobrescritos** em subclasses.

```
2
3  public class Aluno extends Pessoa {
4      private String curso;
5
6      public Aluno(String nome, int idade, String curso) {
7          this.nome = nome;
8          this.idade = idade;
9          this.curso = curso;
10     }
11
12     @Override
13     public void exibirInformacoes() {
14         System.out.println("Nome: " + nome);
15         System.out.println("Idade: " + idade);
16         System.out.println("Curso: " + curso);
17     }
18 }
19
```

```
2
3  public class Professor extends Pessoa {
4      private String disciplina;
5
6      public Professor(String nome, int idade, String disciplina) {
7          this.nome = nome;
8          this.idade = idade;
9          this.disciplina = disciplina;
10     }
11
12     @Override
13     public void exibirInformacoes() {
14         System.out.println("Nome: " + nome);
15         System.out.println("Idade: " + idade);
16         System.out.println("Disciplina: " + disciplina);
17     }
18 }
19
```

Princípios da POO - Abstração

- A **abstração** em POO consiste em ocultar detalhes internos da implementação e expor apenas os comportamentos essenciais.
  - Utilizamos **classes abstratas** para criar um modelo base que define métodos genéricos, sem implementação completa.
  - Subclasses concretas devem implementar esses métodos.

```
 2
 3 public abstract class Pessoa {
 4     protected String nome;
 5     protected int idade;
 6
 7     public Pessoa(String nome, int idade) {
 8         this.nome = nome;
 9         this.idade = idade;
10     }
11
12     public abstract void exibirInformacoes();
13
14 }
15
```

Exemplo Prático

- Criar uma classe **Carro**.
- Adicionar os atributos **marca**, **modelo** e **ano**.
- Adicionar método **exibirDetalhes**().
- Fazer o encapsulamento da classe **Carro**.
- Criar uma classe Filha **CarroEsportivo** que herda a de **Carro** e adiciona o atributo **velocidadeMaxima**.
- Adicione o método acelerar, com implementações diferentes nas classes **Carro** e **CarroEsportivo**.
- Faça a abstração do método **acelerar** na classe **Carro**

Exercício

1. Criar uma Classe **ContaBancaria** com os seguintes métodos:
   - **depositar**(double valor)
   - **sacar**(double valor)
   - **exibirSaldo**()
2. Testar no **main**() e imprimir o saldo após operações.
3. Tentar usar ao menos um dos princípios da POO.

Dica: Utilize modificadores de acesso (private) para proteger os atributos e crie métodos get e set para manipulação dos dados.

Próxima aula

- Conceito e aplicação da sobrecarga
- Benefícios da sobrecarga na reutilização de código