
FIAP

Domain Driven Design
using Java

AGENDA

1. Padrões de Projeto em Java DAO, Factory e Singleton
2. Exercícios

Introdução

- O que são padrões de projeto?
  - Soluções reutilizáveis para problemas comuns no desenvolvimento de software.
  - Melhoram a organização, manutenção e flexibilidade do código.

- Padrões abordados:
  - DAO (Data Access Object).
  - Factory.
  - Singleton

Padrão DAO

- O que é o Padrão DAO?
  - Encapsula o acesso ao banco de dados.
  - Separa a lógica de persistência da lógica de negócios.

- Vantagens:
  - Facilita a manutenção e evolução do sistema.
  - Permite trocar a tecnologia de persistência com facilidade.

Componentes DAO

- Interface DAO
  - Define os métodos para operações de CRUD (Create, Read, Update, Delete).
- Implementação DAO:
  - Contém a lógica de interação com o banco de dados.
- Entidade/Modelo:
  - Representa as tabelas do banco como objetos.

Exemplo DAO

- Entidade

```
public class Carro {
    private int id;
    private String marca;
    private String modelo;
    private int ano;
    private double preco;
    // Getters e setters
}
```

Exemplo DAO

- Interface DAO

```
public interface CarroDAO {
    void salvar(Carro carro);
    Carro buscarPorId(int id);
    void atualizar(Carro carro);
    void deletar(int id);
    List<Carro> listarTodos();
}
```

Exemplo DAO

- Implementação DAO

```java
public class CarroDAOImpl implements CarroDAO {
    private Connection conn;

    public CarroDAOImpl(Connection conn) {
        this.conn = conn;
    }

    @Override
    public void salvar(Carro carro) {
        String sql = "INSERT INTO carros (marca, modelo, ano, preco) VALUES (?, ?, ?, ?)";
        try (PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.setString(1, carro.getMarca());
            stmt.setString(2, carro.getModelo());
            stmt.setInt(3, carro.getAno());
            stmt.setDouble(4, carro.getPreco());
            stmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    // Métodos restantes...
}
```

Padrão Factory

- O que é o Padrão Factory?
  - Cria objetos sem expor a lógica de criação ao cliente.
  - Centraliza a lógica de decisão para instanciar diferentes classes.

- Vantagens:
  - Reduz o acoplamento entre classes.
  - Torna o código mais flexível e fácil de testar.

Exemplo Padrão Factory

- Interface

```
public interface Carro {
    void exibirInfo();
}
```

Exemplo Padrão Factory

- Implementações Concretas:

```
public class Sedan implements Carro {
    @Override
    public void exibirInfo() {
        System.out.println("Carro Sedan");
    }
}

public class SUV implements Carro {
    @Override
    public void exibirInfo() {
        System.out.println("Carro SUV");
    }
}
```

Exemplo Padrão Factory

- Classe Factory:

```
public class CarroFactory {
    public static Carro criarCarro(String tipo) {
        if ("sedan".equalsIgnoreCase(tipo)) {
            return new Sedan();
        } else if ("suv".equalsIgnoreCase(tipo)) {
            return new SUV();
        }
        return null;
    }
}
```

Exemplo Padrão Factory

- Uso:

```
public class Main {
    public static void main(String[] args) {
        Carro carro1 = CarroFactory.criarCarro("sedan");
        carro1.exibirInfo(); // Exibe: Carro Sedan

        Carro carro2 = CarroFactory.criarCarro("suv");
        carro2.exibirInfo(); // Exibe: Carro SUV
    }
}
```

Padrão Singleton

- O que é o Padrão Singleton?
  - Garante que uma classe tenha apenas uma única instância.
  - Fornece um ponto de acesso global para essa instância.

- Vantagens:
  - Útil para gerenciar recursos compartilhados, como conexões de banco de dados.
  - Evita múltiplas instâncias de uma classe.

Exemplo Singleton

```java
public class ConexaoBD {
    private static ConexaoBD instancia;
    private Connection conexao;

    private ConexaoBD() {
        try {
            String url = "jdbc:oracle:thin:@localhost:1521:xe";
            String usuario = "usuario";
            String senha = "senha";
            conexao = DriverManager.getConnection(url, usuario, senha);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public static ConexaoBD getInstancia() {
        if (instancia == null) {
            instancia = new ConexaoBD();
        }
        return instancia;
    }

    public Connection getConexao() {
        return conexao;
    }
}
```

Exemplo Padrão Factory

- Uso:

```
public class Main {
    public static void main(String[] args) {
        ConexaoBD conexaoBD = ConexaoBD.getInstancia();
        Connection conn = conexaoBD.getConexao();
        System.out.println("Conexão obtida: " + conn);
    }
}
```

Resumo da Aula

- **Padrão DAO:** Separa a lógica de persistência da lógica de negócios, facilitando manutenção.
- **Padrão Factory:** Encapsula a criação de objetos, promovendo flexibilidade.
- **Padrão Singleton:** Gerencia instâncias únicas, ideal para recursos globais e compartilhados.

Exercícios – DAO

1- Crie uma tabela no banco chamada produtos com colunas id, nome, quantidade e preco.

2- Implemente:

- Uma classe Produto para representar os dados.
- Uma interface ProdutoDAO com métodos CRUD.
- Uma classe ProdutoDAOImpl que implemente a interface e realize as operações no banco.

3- Teste todas as operações no método main()..

Exercícios – Factory

1 - Crie uma interface FormaGeometrica com o método desenhar().

2 - Implemente as classes Circulo e Quadrado, ambas implementando FormaGeometrica.

3 - Crie uma classe FormaFactory com um método estático que retorne a instância correta com base em uma string ("circulo" ou "quadrado").

4 - No programa principal, teste o método da fábrica para criar e desenhar formas.

Exercícios – Singleton

1 - Implemente uma classe Singleton Logger com:

- Um método log(String mensagem) que grava mensagens em um arquivo de texto.

2 - No programa principal:

- Obtenha a instância do Logger e registre mensagens de diferentes pontos do código.
- Verifique que apenas uma instância é usada.


FIAP