UNIVERSIDADE FEDERAL DE ALAGOAS - CAMPUS A.C.
SIMÕES INSTITUTO DE COMPUTAÇÃO

RELATÓRIO - PROGRAMAÇÃO 2 (MyFood)
Ciência da Computação

Matheus Lopes dos Santos

MACEIÓ - AL
2024

Introdução

O projeto desenvolvido é um sistema de gerenciamento para uma plataforma de pedidos e restaurantes, utilizando Programação Orientada a Objetos (POO) com a linguagem Java. O sistema adota o padrão de projeto Facade para simplificar a interface com o usuário e implementa persistência de dados para garantir que as informações sejam salvas e recuperadas de forma eficiente. Além disso, o código está

1. Programação Orientada a Objetos (POO)

A Programação Orientada a Objetos é um paradigma fundamental para o desenvolvimento do sistema, permitindo a modelagem dos dados e comportamentos de forma organizada e modular. O sistema utiliza conceitos de POO como encapsulamento, herança e polimorfismo:

- **Encapsulamento**: Os dados são encapsulados dentro de classes e objetos. Por exemplo, a classe Usuario encapsula os atributos e comportamentos comuns a todos os usuários, como id, nome, email, senha e endereco. Métodos como getAtributo e podeCriarEmpresa garantem o acesso controlado a essas informações.
- **Herança**: O sistema faz uso de herança para especializar tipos de usuários e produtos. As classes Cliente e DonoRestaurante estendem a classe Usuario, herdando seus atributos e métodos, mas adicionando comportamentos específicos. Da mesma forma, o padrão de projeto Facade é implementado por meio da classe Facade, que fornece uma interface simplificada para interagir com o sistema.
- **Polimorfismo**: O método getAtributo na classe Usuario é um exemplo de polimorfismo, pois pode retornar diferentes tipos de atributos com base na string fornecida. As classes que estendem Usuario implementam o método podeCriarEmpresa de maneira específica, demonstrando o polimorfismo em ação.

2. Uso da Linguagem Java

Java foi escolhido como a linguagem principal para o desenvolvimento do sistema devido a suas características robustas e amplamente aceitas no desenvolvimento de software:

- **Classes e Objetos**: O código foi estruturado em várias classes que representam diferentes entidades do sistema, como Usuario, Restaurante, Produto, e Pedido. Cada classe é responsável por encapsular dados e comportamentos específicos.
- **Exceções**: O sistema usa classes de exceções personalizadas, como AtributoInvalidoException, para lidar com condições de erro e garantir a

robustez do código. Isso permite que o sistema informe ao usuário sobre erros específicos de maneira controlada.
- **Serialização**: A implementação da interface Serializable em classes como Usuario permite que objetos sejam convertidos em um formato que pode ser salvo em disco e recuperado mais tarde, facilitando a persistência de dados.

3. Comentários e Documentação

O código do projeto é enriquecido com comentários detalhados que desempenham um papel crucial na sua manutenção e compreensão. Estes comentários estão presentes em diversas partes do código, incluindo:

- **Descrição de Métodos e Funções**: Cada método e função inclui comentários que explicam seu propósito, parâmetros e o valor retornado. Isso facilita a compreensão do que cada parte do código faz e como ela deve ser utilizada.
- **Exceções e Tratamento de Erros**: Comentários adicionais explicam as exceções lançadas e as condições que podem levar a erros, o que ajuda a identificar e resolver problemas de forma mais eficiente.
- **Explicações de Lógica e Algoritmos**: A lógica por trás de decisões importantes e algoritmos complexos é explicada com comentários que clarificam o raciocínio e a abordagem utilizada. Isso é especialmente útil para novos desenvolvedores que possam trabalhar no projeto no futuro.
- **Seções e Blocos de Código**: Comentários também são usados para dividir o código em seções distintas e descrever o que cada bloco faz. Isso melhora a legibilidade e a organização do código.

Esses comentários não apenas tornam o código mais acessível e compreensível, mas também garantem que a equipe de desenvolvimento possa realizar modificações e melhorias com maior eficiência e menor risco de introduzir erros.

4. Padrão de Projeto Facade

O padrão de projeto Facade é utilizado para fornecer uma interface simplificada para o sistema complexo subjacente:

- **Classe Facade**: A classe Facade atua como uma fachada que simplifica a interação com o sistema, oferecendo métodos para criar usuários, gerenciar produtos e pedidos, e consultar informações. Isso oculta a complexidade do sistema subjacente e fornece uma interface clara e intuitiva para os usuários e desenvolvedores.
- **Integração com o Sistema**: A Facade delega as solicitações para a classe Sistema, que implementa a lógica de negócio detalhada. Essa separação

entre a fachada e o sistema subjacente facilita a manutenção e a escalabilidade do sistema.

5. Persistência de Dados

O sistema utiliza a persistência de dados para garantir que as informações sejam armazenadas e recuperadas de forma eficaz:

- **Serialização de Objetos**: As classes Sistema e UsuarioSave, entre outras, implementam métodos para salvar e carregar dados de objetos em arquivos. Isso permite que o estado do sistema seja preservado entre execuções.
- **Métodos de Salvamento e Carregamento**: Métodos como UsuarioSave.carregarUsuarios() e ProdutoSave.salvarProdutos() são responsáveis por carregar e salvar dados em arquivos, garantindo que as informações persistam e possam ser recuperadas quando o sistema for reiniciado.

Conclusão

O projeto demonstra a aplicação eficaz de Programação Orientada a Objetos, a utilização da linguagem Java, o padrão de projeto Facade e técnicas de persistência de dados. A estrutura modular e a organização do código facilitam a manutenção e a escalabilidade, enquanto a persistência de dados assegura a continuidade do estado do sistema.

Este projeto não só exemplifica boas práticas de design de software, mas também fornece uma base sólida para futuras expansões e melhorias.