Universidade: Estácio de Sá

Campus: Estácio Interlagos

Curso: Desenvolvimento Full Stack

Disciplina: Iniciando o caminho pelo Java

Número da Turma: 2023.3

Semestre Letivo: 3° Semestre

Integrantes da Prática: Lucas Henrique Silva Santos

Relatório de Prática: Implementação de um cadastro de clientes em modo texto, com persistência em arquivos, baseado na tecnologia Java.

Título da Prática: Criação das Entidades e Sistema de Persistência

Objetivo da Prática:

1. Utilizar herança e polimorfismo na definição de entidades.
2. Utilizar persistência de objetos em arquivos binários.
3. Implementar uma interface cadastral em modo texto.
4. Utilizar o controle de exceções da plataforma Java.
5. No final do projeto, o aluno terá implementado um sistema cadastral em Java, utilizando os recursos da programação orientada a objetos e a persistência em arquivos binários.

Códigos da Prática

Este relatório descreve o desenvolvimento de um sistema cadastral em Java, no qual foram aplicados os conceitos de herança, polimorfismo, persistência de objetos em arquivos binários e a implementação de uma interface cadastral em modo texto. O projeto foi criado como parte de um exercício acadêmico com o objetivo de praticar programação orientada a objetos e persistência de dados em Java.

Herança e Polimorfismo: No projeto, utilizei herança para criar uma hierarquia de classes que representam entidades no sistema cadastral. A classe base Pessoa possui os atributos comuns a todas as entidades, como id e nome. As classes PessoaFisica e PessoaJuridica herdam de Pessoa e adicionam campos específicos, como cpf e idade para pessoas físicas e cnpj para pessoas jurídicas.

O polimorfismo foi aplicado no método exibir, que é polimórfico nas classes PessoaFisica e PessoaJuridica.

Persistência em Arquivos Binários: Para persistir objetos no sistema, implementei repositórios separados para PessoaFisica e PessoaJuridica. Esses repositórios utilizam arquivos binários para armazenar e recuperar os dados das entidades. O método persistir grava os objetos em um arquivo binário, enquanto o método recuperar lê os objetos do arquivo binário e os carrega de volta para a memória.

Interface Cadastral em Modo Texto: Desenvolvi uma interface cadastral em modo texto que permite ao usuário interagir com o sistema de forma intuitiva. O programa exibe um menu de opções, incluindo inserção, alteração, exclusão e consulta de entidades. O usuário pode escolher o tipo de entidade (Pessoa Física ou Pessoa Jurídica), fornecer os dados solicitados e realizar as operações desejadas.

Controle de Exceções em Java: Implementei tratamento de exceções para lidar com situações imprevistas durante a execução, garantindo que o programa não quebre inesperadamente.

Resultados da Execução: Durante os testes, o sistema demonstrou eficácia na criação, edição, exclusão e consulta de entidades. Os dados foram corretamente persistidos em arquivos binários, permitindo a recuperação após a reinicialização do programa. O controle de exceções garantiu uma experiência de usuário robusta e livre de falhas.

Este projeto nos proporcionou uma valiosa experiência de desenvolvimento em Java e a oportunidade de aplicar conceitos teóricos em um contexto prático.

Análise e Conclusão

Quais as vantagens e desvantagens do uso de herança?

Reutilização de Código: A herança permite que as classes filhas herdem os atributos e métodos da classe pai, o que promove a reutilização de código.

Polimorfismo: As classes derivadas podem ser tratadas como instâncias da classe base, permitindo a criação de código mais flexível e genérico.

Organização Hierárquica: A herança pode ser usada para criar uma estrutura hierárquica que reflete as relações do mundo real, facilitando o design orientado a objetos.

Por que a interface Serializable é necessária ao efetuar persistência em arquivos binários?

Acoplamento: Herança pode levar a um alto acoplamento entre classes pai e filhas, tornando as mudanças em uma classe pai potencialmente impactantes em todas as classes filhas.

Herança Múltipla Complexa: Em linguagens que suportam herança múltipla, pode surgir complexidade quando uma classe herda de várias classes, levando a ambiguidades e dificuldades de manutenção.

Fragilidade da Hierarquia: Alterações na hierarquia de herança podem afetar inesperadamente o comportamento das classes derivadas, resultando em problemas de manutenção.

Interface Serializable na Persistência em Arquivos Binários: A interface Serializable em Java é necessária ao efetuar persistência em arquivos binários porque ela indica ao mecanismo de serialização que uma classe pode ser convertida em uma sequência de bytes, que pode ser gravada em um arquivo e posteriormente reconstruída. Sem a implementação da interface Serializable, uma classe não pode ser serializada e, portanto, não pode ser armazenada em arquivos binários.

Como o paradigma funcional é utilizado pela API stream no Java?

Paradigma Funcional na API Stream do Java:

A API Stream do Java utiliza o paradigma funcional para executar operações em coleções de dados de forma concisa e expressiva. Ela introduz conceitos como funções de alta ordem, lambda expressions e operações como map, filter, reduce e forEach, que permitem a manipulação de coleções de maneira funcional e declarativa. Isso torna o código mais legível e facilita a execução de operações complexas em coleções de dados.

Quando trabalhamos com Java, qual padrão de desenvolvimento é adotado na persistência de dados em arquivos?

Paradigma Funcional na API Stream do Java:

A API Stream do Java utiliza o paradigma funcional para executar operações em coleções de dados de forma concisa e expressiva. Ela introduz conceitos como funções de alta ordem, lambda expressions e operações como map, filter, reduce e forEach, que permitem a manipulação de coleções de maneira funcional e declarativa. Isso torna o código mais legível e facilita a execução de operações complexas em coleções de dados.

Padrão de Desenvolvimento na Persistência de Dados em Arquivos em Java:

Quando se trabalha com Java, um padrão comum de desenvolvimento na persistência de dados em arquivos é a utilização de classes de entrada e saída (I/O) do Java, como FileInputStream, FileOutputStream, ObjectInputStream e ObjectOutputStream. Essas classes fornecem métodos para ler e escrever dados em arquivos de forma eficiente. Para a persistência em arquivos binários, a implementação da interface Serializable é frequentemente usada para permitir a serialização e desserialização de objetos em arquivos binários. Além disso, práticas como tratamento de exceções e fechamento adequado de recursos são adotadas para garantir a integridade dos dados e a segurança durante a operação de leitura e escrita em arquivos.

Repositório GIT: https://github.com/LucasHSS904/CadastroPOO

Data de Elaboração: 06/09/2023