Universidade Estácio de Sá

Paulo amancio de souza

Desenvolvimento Full Stack

Iniciando o Caminho Pelo Java
Mundo 3

Palhoça
2024

Acesso Github:https://github.com/willsouzaa/Cadastro-de-clientes-java.git

Título da Prática

Implementação de um Cadastro de Clientes em Modo Texto com Persistência em Arquivos Binários Usando Java

Objetivo da Prática

O objetivo desta prática é implementar um sistema de cadastro de clientes em modo texto utilizando a linguagem de programação Java, com persistência em arquivos binários. Os objetivos específicos incluem:

- Utilizar herança e polimorfismo para definir entidades.
- Implementar persistência de objetos em arquivos binários.
- Desenvolver uma interface de cadastro em modo texto.
- Aplicar o controle de exceções da plataforma Java.

Desenvolvimento da Prática

1. Criação do Projeto

1. Criar um projeto Ant Java Application no NetBeans com o nome `CadastroPOO`.
2. Criar um pacote chamado `model` para as entidades e gerenciadores.

2. Criação das Entidades

Classe Pessoa

- Campos: `id` (inteiro) e `nome` (texto)
- Métodos: `exibir`, construtores padrão e completo, getters e setters

Classe PessoaFisica

- Herda de Pessoa
- Campos adicionais: `cpf` (texto) e `idade` (inteiro)
- Método: `exibir` (polimórfico), construtores, getters e setters

Classe PessoaJuridica

- Herda de Pessoa
- Campo adicional: `cnpj` (texto)
- Método: `exibir` (polimórfico), construtores, getters e setters

3. Criação dos Gerenciadores

Classe PessoaFisicaRepo

- ArrayList de `PessoaFisica`

- Métodos: `inserir`, `alterar`, `excluir`, `obter`, `obterTodos`, `persistir`, `recuperar`

Classe PessoaJuridicaRepo

- ArrayList de `PessoaJuridica`
- Métodos: `inserir`, `alterar`, `excluir`, `obter`, `obterTodos`, `persistir`, `recuperar`

4. Testando o Sistema na Classe Principal

- Instanciar um repositório de pessoas físicas (`repo1`)
- Adicionar duas pessoas físicas usando o construtor completo
- Persistir os dados em um arquivo
- Instanciar outro repositório de pessoas físicas (`repo2`)
- Recuperar os dados do arquivo
- Exibir os dados das pessoas físicas recuperadas
- Repetir os mesmos passos para pessoas jurídicas (`repo3` e `repo4`)

Resultados da Execução dos Códigos

Após a execução dos códigos:

- Dados de duas pessoas físicas são armazenados e recuperados do arquivo `pessoasJisicas.dat`.
- Dados de duas pessoas jurídicas são armazenados e recuperados do arquivo `pessoasJuridicas.dat`.
- Os dados recuperados são exibidos no console, confirmando a funcionalidade do sistema de persistência.

Análise e Conclusão

Vantagens e Desvantagens do Uso de Herança

- **Vantagens**:
  - **Reutilização de Código**: Permite compartilhar código comum entre classes, evitando duplicação.
  - **Organização**: Melhora a estrutura e a legibilidade do código.
  - **Extensibilidade**: Facilita a adição de novas funcionalidades sem modificar o código existente.
- **Desvantagens**:
  - **Relação Forte**: Cria uma dependência forte entre a classe base e as classes derivadas.
  - **Complexidade**: Pode adicionar complexidade ao sistema, dificultando a manutenção.
  - **Fragilidade**: Mudanças na classe base podem impactar todas as classes derivadas.

Necessidade da Interface Serializable

A interface `Serializable` é necessária para que os objetos possam ser convertidos em um formato que permita sua escrita em arquivos binários. Sem essa interface, a JVM não sabe como transformar um objeto em uma sequência de bytes, necessária para a persistência em arquivos.

Uso do Paradigma Funcional com API Stream no Java

A API Stream do Java permite operações funcionais sobre coleções de dados. Usando streams, é possível realizar operações como map, filter e reduce de maneira declarativa e concisa. Isso facilita a manipulação e processamento de dados, promovendo um estilo de programação mais funcional e expressivo.

Padrão de Desenvolvimento na Persistência de Dados em Arquivos

Quando trabalhamos com persistência de dados em arquivos usando Java, é comum adotar o padrão de serialização e desserialização de objetos. Esse padrão envolve o uso das classes `ObjectOutputStream` e `ObjectInputStream` para escrever e ler objetos em arquivos binários. A abordagem permite a persistência de objetos complexos e a recuperação de seu estado de forma eficiente.

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