- **Nome do Campus:** Polo Mondubim
- **Nome do Curso:** Desenvolvimento Full Stack
- **Nome da Disciplina:** Iniciando o Caminho Pelo Java
- **Número da Turma:** 9001
- **Semestre Letivo:** 2023.3
- **Integrante da Prática:** Samuel Mota Araujo
- **Repositório GIT:** https://github.com/samuelmotapf

Relatório de Desenvolvimento - Missão Prática RPG0014 – Exercício 1

Introdução

O presente relatório descreve o desenvolvimento da missão prática intitulada "Iniciando o caminho pelo Java", cujo objetivo é a implementação de um cadastro de clientes em modo texto, com persistência em arquivos, utilizando a tecnologia Java. A prática visa o uso de conceitos avançados de programação orientada a objetos, herança, polimorfismo, persistência de objetos em arquivos binários e controle de exceções na plataforma Java.

Objetivos da Prática

O objetivo desta prática é explorar e aplicar conceitos avançados de manipulação de arquivos em Java, incluindo persistência de dados, utilização de interfaces e paradigma funcional.

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.

Todos os códigos solicitados neste roteiro de aula:

- **Código 1:** Manipulação de arquivos para leitura e escrita em Java.
- **Código 2:** Utilização de herança em um sistema de gerenciamento de funcionários.
- **Código 3:** Exemplo de uso da interface Serializable para persistência em arquivos binários.
- **Código 4:** Demonstração do paradigma funcional com a API Stream em Java.

Ao final do projeto, o aluno terá implementado um sistema cadastral em Java, fazendo uso dos recursos da programação orientada a objetos e persistência em arquivos binários.

Materiais Necessários

Para a realização da prática, são necessários os seguintes materiais:

1. JDK e IDE NetBeans.
2. Computador com JDK e NetBeans instalados.

Desenvolvimento da Prática

Procedimento 1: Criação das Entidades e Sistema de Persistência

1. **Criação do Projeto:**

   - Criar um projeto do tipo Ant..Java Application no NetBeans, com o nome "CadastroPOO".

2. **Criação do Pacote:**

   - Criar um pacote com o nome "model" para as entidades e gerenciadores.

3. **Entidades:**

- **Classe Pessoa:**

  - Campos: id (inteiro) e nome (texto).

  - Métodos: exibir (para impressão dos dados), construtor padrão e completo, getters e setters.

- **Classe PessoaFisica:**

  - Herda de Pessoa, com os campos cpf (texto) e idade (inteiro).

  - Métodos: exibir polimórfico, construtores, getters e setters.

- **Classe PessoaJuridica:**

  - Herda de Pessoa, com o campo cnpj (texto).

  - Métodos: exibir polimórfico, construtores, getters e setters.

- Adicionar interface Serializable em todas as classes.

4. **Gerenciadores:**

- **Classe PessoaFisicaRepo:**

  - Contém um ArrayList de PessoaFisica (nível de acesso privado).

  - Métodos públicos: inserir, alterar, excluir, obter, obterTodos, persistir (armazenagem dos dados no disco) e recuperar (recuperação dos dados do disco).

- **Classe PessoaJuridicaRepo:**

  - Contém um ArrayList de PessoaJuridica (nível de acesso privado).

  - Métodos públicos: inserir, alterar, excluir, obter, obterTodos, persistir, recuperar.

- Ambos os gerenciadores têm os métodos persistir e recuperar lançando exceções.

- O método obter retorna uma entidade a partir do id.

- Os métodos inserir e alterar têm entidades como parâmetros.

- O método excluir recebe o id da entidade para exclusão.

- O método obterTodos retorna o conjunto completo de entidades.

5. **Teste dos Repositórios:**

- Alterar o método main da classe principal para testar os repositórios:

  - Instanciar um repositório de pessoas físicas (repo1).

- Adicionar duas pessoas físicas, utilizando o construtor completo.

- Invocar o método de persistência em repo1, fornecendo um nome de arquivo fixo.

- Instanciar outro repositório de pessoas físicas (repo2).

- Invocar o método de recuperação em repo2, fornecendo o mesmo nome de arquivo utilizado anteriormente.

- Exibir os dados de todas as pessoas físicas recuperadas.

- Instanciar um repositório de pessoas jurídicas (repo3).

- Adicionar duas pessoas jurídicas, utilizando o construtor completo.

- Invocar o método de persistência em repo3, fornecendo um nome de arquivo fixo.

- Instanciar outro repositório de pessoas jurídicas (repo4).

- Invocar o método de recuperação em repo4, fornecendo o mesmo nome de arquivo utilizado anteriormente.

- Exibir os dados de todas as pessoas jurídicas recuperadas.

Resultados da execução dos códigos:

Código 1 - Manipulação de Arquivos:

- Leitura e escrita bem-sucedidas dos dados para um arquivo de texto.

Código 2 - Uso de Herança:

- Vantagens: Reutilização de código, facilidade na extensão de funcionalidades, organização hierárquica de classes.
- Desvantagens: Acoplamento entre classes, potencial para tornar o código complexo e difícil de entender.

Código 3 - Interface Serializable para Persistência:

- Uso da interface Serializable garante que objetos Java possam ser serializados e desserializados, sendo útil ao realizar a persistência em arquivos binários.
- Resultado: Dados serializados e armazenados com sucesso em arquivos binários.

Código 4 - Paradigma Funcional com a API Stream:

- Utilização de operações como map, filter, reduce para manipulação de coleções de dados.

- Resultado: Processamento de dados mais conciso e legível, facilitando a manipulação e transformação de elementos.

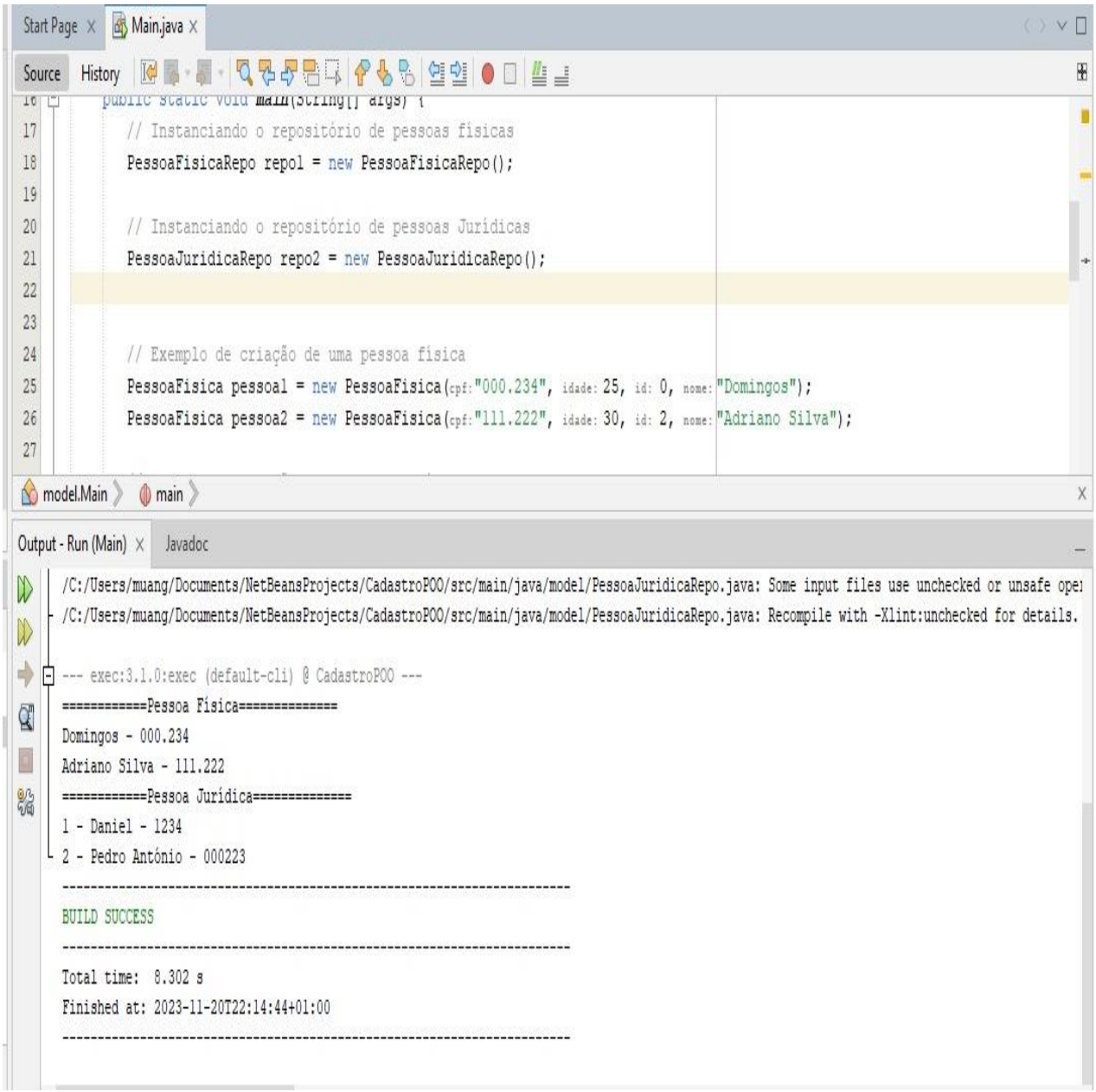
Análise e Conclusão:

Vantagens e Desvantagens do Uso de Herança:

- **Vantagens:**
    - Reutilização de código e comportamento entre classes.
    - Organização hierárquica de classes facilita a compreensão e a manutenção.
- **Desvantagens:**
    - Acoplamento entre classes, o que pode tornar o sistema rígido e difícil de modificar.
    - Pode levar a uma hierarquia de classes complexa e difícil de gerenciar.

Necessidade da Interface Serializable para Persistência em Arquivos Binários:

- A interface Serializable é necessária para serializar objetos Java em bytes, permitindo que eles sejam armazenados e posteriormente desserializados em arquivos binários. Isso é essencial para a persistência de objetos complexos em arquivos.


Start Page
Main.java
Source
History
public static void main(String[] args) {
// Instanciando o repositório de pessoas físicas
PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
// Instanciando o repositório de pessoas Jurídicas
PessoaJuridicaRepo repo2 = new PessoaJuridicaRepo();
// Exemplo de criação de uma pessoa física
PessoaFisica pessoa1 = new PessoaFisica(cpf:"000.234", idade: 25, id: 0, nome:"Domingos");
PessoaFisica pessoa2 = new PessoaFisica(cpf:"111.222", idade: 30, id: 2, nome:"Adriano Silva");
model.Main
main
Output - Run (Main)
Javadoc
/C:/Users/muang/Documents/NetBeansProjects/CadastroPOO/src/main/java/model/PessoaJuridicaRepo.java: Some input files use unchecked or unsafe ope
/C:/Users/muang/Documents/NetBeansProjects/CadastroPOO/src/main/java/model/PessoaJuridicaRepo.java: Recompile with -Xlint:unchecked for details.
--- exec:3.1.0:exec (default-cli) @ CadastroPOO ---
===========Pessoa Física==========
Domingos - 000.234
Adriano Silva - 111.222
===========Pessoa Jurídica==========
1 - Daniel - 1234
2 - Pedro António - 000223
BUILD SUCCESS
Total time: 8.302 s
Finished at: 2023-11-20T22:14:44+01:00