Criação das Entidades e Sistema de Persistência

Nome: Thomaz Alves Da Costa **Matrícula:** 202303196751

Campus: Campus Virtual (SIA)
Iniciando caminho pelo Java – 2023.1 – 3º Semestre

Objetivo da Prática

Criar entidades para realizar cadastro de pessoas físicas e pessoas jurídicas em Java, utilizando conceitos de POO (Programação orientada a objetos), herança, polimorfismo, e persistência de dados com tratamento de exceções.

1º Procedimento | Criação das Entidades e Sistema de Persistência

Classe Pessoa:

```java
package cadastropoo.model;

/**
 *
 * @author Thomaz Alves
 */
import java.io.Serializable;

public class Pessoa implements Serializable{
    // atributos
    private int id;
    private String nome;

    //Construtor padrao
    public Pessoa(){
        this.nome = "";
        this.id = 0;
    }

    // Constructor
    Pessoa(int id, String nome){
        this.id = id;
        this.nome = nome;
    }


    public void setId(int id){
        this.id = id;
    }

    public int getId(){
        return this.id;
    }

    public void setName(String nome){
        this.nome = nome;
    }

    public String getName(){
        return this.nome;
    }

    public void exibir(){
        System.out.println("nome: " + getName());
        System.out.println("Id: " + getId());
    }
}
```

Classe Pessoa fisica:

```java
package cadastropoo.model;

/**
 *
 * @author Thomaz Alves
 */
public class PessoaFisica extends Pessoa{
    // Atributos
    private String cpf;
    private int idade;

    // Constructor
    public PessoaFisica(String nome, int id, String cpf, int idade){
        super(id, nome);

        this.cpf = cpf;
        this.idade = idade;
    }



    public void setCpf(String cpf){
        this.cpf = cpf;
    }

    public String getCpf(){
        return this.cpf;
    }

    public void setIdade(int idade){
        this.idade = idade;
    }

    public Integer getIdade(){
        return this.idade;
    }

    @Override
    public void exibir() {
        System.out.println("Nome: " + getName());
        System.out.println("Id: " + getId());
        System.out.println("Idade: " + getIdade());
        System.out.println("Cpf: " + getCpf());
    }
}
```

Classe PessoaFisicaRepo:

```java
    public PessoaFisica obter(int id) {
        for (PessoaFisica pessoa : pessoaFisicas) {
            if (pessoa.getId() == id) {
                return pessoa;
            }
        }
        return null;
    }

    public ArrayList<PessoaFisica> obterTodos(){
        return pessoaFisicas;
    }

    public void persistir(String prefixo){
        try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("./" + prefixo + "fisica"))) {
            out.writeObject(pessoaFisicas);
        } catch (IOException e) {
            // Lançar uma nova exceção com uma mensagem personalizada

            System.err.println("Erro ao persistir os dados: " + e.getMessage());
        }
    }

    @SuppressWarnings("unchecked")
    public void recuperar(String prefixo) {
        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream("./"+ prefixo + "fisica"))) {
            Object obj = in.readObject();
            if (obj instanceof ArrayList<?>) {
                pessoaFisicas = (ArrayList<PessoaFisica>) obj;
            } else {
                throw new IOException("Arquivo não contém uma lista de pessoas físicas");
            }
        } catch(IOException e){
            System.err.println(e.getMessage());
        } catch(ClassNotFoundException e){
            System.err.println(e.getMessage());
        }
    }


}
```

```java
package cadastropoo.model;

/**
 *
 * @author Thomaz Alves
 */
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;

public class PessoaFisicaRepo {
    private ArrayList<PessoaFisica> pessoaFisicas = new ArrayList<>();

    public void inserir(PessoaFisica pessoaFisica){
        pessoaFisicas.add(pessoaFisica);
    }

    public void alterar(int id, PessoaFisica pessoaFisica){
        for(int i = 0; i < pessoaFisicas.size(); i++){
            if(pessoaFisicas.get(i).getId() == id){
                pessoaFisicas.set(i, pessoaFisica);
                break;
            }
        }
    }

    public void excluir(int id){
        for(int i = 0; i < pessoaFisicas.size(); i++){
            if(pessoaFisicas.get(i).getId() == id){
                pessoaFisicas.remove(i);
                break;
            }
        }
    }
```

Classe pessoaJuridica:

```java
package cadastropoo.model;

/**
 *
 * @author Thomaz Alves
 */
public class PessoaJuridica extends Pessoa{
    // Atributos
    private String cnpj;

    // Constructor
    public PessoaJuridica(String nome, int id, String cnpj){
        super(id, nome);

        this.cnpj = cnpj;
    }

    public void setCnpf(String cnpj){
        this.cnpj = cnpj;
    }

    public String getCnpf(){
        return cnpj;
    }

    @Override
    public void exibir() {
        System.out.println("Nome: " + getName());
        System.out.println("Id: " + getId());
        System.out.println("Cnpj: " + getCnpf());
    }
}
```

Classe PessoaJuridicaRepo

```java
package cadastropoo.model;

/**
 *
 * @author Thomaz Alves
 */
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;

public class PessoaJuridicaRepo {
    private ArrayList<PessoaJuridica> pessoasJuridicas = new ArrayList<>();

    // incluir
    public void incluir(PessoaJuridica pessoaJuridica){
        pessoasJuridicas.add(pessoaJuridica);
    }

    // Alterar
    public void alterar(int id, PessoaJuridica pessoaJuridica){
        for(int i = 0; i < pessoasJuridicas.size(); i++){
            if(pessoasJuridicas.get(i).getId() == id){
                pessoasJuridicas.set(i, pessoaJuridica);
                break;
            }
        }
    }

    // Excluir
    public void excluir(int id){
        for(int i = 0; i < pessoasJuridicas.size(); i++){
            if(pessoasJuridicas.get(i).getId() == id){
                pessoasJuridicas.remove(i);
                break;
            }
        }
    }
```

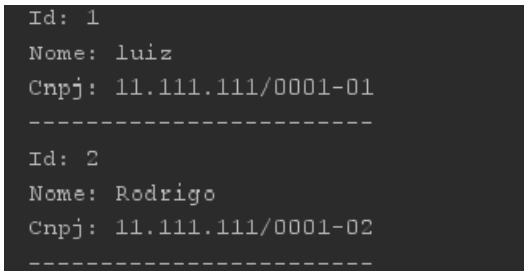
```java
    public PessoaJuridica obter(int id){
        for(PessoaJuridica pessoa : pessoasJuridicas){
            if(pessoa.getId() == id){
                return pessoa;
            }
        }
        return null;
    }

    // Obter todos
    public ArrayList<PessoaJuridica> obterTodos(){
        return pessoasJuridicas;
    }

    // Persistir dados
    public void persistir(String preFixo) {
        try(ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("./" + preFixo + "juridica"))){
            if(preFixo != ""){
                out.writeObject(pessoasJuridicas);
            }else{
                throw new IOException("Erro ao persistir os dados");
            }

        } catch(IOException e){
            System.err.println(e.getMessage());
        }
    }

    // Recuperar dados
    @SuppressWarnings("unchecked")
    public void recuperar(String preFixo) {
        try(ObjectInputStream in = new ObjectInputStream(new FileInputStream("./" + preFixo + "juridica"))){
            Object obj = in.readObject();
            if(obj instanceof ArrayList<?>){
                pessoasJuridicas = (ArrayList<PessoaJuridica>) obj;
            } else{
                throw new IOException("Não foi possivel encontrar o arquivo.");
            }
        }catch(IOException e){
            System.err.println(e.getMessage());
        } catch(ClassNotFoundException e){
            System.err.println(e.getMessage());
        }
    }
}
```

Resultados:

Instanciando um repositório de pessoas físicas, depois persistir os dados por meio do metodo de persistência e depois recuperar arquivo persistido. O resultado esperado está logo abaixo:

```
Nome: thomaz
Idade: 19
Cpf: 999-999-999-99
-------------------------
Id: 2
Nome: Luiz
Idade: 19
Cpf: 999-999-999-98
-------------------------
```

Instanciando um repositório de pessoas juridicas, depois persistir os dados por meio do metodo de persistência e depois recuperar arquivo persistido. O resultado esperado está logo abaixo:



Análise e Conclusão:

a) **Quais as vantagens e desvantagens do uso de herança?**

Reutilização de código, classes filhas podem herdar métodos e atributos da classe pai, reduzindo a redundância de código. Polimorfismo, permite tratar objetos de classes filhas como objetos da classe pai, o que facilita a criação de estruturas flexíveis e genéricas. Organização e abstração, ajuda a organizar e abstrair conceitos em hierarquias, tornando o código mais compreensível e manutenível.

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

Usamos a interface serializable para que os objetos Java sejam convertidos em uma sequência de bytes

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

O paradigma funcional Java fornece operações de processamentos de dados em coleções de forma expressiva e concisa.

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

Ao trabalhar com Java e persistência de dados em arquivos, um padrão comum é o uso de serialização e desserialização de objetos. Isso envolve a implementação da interface Serializable em classes cujas instâncias serão armazenadas nos arquivos. Em seguida, os objetos são gravados em arquivos usando ObjectOutputStream e lidos usando ObjectInputStream. Isso permite que os objetos sejam armazenados em arquivos binários e recuperados posteriormente, mantendo seu estado intacto.

2º Procedimento | **Criação do Cadastro em Modo Texto**

Resultados:

Inserindo Usuário:

Inserindo Pessoa Física:

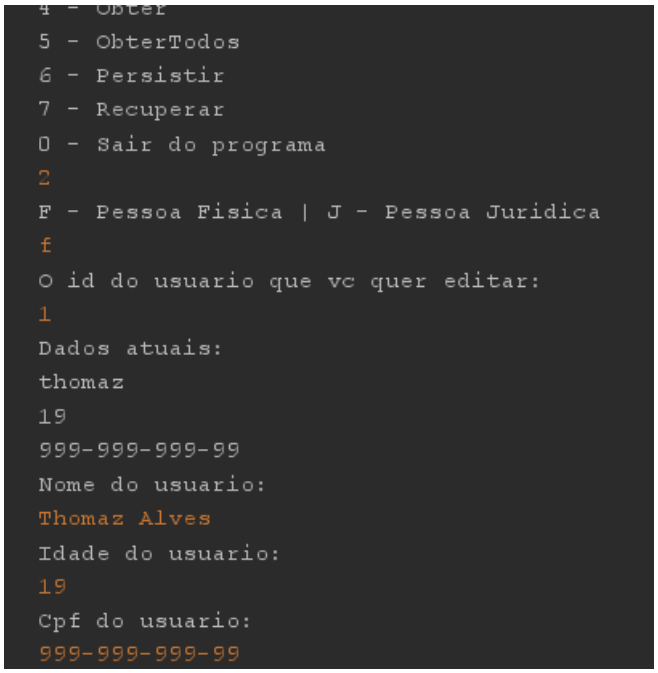
```
Escolha uma opcao:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Obter
5 - ObterTodos
6 - Persistir
7 - Recuperar
0 - Sair do programa
1
F - Pessoa Fisica | J - Pessoa Juridica
f
Id do usuario:
1
Nome do usuario
thomaz
Idade do usuario
19
Cpf do usuario
999-999-999-99
```

Inserindo Pessoa Jurídica:

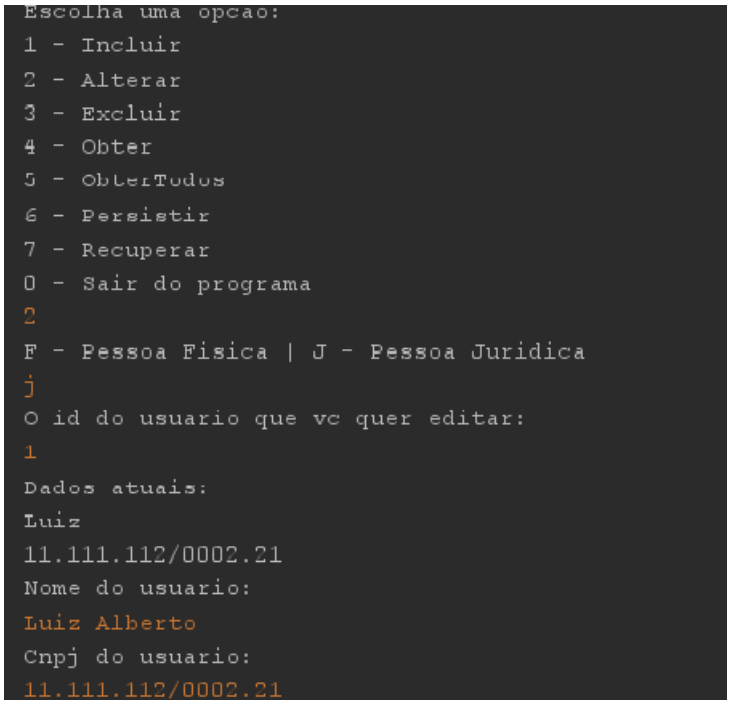
```
Escolha uma opcao:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Obter
5 - ObterTodos
6 - Persistir
7 - Recuperar
0 - Sair do programa
1
F - Pessoa Fisica | J - Pessoa Juridica
j
Id do usuario:
1
Nome do usuario
Luiz
Cnpj do usuario:
11.111.111/0001.21
```

Alterando Usuário:

Alterando Pessoa Física:



Alterando Pessoa Jurídica:



Obtendo Usuário pelo id:

Pessoa Física:

```
Escolha uma opcao:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Obter
5 - ObterTodos
6 - Persistir
7 - Recuperar
0 - Sair do programa
4
F - Pessoa Fisica | J - Pessoa Juridica
f
Digite o id da pessoa que deseja obter:
1
Nome: Thomaz Alves
Idade: 19
Cpf: 999-999-999-99
```

Pessoa Jurídica:

```
Escolha uma opcao:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Obter
5 - ObterTodos
6 - Persistir
7 - Recuperar
0 - Sair do programa
4
F - Pessoa Fisica | J - Pessoa Juridica
j
Digite o id da pessoa que deseja obter:
1
Nome: Luiz Alberto
Cnpj: 11.111.112/0002.21
```

Obtendo Todos os Usuário:

Pessoas Fisica:

```
Escolha uma opcao:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Obter
5 - ObterTodos
6 - Persistir
7 - Recuperar
0 - Sair do programa
5
F - Pessoa Fisica | J - Pessoa Juridica
f
Id: 1
Nome: Thomaz Alves
Idade: 19
Cpf: 999-999-999-99
------------------------
```

Pessoas Jurídica:

```
Escolha uma opcao:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Obter
5 - ObterTodos
6 - Persistir
7 - Recuperar
0 - Sair do programa
5
F - Pessoa Fisica | J - Pessoa Juridica
j
Id: 1
Nome: Luiz Alberto
Cnpj: 11.111.112/0002.21
------------------------
```

Persistindo e Recuperando

Pessoa Física:

Persistindo:

```
Escolha uma opcao:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Obter
5 - ObterTodos
6 - Persistir
7 - Recuperar
0 - Sair do programa
6
F - Pessoa Fisica   J - Pessoa Juridica
f
Digite o nome do arquivo
PessoasFisicas
```

Recuperando:

```
Escolha uma opcao:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Obter
5 - ObterTodos
6 - Persistir
7 - Recuperar
0 - Sair do programa
7
F - Pessoa Fisica | J - Pessoa Juridica
f
Digite o prefixo do arquivo:
PessoasFisicas
Escolha uma opcao:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Obter
5 - ObterTodos
6 - Persistir
7 - Recuperar
0 - Sair do programa
5
F - Pessoa Fisica | J - Pessoa Juridica
f
Id: 1
Nome: Thomaz Alves
Idade: 19
Cpf: 999-999-999-99
-------------------------
```

Pessoa Jurídica:

Persistindo:

```
Escolha uma opcao:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Obter
5 - ObterTodos
6 - Persistir
7 - Recuperar
0 - Sair do programa
6
F - Pessoa Fisica | J - Pessoa Juridica
j
Digite o nome do arquivo
PessoasJuridicas
```

Recuperando:

```
Escolha uma opcao:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Obter
5 - ObterTodos
6 - Persistir
7 - Recuperar
0 - Sair do programa
7
F - Pessoa Fisica | J - Pessoa Juridica
j
Digite o prefixo do arquivo:
PessoasJuridicas
Escolha uma opcao:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Obter
5 - ObterTodos
6 - Persistir
7 - Recuperar
0 - Sair do programa
5
F - Pessoa Fisica | J - Pessoa Juridica
j
Id: 1
Nome: Luiz Alberto
Cnpj: 11.111.112/0002.21
------------------------
```

Análise e Conclusão:

a) O que são elementos estáticos e qual o motivo para o método main adotar esse modificador?

Elementos estaticos são membros de uma classe que pertencem à própria classe, em vez de pertencerem a instâncias individuais dessa classe, no caso java utilizamos métodos estáaticos chamado de main.

O método **main** em Java é declarado como estático por uma razão importante: ele é o ponto de entrada para um programa Java e é invocado pelo Java Virtual Machine (JVM) sem a necessidade de criar uma instância da classe que contém o método **main**. Como resultado, o método **main** precisa ser estático para ser invocado diretamente pela JVM sem a necessidade de criar um objeto.

b) Para que serve a classe Scanner?

Pacote do java.util, responsável por ler entradas de dados dos usuários e armazena-las em variáveis.

c) Como o uso de classes de repositório impactou na organização do código?

Questões como manutenibilidade e organização do código, eu cito como os principais sobre este assunto.

Conclusão

Sobre a missão prática, achei um pouco difícil no começo devido que eu não estar acostumado a desenvolver em orientado a objetos em java, mas depois de praticar os passo a passo da missão conseguir aprender mais sobre programação orientada a obejetos na prática, o seu real motivo de uso, a organização que ele promove ao sistema como um todo, e como ele tambem pode me ajudar em outras linguagens.