Curso
Desenvolvimento full Stack
Disciplina
Nível 1: Iniciando o Caminho Pelo Java
Turma
3°Semestre
Campus
POLO INGL RIO VERM – FLORIANOPOLIS - SC
Nome
Yuri Friederick de Sousa Cunha Bernardo

Criação das Entidades e Sistema de Persistência

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

Classe Pessoa:

```
package model;

import java.io.Serializable;

public class Pessoa implements Serializable{

	private static final long serialVersionUID = 1L;
	private Integer id;
	private String nome;


	public Pessoa(Integer id, String nome) {
		this.id = id;
		this.nome = nome;
	}

	public Integer getId() {
		return id;
	}

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

	public String getNome() {
		return nome;
	}

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

}
```

Classe PessoaFisica:

```java
package model;

import java.io.Serializable;

public class PessoaFisica extends Pessoa implements Serializable {

	private static final long serialVersionUID = 1L;
	private String cpf;
	private int idade;

	public PessoaFisica(Integer id, String nome, String cpf, int idade) {
		super(id, nome);
		this.cpf = cpf;
		this.idade = idade;
	}

	@Override
	public String toString() {
		return String.format("Id: %d\nNome: %s\nCPF: %s\nIdade: %d ", getId(), getNome(), this.cpf, this.idade);
	}

	public String getCpf() {
		return cpf;
	}

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

	public int getIdade() {
		return idade;
	}

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

Classe PessoaJuridica:

```java
package model;

import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements Serializable {

	private static final long serialVersionUID = 1L;
	private String cnpj;


	public PessoaJuridica(Integer id, String nome, String cnpj) {
		super(id, nome);
		this.cnpj = cnpj;
	}

	@Override
	public String toString() {
		return String.format("Id: %d\nNome: %s\nCNPJ: %s", getId(), getNome(), this.cnpj);
	}

	public String getCnpj() {
		return cnpj;
	}

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


}
```

Classe PessoaFisicaRepo:

```java
package model;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class PessoaFisicaRepo {

	private PessoaFisica[] pessoaFisicaArray;

	public PessoaFisica[] getPessoaFisicacaArray() {
		return pessoaFisicaArray;
	}

	public void setPessoaFisicacaArray(PessoaFisica[] pessoaFisicacaArray) {
		this.pessoaFisicaArray = pessoaFisicacaArray;
	}

	public Boolean vetorVazio() {
		if (this.pessoaFisicaArray.length == 0) {
			System.out.println("Array Vazio!!!");
			return true;
		} else {
			return false;
		}
	}

	public void inserirPessoaVazio(Integer id, PessoaFisica... pessoaFisica) {
		if (id == 0) {
			System.out.println("Adição invalida");
		} else {
			this.pessoaFisicaArray = new PessoaFisica[id];
			int i = 0;
			for (PessoaFisica p : pessoaFisica) {
				if (i > (this.pessoaFisicaArray.length - 1)) {
					throw new ArrayIndexOutOfBoundsException("Número de registro excede o tamanho do Array!!!");
				} else {
					this.pessoaFisicaArray[i] = new PessoaFisica(p.getId(), p.getNome(), p.getCpf(), p.getIdade());
					i++;
				}
			}
		}
	}
```

```java
		if (this.pessoaFisicaArray.length == 0) {
			System.out.println("Necessario criar o Array primeiro!!!");
		} else {
			int i = 0;
			while ((i < this.pessoaFisicaArray.length) && (this.pessoaFisicaArray[i] != null)) {
				i++;
			}
			if (i > (this.pessoaFisicaArray.length - 1)) {
				System.out.println("Array cheio!!!");
			} else {
				if ((this.pessoaFisicaArray.length - i) < id) {
					System.out.println("Numero de registros excede o tamanho do Array!!!");
				} else {
					for (PessoaFisica p : pessoaFisica) {
						this.pessoaFisicaArray[i] = new PessoaFisica(p.getId(), p.getNome(), p.getCpf(), p.getIdade());
						i++;
					}
				}
			}
		}
	}

	public void alterar(PessoaFisica pessoaFisica) {
		int achado = 0;
		if (!vetorVazio()) {
			for (int i = 0; i < this.pessoaFisicaArray.length; i++) {
				if ((this.pessoaFisicaArray[i] != null)
						&& (this.pessoaFisicaArray[i].getId() == (pessoaFisica.getId()))) {
					this.pessoaFisicaArray[i] = pessoaFisica;
					achado++;
				}
			}
		}
		if (achado == 0) {
			System.out.println("Pessoa nao encontrada!!!");
		}
	}

	public void excluir(PessoaFisica id) {
		int achado = 0;
		if (!vetorVazio()) {
			for (int i = 0; i < this.pessoaFisicaArray.length; i++) {
				if ((this.pessoaFisicaArray[i] != null) && (this.pessoaFisicaArray[i].getId().equals(id.getId()))) {
					achado++;
					for (int j = i; j < this.pessoaFisicaArray.length; j++) {
						if (this.pessoaFisicaArray[j] != null)
							pessoaFisicaArray[j] = pessoaFisicaArray[j + 1];
					}
				}
			}
		}
		if (achado == 0) {
			System.out.println("Pessoa nao encontrada!!!");
		}
```

```
        if (!vetorVazio()) {
                for (int i = 0; i < this.pessoaFisicaArray.length; i++) {
                        if (pessoaFisicaArray[i] != null) {
                                System.out.println(pessoaFisicaArray[i]);
                        } else {
                        }
                }
        }
}

public static boolean persistir(String nomeArquivo, Object obj) throws IOException {
        File arquivo = new File(nomeArquivo);

        if (!arquivo.exists()) {
                try {
                        arquivo.createNewFile();
                } catch (Exception e) {
                        e.printStackTrace();
                        return false;
                }
        }

        try {
                FileOutputStream escreverArq = new FileOutputStream(arquivo);
                ObjectOutputStream inserirObj = new ObjectOutputStream(escreverArq);
                inserirObj.writeObject(obj);

                inserirObj.flush();
                escreverArq.flush();

                inserirObj.close();
                escreverArq.close();
                System.out.println("Dados de Pessoa Fisica Armazenados.");

                return true;
        } catch (Exception e) {
                e.printStackTrace();
                return false;
        }

}

public static Object recuperar(String caminho) {
        File arquivo = new File(caminho);

        try {
                FileInputStream recDados = new FileInputStream(arquivo);
                ObjectInputStream objInput = new ObjectInputStream(recDados);

                Object retorno = objInput.readObject();

                objInput.close();
                recDados.close();
                System.out.println("Dados de Pessoa Fisica Recuperados.");
                return retorno;
        } catch (Exception e) {
```

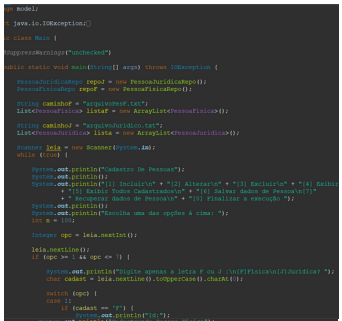
Classe PessoaJuridicaRepo:

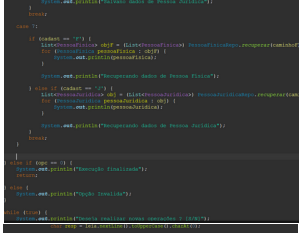
```
package model;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class PessoaJuridicaRepo {

	private PessoaJuridica[] pessoaJuridicaArray;

	public PessoaJuridica[] getPessoaJuridica() {
		return pessoaJuridicaArray;
	}

	public void setPessoaJuridica(PessoaJuridica[] pessoaJuridicaArray) {
		this.pessoaJuridicaArray = pessoaJuridicaArray;
	}

	public Boolean vetorVazio() {
		if (this.pessoaJuridicaArray.length == 0) {
			System.out.println("Array Vazio!!!");
			return true;
		} else {
			return false;
		}
	}

	public void inserirPessoaVazio(Integer id, PessoaJuridica... pessoaJuridica) {
		if (id == 0) {
			System.out.println("Adição invalida");
		} else {
			this.pessoaJuridicaArray = new PessoaJuridica[id];
			int i = 0;
			for (PessoaJuridica p : pessoaJuridica) {
				if (i > (this.pessoaJuridicaArray.length - 1)) {
					throw new ArrayIndexOutOfBoundsException("Número de registro excede o tamanho do Array!!!");
				} else {
					this.pessoaJuridicaArray[i] = new PessoaJuridica(p.getId(), p.getNome(), p.getCnpj());
					i++;
				}
			}
		}
	}
```

```
	public void adicionarPessoa(Integer id, PessoaJuridica... pessoaJuridicas) {
		if (this.pessoaJuridicaArray.length == 0) {
			System.out.println("Necessario criar o Array primeiro!!!");
		} else {
			int i = 0;
			while ((i < this.pessoaJuridicaArray.length) && (this.pessoaJuridicaArray[i] != null)) {
				i++;
			}
			if (i > (this.pessoaJuridicaArray.length - 1)) {
				System.out.println("Array cheio!!!");
			} else {
				if ((this.pessoaJuridicaArray.length - i) < id) {
					System.out.println("Numero de registros excede o tamanho do Array!!!");
				} else {
					for (PessoaJuridica p : pessoaJuridicas) {
						this.pessoaJuridicaArray[i] = new PessoaJuridica(p.getId(), p.getNome(), p.getCnpj());
						i++;
					}
				}
			}
		}
	}

	public void alterar(PessoaJuridica pessoaJuridica) {
		int achado = 0;
		if (!vetorVazio()) {
			for (int i = 0; i < this.pessoaJuridicaArray.length; i++) {
				if ((this.pessoaJuridicaArray[i] != null)
						&& (this.pessoaJuridicaArray[i].getId() == (pessoaJuridica.getId()))) {
					this.pessoaJuridicaArray[i] = pessoaJuridica;
					achado++;
				}
			}
		}
		if (achado == 0) {
			System.out.println("Pessoa nao encontrada!!!");
		}
	}

	public void excluir(PessoaJuridica id) {
		int achado = 0;
		if (!vetorVazio()) {
			for (int i = 0; i < this.pessoaJuridicaArray.length; i++) {
				if ((this.pessoaJuridicaArray[i] != null) && (this.pessoaJuridicaArray[i].getId().equals(id.getId()))) {
					achado++;
					for (int j = i; j < this.pessoaJuridicaArray.length; j++) {
						if (this.pessoaJuridicaArray[j] != null)
							pessoaJuridicaArray[j] = pessoaJuridicaArray[j + 1];
					}
				}
			}
		}
		if (achado == 0) {
			System.out.println("Pessoa nao encontrada!!!");
		}
	}
```

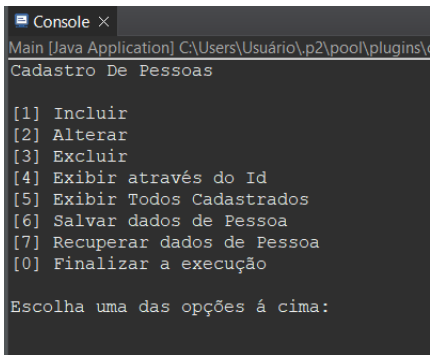
```
public void obterTodos() {
        if (!vetorVazio()) {
                for (int i = 0; i < this.pessoaJuridicaArray.length; i++) {
                        if (pessoaJuridicaArray[i] != null) {
                                System.out.println(pessoaJuridicaArray[i]);
                        } else {
                        }
                }
        }
}

public static boolean persistir(String nomeArquivo, Object obj) throws IOException {
        File arquivo = new File(nomeArquivo);

        if (!arquivo.exists()) {
                try {
                        arquivo.createNewFile();
                } catch (Exception e) {
                        e.printStackTrace();
                        return false;
                }
        }

        try {
                FileOutputStream escreverArq = new FileOutputStream(arquivo);
                ObjectOutputStream inserirObj = new ObjectOutputStream(escreverArq);
                inserirObj.writeObject(obj);

                inserirObj.flush();
                escreverArq.flush();

                inserirObj.close();
                escreverArq.close();
                System.out.println("Dados de Pessoa Juridica Armazenados.");

                return true;
        } catch (Exception e) {
                e.printStackTrace();
                return false;
        }

}

public static Object recuperar(String caminho) {
        File arquivo = new File(caminho);

        try {
                FileInputStream recDados = new FileInputStream(arquivo);
                ObjectInputStream objInput = new ObjectInputStream(recDados);

                Object retorno = objInput.readObject();

                objInput.close();
                recDados.close();
                System.out.println("Dados de Pessoa Juridica Recuperados.");
                return retorno;
        } catch (Exception e) {
```

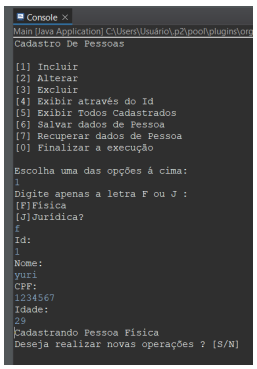
Classe Principal:

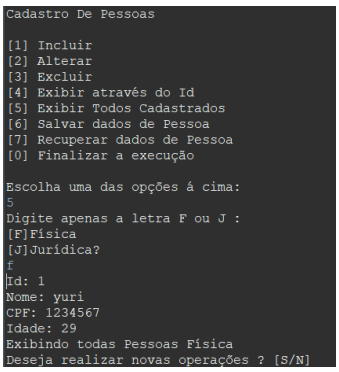


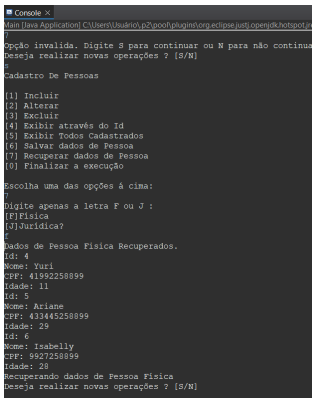


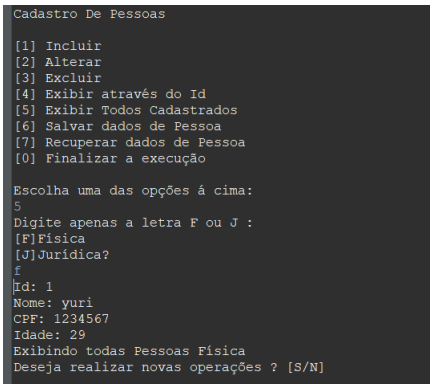
Resultados da execução dos códigos:

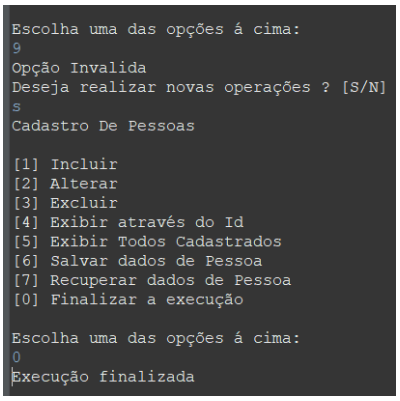












Análise e Conclusão:

Para resumir, a aplicação do cadastro de clientes em formato de texto com armazenamento em arquivos, utilizando a tecnologia Java e as classes Pessoa, PessoaFisica e PessoaJuridica, juntamente com as classes de repositório PessoaFisicaRepo e PessoaJuridicaRepo, resultou em um sistema bem estruturado e modular. A inclusão das classes de repositório teve um impacto positivo na organização do código, facilitando a manutenção e possibilitando a expansão futura do sistema. A utilização de elementos estáticos, como o método main, demonstra a importância do ponto de entrada para o programa. Em conclusão, a cuidadosa estruturação do código e a seleção de componentes adequados refletem a solidez e eficácia do sistema desenvolvido.

O uso de herança em Java tem vantagens como reutilização de código, mas desvantagens como acoplamento forte. A interface Serializable é crucial para persistência em arquivos binários. A API stream aplica o paradigma funcional para operações eficientes em coleções. A persistência de dados em arquivos em Java frequentemente usa a API de E/S. Esses conceitos fundamentais são essenciais para a programação eficaz em Java.

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

Em Java, os elementos estáticos (ou membros estáticos) estão ligados à classe em vez das instâncias individuais da classe. Eles pertencem à classe como um todo, não a objetos específicos criados a partir dela.
Exemplos de elementos estáticos incluem variáveis estáticas (campos), métodos estáticos e blocos estáticos.

O método Main é declarado como estático porque é o ponto de entrada do programa e precisa ser acessível sem a criação de uma instância da classe.

Para que serve a classe Scanner?

A classe Scanner em Java é usada para ler dados de entrada do usuário.

Ela permite que o programa interaja com o usuário, lendo valores digitados no console.

É frequentemente usada para ler entradas do teclado, como números inteiros, números de ponto flutuante, strings, etc.

A classe Scanner é uma ferramenta muito útil para a interação entre o programa e o usuário. Com ela, podemos capturar informações digitadas no console de forma simples e eficiente. Além de ler números e strings, também é possível utilizar o Scanner para interpretar comandos e realizar ações específicas de acordo com o que é inserido pelo usuário.

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

A introdução das classes de repositório (como PessoaFisicaRepo e PessoaJuridicaRepo) ajudou a organizar o código de forma mais modular e coesa.

Cada classe de repositório é responsável por operações específicas relacionadas a pessoas físicas ou jurídicas.

Isso facilita a manutenção, reutilização e expansão do código, pois cada classe tem uma única responsabilidade bem definida.