



Desenvolvimento Full Stack

Aluno: Victor de A. Costa

Missão Prática | Nível 1 | Mundo 3 - 2024.3
INICIANDO O CAMINHO PELO JAVA

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

2º Procedimento | Criação do Cadastro em Modo Texto

---> CÓDIGOS UTILIZADOS

GIT: github.com/VictorDSGNR/missao_mundo03_nivel01

CadastroPOOttext.java

```
package model;
/**
 *
 * @author victoracosta
 */
import java.io.IOException;
import java.util.Scanner;
import java.util.List;
import java.io.File;
import java.io.ObjectOutputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.FileInputStream;

public class CadastroPOOttext {
    private static Scanner scanner = new Scanner(System.in);
    private static PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
    private static PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();
    public static void main(String[] args) {
        try {
            repoFisica.recuperar("pessoasFisicas.dat");
        } catch (IOException | ClassNotFoundException e) {
        }
        try {
            repoJuridica.recuperar("pessoasJuridicas.dat");
        } catch (IOException | ClassNotFoundException e) {
        }
        int opcao;
        do {
            System.out.println("=====");
            System.out.println("1- Incluir Pessoa");
            System.out.println("2- Alterar Pessoa");
            System.out.println("3- Excluir Pessoa");
            System.out.println("4- Buscar pelo Id");
            System.out.println("5- Exibir Todos");
            System.out.println("6- Persistir Dados");
            System.out.println("7- Recuperar Dados");
            System.out.println("0- Finalizar Programa");
            System.out.println("=====");
            opcao = scanner.nextInt();
            switch (opcao) {
                case 1:
                    incluirPessoa();
                    break;
                case 2:
                    alterarPessoa();
                    break;
                case 3:
                    excluirPessoa();
                    break;
                case 4:
                    buscarPeloid();
                    break;
            }
        } while (opcao != 0);
    }
}
```

```
case 5:  
exibirTodos();  
break;  
case 6:  
persistirDados();  
break;  
case 7:  
recuperarDados();  
break;  
case 0:  
System.out.println("Saindo...");  
break;  
default:  
System.out.println("Opção inválida!");  
}  
}  
} while (opcao!= 0);  
}  
private static void incluirPessoa() {  
String tipo;  
int id;  
String nome;  
String cpf;  
int idade;  
String cnpj;  
System.out.println("F- Pessoa Física | J- Pessoa Jurídica");  
tipo = scanner.next().toUpperCase();  
if (!tipo.equals("J") && !tipo.equals("F")) {  
System.out.println("Opção inválida!");  
return;  
}  
System.out.println("Digite o id da Pessoa:");  
id = scanner.nextInt();  
System.out.println("Insira os dados...");  
System.out.println("Nome:");  
nome = scanner.next();  
if (tipo.equals("J")) {  
System.out.println("CNPJ:");  
cnpj = scanner.next();  
PessoaJurídica pessoaJ = new PessoaJurídica(id, nome, cnpj);  
repoJurídica.inserir(pessoaJ);  
saveData("J");  
}  
if (tipo.equals("F")) {  
System.out.println("CPF:");  
cpf = scanner.next();  
System.out.println("Idade:");  
idade = scanner.nextInt();  
PessoaFísica pessoaF = new PessoaFísica(id, nome, cpf, idade);  
repoFísica.inserir(pessoaF);  
saveData("F");  
}  
}  
private static void saveData(String tipo) {  
if (tipo.equals("J")) {  
try {  
repoJurídica.persistir("pessoasJurídicas.dat");  
} catch (IOException e) {  
}  
}  
if (tipo.equals("F")) {  
try {  
repoFísica.persistir("pessoasFísicas.dat");  
} catch (IOException e) {  
}  
}  
}  
private static void exibirTodos() {  
System.out.println("Dados de Pessoas Físicas:");  
List<PessoaFísica> pessoasFísicas = repoFísica.obterTodos();  
if (!pessoasFísicas.isEmpty()) {  
for (PessoaFísica pf : pessoasFísicas) {  
pf.exibir();  
System.out.println("");  
}
```

```

} else {
    System.out.println("Nenhuma Pessoa Fisica");
}
System.out.println("-----");
System.out.println("Dados de Pessoas Juridicas:");
List<PessoaJuridica> pessoaJuridicas = repoJuridica.obterTodos();
if (!pessoaJuridicas.isEmpty()) {
    for (PessoaJuridica pf : pessoaJuridicas) {
        pf.exibir();
        System.out.println("");
    }
} else {
    System.out.println("Nenhuma Pessoa Juridica");
}
}

private static void buscarPeloid() {
String tipo;
int id;
System.out.println("F- Pessoa Fisica | J- Pessoa Juridica");
tipo = scanner.next().toUpperCase();
if (!tipo.equals("J") &&!tipo.equals("F")) {
    System.out.println("Opção inválida!");
    return;
}
System.out.println("Digite o id da Pessoa:");
id = scanner.nextInt();
if (tipo.equals("J")) {
    PessoaJuridica pessoa = repoJuridica.obter(id);
    if (pessoa == null) {
        System.out.println("Pessoa não encontrada");
    } else {
        System.out.println("Pessoa Juridica:");
        pessoa.exibir();
    }
}
if (tipo.equals("F")) {
    PessoaFisica pessoa = repoFisica.obter(id);
    if (pessoa == null) {
        System.out.println("Pessoa não encontrada");
    } else {
        System.out.println("Pessoa Fisica:");
        pessoa.exibir();
    }
}
}

private static void excluirPessoa() {
String tipo;
int id;
System.out.println("F- Pessoa Fisica | J- Pessoa Juridica");
tipo = scanner.next().toUpperCase();
if (!tipo.equals("J") &&!tipo.equals("F")) {
    System.out.println("Opção inválida!");
    return;
}
System.out.println("Digite o id da Pessoa:");
id = scanner.nextInt();
if (tipo.equals("J")) {
    PessoaJuridica pessoa = repoJuridica.obter(id);
    if (pessoa == null) {
        System.out.println("Pessoa não encontrada");
    } else {
        repoJuridica.excluir(id);
        System.out.printf("A pessoa com a id:%d e o nome: %s foi excluída%n",
        pessoa.getId(), pessoa.getNome());
        saveData("J");
    }
}
if (tipo.equals("F")) {
    PessoaFisica pessoa = repoFisica.obter(id);
    if (pessoa == null) {
        System.out.println("Pessoa não encontrada");
    }
}
}

```

```

} else {
    repoFisica.excluir(id);
    System.out.printf("A pessoa com a id:%d e o nome: %s foi excluída%n",
    pessoa.getId(), pessoa.getNome());
    saveData("F");
}
}
}

private static void alterarPessoa() {
String tipo;
int id;
String nome;
String cpf;
int idade;
String cnpj;
PessoaJuridica pessoaJuridica = null;
PessoaFisica pessoaFisica = null;
System.out.println("F- Pessoa Física | J- Pessoa Jurídica");
tipo = scanner.next().toUpperCase();
if (!tipo.equals("J") && !tipo.equals("F")) {
    System.out.println("Opção inválida!");
    return;
}
System.out.println("Digite o id da Pessoa:");
id = scanner.nextInt();

if (tipo.equals("J")) {
    pessoaJuridica = repoJuridica.obter(id);
    if (pessoaJuridica == null) {
        System.out.println("Pessoa não encontrada");
        return;
    }
    if (tipo.equals("F")) {
        pessoaFisica = repoFisica.obter(id);
        if (pessoaFisica == null) {
            System.out.println("Pessoa não encontrada");
            return;
        }
    }
    System.out.println("Insira os dados...");
    if (tipo.equals("J")) {
        System.out.printf("Nome (%s):", pessoaJuridica.getNome());
        System.out.println("");
    }
    if (tipo.equals("F")) {
        System.out.printf("Nome (%s):", pessoaFisica.getNome());
        System.out.println("");
    }
    nome = scanner.next();

    if (tipo.equals("J")) {
        System.out.printf("CNPJ (%s):", pessoaJuridica.getCnpj());
        System.out.println("");
        cnpj = scanner.next();
        PessoaJuridica pessoaJ = new PessoaJuridica(id, nome, cnpj);
        repoJuridica.alterar(id, pessoaJ);
        saveData("J");
    }
    if (tipo.equals("F")) {
        System.out.printf("CPF (%s):", pessoaFisica.getCpf());
        System.out.println("");
        cpf = scanner.next();
        System.out.printf("Idade (%d):", pessoaFisica.getIdade());
        System.out.println("");
        idade = scanner.nextInt();
        PessoaFisica pessoaF = new PessoaFisica(id, nome, cpf, idade);
        repoFisica.alterar(id, pessoaF);
        saveData("F");
    }
}
}
}

```

```

private static void persistirDados() {
    String tipo;
    String prefixo;
    File file;
    System.out.println("F- Pessoa Fisica | J- Pessoa Juridica");
    tipo = scanner.next().toUpperCase();
    if (!tipo.equals("J") && !tipo.equals("F")) {
        System.out.println("Opção inválida!");
        return;
    }
    System.out.println("Insira o prefixo do arquivo:");
    prefixo = scanner.next();
    if (tipo.equals("J")) {
        file = new File(prefixo + ".juridica.bin");
    } else {
        file = new File(prefixo + ".fisica.bin");
    }
    try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(file))) {
        if (tipo.equals("J")) {
            List<PessoaJuridica> pessoas = repoJuridica.obterTodos();
            oos.writeObject(pessoas);
        } else {
            List<PessoaFisica> pessoas = repoFisica.obterTodos();
            oos.writeObject(pessoas);
        }
    } catch (IOException e) {
        System.err.println("Erro ao salvar dados: " + e.getMessage());
        return;
    }
    System.out.println("Dados salvos com êxito");
}
private static void recuperarDados() {
    String tipo;
    String prefixo;
    File file;
    ObjectInputStream ois;
    List<PessoaFisica> pessoasFisicas = null;
    List<PessoaJuridica> pessoasJuridicas = null;
    System.out.println("F- Pessoa Fisica | J- Pessoa Juridica");
    tipo = scanner.next().toUpperCase();
    if (!tipo.equals("J") && !tipo.equals("F")) {
        System.out.println("Opção inválida!");
        return;
    }
    System.out.println("Insira o prefixo do arquivo:");
    prefixo = scanner.next();
    try {
        if (tipo.equals("J")) {
            file = new File(prefixo + ".juridica.bin");
            ois = new ObjectInputStream(new FileInputStream(file));
            pessoasJuridicas = (List<PessoaJuridica>) ois.readObject();
            ois.close();
            repoJuridica.novosDados(pessoasJuridicas);
            saveData("J");
        }
        if (tipo.equals("F")) {
            file = new File(prefixo + ".fisica.bin");
            ois = new ObjectInputStream(new FileInputStream(file));
            pessoasFisicas = (List<PessoaFisica>) ois.readObject();
            ois.close();
            repoFisica.novosDados(pessoasFisicas);
            saveData("F");
        }
    } catch (IOException | ClassNotFoundException e) {
        System.err.println("Erro ao recuperar dados: " + e.getMessage());
    }
    return;
}
System.out.println("Dados recuperados com êxito");
}

```

Pessoa.java

```
package model;
import java.io.Serializable;

public class Pessoa implements Serializable {
    private int id;
    private String nome;
    public Pessoa() {}
    public Pessoa(int id, String nome) {
        this.id = id;
        this.nome = nome;
    }
    public void exibir() {
        System.out.println("Id: " + id);
        System.out.println("Nome: " + nome);
    }
    // Getters e Setters
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
}
```

PessoaFisica.java

```
package model;
import java.io.Serializable;

public class PessoaFisica extends Pessoa implements Serializable {
    private String cpf;
    private int idade;
    public PessoaFisica() {
        super();
    }
    public PessoaFisica(int id, String nome, String cpf, int idade) {
        super(id, nome);
        this.cpf = cpf;
        this.idade = idade;
    }
    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CPF: " + cpf);
        System.out.println("Idade: " + idade);
    }
    public String getCpf() {
        return cpf;
    }
    public int getIdade() {
        return idade;
    }
}
```

PessoaFisicaRepo.java

```
package model;
import java.io.*;
import java.util.ArrayList;
import java.util.List;

public class PessoaFisicaRepo {
    private List<PessoaFisica> pessoasFisicas = new ArrayList<>();
    public void inserir(PessoaFisica pessoaFisica) {
        pessoasFisicas.add(pessoaFisica);
    }
    public void alterar(int id, PessoaFisica novaPessoaFisica) {
        for (int i = 0; i < pessoasFisicas.size(); i++) {
            if (pessoasFisicas.get(i).getId() == id) {
                pessoasFisicas.set(i, novaPessoaFisica);
                break;
            }
        }
    }
    public void excluir(int id) {
        pessoasFisicas.removeIf(pessoa -> pessoa.getId() == id);
    }
    public void novosDados(List<PessoaFisica> pessoas) {
        pessoasFisicas = pessoas;
    }
    public PessoaFisica obter(int id) {
        for (PessoaFisica pessoa : pessoasFisicas) {
            if (pessoa.getId() == id) {
                return pessoa;
            }
        }
        return null;
    }
    public List<PessoaFisica> obterTodos() {
        return new ArrayList<>(pessoasFisicas);
    }
    public void persistir(String fileName) throws IOException {
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(fileName))) {
            oos.writeObject(pessoasFisicas);
        }
    }
    public List<PessoaFisica> recuperar(String fileName) throws IOException,
    ClassNotFoundException {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(fileName))) {
            pessoasFisicas = (List<PessoaFisica>) ois.readObject();
        }
        return pessoasFisicas;
    }
}
```

PessoaJuridica.java

```
package model;
import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements Serializable {
    private String cnpj;
    public PessoaJuridica() {
        super();
    }
    public PessoaJuridica(int id, String nome, String cnpj) {
        super(id, nome);
        this.cnpj = cnpj;
    }
    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CNPJ: " + cnpj);
    }
    public String getCnpj() {
        return cnpj;
    }
}
```

PessoaJuridicaRepo.java

```
package model;
import java.io.*;
import java.util.ArrayList;
import java.util.List;

public class PessoaJuridicaRepo {
    private List<PessoaJuridica> pessoasJuridicas = new ArrayList<>();
    public void inserir(PessoaJuridica pessoaJuridica) {
        pessoasJuridicas.add(pessoaJuridica);
    }
    public void alterar(int id, PessoaJuridica novaPessoaJuridica) {
        for (int i = 0; i < pessoasJuridicas.size(); i++) {
            if (pessoasJuridicas.get(i).getId() == id) {
                pessoasJuridicas.set(i, novaPessoaJuridica);
                break;
            }
        }
    }
    public void excluir(int id) {
        pessoasJuridicas.removeIf(pessoa-> pessoa.getId() == id);
    }
    public void novosDados(List<PessoaJuridica> pessoas) {
        pessoasJuridicas = pessoas;
    }
    public PessoaJuridica obter(int id) {
        for (PessoaJuridica pessoa : pessoasJuridicas) {
            if (pessoa.getId() == id) {
                return pessoa;
            }
        }
        return null;
    }
    public List<PessoaJuridica> obterTodos() {
        return new ArrayList<>(pessoasJuridicas);
    }
    public void persistir(String fileName) throws IOException {
        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(fileName))) {
            oos.writeObject(pessoasJuridicas);
        }
    }
    public List<PessoaJuridica> recuperar(String fileName) throws IOException,
    ClassNotFoundException {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(fileName))) {
            pessoasJuridicas = (List<PessoaJuridica>) ois.readObject();
        }
        return pessoasJuridicas;
    }
}
```

---> RESULTADO

```
Output - CadastroPOO_text (run) ×
▶ run:
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
=====
1
F - Pessoa Fisica | J - Pessoa Juridica
F
Digite o id da Pessoa:
1
Insira os dados...
Nome:
Victor
CPF:
1234567799
Idade:
37
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Persistir Dados
7 - Recuperar Dados
```

```
Output - CadastroPOO_text (run) ×
▶ run:
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
=====
2
F - Pessoa Fisica | J - Pessoa Juridica
J
Digite o id da Pessoa:
2
Insira os dados...
Nome (CREATEE):
CREATEE
CNPJ (BRASIL):
12.345.678/0001-10
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
=====
```

CadastroPOO_text (run) | 9:12 | INS

---> ANÁLISE E CONCLUSÃO

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

Os elementos estáticos em Java são aqueles que pertencem à classe e não a uma instância específica dessa classe. Isso significa que você pode acessá-los sem criar um objeto dessa classe.

Ao marcar o método “main” como “static” (estático), o compilador Java entende que esse método pertence à própria classe e não a qualquer instância específica dessa classe. Isso permite que o programa seja iniciado e executado, mesmo antes de qualquer objeto da classe ser criado.

B. Para que serve a classe Scanner?

.É usada principalmente para receber entrada do usuário e analisá-la em tipos de dados primitivos, como “int”, “double” ou o tipo padrão “String”. É uma classe utilitária que serve para analisar dados usando expressões regulares por meio da geração de tokens. Oferecendo vários construtores que permitem a leitura de diferentes fontes de entrada.

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

O uso de classes de repositório teve um impacto significativo na organização do código, especialmente em projetos de software que envolvem operações CRUD (Create, Read, Update, Delete) com bancos de dados, promovendo clareza, modularidade, reutilização, testabilidade e escalabilidade.