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

Leonardo Schaffer Mota – 2022.05.09098-1

Polo centro – Santo André - SP
Iniciando o caminho pelo Java – Número da Turma – 2023.4

Objetivo da Prática

Utilizar herança e polimorfismo na definição de entidades.
Utilizar persistência de objetos em arquivos binários.
Implementar uma interface cadastral em modo texto.
Utilizar o controle de exceções da plataforma Java.

1º Procedimento – Criação das entidades e Sistema de persistência.

Classe Pessoa:

```java
package model;

import java.io.Serializable;

/**
 *
 * @author leosc
 */
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 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;
    }


}
```

Classe PessoaFisica:

```java
package model;

import java.io.Serializable;

/**
 *
 * @author leosc
 */
public class PessoaFisica extends Pessoa implements Serializable{

    private String cpf;
    private int idade;

    public PessoaFisica() {
    }

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

```java

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append(str: "Id: ").append(i: getId()).append(str: "\n");
        sb.append(str: "Nome: ").append(str: getNome()).append(str: "\n");
        sb.append(str: "CPF: ").append(str: cpf).append(str: "\n");
        sb.append(str: "Idade: ").append(i: idade);
    return sb.toString();
    }

}
```

Classe PessoaJuridica:

```java
package model;

import java.io.Serializable;

/**
 *
 * @author leosc
 */
public class PessoaJuridica extends Pessoa implements Serializable{

    private String cnpj;

    public PessoaJuridica() {
    }

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

    public String getCnpj() {
        return cnpj;
    }

    public void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }
    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append(str: "Id: ").append(i: getId()).append(str: "\n");
        sb.append(str: "Nome: ").append(str: getNome()).append(str: "\n");
        sb.append(str: "CNPJ: ").append(str: cnpj);
    return sb.toString();
}

}
```

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;
import java.util.ArrayList;

/**
 *
 * @author leosc
 */


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

    public void inserir(PessoaFisica pessoaFisica) {
        pessoasFisicas.add(e: pessoaFisica);
    }
```

```
27
28      public void alterar(PessoaFisica pessoaFisica) {
29          int index = pessoasFisicas.indexOf(o: pessoaFisica);
30          pessoasFisicas.set(index, element: pessoaFisica);
31      }
32
33      public void excluir(PessoaFisica pessoaFisica) {
34          pessoasFisicas.remove(o: pessoaFisica);
35      }
36
37      public PessoaFisica obter(int id) {
38          return pessoasFisicas.get(index:id);
39      }
40
41      public ArrayList<PessoaFisica> obterTodos() {
42          return pessoasFisicas;
43      }
44
45      public void persistir(String nomeArquivo) throws IOException {
46          FileOutputStream fos = new FileOutputStream(new File(pathname: nomeArquivo));
47              try (ObjectOutputStream oos = new ObjectOutputStream(out: fos)) {
48                  oos.writeObject(obj: pessoasFisicas);
49              }
50          System.out.println(x: "Dados de Pessoa Fisica Armazenados.");
51      }
52
53      public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
54          FileInputStream fis = new FileInputStream(new File(pathname: nomeArquivo));
55              try (ObjectInputStream ois = new ObjectInputStream(in: fis)) {
56                  pessoasFisicas = (ArrayList<PessoaFisica>) ois.readObject();
57              }
58          System.out.println(x: "Dados de Pessoa Fisica Recuperados.");
59      }
60  }
```

Classe PessoaJuridicaRepo:

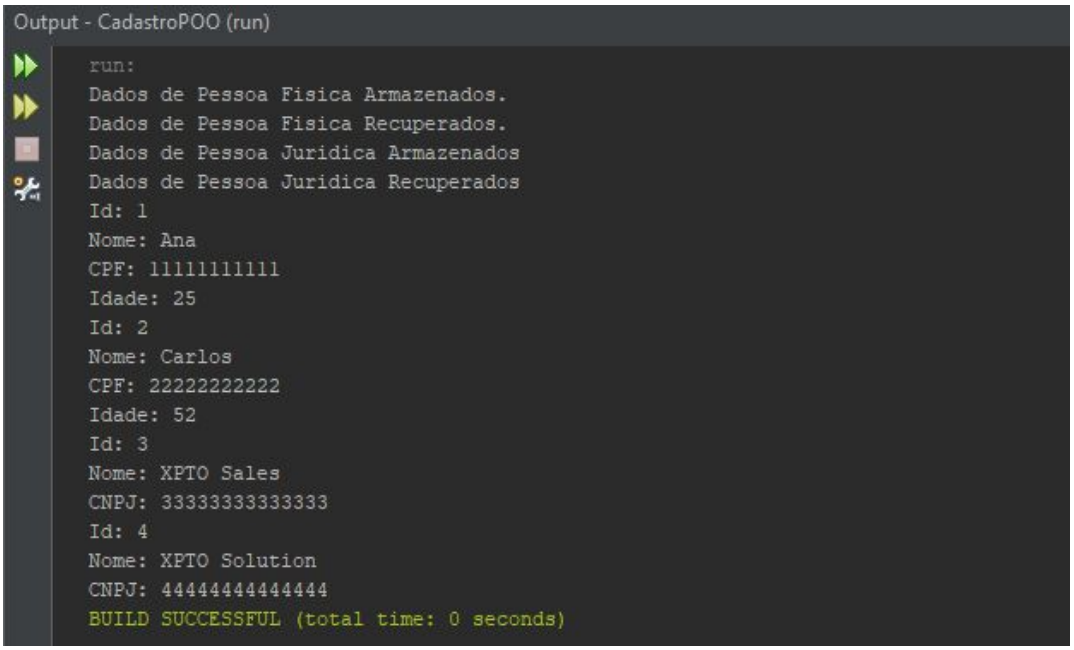
```
5   package model;
6
7   import java.io.File;
8   import java.io.FileInputStream;
9   import java.io.FileOutputStream;
10  import java.io.IOException;
11  import java.io.ObjectInputStream;
12  import java.io.ObjectOutputStream;
13  import java.util.ArrayList;
14
15  /**
16   *
17   * @author leosc
18   */
19
20
21  public class PessoaJuridicaRepo {
22
23  private ArrayList<PessoaJuridica> pessoasJuridicas = new ArrayList<>();
24
25
26
27      public void inserir(PessoaJuridica pessoaJuridica) {
28          pessoasJuridicas.add(e: pessoaJuridica);
29      }
30
31      public void alterar(PessoaJuridica pessoaJuridica) {
32          int index = pessoasJuridicas.indexOf(o: pessoaJuridica);
33          pessoasJuridicas.set(index, element: pessoaJuridica);
34      }
```

```java
    public void excluir(PessoaJuridica pessoaJuridica) {
        pessoasJuridicas.remove(o: pessoaJuridica);
    }

    public PessoaJuridica obter(int id) {
        return pessoasJuridicas.get(index:id);
    }

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

    public void persistir(String nomeArquivo) throws IOException {
        FileOutputStream fos = new FileOutputStream(new File(pathname: nomeArquivo));
    try (ObjectOutputStream oos = new ObjectOutputStream(out: fos)) {
        oos.writeObject(obj: pessoasJuridicas);
    }
        System.out.println(x: "Dados de Pessoa Juridica Armazenados");
    }

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        FileInputStream fis = new FileInputStream(new File(pathname: nomeArquivo));
    try (ObjectInputStream ois = new ObjectInputStream(in: fis)) {
        pessoasJuridicas = (ArrayList<PessoaJuridica>) ois.readObject();
    }
        System.out.println(x: "Dados de Pessoa Juridica Recuperados");
    }
```

Classe CadastroPOO:

```java
package cadastropoo;

import java.io.IOException;
import model.PessoaFisica;
import model.PessoaJuridica;
/**
 * @author leosc
 */
public class CadastroPOO {

    public static void main(String[] args) throws IOException, ClassNotFoundException {

        model.PessoaFisicaRepo repo1 = new model.PessoaFisicaRepo();
        model.PessoaFisicaRepo repo2 = new model.PessoaFisicaRepo();
        model.PessoaJuridicaRepo repo3 = new model.PessoaJuridicaRepo();
        model.PessoaJuridicaRepo repo4 = new model.PessoaJuridicaRepo();
        PessoaFisica pessoa1 = new model.PessoaFisica(id: 1,nome:"Ana",cpf: "11111111111",idade:25);
        PessoaFisica pessoa2 = new model.PessoaFisica(id: 2,nome:"Carlos",cpf: "22222222222" , idade:52);
        PessoaJuridica pessoa3 = new model.PessoaJuridica(id: 3, nome:"XPTO Sales",cnpj:"33333333333333");
        PessoaJuridica pessoa4 = new model.PessoaJuridica(id: 4, nome:"XPTO Solution",cnpj:"44444444444444");

        repo1.inserir(pessoaFisica: pessoa1);
        repo1.inserir(pessoaFisica: pessoa2);

        repo1.persistir(nomeArquivo: "cpf.txt");

        repo2.recuperar(nomeArquivo: "cpf.txt");

        repo3.inserir(pessoaJuridica:pessoa3);
        repo3.inserir(pessoaJuridica:pessoa4);

        repo3.persistir(nomeArquivo: "cnpj.txt");

        repo4.recuperar(nomeArquivo: "cnpj.txt");

        for (PessoaFisica pessoa : repo2.obterTodos()) {
            System.out.println(x: pessoa);
        }

        for (PessoaJuridica pessoa : repo4.obterTodos()) {
            System.out.println(x: pessoa);
        }
    }
}
```

Resultados da execução do Código:



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

Vantagens: reutilização de código, criar novas classes que utilizam uma classe existente, polimorfismo, simplificando o código e facilitando a manutenção, organização, ajuda a criar a criar uma hierarquia nas classes que melhora a organização do código e deixa mais estruturado.
Desvantagens: acoplamento, mudanças em uma classe podem afetar outra classe relacionada, complexidade, uma hierarquia muito profunda pode deixar o código complexo e difícil de entender.

b) Por que a interface Serializable é necessário ao efetuar persistencia em arquivos binários?

É necessário porque a interface Serializable permite que objetos sejam convertidos emu ma sequencia de bytes, fazendo com que esses bytes sejam salvos em um arquivo binário.

c) Como o paradigma functional é utilizado pela API stream Java?

É utilizado a partir das interfaces funcionais, onde essas interfaces fornecem uma maneira mais facil de trabalhar com funções de ordem superios e expresoes lambda, deixando o codigo mais legiel e reduzindo a criação interfaces personalizadas.

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

O padrão de desevolvimento adotado é o padrão DAO (Data Access Object), um padrão que separa a lógica de acesso ao dados da regra de negócio.

2º Procedimento – Criação do Cadastro em Modo Texto.

Classe CadastroPOO:

```
package cadastropoo;

import java.io.IOException;
import model.PessoaFisica;
import model.PessoaJuridica;
import java.util.Scanner;

/**
 * @author leosc
 */
public class CadastroPOO {
    private static int IdCadastro = 1;

    public static int proximoId(){
        return IdCadastro++;
    }

    public static void main(String[] args) throws IOException, ClassNotFoundException {
        boolean inicio = true;

        model.PessoaFisicaRepo repo1 = new model.PessoaFisicaRepo();
        model.PessoaJuridicaRepo repo2 = new model.PessoaJuridicaRepo();

        while(inicio == true){

        System.out.println(x: "================================");
        System.out.println(x: "1 - Incluir Pessoa");
        System.out.println(x: "2 - Alterar Pessoa");
        System.out.println(x: "3 - Excluir Pessoa");
        System.out.println(x: "4 - Buscar pelo Id");
        System.out.println(x: "5 - Exibir Todos");
        System.out.println(x: "6 - Persistir Dados");
        System.out.println(x: "7 - Recuperar Dados");
        System.out.println(x: "0 - Finalizar Programa");
        System.out.println(x: "================================");

        Scanner scanner = new Scanner(source: System.in);
        System.out.println(x: "Digite o numero desejado: ");
        int numero = scanner.nextInt();
        switch (numero){
        case 1 -> {
            boolean caso = true;

            while(caso == true){
                System.out.println(x: "Selecionado Incluir Pessoa");
                System.out.println(x: "F - Pessoa Fisica | J - Pessoa Juridica ");
                String tipoPessoa = scanner.next().toUpperCase();

```

```
                if(tipoPessoa.equals(anObject: "F")){
                    System.out.println(x: "Selecionado Pessoa Fisica");
                    PessoaFisica pessoa = new model.PessoaFisica();
                    pessoa.setId(id: proximoId());

                    /*int Id = scanner.nextInt();*/
                    System.out.println(x: "Digite o Nome: ");
                    String nome = scanner.next();
                    System.out.println(x: "Digite o CPF: ");
                    String cpf = scanner.next();
                    System.out.println(x: "Digite a Idade: ");
                    int idade = scanner.nextInt();

                    pessoa.setNome(nome);
                    pessoa.setCpf(cpf);
                    pessoa.setIdade(idade);

                    repo1.inserir(pessoaFisica: pessoa);
                    caso = false;

                }else if(tipoPessoa.equals(anObject: "J")){
                    System.out.println(x: "Selecionado Pessoa Juridica");
                    PessoaJuridica pessoa = new model.PessoaJuridica();
                    pessoa.setId(id: proximoId());

                    System.out.println(x: "Digite o Nome: ");
                    String nome = scanner.next();
                    System.out.println(x: "Digite o CNPJ: ");
                    String cnpj = scanner.next();

                    pessoa.setNome(nome);
                    pessoa.setCnpj(cnpj);


                    repo2.inserir(pessoaJuridica:pessoa);
                    caso = false;

                }else {
                    System.out.println(x: "Comando Incorreto!!");
                    caso = true;
                    }
                }
            }
        case 2 -> {
                boolean caso = true;
                while(caso == true){
                System.out.println(x: "Selecionado Alterar Pessoa");
                System.out.println(x: "F - Pessoa Fisica | J - Pessoa Juridica ");
                String tipoPessoa = scanner.next().toUpperCase();

```

```
                if(tipoPessoa.equals(anObject: "F")){
105                 System.out.println(x: "Digite o ID da pessoa que deseja alterar: ");
106                 PessoaFisica pessoaAlterada = new model.PessoaFisica();
107                 int id = scanner.nextInt();
108                 scanner.nextLine();
109
110                 PessoaFisica pessoaFisica = repo1.obter(id);
111                 System.out.println(x: pessoaFisica);
112
113                 System.out.println(x: "Digite o nome: ");
114                 String nome = scanner.nextLine();
115
116                 System.out.println(x: "Digite o CPF: ");
117                 String cpf = scanner.nextLine();
118
119                 System.out.println(x: "Digite o Idade: ");
120                 int idade = scanner.nextInt();
121
122                 pessoaAlterada.setNome(nome);
123                 pessoaAlterada.setCpf(cpf);
124                 pessoaAlterada.setIdade(idade);
125
126                 repo1.alterar(pessoaFisica: pessoaAlterada);
127
128                 System.out.println(x: "Pessoa alterada com sucesso!");
129                 caso = false;
130
131             }else if (tipoPessoa.equals(anObject: "J")){
132                 System.out.println(x: "Digite o ID da pessoa que deseja alterar: ");
133                 PessoaJuridica pessoaAlterada = new model.PessoaJuridica();
134                 int id = scanner.nextInt();
135                 scanner.nextLine();
136
137                 PessoaJuridica pessoaJuridica = repo2.obter(id);
138                 System.out.println(x: pessoaJuridica);
139
140                 System.out.println(x: "Digite o nome: ");
141                 String nome = scanner.nextLine();
142
```

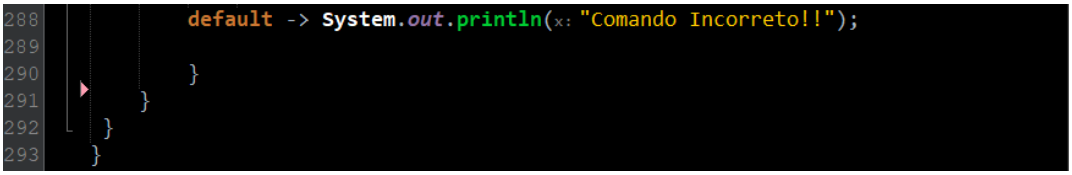
```
143                 System.out.println(x: "Digite o CNPJ: ");
144                 String cnpj = scanner.nextLine();
145
146                 pessoaAlterada.setNome(nome);
147                 pessoaAlterada.setCnpj(cnpj);
148
149                 repo2.alterar(pessoaJuridica:pessoaAlterada);
150
151                 System.out.println(x: "Pessoa alterada com sucesso!");
152                 caso = false;
153
154               }else{
155                 System.out.println(x: "Comando Incorreto!!");
156                 caso = true;
157                 }
158               }
159
160             }
161
162         case 3 ->{
163
```

```
            System.out.println("Selecionado Excluir Pessoa");
            boolean caso = true;
            while(caso == true){
                System.out.println("F - Pessoa Fisica | J - Pessoa Juridica ");
                String tipoPessoa = scanner.next().toUpperCase();
                if(tipoPessoa.equals("F")){
                    System.out.println("Digite o ID da pessoa que deseja alterar: ");
                    int id = scanner.nextInt();
                    scanner.nextLine();
                    repo1.excluir(id);
                    System.out.println("Pessoa excluida com sucesso!");


                    caso = false;
                }else if (tipoPessoa.equals("J")){
                    System.out.println("Digite o ID da pessoa que deseja alterar: ");
                    int id = scanner.nextInt();
                    scanner.nextLine();
                    repo2.excluir(id);
                    System.out.println("Pessoa excluida com sucesso!");

                    caso = false;
                }else{
                    System.out.println("Comando Incorreto!!");
                    caso = true;
                }

            }
        }


        case 4 -> {
            System.out.println("Selecionado Buscar pelo Id");
            boolean caso = true;
            while(caso == true){

                System.out.println("F - Pessoa Fisica | J - Pessoa Juridica ");
                String tipoPessoa = scanner.next().toUpperCase();
```

```
                if(tipoPessoa.equals("F")){

                    System.out.println("Digite o ID da pessoa que deseja obter: ");
                    int id = scanner.nextInt();
                    scanner.nextLine();

                    PessoaFisica pessoaFisica = repo1.obter(id);
                    System.out.println(pessoaFisica);

                    System.out.println("Pessoa obtida com sucesso!");


                    caso = false;
                }else if (tipoPessoa.equals("J")){

                    System.out.println("Digite o ID da pessoa que deseja obter: ");
                    int id = scanner.nextInt();
                    scanner.nextLine();

                    PessoaJuridica pessoaJuridica = repo2.obter(id);
                    System.out.println(pessoaJuridica);

                    System.out.println("Pessoa obtida com sucesso!");
```

```
                    caso = false;
                }else{
                    System.out.println(x: "Comando Incorreto!!");
                    caso = true;
                }
            }
        }
        case 5 -> {
            System.out.println(x: "Selecionado Exibir Todos");

            boolean caso = true;
            while(caso == true){

                System.out.println(x: "F - Pessoa Fisica | J - Pessoa Juridica ");
                String tipoPessoa = scanner.next().toUpperCase();

                if(tipoPessoa.equals(anObject: "F")){

                   for (PessoaFisica pessoa : repo1.obterTodos()) {
                    System.out.println(x: pessoa);

                   }

                    caso = false;
                }else if (tipoPessoa.equals(anObject: "J")){

                    for (PessoaJuridica pessoa : repo2.obterTodos()) {
                    System.out.println(x: pessoa);
                    }

                    caso = false;
                }else{
                    System.out.println(x: "Comando Incorreto!!");
                    caso = true;
                }
            }
        }
```

```
        }
        case 6 -> {
            System.out.println(x: "Selecionado  Persistir Dados");
                scanner.nextLine();
                System.out.println(x: "Digite o prefixo do arquivo:");
                String prefixo = scanner.nextLine();

                repo1.persistir(prefixo + ".fisica.bin");
                repo2.persistir(prefixo + ".fisica.bin");

        }
        case 7 -> {
            System.out.println(x: "Selecionado Recuperar Dados");
                scanner.nextLine();
                System.out.println(x: "Digite o prefixo do arquivo:");
                String prefixo = scanner.nextLine();

                repo1.recuperar(prefixo + ".fisica.bin");
                repo2.recuperar(prefixo + ".fisica.bin");

        }
        case 0 -> {
            System.out.println(x: "Programa Finalizado!");
            inicio = false;
                }
```



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

Elementos estáticos em Java são membros da classe compartilhados por todas as instâncias, incluindo variáveis estáticas, métodos estáticos e blocos estáticos. O método main é estático para servir como ponto de entrada do programa, permitindo que seja chamado diretamente pela JVM sem criar uma instância da classe, facilitando a inicialização do programa Java.

b) Para que serve a classe Scanner?

A classe Scanner é utilizada para ler dados de entrada a partir de diversas fontes, como o teclado ou arquivos. Ela fornece métodos para analisar e extrair valores primitivos e strings.

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

O uso das classes de repositório impactou positivamente na organização do código ao separar a lógica de persistencia e acesso aos dados da lógica de negócio principal.

Conclusão

O código implementa um sistema de cadastro de pessoas físicas e jurídicas utilizando conceitos de programação orientada a objetos e separação de responsabilidades através do uso de classes de repositório. Essa abordagem organizacional permite uma melhor manutenção e escalabilidade do programa, mantendo a lógica de negócio centralizada na classe principal 'main' e o acesso aos dados isolado nos repositórios. Além disso, a interface com o usuário é feita por meio de um menu interativo no console, oferecendo diversas opções de operações CRUD para as pessoas cadastradas. A utilização da classe Scanner permite interação com o usuário, tornando o programa mais prático.