UNIVERSIDADE ESTÁCIO DE SÁ

FULLSTACK

Mundo 03 - Nível 01

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

Herval Rosano Dantas

Matrícula 202205119203

RIO DE JANEIRO – RJ

2023

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 cadastropoo.model;

import java.io.Serializable;

public class Pessoa implements Serializable {
    protected int id;
    protected 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;
    }

    public void exibir() {
        System.out.println("ID: " + id + " | Nome: " + nome);
    }
}
```

Classe PessoaFísica que herda pessoa

```java
package cadastropoo.model;

import java.io.Serializable;
/** *
 * @author HervalDantas
 */
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 void exibir() {
        //super.exibir();
        System.out.println("Nome:      "+nome);
        System.out.println("Código ID: "+id);
        System.out.println("CPF:       "+cpf);
        System.out.println("Idade:     "+idade);
        System.out.println(x: "==================================\n");

    }
}
```

Classe PessoaJurídica que também herda pessoa

```java
package cadastropoo.model;

import java.io.Serializable;

/** *
 * @author HervalDantas
 */
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 void exibir() {

        System.out.println("Nome da Empresa: " + nome);
        System.out.println("Código ID:       " + id);
        System.out.println("CNPJ:            " + cnpj);
        System.out.println(x: "====================================\n");
    }
}
```

Classe **PessoaFisicaRepo**: que gerenciará o conteúdo PessoaFisica, através dos métodos inserir, alterar, excluir, obter e obterTodos do tipo (CRUD) em um banco de dados padrão. Além de implementar os métodos de persistência e recuperação do JPA (Java Persistence Application)

```java
package cadastropoo.model.gerenciadores;

import cadastropoo.model.PessoaFisica;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

import java.util.ArrayList;

/** *
 * @author HervalDantas
 */
public class PessoaFisicaRepo {

    private ArrayList<PessoaFisica> pessoasFisicas;

    //construtor
    public PessoaFisicaRepo() {
        pessoasFisicas = new ArrayList<>();
    }

    //Método inserir que tem como parâmetro tipo = PessoaFisica (classe)
    public void inserir(PessoaFisica pessoaFisica) {
        pessoasFisicas.add(e: pessoaFisica);

        System.out.println(x: "\n ==== Dados Adicionadas ========");
        pessoaFisica.exibir();
    }
```

```java
    //Método alterar que tem como parâmetro tipo = PessoaFisica (classe)
    public void alterar(PessoaFisica pessoaFisica) {
        for (int i = 0; i < pessoasFisicas.size(); i++) {
            if (pessoasFisicas.get(index: i).getId() == pessoaFisica.getId()) {
                pessoasFisicas.set(index: i, element:pessoaFisica);
                System.out.println(x: "\n ==== Dados Alterados ========");
                pessoasFisicas.get(index: i).exibir();
                //break;
            }
        }
    }

```

```java
    //Método excluir que tem como parâmetro um int que identificará o id a ser excluido
    public void excluir(int id) {
        for (int i = 0; i < pessoasFisicas.size(); i++) {
            if (pessoasFisicas.get(index: i).getId() == id) {
                System.out.println("\n Estamos removendo ==> "+ pessoasFisicas.get(index: i).getNome());
                pessoasFisicas.remove(index: i);
                //break;
            } else {
                System.out.println(x: "\n Id inexistente! ");
                //break;
            }
        }
    }

    //Método obter que tem como parâmetro um int que identificará o id a ser excluido
    public PessoaFisica obter(int id) {
        for (PessoaFisica pessoaFisica : pessoasFisicas) {
            if (pessoaFisica.getId() == id) {
                return pessoaFisica;
            } else {
                //System.out.println("ID inexistente!");
                return null;
            }
        }
        return null;
    }

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

```java

    public void persistir(String nomeArquivo) throws IOException {
        // Create a file input stream
        File arquivoPF = new File(pathname: nomeArquivo);
        FileOutputStream fos = new FileOutputStream(file: arquivoPF);

        try (ObjectOutputStream objOutput = new ObjectOutputStream(out:fos)){

            objOutput.writeObject(obj:pessoasFisicas);
            objOutput.close();

            System.out.println(x: "Dados de Pessoa Física amazendados com sucesso!\n");

        } catch (IOException e) {
            System.out.println(x: "An error occurred.");
        }
    }

    // desserialização: recuperando os objetos gravados no arquivo binário "nomeArquivo"
    public ArrayList<PessoaFisica> recuperar(String nomeArquivo) throws IOException, FileNotFoundException {

        //ArrayList<Object> lista = new ArrayList();
        ArrayList<PessoaFisica> listaRecuperada = new ArrayList();
        File arq = new File(pathname: nomeArquivo);
        FileInputStream fis = new FileInputStream(file: arq);
        try {
            if (arq.exists()) {
                try (ObjectInputStream objInput = new ObjectInputStream(in: fis)) {
                    listaRecuperada = (ArrayList<PessoaFisica>) objInput.readObject();

                    System.out.println(x: "Dados de Pessoa Física recuperados com sucesso!");
                    for (PessoaFisica pessoa : listaRecuperada) {
                        pessoasFisicas.add(e: pessoa);
                        pessoa.exibir();
                    }
```

```
110                         }
111                     }
112                 }
113             } catch (IOException erro1) {
114                 System.out.printf(format: "Erro: %s", args: erro1.getMessage());
115             } catch (ClassNotFoundException erro2) {
116                 System.out.printf(format: "Erro: %s", args: erro2.getMessage());
117             }
118             return listaRecuperada;
119         }
120     }
```

Classe **PessoaJuridicaRepo:** que também gerenciará o conteúdo PessoaJuridica, através dos métodos inserir, alterar, excluir, obter e obterTodos . Além de implementar os métodos de persistência e recuperação do JPA.

```java
package cadastropoo.model.gerenciadores;

import cadastropoo.model.PessoaFisica;
import cadastropoo.model.PessoaJuridica;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;

/** *
 * @author HervalDantas
 */
public class PessoaJuridicaRepo {

    private ArrayList<PessoaJuridica> pessoasJuridicas;

    //construtor
    public PessoaJuridicaRepo() {
        pessoasJuridicas = new ArrayList<>();
    }
```

```java
    //Método inserir que tem como parâmetro tipo = PessoaJuridica (classe)
    public void inserir(PessoaJuridica pessoaJuridica) {
        pessoasJuridicas.add(e: pessoaJuridica);
        System.out.println(x: "\n====== Dados Adicionadas =======");
        pessoaJuridica.exibir();
    }

    //Método alterar que tem como parâmetro tipo = PessoaJuridica (classe)
    public void alterar(PessoaJuridica pessoaJuridica) {
        for (int i = 0; i < pessoasJuridicas.size(); i++) {
            if (pessoasJuridicas.get(index: i).getId() == pessoaJuridica.getId()) {
                pessoasJuridicas.set(index: i, element:pessoaJuridica);
                System.out.println(x: "\n ==== Dados Alterados =======");
                pessoasJuridicas.get(index: i).exibir();
               // break;
            }
        }
    }

    //Método excluir que tem como parâmetro um int que identificará o id a ser excluido
    public void excluir(int id) {
        for (int i = 0; i < pessoasJuridicas.size(); i++) {
            if (pessoasJuridicas.get(index: i).getId() == id) {
                System.out.println("\n Estamos removendo ==> "+ pessoasJuridicas.get(index: i).getNome());
                pessoasJuridicas.remove(index: i);
                //break;
            }else {
                System.out.println(x: "\n Id inexistente! ");
                //break;
            }
        }
    }
```

```java
    //Método obter que tem como parâmetro um int que identificará o id a ser excluido
    public PessoaJuridica obter(int id) {
        for (PessoaJuridica pessoajuridica : pessoasJuridicas) {
            if (pessoajuridica.getId() == id) {
                return pessoajuridica;
            }else {
                //System.out.println("ID inexistente!");
                return null;
            }
        }
        return null;
    }

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

    public void persistir(String nomeArquivo) throws IOException  {
        // Create a file input stream
        File arquivoPJ = new File(pathname: nomeArquivo);
        FileOutputStream fos = new FileOutputStream(file: arquivoPJ);

        try (ObjectOutputStream objOutput = new ObjectOutputStream(out:fos)){

            objOutput.writeObject(obj:pessoasJuridicas);
            objOutput.close();

            System.out.println(x: "Dados de Pessoa Jurídica amazendados com sucesso!\n");

        } catch (IOException e) {
            System.out.println(x: "Algo deu errado.");
        }
    }
```

```java

    public ArrayList<PessoaJuridica> recuperar(String nomeArquivo) throws IOException, FileNotFoundException {

        //ArrayList<Object> lista = new ArrayList();
        ArrayList<PessoaJuridica> listaRecuperada = new ArrayList();
        File arq = new File(pathname: nomeArquivo);
        FileInputStream fis = new FileInputStream(file: arq);
        try {
            if (arq.exists()) {
                try (ObjectInputStream objInput = new ObjectInputStream(in: fis)) {
                    listaRecuperada = (ArrayList<PessoaJuridica>) objInput.readObject();

                    System.out.println(x: "Dados de Pessoa Jurídica recuperados com sucesso!");
                    for (PessoaJuridica pessoa : listaRecuperada) {
                        pessoasJuridicas.add(e: pessoa);
                        pessoa.exibir();
                    }
                }
            }
        } catch (IOException erro1) {
            System.out.printf(format: "Erro: %s", args:erro1.getMessage());
        } catch (ClassNotFoundException erro2) {
            System.out.printf(format: "Erro: %s", args:erro2.getMessage());
        }
        return listaRecuperada;
    }
}
```

E por último a classe principal **cadastropoo** onde através de um menu do tipo texto com a captura de input do usuário executará toda a aplicação do cadastro.

```java
package cadastropoo;

import cadastropoo.model.PessoaFisica;
import cadastropoo.model.PessoaJuridica;
import cadastropoo.model.gerenciadores.PessoaFisicaRepo;
import cadastropoo.model.gerenciadores.PessoaJuridicaRepo;
import java.io.IOException;
import java.util.Scanner;

/** *
 * @author HervalDantas
 */
public class CadastroPOO {

    public static void main(String[] args) throws IOException {
        //instanciando os objetos
        PessoaFisica pessoaF = new PessoaFisica();
        PessoaJuridica pessoaJ = new PessoaJuridica();
        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
        PessoaJuridicaRepo repo2 = new PessoaJuridicaRepo();
        //Scannear
        Scanner optionIn = new Scanner(source: System.in);
        //iniciando as 2 variáveis do looping while
        boolean inicio = true;
        boolean caso = true;
```

```java
        while (inicio == true) {

            //Display Menu
            System.out.println(x: "           M  E  N  U");
            System.out.println(x: "==============================");
            System.out.println(x: "1 - Incluir");
            System.out.println(x: "2 - Alterar");
            System.out.println(x: "3 - Excluir");
            System.out.println(x: "4 - Exibir conforme ID");
            System.out.println(x: "5 - Exibir todos");
            System.out.println(x: "6 - Salvar");
            System.out.println(x: "7 - Recuperar");
            System.out.println(x: "0 - Finalizar programa");
            System.out.println(x: "==============================");

            String opcao = optionIn.nextLine();
```

```
43
            switch (opcao) {
45
                case "1":
46
                    caso = true;
47
                    while (caso == true) {
48
                        System.out.println(x: "Foi Selecionado ==> Incluir Pessoa");
49
                        System.out.println(x: "F - Pessoa Física  |  J - Pessoa Jurídica");
50
                        String entity = optionIn.nextLine().toUpperCase();
51
52
                        if (entity.equals(anObject: "F")) {
54                          System.out.println(x: "Digite o Número do ID: ");
55                          int inputId = optionIn.nextInt();
56                          //Consumir a linha pendente
57                          optionIn.nextLine();
58
59                          System.out.println(x: "Digite o Nome: ");
60                          String inputNome = optionIn.nextLine();
61                          System.out.println(x: "Digite o CPF: ");
62                          String inputCPF = optionIn.nextLine();
63                          System.out.println(x: "Digite sua Idade: ");
64                          int inputIdade = optionIn.nextInt();
65                          optionIn.nextLine();
66
67                          //atribui os valores no objeto pessoaFisica
68                          pessoaF.setNome(nome: inputNome);
69                          pessoaF.setId(id: inputId);
70                          pessoaF.setCpf(cpf:inputCPF);
71                          pessoaF.setIdade(idade: inputIdade);
72
73                          //usa o pessoaFisicaRepo para inserir os dados
74                          repo1.inserir(pessoaFisica: pessoaF);
75
76                          caso = false;
```

```
77                          } else if (entity.equals(anObject: "J")) {
78                              System.out.println(x: "Digite o Número do ID: ");
79                              int inputId = optionIn.nextInt();
80                              optionIn.nextLine();
81                              System.out.println(x: "Digite o Nome da Empresa: ");
82                              String inputNome = optionIn.nextLine();
83                              System.out.println(x: "Digite o CNPJ: ");
84                              String inputCNPJ = optionIn.nextLine();
85
86                              pessoaJ.setNome(nome: inputNome);
87                              pessoaJ.setId(id: inputId);
88                              pessoaJ.setCnpj(cnpj: inputCNPJ);
89
90                              repo2.inserir(pessoaJuridica:pessoaJ);
91
92                              caso = false;
93                          } else {
94                              System.out.println(x: "Opção inválida!");
95                              caso = true;
96                          }
97                      }
98                      break;
```

```
99                  case "2":
100                     caso = true;
101                     while (caso == true) {
102                         System.out.println(x: "Foi Selecionado ==> Alterar Pessoa");
103                         System.out.println(x: "F - Pessoa Fisica | J - Pessoa Juridica ");
104                         String entity = optionIn.nextLine().toUpperCase();
105
106                         if (entity.equals(anObject: "F")) {
107                             System.out.println(x: "Informe o ID da pessoa que deseja alterar: ");
108                             PessoaFisica pessoaFisAlterada = new PessoaFisica();
109                             int inputId = optionIn.nextInt();
110                             PessoaFisica pessoaFisicaMostrar = repo1.obter(id: inputId);
111                             if (pessoaFisicaMostrar != null) {
112                                 pessoaFisicaMostrar.exibir();
113
114                                 //Consumir a linha pendente
115                                 optionIn.nextLine();
116
117                                 System.out.println(x: "Digite o Nome: ");
118                                 String inputNome = optionIn.nextLine();
119                                 System.out.println(x: "Digite o CPF: ");
120                                 String inputCPF = optionIn.nextLine();
121                                 System.out.println(x: "Digite sua Idade: ");
122                                 int inputIdade = optionIn.nextInt();
123                                 optionIn.nextLine();
124
125                                 //Alterar os valores no objeto pessoaFisica
126                                 pessoaFisAlterada.setId(id: inputId);
127                                 pessoaFisAlterada.setNome(nome: inputNome);
128                                 pessoaFisAlterada.setCpf(cpf:inputCPF);
129                                 pessoaFisAlterada.setIdade(idade: inputIdade);
130
```

```
                        //usa o pessoaFisicaRepo para inserir os dados
                        repo1.alterar(pessoaFisica: pessoaFisAlterada);

                    } else {
                        System.out.println(x: "ID inexistente!");
                    }

                    caso = false;

                } else if (entity.equals(anObject: "J")) {
                    System.out.println(x: "Informe o ID da Empresa que deseja alterar: ");
                    PessoaJuridica pessoaJurAlterada = new PessoaJuridica();
                    int inputId = optionIn.nextInt();
                    PessoaJuridica pessoaJuridicaMostrar = repo2.obter(id: inputId);
                    if (pessoaJuridicaMostrar != null) {
                        pessoaJuridicaMostrar.exibir();

                        //Consumir a linha pendente
                        optionIn.nextLine();

                        System.out.println(x: "Digite o Nome da Empresa: ");
                        String inputNome = optionIn.nextLine();
                        System.out.println(x: "Digite o CNPJ: ");
                        String inputCNPJ = optionIn.nextLine();

                        pessoaJurAlterada.setNome(nome: inputNome);
                        pessoaJurAlterada.setId(id: inputId);
                        pessoaJurAlterada.setCnpj(cnpj: inputCNPJ);

                        repo2.alterar(pessoaJuridica:pessoaJurAlterada);

                    } else {
                        System.out.println(x: "ID inexistente!");
                    }
```

```
                        caso = false;

                    } else {
                        System.out.println(x: "Opção inválida!");
                        caso = true;
                    }
                }
                break;
            case "3":
                caso = true;
                while (caso == true) {
                    System.out.println(x: "Foi selecionado ==> Excluir Pessoa");
                    System.out.println(x: "F - Pessoa Fisica | J - Pessoa Juridica ");
                    String entity = optionIn.nextLine().toUpperCase();
                    if (entity.equals(anObject: "F")) {
                        System.out.println(x: "Digite o ID da pessoa que deseja excluir: ");
                        int inputId = optionIn.nextInt();
                        repo1.excluir(id: inputId);

                        caso = false;
                    } else if (entity.equals(anObject: "J")) {
                        System.out.println(x: "Digite o ID da Empresa que deseja excluir: ");
                        int inputId = optionIn.nextInt();
                        repo2.excluir(id: inputId);
                        caso = false;
                    } else {
                        System.out.println(x: "Opção inavalida!");
                        caso = true;
                    }
                }
                break;
```

```
197                 case "4":
198                     caso = true;
199                     while (caso == true) {
200                         System.out.println(x: "Foi Selecionado ==> Exibir pelo Id");
201                         System.out.println(x: "F - Pessoa Fisica | J - Pessoa Juridica ");
202                         String entity = optionIn.nextLine().toUpperCase();
203 
204                         if (entity.equals(anObject: "F")) {
205                             System.out.println(x: "Digite o ID da pessoa que deseja obter: ");
206                             int inputId = optionIn.nextInt();
207 
208                             PessoaFisica pessoaFisica = repo1.obter(id: inputId);
209                             //System.out.println(pessoaFisica);
210                             pessoaFisica.exibir();
211 
212                             System.out.println(x: "Pessoa obtida com sucesso!");
213 
214                             caso = false;
215 
216                         } else if (entity.equals(anObject: "J")) {
217 
218                             System.out.println(x: "Digite o ID da Empresa que deseja obter: ");
219                             int inputId = optionIn.nextInt();
220 
221                             PessoaJuridica pessoaJuridica = repo2.obter(id: inputId);
222                             //System.out.println(pessoaJuridica);
223                             pessoaJuridica.exibir();
224                             System.out.println(x: "Pessoa obtida com sucesso!");
225 
226                             caso = false;
```

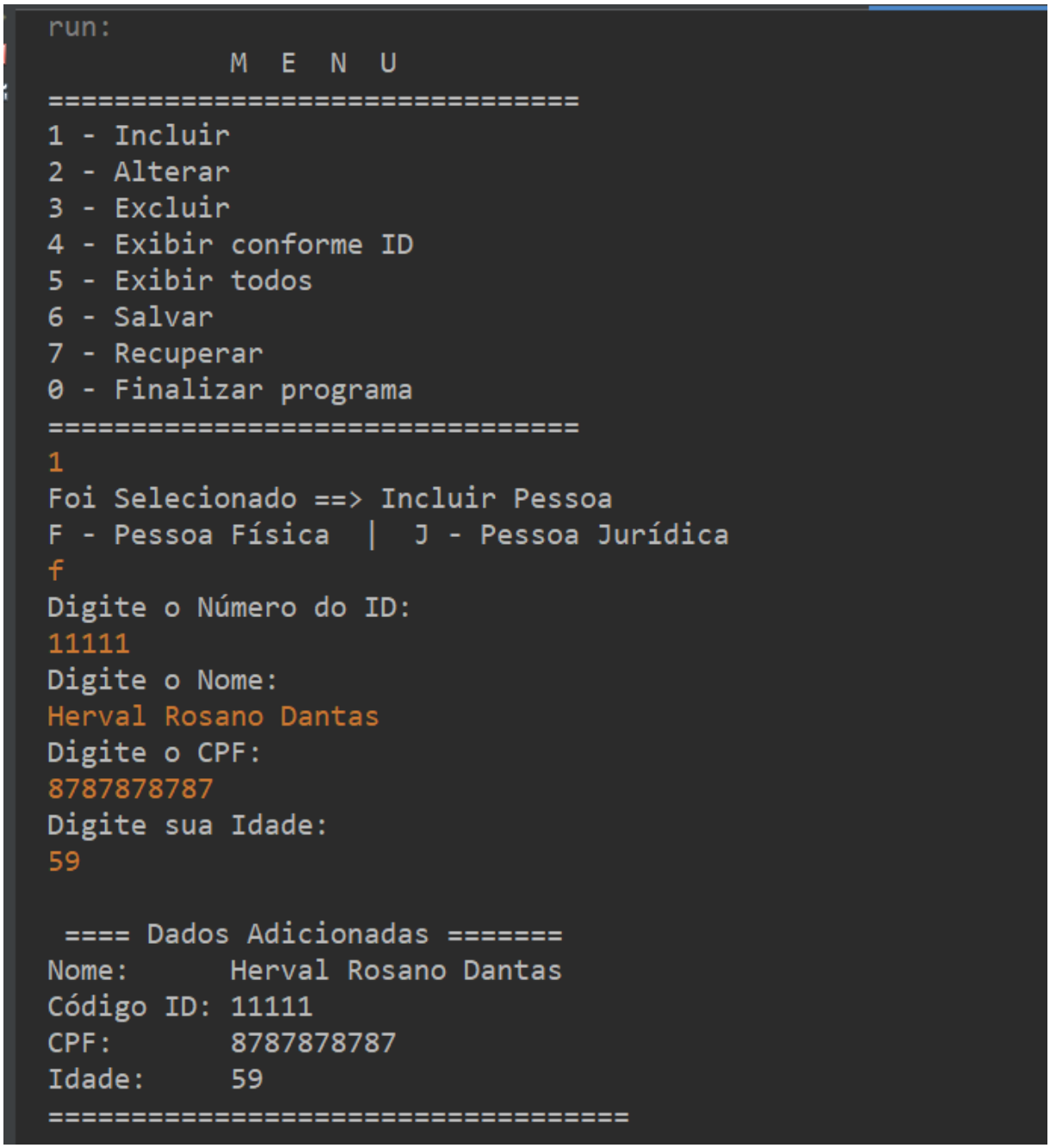
```
227                         } else {
228                             System.out.println(x: "Opção inválida!");
229                             caso = true;
230                         }
231                     }
232                     break;
233                 case "5":
234                     caso = true;
235                     while (caso == true) {
236                         System.out.println(x: "Foi Selecionado ==> Exibir Todos");
237                         System.out.println(x: "F - Pessoa Fisica | J - Pessoa Juridica ");
238                         String entity = optionIn.nextLine().toUpperCase();
239                         if (entity.equals(anObject: "F")) {
240 
241                             for (PessoaFisica pessoa : repo1.obterTodos()) {
242                                 pessoa.exibir();
243                             }
244                             caso = false;
245                         } else if (entity.equals(anObject: "J")) {
246                             for (PessoaJuridica pessoa : repo2.obterTodos()) {
247                                 pessoa.exibir();
248                             }
249                             caso = false;
250                         } else {
251                             System.out.println(x: "Opção inválida!");
252                             caso = true;
253                         }
254                     }
255                     break;
256 
```

```
                case "6":
                    //optionIn.nextLine();
                    System.out.println(x: "Foi Selecionado ==> Salvar Dados");
                    System.out.println(x: "F - Pessoa Fisica | J - Pessoa Juridica ");
                    String entity = optionIn.nextLine().toUpperCase();
                    String prefixo = " ";

                    if (entity.equals(anObject: "F")) {
                        prefixo = "fisica";
                        repo1.persistir(prefixo + "_persistente.bin");

                    } else if (entity.equals(anObject: "J")) {
                        prefixo = "juridica";
                        repo2.persistir(prefixo + "_persistente.bin");
                    } else {
                        System.out.println(x: "Opção inválida!");
                    }

                    //repo2.persistir(prefixo + "_persistente.bin");
                    break;

                case "7":
                    System.out.println(x: "Foi Selecionado ==> Recuperar Dados");
                    //optionIn.nextLine();
                    System.out.println(x: "F - Pessoa Fisica | J - Pessoa Juridica ");
                    entity = optionIn.nextLine().toUpperCase();
                    prefixo = " ";

                    if (entity.equals(anObject: "F")) {
                        prefixo = "fisica";
                        repo1.recuperar(prefixo + "_persistente.bin");
```

```
                    } else if (entity.equals(anObject: "J")) {
                        prefixo = "juridica";
                        repo2.recuperar(prefixo + "_persistente.bin");

                    } else {
                        System.out.println(x: "Opção inválida!");
                    }

                    break;

                case "0":
                    System.out.println(x: "Programa Finalizado!");
                    inicio = false;
                    break;
            }

        }
    }
}
```

A seguir as telas da aplicação em todas as suas etapas do menu:

1- Inserir



2- Alterar

```
2
Foi Selecionado ==> Alterar Pessoa
F - Pessoa Fisica | J - Pessoa Juridica
f
Informe o ID da pessoa que deseja alterar:
11111
Nome:       Herval Rosano Dantas
Código ID: 11111
CPF:        8787878787
Idade:      59
=====================================

Digite o Nome:
Herval R. Dantas
Digite o CPF:
8888888
Digite sua Idade:
22

 ==== Dados Alterados =======
Nome:       Herval R. Dantas
Código ID: 11111
CPF:        8888888
Idade:      22
=====================================
```

3 – Excluir

```
3
Foi selecionado ==> Excluir Pessoa
F - Pessoa Fisica | J - Pessoa Juridica
f
Digite o ID da pessoa que deseja excluir:
9999
```

4 - Exibir conforme ID

```
4
Foi Selecionado ==> Exibir pelo Id
F - Pessoa Fisica | J - Pessoa Juridica
f
Digite o ID da pessoa que deseja obter:
7777
Nome:       Angela Agostinho da Costa Dantas
Código ID: 7777
CPF:        555555
Idade:      77
====================================

Pessoa obtida com sucesso!
```

5 - Exibir todos

```
5
Foi Selecionado ==> Exibir Todos
F - Pessoa Fisica | J - Pessoa Juridica
j
Nome da Empresa: Mei Antonio Dantas
Código ID:       10004
CNPJ:            66648541111
====================================

Nome da Empresa: Mei Antonio Dantas
Código ID:       10004
CNPJ:            66648541111
====================================

Nome da Empresa: Mei Antonio Dantas
Código ID:       10004
CNPJ:            66648541111
====================================
```

6 - Salvar

```
6
Foi Selecionado ==> Salvar Dados
F - Pessoa Fisica | J - Pessoa Juridica
j
Dados de Pessoa Jurídica amazendados com sucesso!
```

7 - Recuperar

```
7
Foi Selecionado ==> Recuperar Dados
F - Pessoa Fisica | J - Pessoa Juridica
f
Dados de Pessoa Física recuperados com sucesso!
Nome:       Angela Agostinho da Costa Dantas
Código ID: 7777
CPF:        555555
Idade:      77
====================================

Nome:       Antonio Dantas
Código ID: 3333
CPF:        77777
Idade:      16
====================================

Nome:       Herval Rosano Dantas
Código ID: 123455
CPF:        555521212
Idade:      59
====================================
```

0 – Finalizar Programa

```
          M  E  N  U
================================
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir conforme ID
5 - Exibir todos
6 - Salvar
7 - Recuperar
0 - Finalizar programa
================================
0
Programa Finalizado!
BUILD SUCCESSFUL (total time: 6 minutes 49 seconds)
```

Conclusão

1 - Quais as vantagens e desvantagens do uso de herança?

Uma das vantagens é a reutilização de códigos já criado nos métodos assim como a reutilização dos campos através dos atributos. A modulação também é mais uma outra vantagem, pois facilita a blocagem (dividir em diversas classes) dos códigos, o que facilita em muito a manutenção/alteração.

Já as desvantagens é que estes acoplamentos quando uma mudança é feita afeta as classes filhas, e conforme estas heranças vão crescendo se tornam mais complexas podendo deixa-las com uma manutenção e até mesmo entendimento mais difícil.

2 - Por que a interface Serializable é necessário ao efetuar persistência em arquivos binários?

É necessário para que os objetos sejam convertidos numa sequência binária de forma que possam ser gravados e lidos em arquivos do tipo .txt, .bin .ser... na própria máquina sem fazer uso de um DBS (Data Base System) sistema de banco de dados.

3 - Como o paradigma functional é utilizado pela API stream Java?

No paradigma functional o uso da API stream que ajuda a realizar operações de processamento de dados em coleções do tipo Lista, ArrayList, Maps onde é possível fazer filtragens, ordenagens e mapeamentos de maneira mais legível.

5 - Quando trabalhamos com Java, qual o padrão de desenvolvimento é adotado na persistência de dados em arquivos?

Serialização Java e Leitura e Gravação de Dados de Forma Manual. A serialização é mais conveniente pra armazenar objetos complexos de forma simples, mas tem a limitações de compatibilidade

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

Elementos estáticos são componentes de uma classe tipo variáveis, métodos e blocos que pertencem à própria classe. Ou seja, eles são compartilhados de forma que não precisam ser instanciados e podem ser acessados diretamente. Por isso que o método main a adota, pois possa ser que certa variável ou métodos precisem estar sempre disponíveis.

5- Para que serve a classe Scanner?

A classe Scanner é uma ferramenta versátil para a leitura de dados de entrada de várias fontes, principalmente para capturar dados imputados pelo usuário via teclado. Mas também pode ler strings e arquivos.

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

O principal impacto foi a organização em um só lugar de todos os componentes (métodos, variáveis) que processam o cadastro imputado na aplicação

Conclusão

Esta é uma aplicação muito básica que consiste em fazer um cadastro de dados onde os mesmos são imputados na forma textual via terminal. Mas o seu intuito principal é mostrar didaticamente a persistência de dados de forma simples sem a necessidade de uso do SGBD (Sistema de gerenciamento de banco de dados). E também nos fez entender e praticar o conceito de POO (Programação Orientada a objeto) onde foi possível fazer uso de herança e poliformismo de classe.