Estácio

Campus: Madureira **Curso:** Desenvolvimento Full Stack
Disciplina: Iniciando o caminho pelo Java **Turma:** 9001
Semestre: 2024.1 **Integrante:** Eduarda Amaral Rizzuto

[1] Missão Prática | Nível 1 | Mundo 3

[2] Objetivos da prática

1. Utilizar herança e polimorfismo na definição de entidades.
2. Utilizar persistência de objetos em arquivos binários.
3. Implementar uma interface cadastral em modo texto.
4. Utilizar o controle de exceções da plataforma Java.
5. No final do projeto, o aluno terá implementado um sistema cadastral em Java, utilizando os recursos da programação orientada a objetos e a persistência em arquivos binários.

[3] Códigos solicitados:

//Main

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

public class Main {

   public static void main(String args[]) {

      PessoaFisicaRepo repo1 = new PessoaFisicaRepo();

      repo1.inserir(new PessoaFisica(1, "Analais", 34, "203.444.567-87"));
      repo1.inserir(new PessoaFisica(2, "Baldur", 43, "787-654-443-02"));

      try {
         repo1.persistir("dadosF.ser");
      } catch (IOException err) {
         err.printStackTrace();
      }

      PessoaFisicaRepo repo2 = new PessoaFisicaRepo();

      try {
         repo2.recuperar("dadosF.ser");
      } catch (IOException | ClassNotFoundException err) {
         err.printStackTrace();
      }
```

```
        for (Object obj : repo2.obterTodos()) {
            PessoaFisica pessoa = (PessoaFisica) obj;
            pessoa.exibir();
        }

        PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
        repo3.inserir(new PessoaJuridica(1, "Camparas CIA", "32.443.867/0001-54"));
        repo3.inserir(new PessoaJuridica(2, "Casa do Anfitrião", "66.545.781/0001-11"));

        try {
            repo3.persistir("dadosJ.ser");
        } catch (IOException err) {
            err.printStackTrace();
        }

        PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();

        try {
            repo4.recuperar("dadosJ.ser");
        } catch (IOException | ClassNotFoundException err) {
            err.printStackTrace();
        }

        for (Object obj : repo4.obterTodos()) {
            PessoaJuridica pessoa = (PessoaJuridica) obj;
            pessoa.exibir();
        }
    }
}
```

//Pessoa

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

public class Main {

    public static void main(String args[]) {

        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();

        repo1.inserir(new PessoaFisica(1, "Analais", 34, "203.444.567-87"));
        repo1.inserir(new PessoaFisica(2, "Baldur", 43, "787-654-443-02"));

        try {
            repo1.persistir("dadosF.ser");
```

```
        } catch (IOException err) {
            err.printStackTrace();
        }

        PessoaFisicaRepo repo2 = new PessoaFisicaRepo();

        try {
            repo2.recuperar("dadosF.ser");
        } catch (IOException | ClassNotFoundException err) {
            err.printStackTrace();
        }

        for (Object obj : repo2.obterTodos()) {
            PessoaFisica pessoa = (PessoaFisica) obj;
            pessoa.exibir();
        }

        PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
        repo3.inserir(new PessoaJuridica(1, "Camparas CIA", "32.443.867/0001-54"));
        repo3.inserir(new PessoaJuridica(2, "Casa do Anfitrião", "66.545.781/0001-11"));

        try {
            repo3.persistir("dadosJ.ser");
        } catch (IOException err) {
            err.printStackTrace();
        }

        PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();

        try {
            repo4.recuperar("dadosJ.ser");
        } catch (IOException | ClassNotFoundException err) {
            err.printStackTrace();
        }

        for (Object obj : repo4.obterTodos()) {
            PessoaJuridica pessoa = (PessoaJuridica) obj;
            pessoa.exibir();
        }
    }
}
```

//Pessoa Física

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

```
public class PessoaFisica extends Pessoa implements Serializable{
   //atributos
   String cpf;
   int idade;

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

   //métodos
   @Override
   public void exibir(){
       System.out.println("ID: " + getId() + ", Nome: " + getNome() + ", CPF: " + getCpf()
+ ", Idade: " + getIdade());
   }

   //acessores
   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;
   }

}
```

//Pessoa Jurídica

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

public class PessoaJuridica extends Pessoa implements Serializable{
   //atributos
   String cnpj;
```

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

    //métodos
    @Override
    public void exibir(){
            System.out.println("ID: " + getId() + ", Nome: " + getNome() + ", CNPJ: " +
getCnpj());
    }

    //acessores
    public String getCnpj() {
        return cnpj;
    }
    public void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }

}
```

//Pessoa Física Repositório

```
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;
import java.util.Scanner;

public class PessoaFisicaRepo {
    Scanner entrada = new Scanner(System.in);

    private ArrayList <PessoaFisica> pessoasFisicas= new ArrayList<>();

    //métodos
    public void inserir(PessoaFisica pessoaf){
        pessoasFisicas.add(pessoaf);
    }
    public void alterar(int id){
        boolean encontrada = false;
```

```
    for (PessoaFisica pes : pessoasFisicas){
        if (pes.getId() == id){

            encontrada = true;
            System.out.print("Pessoa encontrada: ");
            pes.exibir();
            System.out.println("Insira os novos dados:");
            System.out.print("Id: ");
            int ID = entrada.nextInt();
            System.out.print("Nome: ");
            String nome = entrada.next();
            System.out.print("Idade: ");
            int idade = entrada.nextInt();
            System.out.print("CPF: ");
            String cpf = entrada.next();

            pes.setId(ID);
            pes.setNome(nome);
            pes.setIdade(idade);
            pes.setCpf(cpf);

            System.out.println("Pessoa Física alterada com sucesso");
            pes.exibir();
        }
    }
    if (!encontrada){
        System.out.println("Pessoa Física não encontrada");
    }
}
public void excluir(int id){
    boolean encontrada = false;
    for (PessoaFisica pes: pessoasFisicas){
        if (pes.getId() == id){
            pessoasFisicas.remove(pessoasFisicas.indexOf(pes));
            System.out.println("Pessoa Física excluída com sucesso");
            encontrada = true;
            break;
        }
    }
    if (!encontrada){
        System.out.println("Pessoa Física não encontrada");
    }
}
public void obter(int id){
    boolean encontrada = false;
```

```
        for (PessoaFisica pes: pessoasFisicas){
            if (pes.getId() == id){
                pes.exibir();
                encontrada = true;
                break;
            }
        }
        if (!encontrada){
            System.out.println("Pessoa Física não encontrada");
        }
    }
    public ArrayList obterTodos(){
        return pessoasFisicas;
    }

    public void persistir(String prefixo) throws IOException{
    String arquivoFisica = prefixo + ".fisica.bin";
    try (FileOutputStream salvaFisica = new FileOutputStream(new File(arquivoFisica));
         ObjectOutputStream oosFisica = new ObjectOutputStream(salvaFisica)) {

        oosFisica.writeObject(pessoasFisicas);

        System.out.println("Dados de Pessoa Física Armazenados em " + arquivoFisica);
    } catch (IOException e) {
            throw new IOException("Erro ao persistir os dados de Pessoa Física: " +
e.getMessage());
    }
}

public void recuperar(String prefixo) throws IOException, ClassNotFoundException {
    String arquivoFisica = prefixo + ".fisica.bin";
    try (FileInputStream recuperaFisica = new FileInputStream(arquivoFisica);
         ObjectInputStream oisFisica = new ObjectInputStream(recuperaFisica)) {

        pessoasFisicas = (ArrayList<PessoaFisica>) oisFisica.readObject();

        System.out.println("Dados de Pessoa Física Recuperados de " + arquivoFisica);
    } catch (IOException | ClassNotFoundException e) {
            throw new IOException("Erro ao recuperar os dados de Pessoa Física: " +
e.getMessage());
    }
}
}
```

//Pessoa Jurídica Repositório

```
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;
import java.util.Scanner;

public class PessoaJuridicaRepo {
   Scanner entrada = new Scanner(System.in);

   private ArrayList <PessoaJuridica> pessoasJuridicas = new ArrayList<>();

   //métodos
   public void inserir(PessoaJuridica pessoaj){
      pessoasJuridicas.add(pessoaj);
   }
   public void alterar(int id){
      boolean encontrada = false;
      for (PessoaJuridica pes : pessoasJuridicas){
         if (pes.getId() == id){

            encontrada = true;
            System.out.print("Pessoa encontrada: ");
            pes.exibir();
            System.out.println("Insira os novos dados:");
            System.out.print("Id: ");
            int ID = entrada.nextInt();
            System.out.print("Nome: ");
            String nome = entrada.next();
            System.out.print("CNPJ: ");
            String cnpj = entrada.next();

            pes.setId(ID);
            pes.setNome(nome);
            pes.setCnpj(cnpj);

            System.out.println("Pessoa Jurídica alterada com sucesso");
            pes.exibir();
         }
      }
      if (!encontrada){
```

```
            System.out.println("Pessoa Jurídica não encontrada");
        }
    }
    public void excluir(int id){
        boolean encontrada = false;
        for (PessoaJuridica pes: pessoasJuridicas){
            if (pes.getId() == id){
                pessoasJuridicas.remove(pessoasJuridicas.indexOf(pes));
                System.out.println("Pessoa Jurídica excluída com sucesso");
                encontrada = true;
                break;
            }
        }
        if (!encontrada){
            System.out.println("Pessoa Jurídica não encontrada");
        }
    }

    public void obter(int id){
        boolean encontrada = false;
        for (PessoaJuridica pes: pessoasJuridicas){
            if (pes.getId() == id){
                pes.exibir();
                encontrada = true;
                break;
            }
        }
        if (!encontrada){
            System.out.println("Pessoa Jurídica não encontrada");
        }
    }
    public ArrayList obterTodos(){
        return pessoasJuridicas;
    }

    public void persistir(String prefixo) throws IOException {
    String arquivoJuridica = prefixo + ".juridica.bin";
                try (FileOutputStream salvaJuridica = new FileOutputStream(new
File(arquivoJuridica));
         ObjectOutputStream oosJuridica = new ObjectOutputStream(salvaJuridica)) {

        oosJuridica.writeObject(pessoasJuridicas);

                    System.out.println("Dados de Pessoa Jurídica Armazenados em " +
arquivoJuridica);
```

```
    } catch (IOException e) {
            throw new IOException("Erro ao persistir os dados de Pessoa Física: " +
e.getMessage());
    }
}

public void recuperar(String prefixo) throws IOException, ClassNotFoundException {
        String arquivoJuridica = prefixo + ".juridica.bin";
        try (FileInputStream recuperaJuridica = new FileInputStream(arquivoJuridica);
            ObjectInputStream oisJuridica = new ObjectInputStream(recuperaJuridica)) {

           pessoasJuridicas = (ArrayList<PessoaJuridica>) oisJuridica.readObject();

                    System.out.println("Dados de Pessoa Jurídica Recuperados de " +
arquivoJuridica);
        } catch (IOException | ClassNotFoundException e) {
                throw new IOException("Erro ao recuperar os dados de Pessoa Física: " +
e.getMessage());
        }
    }
}
```

[4] Resultados

```
run:
Dados de Pessoa Física Armazenados em dadosF.ser.fisica.bin
Dados de Pessoa Física Recuperados de dadosF.ser.fisica.bin
ID: 1, Nome: Analais, CPF: 203.444.567-87, Idade: 34
ID: 2, Nome: Baldur, CPF: 787-654-443-02, Idade: 43
Dados de Pessoa Jurídica Armazenados em dadosJ.ser.juridica.bin
Dados de Pessoa Jurídica Recuperados de dadosJ.ser.juridica.bin
ID: 1, Nome: Camparas CIA, CNPJ: 32.443.867/0001-54
ID: 2, Nome: Casa do Anfitrião, CNPJ: 66.545.781/0001-11
BUILD SUCCESSFUL (total time: 0 seconds)
```

[5] Análise e Conclusão

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

As vantagens são o reúso de atributos e métodos com economia de linha de código, de tal forma que respeita a simplificação que a POO busca (as subclasses agem como “filhos” que puxam características de sua superclasse mãe, porém ainda podendo alterá-las). Uma desvantagem pode ser seu uso em ocasiões em que apenas deixará o código mais complexo e haverá dificuldade na administração do encapsulamento.

b. Por que a interface Seriazable é necessária ao efetuar persistência em arquivos binários?

Porque ela é responsável por “dizer” ao motor do java que sua respectiva classe implementada possui a capacidade de ter seus dados gravados em disco e recuperados.

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

A API stream fornece ferramentas que tornam a escrita do código mais enxuta. Para isso ele utiliza métodos do paradigma funcional (map, filter, etc). Essas funções costumam reduzir bastante a quantidade de linhas de código, além de simplifica laços e estruturas condicionais.

d. Quando trabalhamos com Java, qual padrão de desenvolvimento é adotado na persistência de arquivos.

O padrão DAO (Data Access Object).

[link]:https://github.com/duda6134/MissaoPratica1/tree/main/CadastroPOO/build/classes/model

obs.: a classe Main está alterada para a segunda parte do projeto.