



Criação das Entidades e Sistema de Persistência

Wesley Borges do Carmo de Oliveira

Polo Centro - Barão de Cocais – MG

Iniciando o Caminho Pelo Java – 9001 – 3º Semestre

Objetivo da Prática

O objetivo desta prática é o desenvolvimento de um sistema de cadastro em java, fazendo o uso de paradigmas de programação como a programação orientada a objetos, fazendo a manutenção, reuso, e organização dos códigos, mais suave e descomplicada, e a persistência em arquivos binários, para garantir que as informações sejam armazenadas em um meio que possam ser recuperadas de forma consistente.

1º Procedimento | Criação das Entidades e Sistema de Persistência

CadastroPOO

```
package cadastropoo;

import java.io.IOException;
import model.PessoaFisica;
import model.PessoaFisicaRepo;
import model.PessoaJuridica;
import model.PessoaJuridicaRepo;

public class CadastroPOO {

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

        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();

        PessoaFisica pessoa1 = new PessoaFisica(0,"Marco", "2122", 25);
```

```
PessoaFisica pessoa2 = new PessoaFisica(25,"Merlyn", "5932", 23);
```

```
repo1.inserir(pessoa1);
```

```
repo1.inserir(pessoa2);
```

```
repo1.persistir("repositorioFis1");
```

```
PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
```

```
repo2.recuperar("repositorioFis1");
```

```
for(PessoaFisica p : repo2.obterTodos()){
```

```
    p.exibir();
```

```
}
```

```
PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
```

```
PessoaJuridica pessoa3 = new PessoaJuridica(63,"Marco", "2122");
```

```
PessoaJuridica pessoa4 = new PessoaJuridica(96,"Merlyn", "5932");
```

```
repo3.inserir(pessoa3);
```

```
repo3.inserir(pessoa4);
```

```
repo3.persistir("repositorioJus1");
```

```
PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
```

```
repo4.recuperar("repositorioJus1");
```

```
for(PessoaJuridica p : repo4.obterTodos()){
```

```
    p.exibir();
```

```
    }  
  }  
}
```

Pessoa

```
package model;
```

```
import java.io.Serializable;
```

```
public class Pessoa implements Serializable {
```

```
    private int id;
```

```
    private String nome;
```

```
    public Pessoa(int id, String nome){
```

```
        this.id = id;
```

```
        this.nome = nome;
```

```
    }
```

```
    protected void exibir(){
```

```
        System.out.println("Id: " + this.getId() + "\nNome: " + this.getNome());
```

```
    }
```

```
    protected void setId(int id){
```

```
        this.id = id;
```

```
    }
```

```
    protected int getId(){
```

```
        return this.id;
```

```
    }
```

```
    protected void setNome(String nome){
```

```

        this.nome = nome;
    }

    protected String getNome(){
        return this.nome;
    }
}

```

PessoaFisica

```

package model;

import java.io.Serializable;

public class PessoaFisica extends Pessoa implements Serializable{

    private String cpf;
    private int idade;

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

    @Override
    public void exibir(){
        System.out.println("Id: " + this.getId() + "\nNome: " + this.getNome() +
"\nCPF: " + this.getCPF() + "\nIdade:" + this.getIdade());
    }

    protected void setCPF(String cpf){
        this.cpf = cpf;
    }
}

```

```

protected String getCPF(){
    return this.cpf;
}

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

protected int getIdade(){
    return this.idade;
}
}

```

PessoaFisicaRepo

```

package model;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;

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

    public void persistir(String nome) throws IOException{
        try(ObjectOutputStream oStream = new ObjectOutputStream(new
        FileOutputStream(nome))){
            System.out.println("Dados de Pessoa Fisica Armazenados.");
            oStream.writeObject(lista);
        }
    }
}

```

```
    }  
}
```

```
    public void recuperar(String nome) throws IOException,  
ClassNotFoundException{  
        try(ObjectInputStream oStream = new ObjectInputStream(new  
FileInputStream(nome))){  
            System.out.println("Dados de Pessoa Fisica Recuperados.");  
            lista = (ArrayList<PessoaFisica>) oStream.readObject();  
        }  
    }  
}
```

```
public void inserir(PessoaFisica pessoa){  
    lista.add(pessoa);  
}
```

```
public void alterar(PessoaFisica pessoa){  
    for (PessoaFisica p : lista) {  
        if (p.getId() == pessoa.getId()) {  
            p.setNome(pessoa.getNome());  
            p.setCPF(pessoa.getCPF());  
            p.setIdade(pessoa.getIdade());  
            return;  
        }  
    }  
}
```

```
public void excluir(int id){  
    lista.removeIf(p -> p.getId() == id);  
}
```

```

public PessoaFisica obter(int id){
    for(PessoaFisica p: lista){
        if(p.getId() == id){
            return p;
        }
    }
    return null;}

public ArrayList<PessoaFisica> obterTodos(){
    return new ArrayList<>(lista);
}

}

```

PessoaJuridica

```

package model;
import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements Serializable {

    private String cnpj;

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

    @Override
    public void exibir(){
        System.out.println("Id: " + this.getId() + "\nNome: " + this.getNome() +
            "\nCNPJ: " + this.getCNPJ());
    }
}

```

```

    }

    protected void setCNPJ(String cnpj){
        this.cnpj = cnpj;
    }

    protected String getCNPJ(){
        return this.cnpj;
    }

}

```

PessoaJuridicaRepo

```

package model;

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.Optional;

public class PessoaJuridicaRepo {

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

    public void persistir(String nome) throws IOException{
        try(ObjectOutputStream oStream = new ObjectOutputStream(new
        FileOutputStream(nome))){
            System.out.println("Dados de Pessoa Juridica Armazenados.");
            oStream.writeObject(lista);
        }
    }
}

```



```
}
```

```
public void recuperar(String nome) throws IOException,  
ClassNotFoundException{
```

```
    try(ObjectInputStream oStream = new ObjectInputStream(new  
        FileInputStream(nome))){
```

```
        System.out.println("Dados de Pessoa Juridica Recuperados.");
```

```
        lista = (ArrayList<PessoaJuridica>) oStream.readObject();
```

```
    }
```

```
}
```

```
public void inserir(PessoaJuridica pessoa){
```

```
    lista.add(pessoa);
```

```
}
```

```
public void alterar(PessoaJuridica pessoa){
```

```
    for (PessoaJuridica p : lista) {
```

```
        if (p.getId() == pessoa.getId()) {
```

```
            p.setNome(pessoa.getNome());
```

```
            p.setCNPJ(pessoa.getCNPJ());
```

```
            return;
```

```
        }
```

```
    }
```

```
}
```

```
public void excluir(int id){
```

```
    lista.removeIf(p -> p.getId() == id);
```

```
}
```

```
public PessoaJuridica obter(int id){
```

```
    for(PessoaJuridica p: lista){
```

```

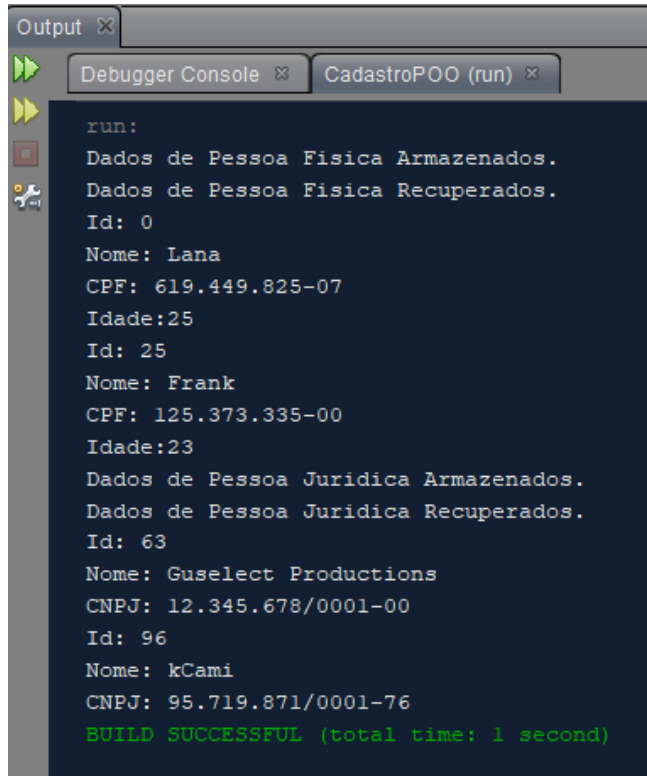
        if(p.getId() == id){
            return p;
        }
    }
    return null;
}

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

public void main(String[] args){
    System.out.println(obter(25));
}
}

```

Resultados



```

run:
Dados de Pessoa Fisica Armazenados.
Dados de Pessoa Fisica Recuperados.
Id: 0
Nome: Lana
CPF: 619.449.825-07
Idade:25
Id: 25
Nome: Frank
CPF: 125.373.335-00
Idade:23
Dados de Pessoa Juridica Armazenados.
Dados de Pessoa Juridica Recuperados.
Id: 63
Nome: Guselect Productions
CNPJ: 12.345.678/0001-00
Id: 96
Nome: kCami
CNPJ: 95.719.871/0001-76
BUILD SUCCESSFUL (total time: 1 second)

```

Análise e Conclusão

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

A reutilização de código, que permite compartilhar similaridades enquanto preserva as diferenças, e a manutenção do sistema se torna mais fácil, proporcionando maior legibilidade do código existente, reduzindo a quantidade de linhas de código e limitando as alterações a poucas partes do código, são algumas das vantagens apresentadas pela herança, porém, O uso desta técnica compromete o princípio do encapsulamento, pois os atributos são compartilhados entre várias classes, criando uma interdependência entre elas e dificultando mudanças nas superclasses.

b) Por que a interface `Serializable` é necessária ao efetuar persistência em arquivos binários?

A serialização de um objeto, faz a conversão dos valores de uma instância, em uma sequência de bytes, fazendo com que o estado do objeto possa ser recuperado.

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

A API `stream` faz uso do paradigma funcional, com o uso de expressões `lambda` para passar funções como argumentos, e `streams` imutáveis, ou seja, não modificam a coleção original.

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

O Padrão MVC (Model-View-Controller),

2º Procedimento | Criação do Cadastro em Modo Texto

CadastroPOO

```
package cadastrapoo;

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

public class CadastroPOO {

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

        Scanner in = new Scanner(System.in);
        PessoaFisicaRepo repoPFisica = new PessoaFisicaRepo();
        PessoaJuridicaRepo repoPJuridica = new PessoaJuridicaRepo();

        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("=====");

            int input = in.nextInt();

            switch(input){
                case(1):
                    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
                    String pessoaTipol = in.next();
                    if(pessoaTipol.equals("F")){
                        System.out.println("Insira os dados...");

                        System.out.print("Nome: ");
                        String nome = in.next();

                        System.out.print("CPF: ");
                        String cpf = in.next();

                        System.out.print("Idade: ");
```

```

        int idade = in.nextInt();

        int id = repoPFisica.obterTodos().size() + 1;

        PessoaFisica pessoaFNova = new PessoaFisica(id, nome, cpf, idade);
        repoPFisica.inserir(pessoaFNova);

        System.out.println("Dados adicionados com sucesso.");
    }
    else if(pessoaTipo.equals("J")){
        System.out.println("Insira os dados...");

        System.out.print("Nome: ");
        String nome = in.next();

        System.out.print("CNPJ: ");
        String cnpj = in.next();

        int id = repoPJuridica.obterTodos().size() + 1;

        PessoaJuridica pessoaJNova = new PessoaJuridica(id,nome, cnpj);
        repoPJuridica.inserir(pessoaJNova);
        System.out.println("Dados adicionados com sucesso.");
    }
    break;

case(2):
    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
    String pessoaTipoA = in.next();
    if(pessoaTipoA.equals("F")){
        System.out.print("Insira um id: ");
        int id = in.nextInt();
        try{
            PessoaFisica pessoaAlterar = repoPFisica.obter(id);
            pessoaAlterar.exibir();

            System.out.println("Insira os dados...");

            System.out.print("Novo Nome: ");
            String nome = in.next();

            System.out.print("Novo CPF: ");
            String cpf = in.next();

            System.out.print("Nova Idade: ");
            int idade = in.nextInt();

            PessoaFisica pessoaFNova = new PessoaFisica(id, nome, cpf, idade);
            repoPFisica.alterar(pessoaFNova);
            pessoaAlterar.exibir();

```

```

        System.out.print("Dados alterados com sucesso.");
    }catch(Throwable e){
        System.out.println("Erro ao obter dados: " + e.getMessage());
        e.printStackTrace();
    }
}
}
else if(pessoaTipoA.equals("J")){

    System.out.print("Insira um id: ");
    int id = in.nextInt();
    try{
        PessoaJuridica pessoaAlterar = repoPJuridica.obter(id);
        pessoaAlterar.exibir();

        System.out.println("Insira os dados...");

        System.out.print("Novo Nome: ");
        String nome = in.next();

        System.out.print("Novo CNPJ: ");
        String cnpj = in.next();

        PessoaJuridica pessoaJNova = new PessoaJuridica(id,nome, cnpj);
        repoPJuridica.alterar(pessoaJNova);
        pessoaAlterar.exibir();
        System.out.print("Dados alterados com sucesso.");
    }catch(Throwable e){
        System.out.println("Erro ao obter dados: " + e.getMessage());
        e.printStackTrace();
    }
}
break;
case(3):
    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
    String pessoaTipoE = in.next();
    if(pessoaTipoE.equals("F")){
        System.out.print("Insira um id: ");
        int id = in.nextInt();
        try{
            repoPFisica.excluir(id);
            System.out.println("Dados removidos.");
        }catch(Throwable e){
            System.out.println("Erro ao excluir dados: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
else if(pessoaTipoE.equals("J")){
    System.out.print("Insira um id: ");
    int id = in.nextInt();
    try{

```

```

        repoPJuridica.excluir(id);
        System.out.println("Dados removidos.");
    }catch(Throwable e){
        System.out.println("Erro ao excluir dados: " + e.getMessage());
        e.printStackTrace();
    }
}
break;

```

case(4):

```

        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
        String pessoaTipoO = in.next();
        if(pessoaTipoO.equals("F")){
            System.out.print("Insira um id: ");
            int id = in.nextInt();

            try{
                PessoaFisica pessoaObtida = repoPFisica.obter(id);
                pessoaObtida.exibir();
            }catch(Throwable e){
                System.out.println("Erro ao obter dados: " + e.getMessage());
                e.printStackTrace();
            }
        }
        else if(pessoaTipoO.equals("J")){
            System.out.print("Insira um id: ");
            int id = in.nextInt();

            try{
                PessoaJuridica pessoaObtida = repoPJuridica.obter(id);
                pessoaObtida.exibir();
            }catch(Throwable e){
                System.out.println("Erro ao obter dados: " + e.getMessage());
                e.printStackTrace();
            }
        }
    }
    break;

```

case(5):

```

        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
        String pessoaTipoOT = in.next();
        if(pessoaTipoOT.equals("F")){
            ArrayList<PessoaFisica> pessoaObtida = repoPFisica.obterTodos();
            for(PessoaFisica p: pessoaObtida){
                p.exibir();
            }
        }
        else if(pessoaTipoOT.equals("J")){
            ArrayList<PessoaJuridica> pessoaObtida = repoPJuridica.obterTodos();
            for(PessoaJuridica p: pessoaObtida){

```

```

        p.exibir();
    }
}
break;

case(6):
    System.out.print("Insira um prefixo para salvar o arquivo: ");
    String arquivoNomeS = in.next();
    try{
        repoPFisica.persistir(arquivoNomeS + ".fisica.bin");
        repoPJuridica.persistir(arquivoNomeS + ".juridica.bin");
        System.out.println("Dados salvos com sucesso.");
    }catch(IOException e){
        System.out.println("Erro ao salvar dados: " + e.getMessage());
        e.printStackTrace();
    }
    break;

case(7):
    System.out.print("Insira um prefixo para recuperar o arquivo: ");
    String arquivoNomeR = in.next();
    try{
        repoPFisica.recuperar(arquivoNomeR + ".fisica.bin");
        repoPJuridica.recuperar(arquivoNomeR + ".juridica.bin");
        System.out.println("Dados recuperados com sucesso.");
    }catch(IOException | ClassNotFoundException e){
        System.out.println("Erro ao recuperar dados: " + e.getMessage());
        e.printStackTrace();
    }
    break;

case(0):
    System.out.println("Finalizando a execução...");
    in.close();
    return;

default:
    System.out.println("Escolha uma opção entre 0 a 7.");
    break;
}
}while(true);

}

}

```


Resultados

```
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
=====

6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
=====
1
F - Pessoa Fisica | J - Pessoa Juridica
F
Insira os dados...
Nome: Fran
CPF: 256.653.635-63
Idade: 23
Dados adicionados com sucesso.
=====

5 - Exibir Todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
=====
4
F - Pessoa Fisica | J - Pessoa Juridica
J
Insira um id: 1
Id: 1
Nome: GuselectProd
CNPJ: 12.345.678/0001-00
=====

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
=====
6
Insira um prefixo para salvar o arquivo: PessoasPratica
Dados de Pessoa Fisica Armazenados.
Dados de Pessoa Juridica Armazenados.
Dados salvos com sucesso.
=====
```

Análise e Conclusão

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

elementos estáticos são elementos que "existem", ou seja, estão disponíveis para uso, sem a necessidade de serem instanciados, a palavra-chave "static" é usada para indicar que o método "main" pertence à classe, e não a uma instância específica da classe, isso é necessário porque o método "main" é chamado pelo sistema de execução do programa, antes que qualquer objeto seja criado.

- b) Para que serve a classe Scanner?

A classe Scanner tem como objetivo separar a entrada dos textos em blocos, gerando os conhecidos tokens, que são sequências de caracteres separados por delimitadores que por padrão correspondem aos espaços em branco, tabulações e mudança de linha.

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

O uso de classes de repositório impactou de forma positiva o desenvolvimento, apresentando alguns benefícios para ajudar na organização do código como, o encapsulamento, reusabilidade, assim como facilitando mudanças e a manutenção.