Campus Volta Redonda

Curso de Desenvolvimento Full Stack

Nivel 1: Iniciando o Caminho pelo Java

Turma 9001

Semestre 2025.1

Nicolas Calheiros Maciel

Vantagens do uso da herança: reutilização do código através do uso de subclasses e superclasses; hierarquização do código e fácil manutenção. Desvantagens: o uso excessivo ou mal uso da hierarquização pode tornar o código extenso e difícil de entender.

A interface Serializable e necessária para informar ao JVM que um objeto em questão pode ser serializado, provendo segurança para os dados inseridos.

Seguem os codigos:

Main.Java:

```
import model.PessoaFisica;
import model.PessoaFisicaRepo;
import model.PessoaJuridica;
import model.PessoaJuridicaRepo;

import java.io.IOException;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
        PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();

        int opcao;
```

```
do {
    // Exibir menu
    System.out.println("\n--- Menu ---");
    System.out.println("1. Incluir");
    System.out.println("2. Alterar");
    System.out.println("3. Excluir");
    System.out.println("4. Exibir pelo ID");
    System.out.println("5. Exibir todos");
    System.out.println("6. Salvar dados");
    System.out.println("7. Recuperar dados");
    System.out.println("0. Sair");
    System.out.print("Escolha uma opção: ");
    opcao = scanner.nextInt();
    scanner.nextLine(); // Consumir a nova linha

    switch (opcao) {
        case 1: // Incluir
            incluirEntidade(scanner, repoFisica, repoJuridica);
            break;

        case 2: // Alterar
            alterarEntidade(scanner, repoFisica, repoJuridica);
            break;

        case 3: // Excluir
            excluirEntidade(scanner, repoFisica, repoJuridica);
            break;

        case 4: // Exibir pelo ID
```

```
                exibirPorId(scanner, repoFisica, repoJuridica);
                break;

            case 5: // Exibir todos
                exibirTodos(scanner, repoFisica, repoJuridica);
                break;

            case 6: // Salvar dados
                salvarDados(scanner, repoFisica, repoJuridica);
                break;

            case 7: // Recuperar dados
                recuperarDados(scanner, repoFisica, repoJuridica);
                break;

            case 0: // Sair
                System.out.println("Finalizando a execução...");
                break;

            default:
                System.out.println("Opção inválida. Tente novamente.");
        }
    } while (opcao != 0);

    scanner.close();
  }

  // Método para incluir uma entidade
  private static void incluirEntidade(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo
repoJuridica) {
```

```
System.out.println("\n--- Incluir ---");
System.out.println("Escolha o tipo:");
System.out.println("1. Pessoa Física");
System.out.println("2. Pessoa Jurídica");
System.out.print("Opção: ");
int tipo = scanner.nextInt();
scanner.nextLine(); // Consumir a nova linha

if (tipo == 1) {
    System.out.print("ID: ");
    int id = scanner.nextInt();
    scanner.nextLine();
    System.out.print("Nome: ");
    String nome = scanner.nextLine();
    System.out.print("CPF: ");
    String cpf = scanner.nextLine();
    System.out.print("Idade: ");
    int idade = scanner.nextInt();
    scanner.nextLine();

    PessoaFisica pessoa = new PessoaFisica(id, nome, cpf, idade);
    repoFisica.inserir(pessoa);
    System.out.println("Pessoa Física adicionada com sucesso!");
} else if (tipo == 2) {
    System.out.print("ID: ");
    int id = scanner.nextInt();
    scanner.nextLine();
    System.out.print("Nome: ");
    String nome = scanner.nextLine();
```

```
            System.out.print("CNPJ: ");
            String cnpj = scanner.nextLine();

            PessoaJuridica pessoa = new PessoaJuridica(id, nome, cnpj);
            repoJuridica.inserir(pessoa);
            System.out.println("Pessoa Jurídica adicionada com sucesso!");
        } else {
            System.out.println("Opção inválida.");
        }
    }

    // Método para alterar uma entidade
    private static void alterarEntidade(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica) {
        System.out.println("\n--- Alterar ---");
        System.out.println("Escolha o tipo:");
        System.out.println("1. Pessoa Física");
        System.out.println("2. Pessoa Jurídica");
        System.out.print("Opção: ");
        int tipo = scanner.nextInt();
        scanner.nextLine(); // Consumir a nova linha

        System.out.print("ID da entidade a ser alterada: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        if (tipo == 1) {
            PessoaFisica pessoa = repoFisica.obter(id);
            if (pessoa != null) {
                System.out.println("Dados atuais:");
```

```
            pessoa.exibir();

            System.out.print("Novo Nome: ");
            String nome = scanner.nextLine();
            System.out.print("Novo CPF: ");
            String cpf = scanner.nextLine();
            System.out.print("Nova Idade: ");
            int idade = scanner.nextInt();
            scanner.nextLine();

            PessoaFisica pessoaAtualizada = new PessoaFisica(id, nome, cpf, idade);
            repoFisica.alterar(pessoaAtualizada);
            System.out.println("Pessoa Física alterada com sucesso!");
        } else {
            System.out.println("Pessoa Física não encontrada.");
        }
    } else if (tipo == 2) {
        PessoaJuridica pessoa = repoJuridica.obter(id);
        if (pessoa != null) {
            System.out.println("Dados atuais:");
            pessoa.exibir();

            System.out.print("Novo Nome: ");
            String nome = scanner.nextLine();
            System.out.print("Novo CNPJ: ");
            String cnpj = scanner.nextLine();

            PessoaJuridica pessoaAtualizada = new PessoaJuridica(id, nome, cnpj);
            repoJuridica.alterar(pessoaAtualizada);
```

```
            System.out.println("Pessoa Jurídica alterada com sucesso!");
        } else {
            System.out.println("Pessoa Jurídica não encontrada.");
        }
    } else {
        System.out.println("Opção inválida.");
    }
}

// Método para excluir uma entidade
private static void excluirEntidade(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica) {
    System.out.println("\n--- Excluir ---");
    System.out.println("Escolha o tipo:");
    System.out.println("1. Pessoa Física");
    System.out.println("2. Pessoa Jurídica");
    System.out.print("Opção: ");
    int tipo = scanner.nextInt();
    scanner.nextLine(); // Consumir a nova linha

    System.out.print("ID da entidade a ser excluída: ");
    int id = scanner.nextInt();
    scanner.nextLine();

    if (tipo == 1) {
        repoFisica.excluir(id);
        System.out.println("Pessoa Física excluída com sucesso!");
    } else if (tipo == 2) {
        repoJuridica.excluir(id);
        System.out.println("Pessoa Jurídica excluída com sucesso!");
```

```
        } else {
            System.out.println("Opção inválida.");
        }
    }

    // Método para exibir uma entidade pelo ID
    private static void exibirPorId(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica) {
        System.out.println("\n--- Exibir pelo ID ---");
        System.out.println("Escolha o tipo:");
        System.out.println("1. Pessoa Física");
        System.out.println("2. Pessoa Jurídica");
        System.out.print("Opção: ");
        int tipo = scanner.nextInt();
        scanner.nextLine(); // Consumir a nova linha

        System.out.print("ID da entidade: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        if (tipo == 1) {
            PessoaFisica pessoa = repoFisica.obter(id);
            if (pessoa != null) {
                pessoa.exibir();
            } else {
                System.out.println("Pessoa Física não encontrada.");
            }
        } else if (tipo == 2) {
            PessoaJuridica pessoa = repoJuridica.obter(id);
            if (pessoa != null) {
```

```
            pessoa.exibir();
        } else {
            System.out.println("Pessoa Jurídica não encontrada.");
        }
    } else {
        System.out.println("Opção inválida.");
    }
}

// Método para exibir todas as entidades
private static void exibirTodos(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica) {
    System.out.println("\n--- Exibir Todos ---");
    System.out.println("Escolha o tipo:");
    System.out.println("1. Pessoa Física");
    System.out.println("2. Pessoa Jurídica");
    System.out.print("Opção: ");
    int tipo = scanner.nextInt();
    scanner.nextLine(); // Consumir a nova linha

    if (tipo == 1) {
        System.out.println("Pessoas Físicas Cadastradas:");
        for (PessoaFisica pessoa : repoFisica.obterTodos()) {
            pessoa.exibir();
            System.out.println("------");
        }
    } else if (tipo == 2) {
        System.out.println("Pessoas Jurídicas Cadastradas:");
        for (PessoaJuridica pessoa : repoJuridica.obterTodos()) {
            pessoa.exibir();
```

```
            System.out.println("------");
        }
    } else {
        System.out.println("Opção inválida.");
    }
}

// Método para salvar dados
private static void salvarDados(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica) {
    System.out.println("\n--- Salvar Dados ---");
    System.out.print("Digite o prefixo dos arquivos: ");
    String prefixo = scanner.nextLine();

    try {
        repoFisica.persistir(prefixo + ".fisica.bin");
        repoJuridica.persistir(prefixo + ".juridica.bin");
        System.out.println("Dados salvos com sucesso!");
    } catch (IOException e) {
        System.out.println("Erro ao salvar dados: " + e.getMessage());
    }
}

// Método para recuperar dados
private static void recuperarDados(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica) {
    System.out.println("\n--- Recuperar Dados ---");
    System.out.print("Digite o prefixo dos arquivos: ");
    String prefixo = scanner.nextLine();
```

```
        try {
            repoFisica.recuperar(prefixo + ".fisica.bin");
            repoJuridica.recuperar(prefixo + ".juridica.bin");
            System.out.println("Dados recuperados com sucesso!");
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Erro ao recuperar dados: " + e.getMessage());
        }
    }
}
```

Pessoa.java:

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

public class Pessoa implements Serializable {
    // Atributos
    private int id;
    private String nome;

    // Construtor padrão
    public Pessoa() {
    }

    // Construtor completo
    public Pessoa(int id, String nome) {
        this.id = id;
        this.nome = nome;
    }
```

```
    // Método exibir para impressão dos dados
    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 {
    // Atributos adicionais
    private String cpf;
    private int idade;

    // Construtor padrão
    public PessoaFisica() {
    }

    // Construtor completo
    public PessoaFisica(int id, String nome, String cpf, int idade) {
        super(id, nome); // Chama o construtor da classe Pessoa
        this.cpf = cpf;
        this.idade = idade;
    }

    // Método exibir polimórfico (sobrescrito)
    @Override
    public void exibir() {
        super.exibir(); // Chama o método exibir da classe Pessoa
        System.out.println("CPF: " + cpf);
        System.out.println("Idade: " + idade);
    }

    // Getters e Setters
    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;
    }
}
```

PessoaJuridica.java:

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

public class PessoaJuridica extends Pessoa implements Serializable {
    // Atributo adicional
    private String cnpj;

    // Construtor padrão
    public PessoaJuridica() {
    }

    // Construtor completo
    public PessoaJuridica(int id, String nome, String cnpj) {
```

```
        super(id, nome); // Chama o construtor da classe Pessoa
        this.cnpj = cnpj;
    }

    // Método exibir polimórfico (sobrescrito)
    @Override
    public void exibir() {
        super.exibir(); // Chama o método exibir da classe Pessoa
        System.out.println("CNPJ: " + cnpj);
    }

    // Getters e Setters
    public String getCnpj() {
        return cnpj;
    }

    public void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }
}
```

PessoaFisicaRepo.java:

```
package model;

import java.io.FileOutputStream;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
```

```
import java.io.ObjectOutputStream;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class PessoaFisicaRepo implements Serializable {
  // Lista para armazenar objetos do tipo PessoaFisica
  private List<PessoaFisica> pessoasFisicas;

  // Construtor
  public PessoaFisicaRepo() {
    this.pessoasFisicas = new ArrayList<>();
  }

  // Método para inserir uma PessoaFisica
  public void inserir(PessoaFisica pessoaFisica) {
    pessoasFisicas.add(pessoaFisica);
  }

  // Método para alterar uma PessoaFisica existente
  public void alterar(PessoaFisica pessoaFisicaAtualizada) {
    for (int i = 0; i < pessoasFisicas.size(); i++) {
      if (pessoasFisicas.get(i).getId() == pessoaFisicaAtualizada.getId()) {
        pessoasFisicas.set(i, pessoaFisicaAtualizada); // Substitui a entidade existente
        break;
      }
    }
  }
```

```
    // Método para excluir uma PessoaFisica pelo ID
    public void excluir(int id) {
        pessoasFisicas.removeIf(pessoa -> pessoa.getId() == id);
    }

    // Método para obter uma PessoaFisica pelo ID
    public PessoaFisica obter(int id) {
        for (PessoaFisica pessoa : pessoasFisicas) {
            if (pessoa.getId() == id) {
                return pessoa; // Retorna a pessoa física encontrada
            }
        }
        return null; // Retorna null se não encontrar
    }

    // Método para obter todas as pessoas físicas (cópia defensiva)
    public List<PessoaFisica> obterTodos() {
        return new ArrayList<>(pessoasFisicas); // Retorna uma cópia da lista
    }

     // Método para persistir os dados em um arquivo binário (ecoando exceções)
    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream outputStream = new ObjectOutputStream(new
FileOutputStream(nomeArquivo))) {
            outputStream.writeObject(pessoasFisicas); // Serializa a lista de pessoas físicas
            System.out.println("Dados de Pessoa Física persistidos com sucesso em " + nomeArquivo);
        }
    }

    // Método para recuperar os dados de um arquivo binário (ecoando exceções)
```

```
    @SuppressWarnings("unchecked")
    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
            pessoasFisicas = (List<PessoaFisica>) inputStream.readObject(); // Desserializa a lista de pessoas físicas
            System.out.println("Dados de Pessoa Física recuperados com sucesso de " + nomeArquivo);
        }
    }
}
```

PessoaJuridicaRepo.java:

```
package model;

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

public class PessoaJuridicaRepo implements Serializable {
    // Lista para armazenar objetos do tipo PessoaJuridica
    private List<PessoaJuridica> pessoasJuridicas;

    // Construtor
    public PessoaJuridicaRepo() {
```

```
    this.pessoasJuridicas = new ArrayList<>();
}

// Método para inserir uma PessoaJuridica
public void inserir(PessoaJuridica pessoaJuridica) {
    pessoasJuridicas.add(pessoaJuridica);
}

// Método para alterar uma Pesso    aJuridica existente
public void alterar(PessoaJuridica pessoaJuridicaAtualizada) {
    for (int i = 0; i < pessoasJuridicas.size(); i++) {
        if (pessoasJuridicas.get(i).getId() == pessoaJuridicaAtualizada.getId()) {
            pessoasJuridicas.set(i, pessoaJuridicaAtualizada); // Substitui a entidade existente
            break;
        }
    }
}

// Método para excluir uma PessoaJuridica pelo ID
public void excluir(int id) {
    pessoasJuridicas.removeIf(pessoa -> pessoa.getId() == id);
}

// Método para obter uma PessoaJuridica pelo ID
public PessoaJuridica obter(int id) {
    for (PessoaJuridica pessoa : pessoasJuridicas) {
        if (pessoa.getId() == id) {
            return pessoa;
        }
```

```
        }
        return null; // Retorna null se não encontrar
    }

    // Método para obter todas as pessoas jurídicas (cópia defensiva)
    public List<PessoaJuridica> obterTodos() {
        return new ArrayList<>(pessoasJuridicas); // Retorna uma cópia da lista
    }

     // Método para persistir os dados em um arquivo binário (ecoando exceções)
    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream outputStream = new ObjectOutputStream(new
FileOutputStream(nomeArquivo))) {
            outputStream.writeObject(pessoasJuridicas); // Serializa a lista de pessoas jurídicas
            System.out.println("Dados de Pessoa Jurídica persistidos com sucesso em " + nomeArquivo);
        }
    }

    // Método para recuperar os dados de um arquivo binário (ecoando exceções)
    @SuppressWarnings("unchecked")
    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
            pessoasJuridicas = (List<PessoaJuridica>) inputStream.readObject(); // Desserializa a lista de pessoas
jurídicas
            System.out.println("Dados de Pessoa Jurídica recuperados com sucesso de " + nomeArquivo);
        }
    }
}
```