POLO MADUREIRA - RIO DE JANEIRO - RJ
NIVEL 1 : INICIANDO O CAMINHO PELO JAVA
TURMA 2024.1 - MUNDO 3
ALUNO: MAYLSON SIMOES

Relatório discente de acompanhamento

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.

1º Procedimento | Criação das Entidades e Sistema de Persistência

A primeira etapa do trabalho foi implementar as classes Pessoa, PessoaFisica e PessoaJuridica.

```java
package cadastropoo.model.entidades;

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;
    }

    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);
        System.out.println("Nome: " + nome);
    }
}
```

```java
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;
    }

    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;
    }

    @Override
    public void exibir() {
        System.out.println("ID: " + getId());
        System.out.println("Nome: " + getNome());
        System.out.println("CPF: " + cpf);
        System.out.println("Idade: " + idade);
        System.out.println("_________________" + "\n");
    }
```

```java
package cadastropoo.model.entidades;

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;
    }

    public String getCnpj() {
        return cnpj;
    }

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

    @Override
    public void exibir() {
        System.out.println("ID: " + getId());
        System.out.println("Nome: " + getNome());
        System.out.println("CNPJ: " + cnpj);
        System.out.println("__________________" + "\n");
    }
}
```

Após isso foi criado as classes gerenciadoras para controlar a persistência dos dados.

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

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

    public void alterar(PessoaFisica pessoa) {
        for (int i = 0; i < pessoasFisicas.size(); i++) {...6 lines }
    }

    public boolean excluir(int id) {
        for (int i = 0; i < pessoasFisicas.size(); i++) {...6 lines }
        return false;
    }

    public PessoaFisica obter(int id) {
        for (PessoaFisica pessoa : pessoasFisicas) {...5 lines }
        return null;
    }

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

    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
            out.writeObject(pessoasFisicas);
        }
    }

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
            pessoasFisicas = (ArrayList<PessoaFisica>) in.readObject();
        }
    }
```

```java
import ...7 lines

public class PessoaJuridicaRepo {
    private ArrayList<PessoaJuridica> pessoasJuridicas = new ArrayList<>();

    public void inserir(PessoaJuridica pessoa) {...3 lines }

    public void alterar(PessoaJuridica pessoa) {
        for (int i = 0; i < pessoasJuridicas.size(); i++) {...6 lines }
    }

    public boolean excluir(int id) {
        for (int i = 0; i < pessoasJuridicas.size(); i++) {...6 lines }
        return false;
    }

    public PessoaJuridica obter(int id) {
        for (PessoaJuridica pessoa : pessoasJuridicas) {...5 lines }
        return null;
    }

    public ArrayList<PessoaJuridica> obterTodos() {...3 lines }

    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream output = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
            output.writeObject(pessoasJuridicas);
        }
    }

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream input = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
            pessoasJuridicas = (ArrayList<PessoaJuridica>) input.readObject();
        }
    }
}
```

E por último foi implementada os testes de persistência no método Main()

```java
public class CadastroPOOTestes {
    public static void main(String[] args) {
        try {
            PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
            PessoaFisica pessoa1 = new PessoaFisica(1, "Ewerton", "938.489.620-98", 18);
            PessoaFisica pessoa2 = new PessoaFisica(2, "Ricardo", "277.047.440-54", 41);
            repo1.inserir(pessoa1);
            repo1.inserir(pessoa2);
            repo1.persistir("pessoasFisicas.bin");
            System.out.println("Dados de Pessoas Físicas Armazenados");
            PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
            repo2.recuperar("pessoasFisicas.bin");
            System.out.println("Dados de Pessoas Físicas Recuperados");
            for (PessoaFisica pessoa : repo2.obterTodos()) {
                pessoa.exibir();
            }

            // Testando repositório de pessoas jurídicas
            PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
            PessoaJuridica empresa1 = new PessoaJuridica(1, "Empresa A", "77.154.419/0001-22");
            PessoaJuridica empresa2 = new PessoaJuridica(2, "Empresa B", "76.111.810/0001-87");
            repo3.inserir(empresa1);
            repo3.inserir(empresa2);
            repo3.persistir("pessoasJuridicas.bin");
            System.out.println("Dados de Pessoas Juridicas Armazenados");
            PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
            repo4.recuperar("pessoasJuridicas.bin");
            System.out.println("Dados de Pessoas Juridicas Recuperados");

            for (PessoaJuridica empresa : repo4.obterTodos()) {
                empresa.exibir();
            }
        } catch (IOException | ClassNotFoundException e) {
            System.err.println("Erro: " + e.getMessage());
        }
    }
```

it - CadastroPOO (run)

```
run:
Dados de Pessoas Físicas Armazenados
Dados de Pessoas Físicas Recuperados
ID: 1
Nome: Ewerton
CPF: 938.489.620-98
Idade: 18
_____________

ID: 2
Nome: Ricardo
CPF: 277.047.440-54
Idade: 41
_____________

Dados de Pessoas Juridicas Armazenados
Dados de Pessoas Juridicas Recuperados
ID: 1
Nome: Empresa A
CNPJ: 77.154.419/0001-22
_____________

ID: 2
Nome: Empresa B
CNPJ: 76.111.810/0001-87
_____________
```

Análise e Conclusão:

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

Vantagens:

Reutilização de código: As subclasses herdam atributos e comportamentos da classe pai, permitindo a reutilização de código.

Desvantagens:

Acoplamento: A herança cria um acoplamento entre a classe pai e suas subclasses, o que pode tornar o código mais difícil de manter e modificar.

Hierarquias profundas: Hierarquias de herança muito profundas podem tornar o código complexo e difícil de entender.

2. Por que a interface Serializable é necessária ao efetuar persistência em arquivos binários?

A interface Serializable é necessária ao efetuar persistência em arquivos binários porque ela permite que os objetos Java sejam convertidos em uma sequência de bytes, que podem ser gravados em um arquivo e posteriormente lidos e reconstruídos em objetos Java novamente. Isso é essencial para a serialização e desserialização de objetos ao persistir e recuperar dados de arquivos binários.

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

A API Stream no Java utiliza o paradigma funcional para realizar operações em coleções de dados de forma declarativa e sem efeitos colaterais. Isso é alcançado através de operações que permitem processar e manipular os dados de forma eficiente e concisa. O paradigma funcional promove o uso de funções como cidadãos de primeira classe e permite escrever código mais expressivo e fácil de entender.

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

No Java, o padrão de desenvolvimento mais comum na persistência de dados em arquivos é o uso de classes de entrada e saída de dados, como FileInputStream e FileOutputStream, combinadas com classes de serialização, como ObjectInputStream e ObjectOutputStream. Este padrão permite a leitura e gravação de objetos Java em arquivos binários, proporcionando uma forma eficaz de persistência de dados.

2º Procedimento | Criação do Cadastro em Modo Texto

Essa etapa constitui em criar uma interface para o usuário realizar as ações no programa.

Segue a implementação no método Main().

```
public class CadastroPOO {
    public static void main(String[] args) {
        PessoaFisicaRepo repoPessoaFisica = new PessoaFisicaRepo();
        PessoaJuridicaRepo repoPessoaJuridica = new PessoaJuridicaRepo();
        int opcao;
        String tipoPessoa;
        String prefixo;
        Integer id;
        do {
            Scanner scanner = new Scanner(System.in);
            System.out.println("Selecione uma opção:");
            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 - Finalizar");
            opcao = scanner.nextInt();

            switch (opcao) {
case 1:
    tipoPessoa = pessoaFisicaOuPessoaJuridica(scanner);
    switch (tipoPessoa) {
        case "F":
            try{
                PessoaFisica pessoaFisica = lerDadosPessoaFisica(scanner);
                repoPessoaFisica.inserir(pessoaFisica);
            } catch(Exception e){
                System.out.println(e.getMessage());
            }
            break;
        case "J":
            try{
                PessoaJuridica pessoaJuridica = lerDadosPessoaJuridica(scanner);
                repoPessoaJuridica.inserir(pessoaJuridica);
            } catch(Exception e){
                System.out.println(e.getMessage());
            }
            break;
        default:
            alertaOpçãoInvalida();
            break;
    }
    break;
```

```
case 2:
    tipoPessoa = pessoaFisicaOuPessoaJuridica(scanner);
    id = getId(scanner);
    if (id == null){
        alertaOpçãoInvalida();
        break;
    }
    switch (tipoPessoa) {
        case "F":
            PessoaFisica pessoaFisica = repoPessoaFisica.obter(id);
            if (isPessoaValida(pessoaFisica)){
                pessoaFisica.exibir();

                pessoaFisica = alterarDadosPessoaFisica(scanner, pessoaFisica);
                repoPessoaFisica.alterar(pessoaFisica);
            } else alertaPessoaInvalida();

            break;
        case "J":
            PessoaJuridica pessoaJuridica = repoPessoaJuridica.obter(id);
            if (isPessoaValida(pessoaJuridica)){
                pessoaJuridica.exibir();

                pessoaJuridica = alterarDadosPessoaJuridica(scanner, pessoaJuridica);
                repoPessoaJuridica.alterar(pessoaJuridica);
            } else alertaPessoaInvalida();

            break;
        default:
            alertaOpçãoInvalida();
            break;
    }
    break;
```

```
case 3:
    tipoPessoa = pessoaFisicaOuPessoaJuridica(scanner);
    id = getId(scanner);
    if (id == null){
        alertaOpçãoInvalida();
        break;
    }

    switch (tipoPessoa) {
        case "F":
            if (repoPessoaFisica.excluir(id))
                System.out.println("Pessoa excluida com sucesso.");
            else
                System.out.println("Falha ao excluir.");

            break;
        case "J":
            if (repoPessoaJuridica.excluir(id))
                System.out.println("Empresa excluida com sucesso.");
            else
                System.out.println("Falha ao excluir.");

            break;
        default:
            alertaOpçãoInvalida();
            break;
    }

    break;
```

```
case 4:
    tipoPessoa = pessoaFisicaOuPessoaJuridica(scanner);
    id = getId(scanner);
    if (id == null){
        alertaOpçãoInvalida();
        break;
    }

    switch (tipoPessoa) {
        case "F":
            PessoaFisica pessoaFisica = repoPessoaFisica.obter(id);
            if (isPessoaValida(pessoaFisica))
                pessoaFisica.exibir();
            else alertaPessoaInvalida();

            break;
        case "J":
            PessoaJuridica pessoaJuridica = repoPessoaJuridica.obter(id);
            if (isPessoaValida(pessoaJuridica))
                pessoaJuridica.exibir();
            else alertaPessoaInvalida();

            break;
        default:
            alertaOpçãoInvalida();
            break;
    }

    break;
```

```
case 5:
    tipoPessoa = pessoaFisicaOuPessoaJuridica(scanner);
    switch (tipoPessoa) {
        case "F":
            for (PessoaFisica pessoa : repoPessoaFisica.obterTodos()) {
                pessoa.exibir();
            }

            break;
        case "J":
            for (PessoaJuridica empresa : repoPessoaJuridica.obterTodos()) {
                empresa.exibir();
            }

            break;
        default:
            alertaOpçãoInvalida();
            break;
    }

    break;
case 6:
    prefixo = obterPrefixo(scanner);

    try {
        repoPessoaFisica.persistir(prefixo +".fisica.bin");
        System.out.println("Dados de Pessoas Físicas Armazenados");

        repoPessoaJuridica.persistir(prefixo +".juridica.bin");
        System.out.println("Dados de Pessoas Juridica Armazenados");
    } catch (IOException ex) {
        Logger.getLogger(CadastroPOO.class.getName()).log(Level.SEVERE, null, ex);
    }

    break;
```

```
            case 7:
                prefixo = obterPrefixo(scanner);

                try {
                    repoPessoaFisica.recuperar(prefixo +".fisica.bin");
                    System.out.println("Dados de Pessoas Físicas Recuperados");

                    repoPessoaJuridica.recuperar(prefixo +".juridica.bin");
                    System.out.println("Dados de Pessoas Juridica Armazenados");
                } catch (IOException | ClassNotFoundException ex) {
                    Logger.getLogger(CadastroPOO.class.getName()).log(Level.SEVERE, null, ex);
                }

                break;
            case 0:
                System.out.println("Finalizando...");
                break;
            default:
                alertaOpçãoInvalida();
                break;
        }
    } while (opcao != 0);
}

private static void alertaOpçãoInvalida(){
    System.out.println("Opção inválida. Tente novamente.");
}

private static void alertaPessoaInvalida(){
    System.out.println("Pessoa/Empresa não encontrada.");
}

private static boolean isPessoaValida(Object obj){
    return obj != null;
}

private static String pessoaFisicaOuPessoaJuridica(Scanner scanner){
    System.out.println("F - Pessoa Física | J - Pessoa Jurídica");
    return scanner.next();
}

private static String obterPrefixo(Scanner scanner){
    System.out.println("Informe o prefixo do arquivo a ser salvo");
    return scanner.next();
}

private static Integer getId(Scanner scanner){
    System.out.println("Digite o id da pessoa:");
    try{
        return scanner.nextInt();
    } catch (Exception e) {
        return null;
    }
}
```

```java
private static PessoaFisica lerDadosPessoaFisica(Scanner scanner) throws Exception{
    try{
        System.out.println("Insira os dados...");
        System.out.println("Digite o id da pessoa");
        int id = scanner.nextInt();
        System.out.println("Digite o nome da pessoa");
        String nome = scanner.next();
        System.out.println("Digite o cpf da pessoa");
        String cpf = scanner.next();
        System.out.println("Digite a idade da pessoa");
        int idade = scanner.nextInt();
        return new PessoaFisica(id, nome, cpf, idade);
    } catch(Exception e){
        throw new Exception("Dado digitado está incorreto. Tente novamente.");
    }
}

private static PessoaJuridica lerDadosPessoaJuridica(Scanner scanner) throws Exception{
    try{
        System.out.println("Digite o id da empresa");
        int id = scanner.nextInt();
        System.out.println("Digite o nome da empresa");
        String nome = scanner.next();
        System.out.println("Digite o cnpj da empresa");
        String cnpj = scanner.next();
        return new PessoaJuridica(id, nome, cnpj);
    } catch(Exception e){
        throw new Exception("Dado digitado está incorreto. Tente novamente.");
    }
}

private static PessoaFisica alterarDadosPessoaFisica(Scanner scanner, PessoaFisica pessoa){
    boolean continuar = true;
    while(continuar){
        System.out.println("Selecione uma opção:");
        System.out.println("N - Nome");
        System.out.println("C - CPF");
        System.out.println("I - Idade");
        System.out.println("F - Finalizar");
        String opcao = scanner.next();
        try{
            switch (opcao) {
                case "N":
                    System.out.println("Digite o novo nome:");
                    String nome = scanner.next();
                    pessoa.setNome(nome);
                    break;
                case "C":
                    System.out.println("Digite o novo CPF:");
                    String cpf = scanner.next();
                    pessoa.setCpf(cpf);
                    break;
                case "I":
                    System.out.println("Digite a nova idade:");
                    int idade = scanner.nextInt();
                    pessoa.setIdade(idade);
                    break;
                case "F":
                    continuar = false;
                    break;
                default:
                    alertaOpçãoInvalida();
            }
        } catch(Exception e){...3 lines }
    }
    return pessoa;
```

```java
private static PessoaJuridica alterarDadosPessoaJuridica(Scanner scanner, PessoaJuridica empresa){
    boolean continuar = true;

    while(continuar){
        System.out.println("Selecione uma opção:");
        System.out.println("N - Nome");
        System.out.println("C - CNPJ");
        System.out.println("F - Finalizar");
        String opcao = scanner.next();

        try{
            switch (opcao) {
                case "N":
                    System.out.println("Digite o novo nome:");
                    String nome = scanner.next();
                    empresa.setNome(nome);
                    break;
                case "C":
                    System.out.println("Digite o novo CNPJ:");
                    String cnpj = scanner.next();
                    empresa.setCnpj(cnpj);
                    break;
                case "F":
                    continuar = false;
                    break;
                default:
                    alertaOpçãoInvalida();
            }
        } catch(Exception e){
            alertaOpçãoInvalida();
        }
    }

    return empresa;
}
```

t - CadastroPOO (run)

```
run:
Selecione uma opção:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
6 - Salvar dados
7 - Recuperar dados
0 - Finalizar
1
F - Pessoa Física | J - Pessoa Jurídica
F
Insira os dados...
Digite o id da pessoa
1
Digite o nome da pessoa
NomeTeste
Digite o cpf da pessoa
123.456.789-55
Digite a idade da pessoa
29
Selecione uma opção:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
6 - Salvar dados
7 - Recuperar dados
0 - Finalizar
5
F - Pessoa Física | J - Pessoa Jurídica
F
ID: 1
Nome: NomeTeste
CPF: 123.456.789-55
Idade: 29
```

Análise e Conclusão:

1. **O que são elementos estáticos e qual o motivo para o método main adotar esse modificador?**

Os elementos estáticos em Java são atributos e métodos que pertencem à classe em si, e não a instâncias individuais da classe. Isso significa que eles podem ser acessados diretamente através do nome da classe, sem a necessidade de criar um objeto da classe

2. **Para que serve a classe Scanner?**

A classe Scanner em Java é uma classe que fornece métodos para ler entrada de dados do usuário a partir do console ou de outros fluxos de entrada, como arquivos

3. **Como o uso de classes de repositório impactou na organização do código?**

O uso de classes de repositório impactou positivamente na organização do código, pois ela ajudou a separar as preocupações relacionadas à persistência de dados das outras partes do sistema.

Além disso, as classes de repositório promovem o princípio de responsabilidade única, onde cada classe é responsável por uma única funcionalidade específica. Isso facilita a manutenção do código, pois as alterações em uma parte do sistema não afetam necessariamente outras partes, desde que a interface pública das classes de repositório permaneça inalterada.

Link repositório: https://github.com/VascoMay/CadastroPOO