



**Campus:** Polo Ingleses

**Curso:** Desenvolvimento Full Stack

**Disciplina:** Nível 1 - Iniciando o Caminho Pelo Java

**Turma:** 9001

**Semestre:** 23.3

**Aluno:** Alexandre Henrique Fernandes Nolla

## 1º Procedimento | Criação das Entidades e Sistema de Persistência

**Objetivo da prática:** Executar uma carga inicial de dados nas classes 'Pessoa Física' e 'Pessoa Jurídica' na execução da aplicação através do método main da classe principal.

### Resultados da execução dos códigos:

```
PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS Run: CadastroPooApplication + - [ ] ...

Dados de Pessoa Fisica Armazenados.
Dados de Pessoa Fisica Recuperados.
ID: 1
Nome: Ana
CPF: 11111111111
Idade: 25
ID: 2
Nome: Carlos
CPF: 22222222222
Idade: 52
Dados de Pessoa Juridica Armazenados.
Dados de Pessoa Juridica Recuperados.
ID: 3
Nome: XPT0 Sales
CNPJ: 33333333333
ID: 4
Nome: XPT0 Solutions
CNPJ: 44444444444
=====
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
=====

CadastroPooApplication<CadastroPOO> (MissaoPratica1)  You, 32 minutes ago Ln 21, Col 1 Spaces: 4 UTF-8 LF {} Java Go Live
```

### Análise e Conclusão:

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

*A herança, na programação orientada a objetos, oferece vantagens como a reutilização de código, onde as classes filhas herdam atributos e métodos da classe pai, reduzindo redundâncias e simplificando a manutenção. Também, as modificações na classe pai são automaticamente refletidas nas filhas, facilitando a coesão e consistência do sistema. E, por fim, a herança proporciona polimorfismo, permitindo tratar objetos de classes distintas de maneira uniforme.*

*No entanto, as desvantagens incluem o potencial acoplamento forte entre as classes pai e filhas, tornando o sistema mais sensível a mudanças e dificultando a manutenção. A herança também pode complicar os testes unitários, uma vez que é necessário considerar o comportamento integrado das classes pai e filhas.*

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

*A interface Serializable é necessária ao efetuar a persistência em arquivos binários em Java, pois está envolvida no processo de serialização de objetos. A serialização, que consiste na conversão de um objeto em uma sequência de bytes, permite que o objeto seja armazenado em arquivos binários ou transmitido pela rede.*

*Quando um objeto de uma classe que implementa Serializable é persistido em um arquivo binário, o Java realiza a conversão dos dados do objeto em uma forma serializada. Essa forma serializada pode ser facilmente armazenada e, posteriormente, reconstruída quando necessário.*

*Assim, ao realizar a persistência em arquivos binários, a presença da interface Serializable garante que os objetos possam ser transformados em bytes e, posteriormente, restaurados para sua forma original.*

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

*O paradigma funcional na API Stream do Java, introduzido no Java 8, oferece uma abordagem declarativa e concisa para processar coleções de dados, como Listas e Conjuntos. A API Stream utiliza operações de alto nível, como map, filter e reduce, permitindo expressar operações complexas de forma legível e concisa, sem a necessidade de loops explícitos.*

*Expressões lambda são amplamente utilizadas na API Stream, proporcionando uma forma concisa de representar comportamentos que podem ser passados como argumentos para seus métodos. O paradigma funcional na API Stream incentiva a imutabilidade, evitando efeitos colaterais e promovendo o uso de funções puras, resultando em código mais legível, modular e fácil de entender.*

*Essa combinação de funcionalidades facilita a exploração eficiente de operações paralelas. Em resumo, a API Stream com o paradigma funcional no Java oferece uma abordagem poderosa e*

expressiva para lidar com operações em coleções de dados, proporcionando benefícios em termos de concisão, clareza e potencial de paralelismo.

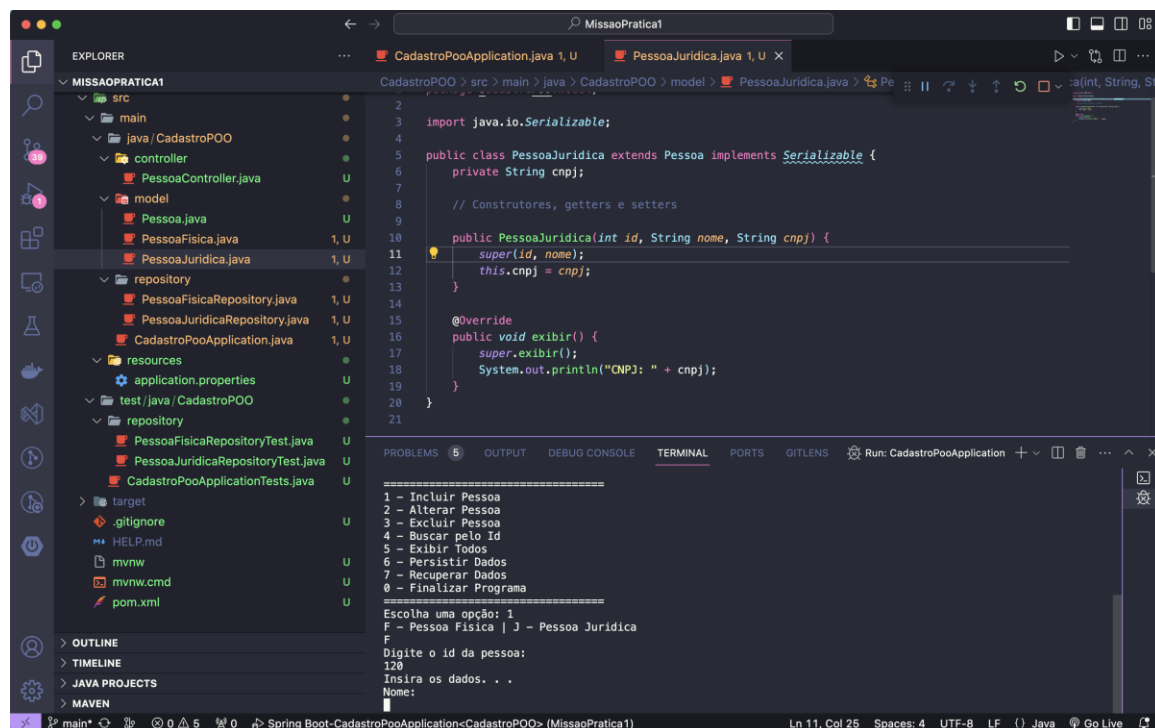
d) Quando trabalhamos com Java, qual padrão de desenvolvimento é adotado na persistência de dados em arquivos?

O padrão predominante na persistência de dados em Java depende do tipo de persistência, seja em bancos de dados relacionais, onde o uso de ORM é comum, ou em arquivos locais, como arquivos de texto, binários ou formato JSON, o uso de bibliotecas de manipulação de arquivos ou serialização/desserialização manual também é comum. Bibliotecas como Jackson para JSON ou `BufferedWriter/BufferedReader` para arquivos de texto são exemplos de abordagens nesse contexto.

## 2º Procedimento | Criação do Cadastro em Modo Texto

**Objetivo da prática:** Gerar um CRUD (create, read, update e delete) das classes 'Pessoa Física' e 'Pessoa Jurídica' e manipulá-las através do console, gerando persistência de dados em arquivo binário.

Resultados da execução dos códigos:



The screenshot displays an IDE with the following components:

- EXPLORER:** Shows the project structure for 'MISSAO PRATICA 1'. The 'model' package contains `Pessoa.java`, `PessoaFisica.java`, and `PessoaJuridica.java`. The 'repository' package contains `PessoaFisicaRepository.java` and `PessoaJuridicaRepository.java`. The 'controller' package contains `PessoaController.java`. The 'main' package contains `CadastroPooApplication.java`.
- EDITOR:** Displays the code for `PessoaJuridica.java`. The code includes an import for `java.io.Serializable`, a class declaration `public class PessoaJuridica extends Pessoa implements Serializable`, a private field `private String cnpj;`, and a constructor `public PessoaJuridica(int id, String nome, String cnpj) { super(id, nome); this.cnpj = cnpj; }`. It also includes an `@Override` method `public void exibir() { super.exibir(); System.out.println("CNPJ: " + cnpj); }`.
- TERMINAL:** Shows the output of the application. It displays a menu with options: 1 - Incluir Pessoa, 2 - Alterar Pessoa, 3 - Excluir Pessoa, 4 - Buscar pelo Id, 5 - Exibir Todos, 6 - Persistir Dados, 7 - Recuperar Dados, and 0 - Finalizar Programa. The user has selected option 1, and the program prompts for the person's ID and name.

```
PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS Run: CadastroPooApplication + - [ ] [ ] ...

=====
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
=====
Escolha uma opção: 1
F - Pessoa Física | J - Pessoa Juridica
F
Digite o id da pessoa:
120
Insira os dados. . .
Nome:
|
CadastroPooApplication<CadastroPOO> (MissaoPratica1) Ln 11, Col 25 Spaces: 4 UTF-8 LF {} Java Go Live
```

```
PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS Run: CadastroPooApplication

Insira os dados. . .
Nome:
ale
CPF:
05193149995
Idade:
29
Pessoa Física adicionada com sucesso!
=====
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
=====
Escolha uma opção: 5
F - Pessoa Física | J - Pessoa Juridica
F
Pessoas Físicas cadastradas:
ID: 1
Nome: ale
CPF: 05193149995
Idade: 29
=====
CadastroPooApplication<CadastroPOO> (MissaoPratica1) Ln 121, Col 41 Spaces: 4 UTF-8 LF (
```

## Análise e Conclusão:

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

*Em Java, elementos estáticos, como variáveis e métodos, estão associados à classe em vez de instâncias específicas da classe. As variáveis estáticas são compartilhadas entre todas as instâncias, enquanto os métodos estáticos pertencem à classe e não exigem uma instância específica.*

*O método main é frequentemente declarado como estático porque serve como ponto de entrada para um programa Java. Sendo estático, pode ser invocado diretamente pela JVM, não necessitando de uma instância da classe que o contém. Essa declaração estática permite que o método main seja acessado diretamente pela classe, essencial para a execução do programa, pois fornece à JVM um ponto de partida sem a necessidade de criar instâncias da classe.*

b) Para que serve a classe Scanner?

*A classe Scanner em Java desempenha um papel fundamental ao simplificar a leitura de dados provenientes de diferentes fontes, como teclado, arquivos ou strings. Pertencente ao pacote java.util, essa classe oferece métodos que facilitam a leitura de uma variedade de tipos de dados, incluindo inteiros, ponto flutuante e caracteres.*

*A principal função do Scanner é tornar mais fácil a captura de dados fornecidos pelo usuário ou presentes em fluxos de entrada. Ao criar uma instância do objeto Scanner, é possível utilizar seus métodos para ler e converter dados de maneira conveniente, otimizando a interação com o usuário e a manipulação de informações em arquivos.*

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

*O uso de classes de repositório teve um impacto significativo na organização do código, proporcionando uma separação clara entre a lógica de negócios e as operações de persistência de dados.*