***Curso: Desenvolvimento Full Stack Disciplina:***
***Iniciando o caminho pelo Java***
***Turma: 2024.3 Semestre: 3°***
***Integrante(s): Daniel Pio pereira***
***Estácio - Catu, BA***
***Matricula: 2023 0702 0151***

**Vantagens e Desvantagens do Uso de Herança**

**Vantagens:**

1. Reutilização de Código: A herança permite que uma classe (subclasse) herde atributos e métodos de outra classe (superclasse). Isso promove a reutilização de código, evitando duplicações e facilitando a manutenção.
2. Organização e Estrutura: Herança ajuda a estruturar o código de maneira hierárquica, o que torna mais fácil entender e gerenciar grandes sistemas.
3. Polimorfismo: Com a herança, as subclasses podem ser tratadas como instâncias de suas superclasses, permitindo o uso de polimorfismo, onde um método pode agir de maneira diferente dependendo da classe que o invoca.

**Desvantagens:**

1. Acoplamento Aumentado: A herança pode aumentar o acoplamento entre classes, tornando o sistema menos flexível e mais difícil de modificar.
2. Dificuldade na Manutenção: Mudanças na superclasse podem afetar todas as subclasses, o que pode introduzir erros e aumentar o custo de manutenção.
3. Problemas de Design: O uso excessivo de herança pode levar a um design de código complexo e difícil de entender. Isso é frequentemente referido como "herança em árvore" que pode se tornar difícil de navegar.


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

A interface Serializable é necessária em Java para permitir que objetos sejam convertidos em um fluxo de bytes, o que é essencial para a persistência em arquivos binários. Quando um objeto é serializado, seus dados podem ser gravados em um arquivo ou transmitidos por uma rede. A serialização é crucial em várias aplicações, pois possibilita a gravação do estado de um objeto e a sua recuperação posterior. Sem essa interface, o Java não saberia como transformar um objeto em um formato que possa ser armazenado ou transmitido, o que limitaria severamente a funcionalidade de aplicações que requerem persistência de dados.

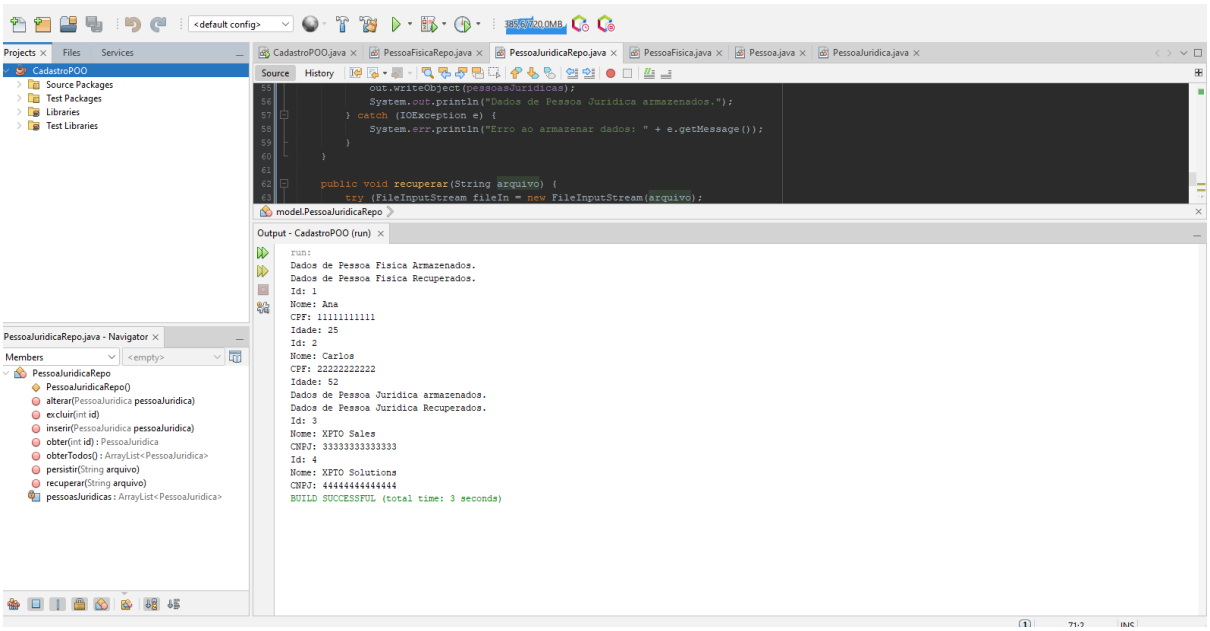
**Uso do Paradigma Funcional pela API Stream no Java**

A API Stream do Java implementa conceitos do paradigma funcional, permitindo operações sobre coleções de dados de maneira mais declarativa. Isso é feito através de uma série de métodos que podem ser encadeados, como map, filter e reduce, possibilitando a transformação e a agregação de dados de forma clara e concisa. Por exemplo, usando streams, um desenvolvedor pode filtrar, transformar e coletar dados de uma lista em poucas linhas de código, tornando a leitura e a manutenção mais fáceis. Essa abordagem também permite aproveitar melhor os processadores multi-core, pois operações em streams podem ser executadas em paralelo.

**Padrão de Desenvolvimento na Persistência de Dados em Arquivos**

Ao trabalhar com Java, o padrão de desenvolvimento mais adotado na persistência de dados em arquivos é o padrão **Data Access Object (DAO)**. Esse padrão separa a lógica de acesso a dados da lógica de negócios, proporcionando uma maneira clara e organizada de interagir com fontes de dados, como arquivos ou bancos de dados. Através de DAOs, a implementação de persistência pode ser alterada sem afetar o restante do aplicativo, o que aumenta a flexibilidade e facilita a manutenção.

A utilização desse padrão é recomendada para manter o código limpo, modular e testável.




**codigo**: https://github.com/DartEe/iniciando-o-caminho-pelo-Java-.git

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

Elementos estáticos em Java referem-se a atributos e métodos que pertencem à classe em si, e não a instâncias específicas dessa classe. Isso significa que todos os objetos da classe compartilham o mesmo valor para atributos estáticos, e os métodos estáticos podem ser chamados sem a necessidade de criar um objeto da classe. Para declarar um elemento como estático, utiliza-se a palavra-chave static.

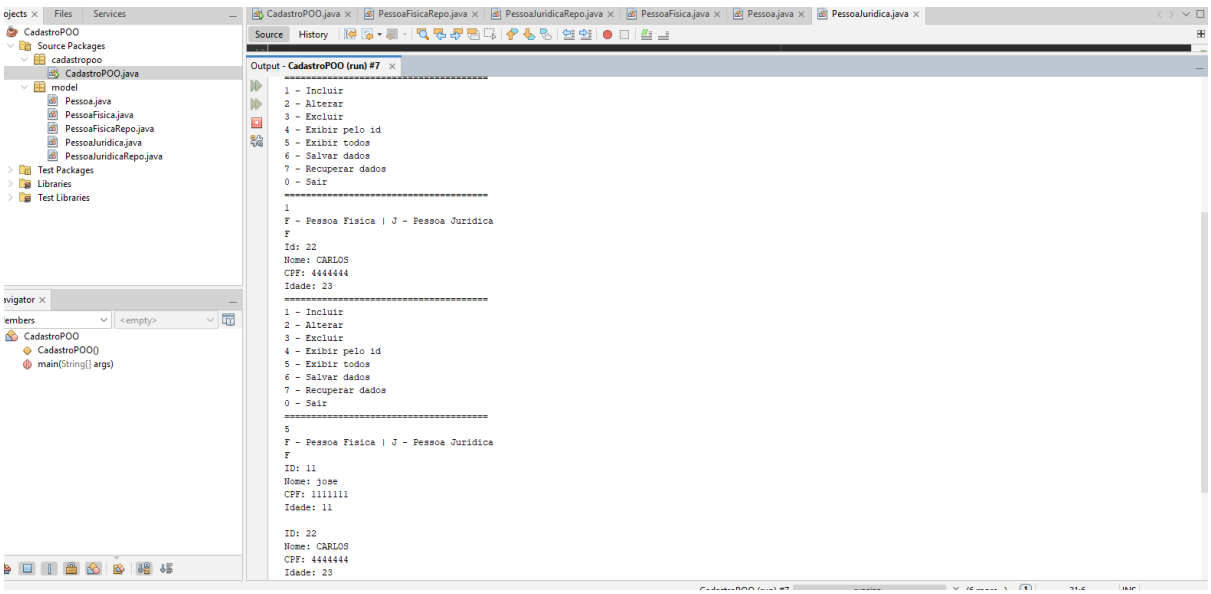

O método main é o ponto de entrada de qualquer aplicação Java. Ele é declarado como estático para que a JVM (Java Virtual Machine) possa invocá-lo sem ter que instanciar a classe. Como o main é o primeiro método a ser executado, não haveria objetos criados antes de sua execução.

## Para que serve a classe Scanner?

A classe Scanner em Java é utilizada para a leitura de entrada de dados. Ela pode ler a entrada de diferentes fontes, como o teclado (entrada padrão), arquivos e strings. A classe oferece métodos convenientes para capturar diferentes tipos de dados, como inteiros, strings, e outros tipos primitivos, tornando-a extremamente útil para interações com o usuário.

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

1. **Separação de Preocupações:** As classes de repositório encapsulam a lógica de acesso a dados, permitindo que a lógica de negócios permaneça independente das operações de persistência. Isso melhora a legibilidade e a manutenção do código.
2. **Modularidade:** As classes de repositório promovem um design modular, onde cada classe é responsável por um conjunto específico de operações de dados. Isso facilita o teste e a reutilização de código.
3. **Facilidade de Manutenção:** Se precisar modificar a forma como os dados são armazenados ou recuperados. Por exemplo, adicionando uma integração com banco de dados.




**Código**:
https://github.com/DartEe/iniciando-o-caminho-pelo-Java-.git