



**Universidade Estácio de Sá**

**Campus Santa Cruz**

**Relatório Discente de Acompanhamento**

**Nome: Tamires de Souza Alves**

**Curso: Desenvolvimento Full Stack**

**Disciplina: Iniciando Caminho Pelo Java**

**Número da Turma: 9001**

**Semestre Letivo: 3º Semestre**

**GitHub:** [**https://github.com/tamiresalves024/MissaoPratica\_Nivel1\_Mundo3.git**](https://github.com/tamiresalves024/MissaoPratica_Nivel1_Mundo3.git)

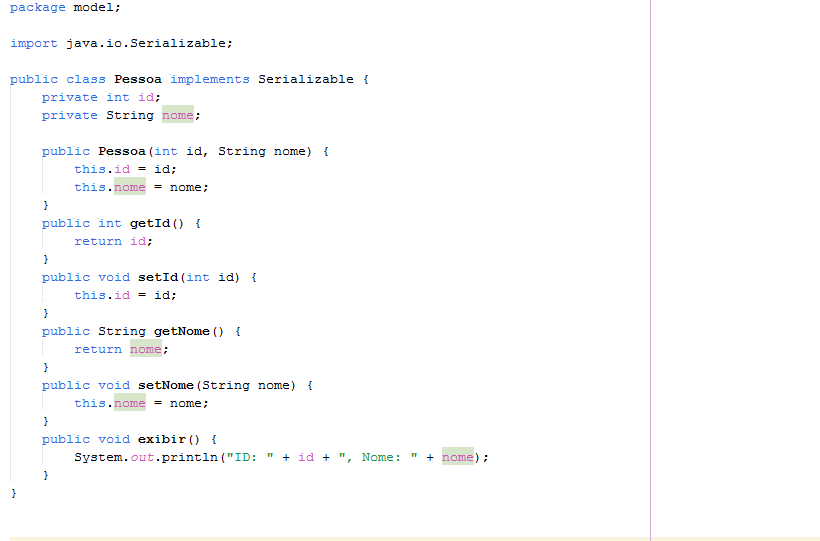
**Implementação de um cadastro de clientes em modo texto, com persistência em arquivos, baseado na tecnologia Java.**

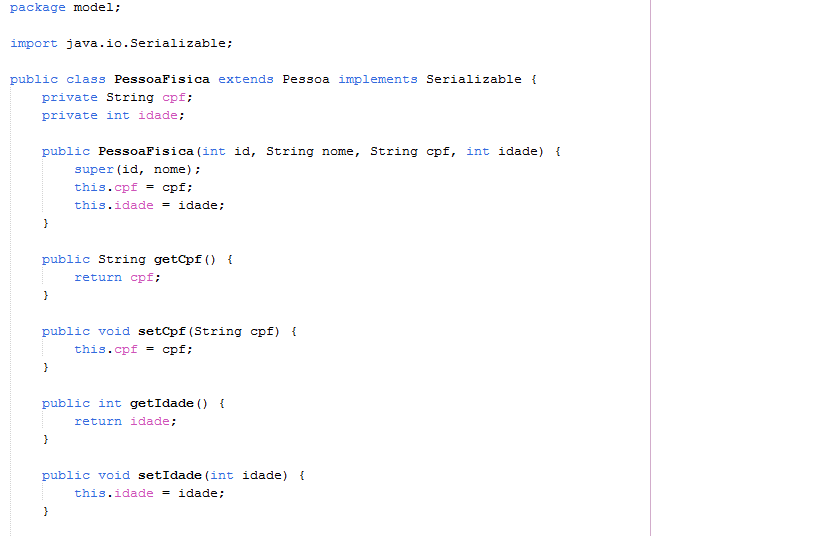
**Objetivo da Prática**

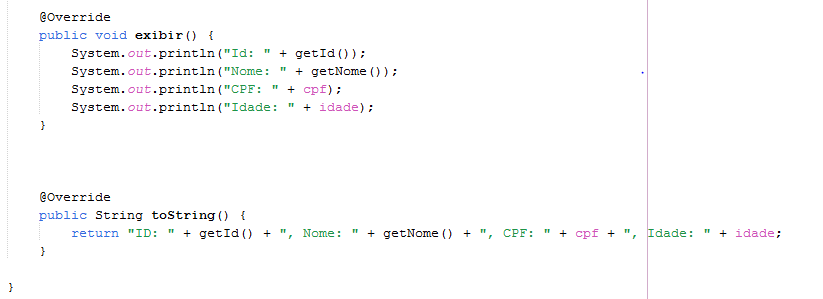
Implementar um sistema cadastral em Java, utilizando os recursos da programação orientada à objetos e a persistência em arquivos binários.

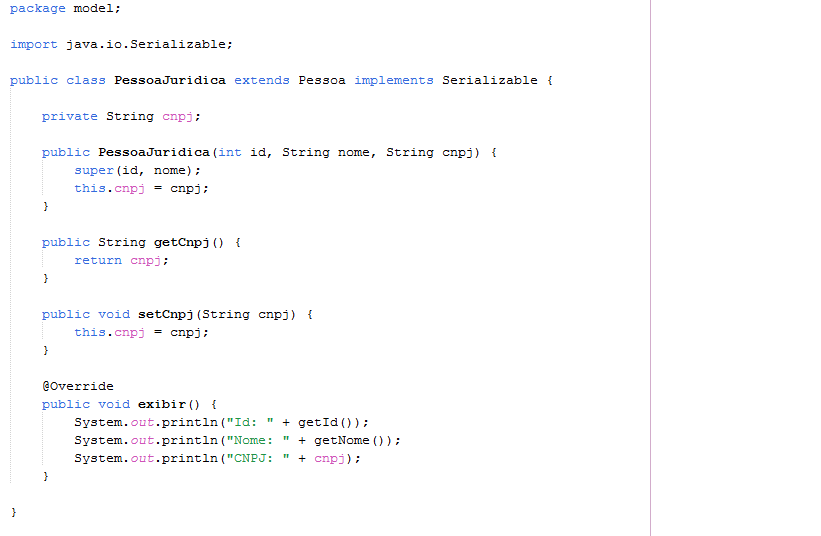
**Procedimento 1 – Criação das Entidades e Sistema de Persistência**

**Códigos do Procedimento 1**

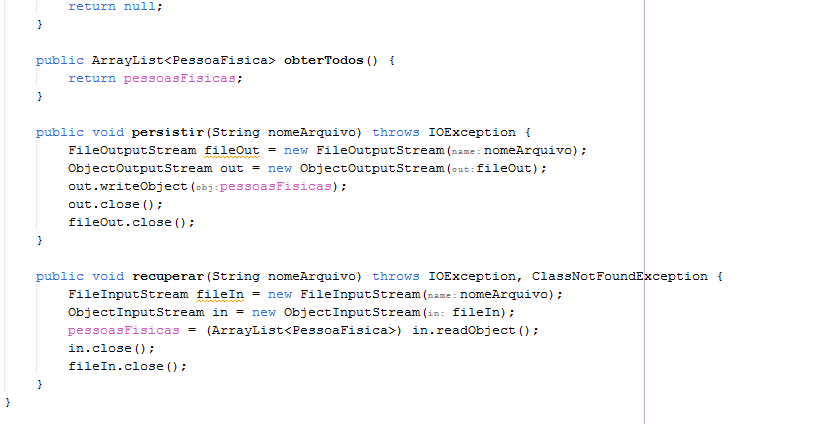






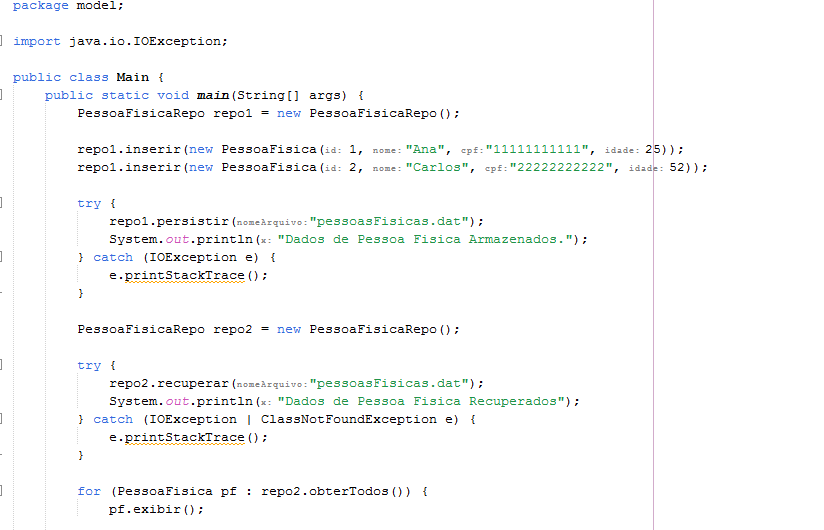


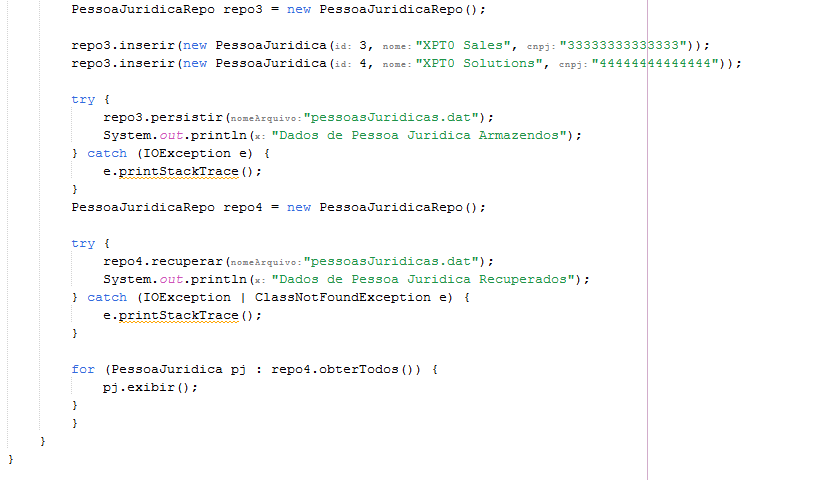




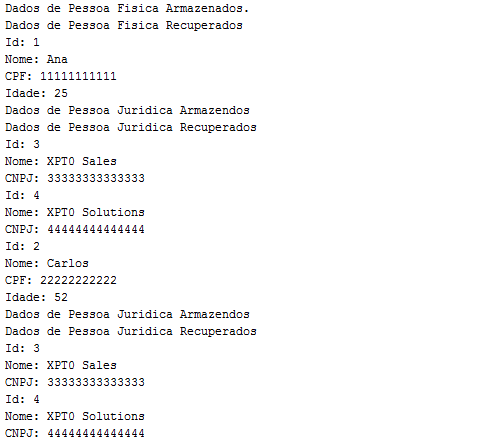








**Resultado do Procedimento 1**

****

**Análise e Conclusão**

**Quais as vantagens e desvantagens do uso de herança?**

Um dos principais benefícios da herança é minimizar a quantidade de código duplicado em um aplicativo, através de compartilhamento de código comum entre várias subclasses. Herança também pode tornar o código do aplicativo mais flexível a mudanças porque as classes que herdam uma superclasse comum podem ser usados ​​alternadamente. Se o tipo de retorno de um método é superclasse. A principal desvantagem de usar a herança é que as duas classes (base e classe herdada) são fortemente acoplados. Isto significa que não podem ser usados independentemente umas das outras.

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

Porque informa ao Java que a classe pode ser convertida e reconstruída posteriormente. Isto é importante ao salvar e recuperar objetos Java em arquivos binários. Se esta interface não for usada, Java não permitirá que classes de objetos sejam “serializados”.

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

A API Stream do Java utiliza o paradigma funcional para manipular e transformar coleções de elementos de forma declarativa e imutável. Isso significa que os elementos são processados sem alterar o estado original da coleção. As operações são realizadas utilizando funções lambda, que representam expressões e comportamentos a serem executados nos elementos da coleção. O paradigma funcional permite uma maior flexibilidade e redução de código, tornando o trabalho com coleções mais eficiente e legível.

1. Quando trabalhamos com Java, qual padrão de desenvolvimento é adotado

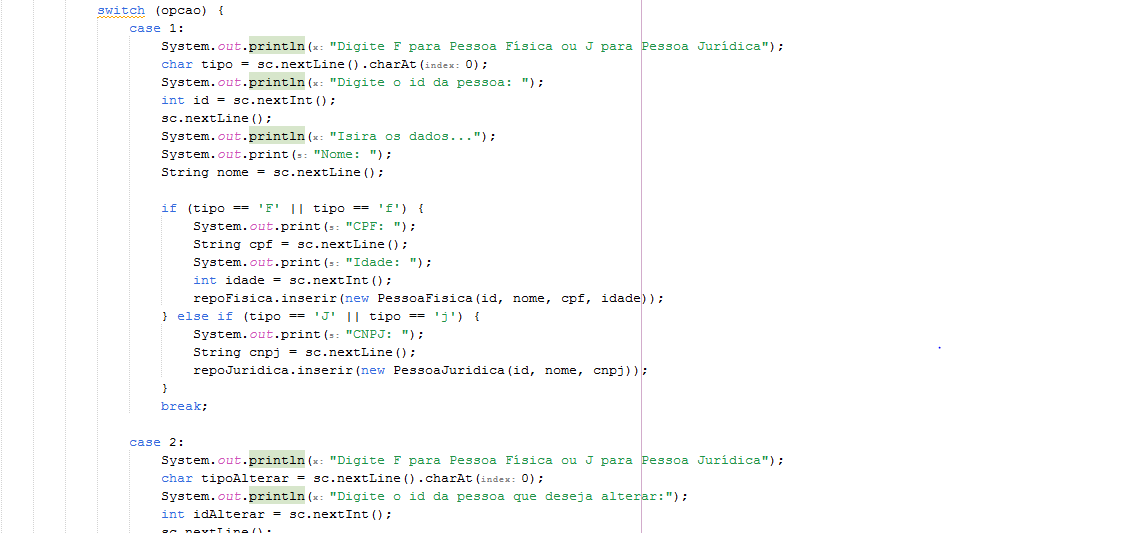
**Quando trabalhamos com Java, qual padrão de desenvolvimento é adotado na persistência de dados em arquivos?**

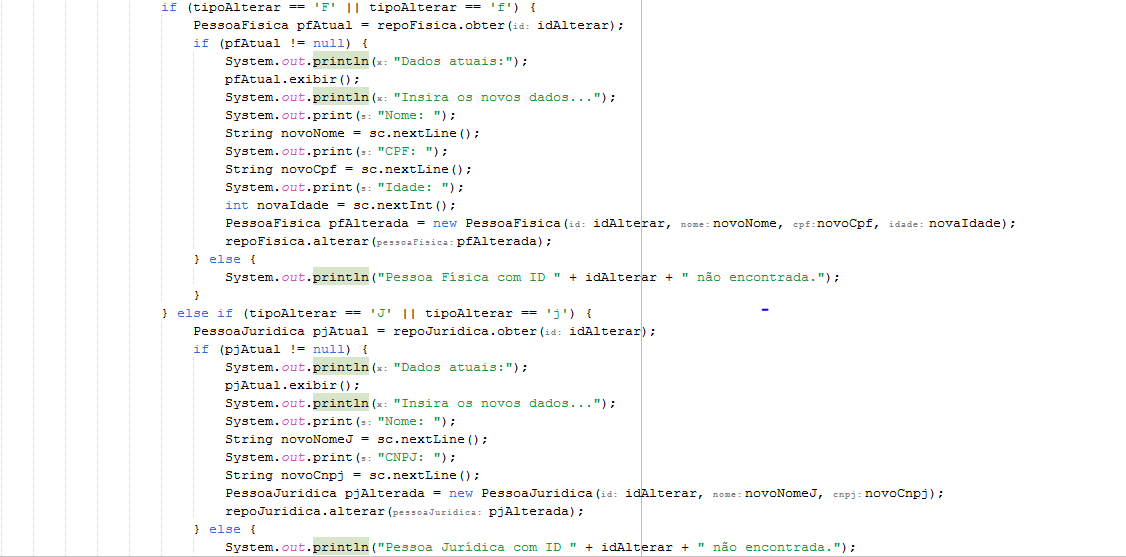
Em Java, um padrão comum para a persistência de dados em arquivos é o pardrão DAO (Data Access Object). O padrão DAO é usado para separar a lógica de acesso aos dados, permitindo que o código seja mais modular e facilmente testado.

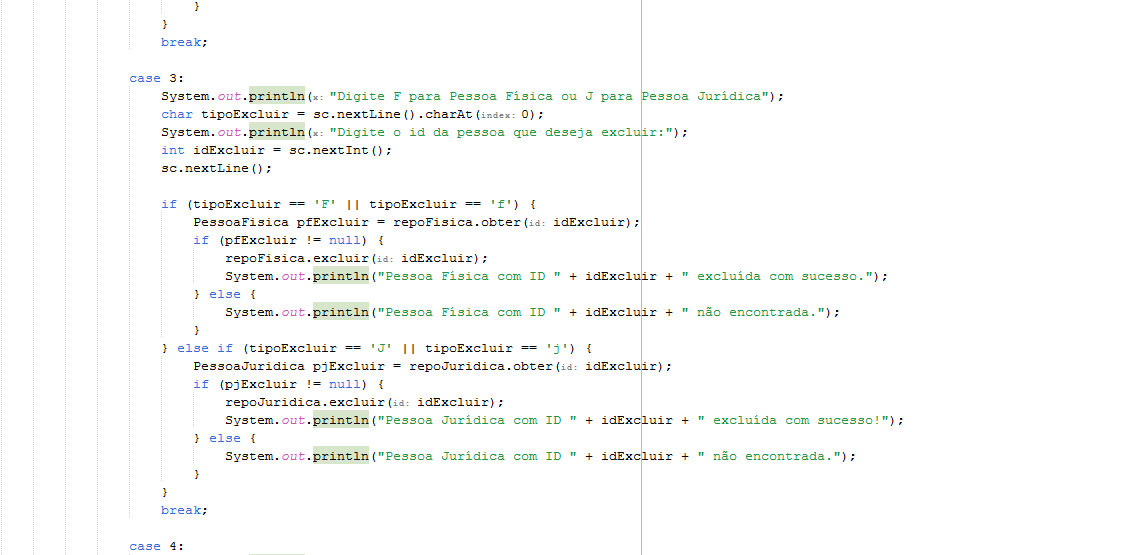
**Procedimento 2 – Criação de Cadastro em Modo Texto**

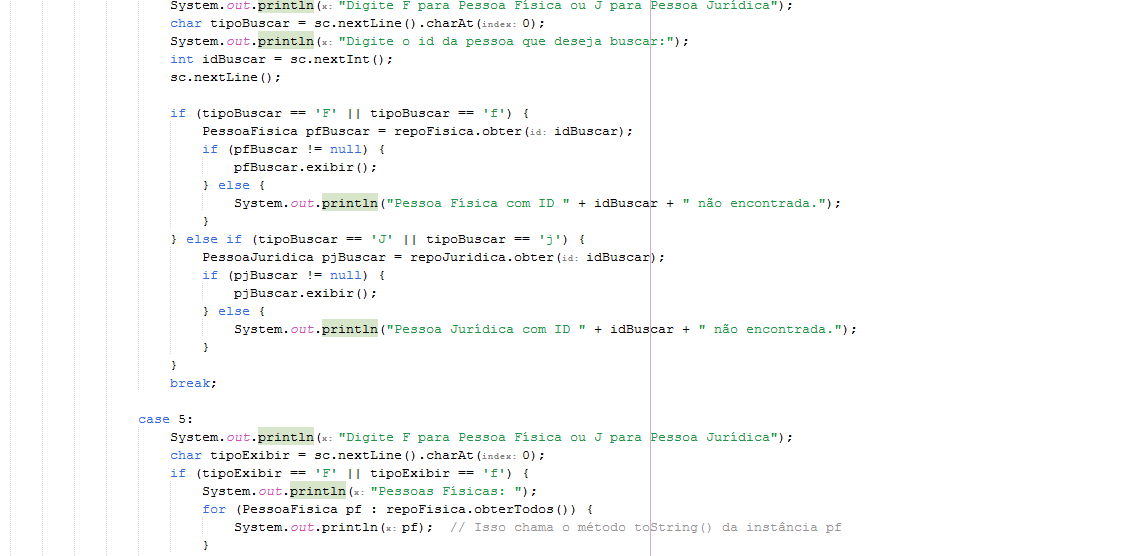
**Códigos do Procedimento 2**

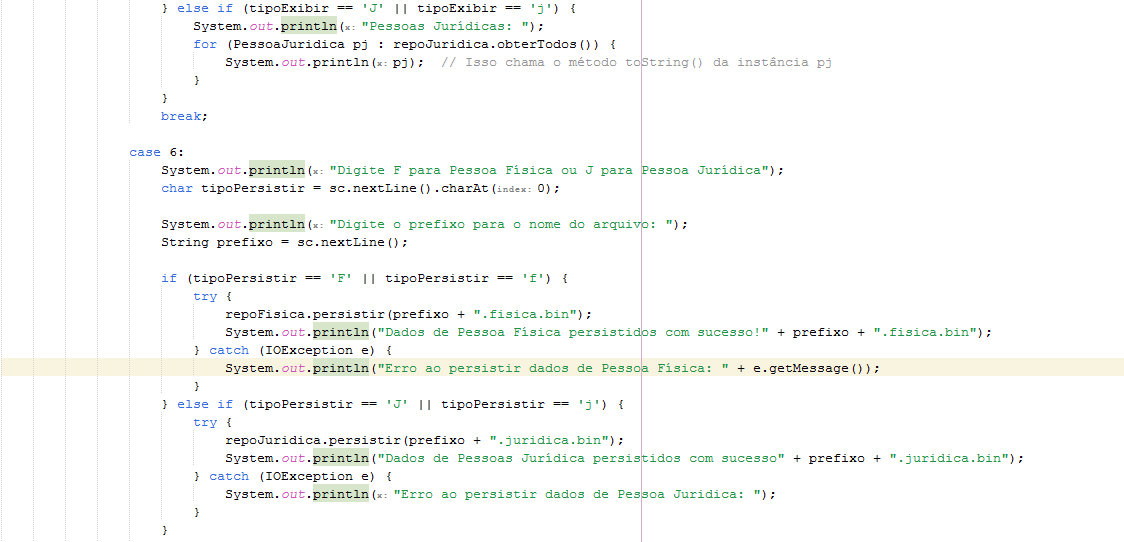


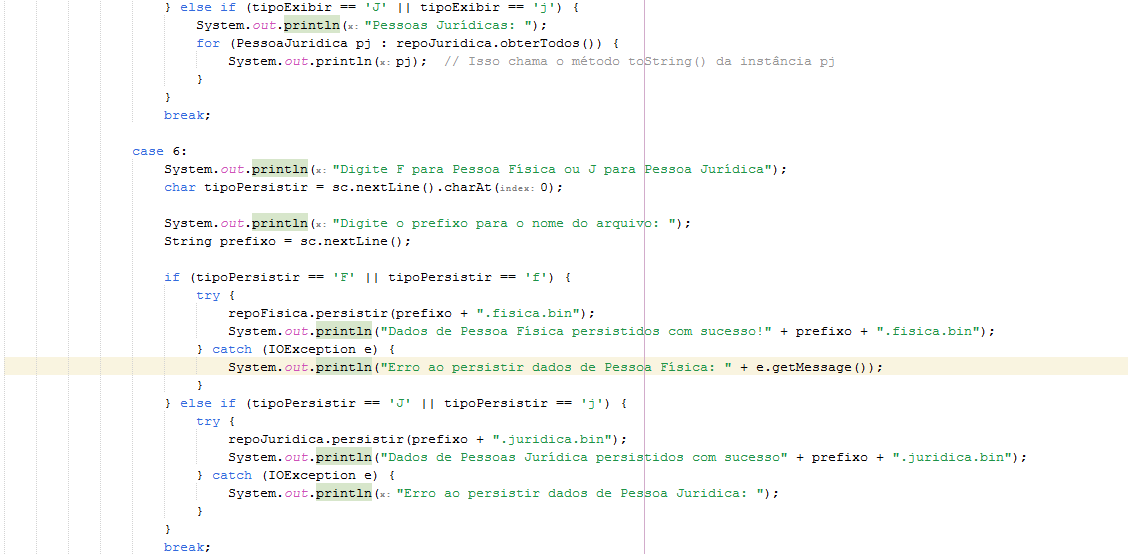


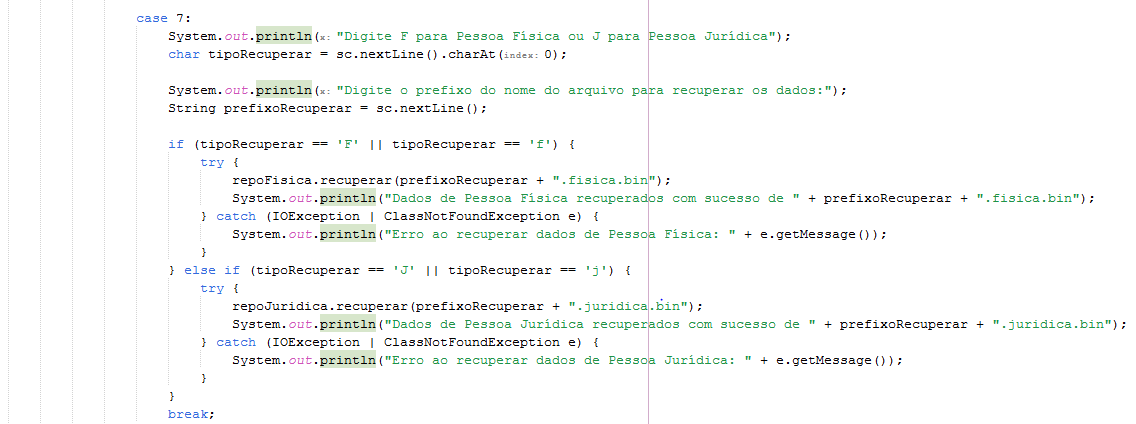


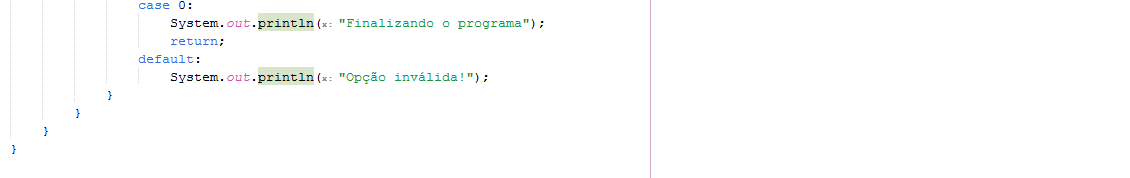




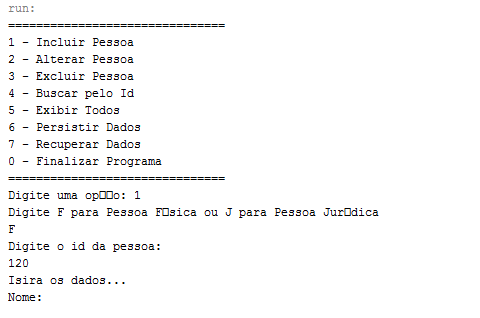








**Resultado do Procedimento 2**



**Análise e Conclusão – Procedimento 2**

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

Os elementos estáticos são recursos que podem ser compartilhados entre todas as instâncias de uma classe, sendo acessíveis diretamente através do nome da classe.

**Para que serve a classe Scanner?**

Ela é útil em situações onde precisamos obter dados de entrada do usuário ou de outros tipos de Input Streams. Ela simplifica e agiliza o processo de leitura e manipulação desses dados.

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

O uso de classes de repositório permitiu separar a lógica de acesso aos dados restantes da aplicação. Com elas, todo o acesso aos dados é centralizado em uma única classe, tornando o código mais organizado e fácil de dar manutenção.

Em resumo, o uso de classes de repositório melhorou a organização do código, tornando-o mais fácil de entender, modificar e reutilizar.