**IFBA – Instituto Federal de Educação, Ciência e Tecnologia da Bahia**

**Departamento de Ciência da Computação**

**Graduação Tecnológica em Análise e Desenvolvimento de Sistemas**

INF011 – Padrões de Projeto

**Prof.:** Frederico Barboza **Data:** 30/07/2024

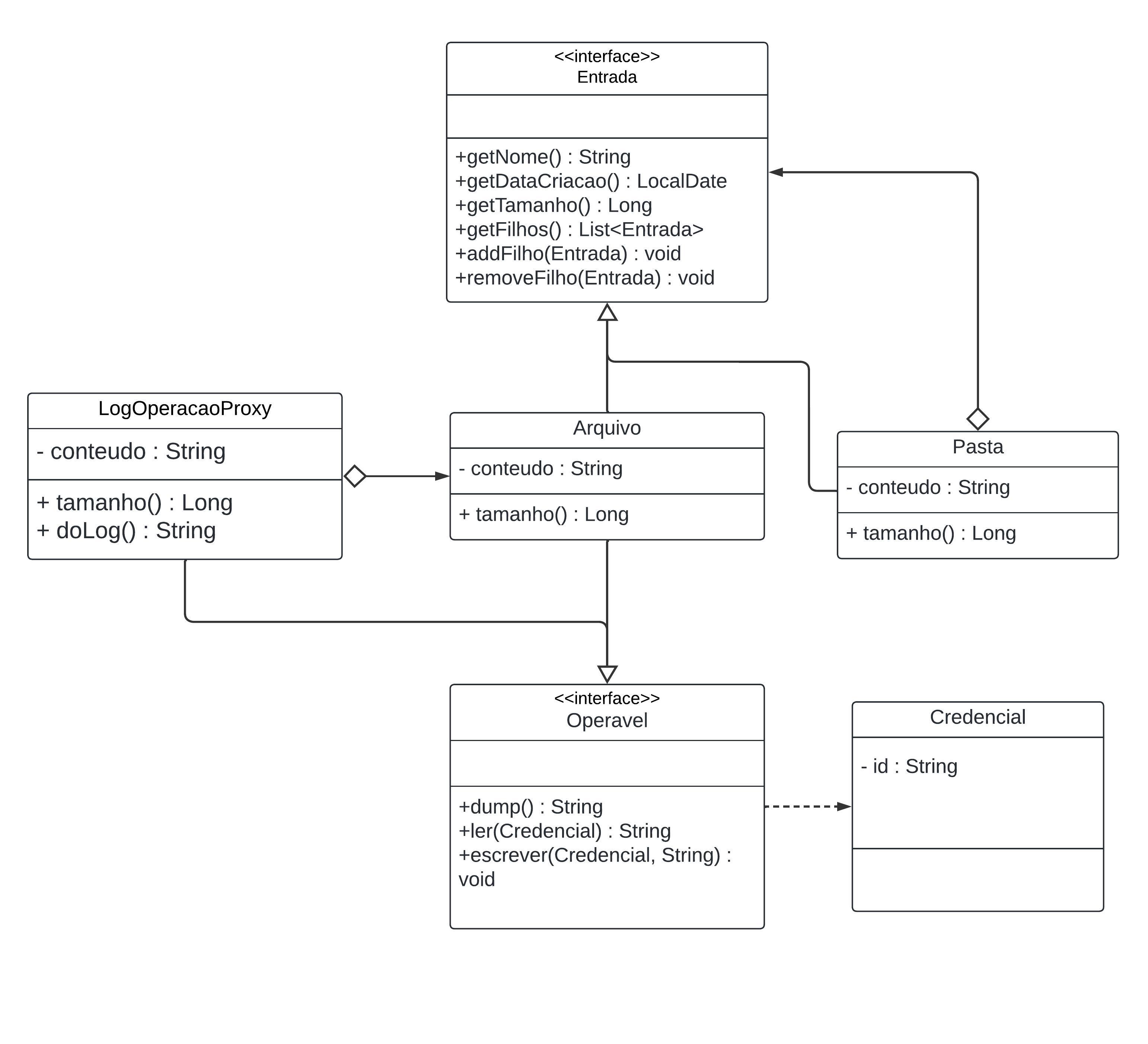
Aluno: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Nota: \_\_\_\_\_\_\_\_\_\_\_

**IIIa Avaliação – Implementação – 2024.1**

*Você está trabalhando no desenvolvimento de um sistema de arquivos para um novo sistema operacional. O sistema precisa ser capaz de organizar arquivos e pastas de forma hierárquica, de forma similar aos sistemas de arquivos tradicionais, onde pastas podem armazenar arquivos, além de outras pastas. A forma de tratar as pastas e arquivos deve ser a mais transparente possível, para a classe cliente (Sistema Operacional).*

*Além disso, o projeto deve contemplar mecanismos de log para garantir o registro de acesso (leitura, escrita e dump) de alguns arquivos (protegidos). Estes registro de acesso deve registrar a credencial que realizou o acessou para leitura de cada arquivo protegido, bem como o número de leituras de cada arquivo protegido por cada credencial. O número de credenciais que podem acessar cada arquivo é arbitrário.*

*De modo a garantir os requisitos apresentados, a equipe de projeto optou pelo uso de Composite e Proxy, o que resultou no seguinte diagrama de classes*

****

*Nessa solução os participantes do Composite são: a interface Entrada (Component), a classe Pasta (Composite) e a classe Arquivo (Leaf); enquanto os participantes do Proxy são: a interface Operavel (ServiceInterface), a classe LogOperacaoProxy (Proxy) e a classe Arquivo (Service).*

*Sua tarefa agora é evoluir o Sistema de Arquivo em questão, mantendo as funcionalidades já implementadas, mas acrescentando os seguintes requisitos:*

O Sistema deve implementar três operações com seus arquivos:

1. **public** String ler(Credencial credencial) **throws** IllegalAccessException; - Esta operação deve recuperar um conteudo previamente armazenado em um arquivo.

2. **public** void escrever(Credencial credencial, String conteudo) **throws** IllegalAccessException; - Esta operação deve permitir armazenar o parâmetro conteudo no arquivo.

3. public **String** dump() throws IllegalAccessException; - Esta é uma operação restrita do sistema operacional, que retorna o conteudo do arquivo da forma com ele está codificado internamente. Caso o arquivo seja binário o retorno é uma sequência de 0’s e 1’s; caso seja texto, o retorno é idêntico ao armazenado.

A forma como o conteúdo dos arquivos é mantida internamente depende do seu tipo. Atualmente implementaremos codificações binárias e textos. Neste cenário, cada caracter é representado por si próprio em caso de codificação texto, ou por uma sequencia de oito bits, de sua codificação ASCII, em caso de codificação binária (A classe **Conversor2Bin** fornece métodos de transformação nos dois sentidos - de caracter para string binária e vice versa). Por exemplo o arquivo com a string CINCO, pode ser representado como

CINCO (no caso de tipo de codificação texto) ou por

0100001101001001010011100100001101001111 (no caso de tipo de codificação binário)

Futuramente a codificação poderá vir a ser realizada em outros formatos, como Hexa, Octal ou Cripto, por exemplo.

O sistema de arquivos deve fornecer um tipo especial de arquivo (ArquivoHistorico), que deve possibilitar ao sistema operacional a manutenção de uma lista de desfazer. Desta forma, este tipo de arquivo deve possuir em sua interface, operações de checkpoint e restore. É importante, que o encapsulamento seja preservado, e que os tipos de codificação de arquivo sejam aplicáveis também ao ArquivoHistorico. Projete o seu sistema evitando a explosão de classes resultante desta combinação entre tipos de arquivo e tipos de codificação.

O ciclo de vida de um arquivo é mostrado no diagrama abaixo.

As operações de ler/escrever/tamanho tem comportamento diferente nestes estados.

Normal – ler/escrever são permitidas, tamanho retorna o número de caracteres do conteúdo.

SomenteLeitura – ler é permitido e tamanho retorna o número de caracteres do conteúdo. Mas, escrever resulta em IllegalAccessException.

Bloqueado – tanto ler, quanto escrever resultam em IllegalAccessException. Como o arquivo está disponível, tamanho retorna o número de caracteres do conteúdo.

Excluído – tanto ler, quanto escrever resultam em IllegalAccessException. Como o arquivo está marcado para exclusão, tamanho retorna o 0, independente do tamanho do conteúdo.

