RELATÓRIO: HIERARQUIA DE CLASSES

Curso: Tópicos em Ciência de Dados A (MAE013)

Professor: Milton R Ramirez

Integrantes

Leonardo Domingos (DRE: 120168324);

Calculadora de Matrizes

1. Introdução

Nesse projeto, iremos elaborar uma calculadora de matrizes, utilizando conceitos de **OOP** (*Object Oriented Programming*) para estruturar o arquivo.

A calculadora terá as seguintes funções:

1. **Imprimir matrizes**
2. **Inserir nova matriz (teclado)**
3. **Inserir matriz identidade**
4. **Inserir matriz de arquivo**
5. **Alterar/remover matriz**
6. **Listar todas as matrizes**
7. **Realizar operações entre matrizes**
    a. **Soma, multiplicação entre matrizes e escalar, transposição, calcular traço e determinante.**
8. **Salvar lista de matrizes**
9. **Carregar lista de matrizes**
10. **Zerar lista de matrizes**

O intuito é que o programa tenha formas de aceitar matrizes como input (teclado, arquivo), e ele irá armazená-las em sua memória *runtime* em uma lista. Para interagir com essas matrizes, o usuário irá sempre selecionar dentre a(s) matriz(es) que já estão armazenadas. O resultado de operações poderá também ser armazenado nessa lista de matrizes. Por fim, também será possível guardar a lista de matrizes em um arquivo, que poderá ser importado pela calculadora.

2. Metodologia

O projeto foi desenvolvido apenas com Python e as seguintes bibliotecas padrão:

- **typing**
    - importamos *List* e *Union,* para usar o tipo “lista” no Python.
- **abc**
    - importamos *abstractmethod*, para usarmos em classes abstratas.

Como ambas bibliotecas são nativas do Python, não é necessário usar nenhum ambiente virtual para rodar o programa.

O desenvolvimento do código em si foi feito com base no paradigma de programação orientada a objetos. A estrutura principal do código se organiza em torno de uma classe base chamada **Matriz**, que define as operações fundamentais que todas as matrizes devem

possuir. A partir dela, outras classes especializadas foram criadas, como **MatrizRegular**, **MatrizQuadrada**, **MatrizTriangularSuperior**, **MatrizTriangularInferior** e **MatrizDiagonal**.

Cada classe implementa seus próprios métodos para lidar com operações específicas. O programa detecta automaticamente o tipo da matriz inserida e instancia a classe correspondente, o que facilita o gerenciamento das matrizes dentro da calculadora.

3. Implementação da calculadora

A calculadora foi implementada em um único arquivo Python. Os principais componentes da implementação são descritos a seguir.

3.1. Implementação das Classes

A estrutura de classes segue a seguinte hierarquia:

- **Matriz** (classe base abstrata): define a interface comum para todas as operações matriciais.
- **MatrizRegular**: representa matrizes gerais m×n, com todos os elementos armazenados.
- **MatrizQuadrada**: especializa MatrizRegular para matrizes n×n.
- **MatrizTriangular** (abstrata): base para matrizes triangulares:
    - **MatrizTriangularSuperior**
    - **MatrizTriangularInferior**

- **MatrizDiagonal**: subclasse de MatrizTriangular, armazena apenas os elementos da diagonal.

3.2. Interface do Usuário

A interface é feita através de um menu no terminal. O usuário pode adicionar matrizes, visualizar as existentes, realizar operações, salvar ou carregar listas de matrizes de arquivos, entre outras funcionalidades.

A interação ocorre por meio de textos no terminal e entrada de dados pelo teclado. Foram implementadas mensagens de erro para entradas inválidas, como índices fora da faixa ou tipos incompatíveis de matrizes.

4. Comentários sobre o Desenvolvimento

Durante o desenvolvimento, algumas decisões foram tomadas para simplificar a implementação do projeto. Por exemplo, embora o enunciado recomendasse armazenar

apenas os elementos não nulos em matrizes triangulares e diagonais, neste projeto todos os elementos foram armazenados em listas completas.

O código está todo contido em um único arquivo, o que foi suficiente para um projeto deste porte, mas, para futuras implementações, seria ideal melhorar essa organização. Uma estratégia poderia ser separar o projeto em um arquivo por classe, por exemplo.

Devido ao tempo gasto em desenvolver o programa, não foi possível implementar as otimizações propostas para as operações. Assim, em termos de complexidade, as operações sempre seguem a lógica padrão para matrizes. As operações de soma e subtração têm complexidade O(m×n), e a multiplicação matricial segue a lógica tradicional de três laços aninhados. O cálculo do determinante para matrizes maiores foi feito de forma recursiva. Portanto, a implementação aqui apresentada, apesar de funcional, não é a mais otimizada.

5. Comentários sobre o Desenvolvimento

O projeto foi concluído dentro das limitações de tempo e escopo. A calculadora é capaz de realizar todas as operações solicitadas, e trata corretamente os diferentes tipos de matrizes. A interação pelo terminal também ficou satisfatória e intuitiva.

Os principais desafios encontrados foram garantir que as verificações de tipo de matriz estivessem corretas e implementar o menu de forma que fosse usável para o usuário final. Dentre as operações, foi necessário implementar as de exportar/importar matrizes e a de criar matriz identidade o quanto antes, para facilitar futuros testes e o desenvolvimento do resto do programa.

Por fim, o uso de OOP foi essencial para esse projeto. Os diferentes tipos de matrizes e suas operações tornaram o código praticamente ilegível caso o sistema de classes não fosse usado.