Documentação de Implementação
Trabalho Intermediário - Programação Orientada a Objetos
Aluno: Lucca Garcia Leão - 2016020967

Introdução

Neste trabalho prático, foi feita a implementação em Java de uma classe de Matriz. Essa classe possui métodos que permitem a manipulação e operação entre matrizes de double, similares à forma feita pelo Matlab. Neste relatório, serão apresentados a classe, seus atributos, os métodos implementados, e as decisões de implementação tomadas.

Classe: Atributos e métodos

A classe Matriz possui 3 atributos, detalhados a seguir:

- private final int ***i***: representa o número de linhas da matriz, para manter a integridade dos dados na memória, este atributo foi definido como final, para que não possam haver mudanças nas dimensões da matriz após ela ser iniciada.
- private final int ***j***: representa o número de colunas da matriz, assim como o número de linhas, foi definido como final.
- private double[][] *elementos*: vetor bidimensional de doubles que armazena os elementos da matriz.

A classe Matriz conta com vários métodos que permitem fazer operações entre matrizes com facilidade. Os métodos são listados e explicados a seguir:

- Métodos construtores:
    - public Matriz(int linhas, int colunas): O primeiro método construtor inicializa os campos *i* e *j* com os valores de linhas e colunas passados para o método. Além disso, inicializa *elementos,* mas não atribui valores.
    - public Matriz(Matriz A): O segundo método construtor, inicializa um novo objeto Matriz, com os mesmos atributos da matriz recebida (A), com os mesmos valores de *elementos*.
- public String toString(): este método sobrescreve o método toString() que existe por padrão em todas as classes Java. Retorna uma String composta por todos

os elementos da matriz, organizados na forma matricial padrão. Com este método, é possível imprimir um objeto Matriz simplesmente utilizando System.out.println(A), sendo A uma Matriz.

- public Matriz mult(Matriz B): retorna uma matriz que é o resultado da multiplicação da matriz corrente por B. Se o número de colunas da matriz corrente for diferente do número de linhas de B, o método lança uma exceção em tempo de execução.
- public Matriz zeros(): inicializa todos os elementos da matriz corrente como zeros, retornando o ponteiro this.
- public Matriz ones(): inicializa todos os elementos da matriz corrente como 1s, retornando o ponteiro this.
- public Matriz unit(): inicializa a matriz corrente como uma matriz identidade, retornando o ponteiro this.
- public Matriz transposta(): retorna um objeto Matriz que é a transposta da Matriz que invocou este método.
- public Matriz soma(Matriz B): retorna um objeto Matriz que é igual a soma da matriz corrente e B. Lança uma exceção em tempo de execução caso as dimensões das matrizes não sejam iguais.
- public Matriz menos(Matriz B): retorna um objeto Matriz que é igual a subtração da matriz por B. Lança uma exceção em tempo de execução caso as dimensões das matrizes não sejam iguais.
- public Matriz vezesConst(double k): este método retorna um objeto Matriz que é igual a matriz corrente multiplicada por uma constante k.
- public Matriz alteraValor(int linha, int coluna, double valor): altera o valor do elemento na posição linha x coluna da Matriz corrente, e retorna o ponteiro this.
- public int getRows(): retorna o número de linhas (i) da matriz corrente.
- public int getCols(): retorna o número de linhas (j) da matriz corrente.

Uma importante decisão de implementação é retornar uma matriz nova nos métodos soma(), mult(), menos(), transposta(), e vezesConst(). Isso permite que sejam feitas atribuições, como feito a seguir, mas sem alterar as matrizes que invocaram tais métodos:

```
C = A.mult(B)
D = A.transposta()
E = B.vezesConst(4.7)
X = A.soma(B)
```