Programação Orientada a Objetos
Trabalho Intermediário 1

André Lage e Augusto Mafra

[1]Universidade Federal de Minas Gerais

1. Introdução

Este trabalho apresenta uma implementação de um tipo abstrato de dados para manipulação de Matrizes na linguagem Java. A manipulação das matrizes é feita por meio do encapsulamento de arranjos bidimensionais em objetos do tipo Matriz que suportam operações aritméticas comumente aplicadas a esse tipo de dado.

O conjunto de operações sobre o tipo Matriz inclui a adição, subtração, a multiplicação por escalar e a multiplicação de matrizes. A interface para os objetos fornece ainda funções para inicializar a matriz com 0, 1 ou com a matriz identidade e um método para imprimir a matriz no *stdout*.

2. Implementação

A interface para os objetos do tipo Matriz é definida a partir das funções descritas a seguir:

```
1  public class Matriz {
2      public Matriz(int nrow, int ncol) throws
           IndexOutOfBoundsException;
3      public int getRows();
4      public int getCols();
5      public void zeros();
6      public void ones();
7      public void unit() throws IndexOutOfBoundsException;
8      public Matriz mult(double valor);
9      public Matriz mult(Matriz A) throws IndexOutOfBoundsException
           ;
10     public Matriz add(Matriz A) throws IndexOutOfBoundsException;
11     public Matriz sub(Matriz A) throws IndexOutOfBoundsException;
12     public Matriz transp();
13     public double at();
14     public void set(int i, int j, double value);
15     public void print();
16     private void expand(int new_nrow, int new_ncol) throw
           IndexOutOfBoundsException;
17     private double[][] m;
18     private int ncol, nrow;
19 }
```

2.1. Construtor

A classe Matriz não suporta um construtor padrão. Toda inicialização deve especificar um par de valores *nrow* e *ncol* para inicializar a matriz. O objeto criado representa uma matriz com *nrow* linhas e *ncol* colunas.

Caso seja passado como parâmetro ao construtor valores menores ou iguais a zero, a execução do programa é interrompida com a exceção *IndexOutOfBoundsException*.

2.2. *Getters*

Os objetos Matriz suportam duas funções *getters* para acessar seus atributos privados. A função *getRows()* retorna o número de linhas da matriz e a função *getCols()* retorna seu número de colunas.

2.3. Funções de inicialização

A interface provê três funções para inicialização da Matriz. As duas primeiras são as *zeros* e *ones*, que inicializam todos os elementos da matriz com 0 e 1, respectivamente. A última função de inicialização é a *unit*, que inicializa a matriz com a matriz identidade.

A função *unit* funciona somente se a matriz inicializada for quadrada. Caso contrário, a execução é interrompida com a exceção *IndexOutOfBoundsException*.

2.4. Funções *mult*

A classe Matriz suporta um método para multiplicação que é sobrecarregado para multiplicação por valores escalares e para multiplicação por outra matriz. Em toda chamada de *mult*, é alocado um novo objeto Matriz para ser retornado com o resultado da multiplicação.

Para o método de multiplicação por matriz, *mult(Matriz A)*, a execução do programa é interrompida com a exceção *IndexOutOfBoundsException* caso as dimensões das matrizes sejam incompatíveis para multiplicação.

2.5. Funções *add* e *sub*

Os métodos add e sub realizam respectivamente as adições e subtrações entre matrizes, de forma que utilizamos o método at para obter os valores de cada linha e coluna da matriz passada por parâmetro.

Essas operações, contudo, serão realizadas apenas se as duas matrizes possuírem as mesmas dimensões, caso contrário, a execução será interrompida com a exceção *IndexOutOfBoundsException*.

2.6. Função *transp*

Para situações em que se deseja obter a transposta do objeto do tipo matriz, faz-se uso do método *transp*.

Esse método retorna a matriz transposta e realiza a alteração do *ncol* e *nrow* da matriz.

2.7. Funções de acesso: *at* e *set*

O acesso e a modificação dos elementos de um objeto da classe Matriz são feitos pelos métodos *at* e *set*. Diferentemente do padrão nas linguagens C e Java, nas quais matrizes são indexadas a partir do índice 0, na implementação da classe toda matriz é indexada a partir do índice 1. Dessa maneira, *at(i, j)* retorna um valor *double* igual ao elemento na posição i,j da matriz e *set(i, j, x)* atribui x ao elemento na posição i,j na matriz.

É importante notar que, de modo a reproduzir a implementação da manipulação de matrizes em *Matlab*, a função *set* não acusa erro caso a atribuição seja feita a um par de índices inexistente. Nessa situação, a matriz é expandida por meio do método privado *expand*, e a atribuição é feita normalmente no novo espaço alocado.

2.8. Função *print*

A função *print* provê um método padronizado para imprimir o conteúdo do objeto Matriz no *stdout*. Todos os elementos da matriz são impressos no formato de ponto flutuante separados por *tab*. Além disso, cada linha da matriz impressa é mostrada entre colchetes.

2.9. Função privada *expand*

O método *expand* é uma função privada utilizada internamente pela função *set* quando é feita uma atribuição a um par de índices inexistente. Nesse caso, essa função é usada para alocar uma nova matriz cujo tamanho comporte a nova atribuição realizada e copiar os elementos do objeto corrente para a nova matriz alocada.

Esse método deve ser utilizado sempre para aumentar o tamanho da matriz, nunca para diminuir. A chamada de *expand* com *newnrow* ou *newncol* menores que os valores originais causa a interrupção do programa com uma exceção *IndexOutOfBoundsException*.