



Técnicas de Programação

Aula 12 – Vetores e matrizes



Objetivos de Aprendizagem

- Descrever estruturas de dados do tipo vetor e matriz.
- Desenvolver programas usando vetor e matriz.



Lendo dados de entrada

- Há três diferentes maneiras de ler dados de entrada vindos do Console Java:
 - Usando a classe Java BufferedReader
 - Classe Scanner em Java
 - Classe Console em Java



Java BufferedReader Class

- Essa é a técnica tradicional Java, introduzida no JDK1.0. Essa estratégia é utilizada ao agrupar o `System.in` (fluxo de informações padrão) em um `InputStreamReader` que é agrupado em um `Java BufferedReader`, podemos ler a inclusão do resultado do usuário na linha de pedido.
- Prós
 - A informação é embalada para leitura produtiva.
- Contras
 - O código de quebra é difícil de lembrar.



Java BufferedReader Class

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
public class Test
{
    public static void main(String[] args) throws IOException
    {
        BufferedReader reader =
        InputStreamReader(System.in).new BufferedReader(new
        String name = reader.readLine();
        System.out.println(name);
    }
}
```



Scanner Class

- Essa é presumivelmente a técnica mais favorecida para receber sugestões.
- O principal motivo da classe Scanner é analisar composições e sequências primitivas utilizando expressões gerais; em qualquer caso, ela pode ser utilizada para examinar a contribuição do cliente na linha de pedido.
- Prós:
 - Estratégias úteis para analisar nativos (`nextInt()`, `nextFloat()`,...) da entrada tokenizada.
 - Articulações gerais podem ser utilizadas para descobrir tokens.
- Contras:
 - Os métodos de leitura não são sincronizados.



Scanner Class

```
import java.util.Scanner;
class GetInputFromUser
{
    public static void main(String args[])
    {
        Scanner in = new Scanner(System.in);
        String s = in.nextLine();
        System.out.println("You entered string "+s);
        int a = in.nextInt();
        System.out.println("You entered integer "+a);
        float b = in.nextFloat();
        System.out.println("You entered float "+b);
    }
}
```



Console Class

- Ele está se transformando em uma rota preferida para examinar a contribuição do cliente na linha de comando.
- Além disso, ele pode utilizar para chave de senha como contribuição, sem ressonar os caracteres digitados pelo cliente, a estrutura da sintaxe da cadeia de configuração também pode utilizar (como `System.out.printf()`).
- Prós:
 - Ler palavra secreta sem reverberar os caracteres digitados.
 - Estratégias de leitura sincronizadas.
 - A estrutura da sentença de cadeia de formato pode ser utilizada.
- Contras:
 - Não funciona em condições não inteligentes (por exemplo, em um IDE).



Console Class

```
public class Sample
{
    public static void main(String[] args)
    {
        // Using Console to input data from user
        String name = System.console().readLine();
        System.out.println(name);
    }
}
```



Vetores

- Em Java podemos criar vetores tanto de tipos primitivos quanto de objetos.
- Os vetores em Java são objetos, isto faz com que tenhamos que criar uma instância para permitir a sua utilização.
- Para declarar um vetor, colocamos os símbolos de colchetes [e] após a declaração do tipo da variável ou da própria variável.



Vetores

```
// declaração de vetores
int[] x, y;
int z[];
// declaração com atribuição
int k[] = { 1, 2, 3 };
// criação de 10 posições
x = new int[10];
// atribuição dos elementos
for (int i = 0; i < 10; i++) {
    x[i] = i;
}
```

Vetores

- É possível criar vetores com duas dimensões (matriz), sejam eles retangulares ou não.
- O código abaixo mostra a criação de um vetor (não retangular) onde a primeira linha tem três colunas e a segunda linha tem quatro.

```
int x[][];  
x = new int[2][];  
x[0] = new int[3];  
x[1] = new int[4];
```





Vetores

- Existe uma outra maneira de declarar vetores (retangulares) com duas dimensões:

```
int x[] [] = new int[4][5];
```

- Na declaração anterior foi criado um vetor com 4 linhas e 5 colunas. Usando essa declaração não é necessário utilizar `new` para cada uma das linhas.

Vetores

```
public static void main(String[] args) {  
    int x[][] = new int[4][5];  
    for (int i = 0; i <= 3; i++) {  
        for (int j = 0; j <= 4; j++) {  
            x[i][j] = i * j;  
        }  
    }  
    for (int i = 0; i <= 3; i++) {  
        for (int j = 0; j <= 4; j++) {  
            System.out.print(x[i][j]+" ");  
        }  
        System.out.println(" ");  
    }  
}
```



Vetores

- O primeiro elemento do vetor é indexado por zero e o seu tamanho é dado pelo atributo `length`.
- Se for acessado um índice fora da faixa será gerada uma exceção.

```
int[] x = {1,2,3};  
for (int i = 0; i < x.length; i++) {  
    System.out.println(x[i]);  
}
```



Vetores

- Depois de criado o vetor não pode ser redimensionado.
- O que podemos fazer é copiar os elementos de um vetor para outro.

```
int[] x = {1,2,3};  
int[] y = new int[5];  
System.arraycopy(x, 0, y, 0, 3);  
for (int i = 0; i < y.length; i++) {  
    System.out.println(y[i]);  
} // será impresso 1,2,3,0 e 0
```




Exercício

1. Construa um programa que preenche um vetor de inteiros de 100 números, colocando 0 nas posições par e 1 ímpar.
2. Construa um programa que lê, soma e imprime o resultado da soma de um vetor de inteiros de 10 posições.
3. Construa um programa que multiplique os valores de um vetor de reais de 20 posições pelo valores de um outro vetor de reais de 20 posições. Os resultados das multiplicações devem ser armazenados num terceiro vetor.

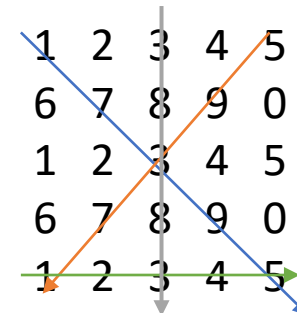


Exercício Desafio

- Construa um programa que leia e guarde os elementos em um vetor de 20 posições.
- Em seguida o algoritmo ordena os elementos do vetor de acordo com a seguinte estratégia:
 - Selecione o elemento do vetor que guarda o menor e o maior valor

Exercício

1. Construa um algoritmo que efetue a soma e a impressão do resultado da soma entre duas matrizes 3×5 .
2. Faça um programa que multiplica uma matriz 3×3 de inteiros por um escalar $k = 5$.
3. Dada uma matriz 5×5 , elabore um algoritmo que escreva:
 - a) A diagonal principal ($i == j$)
 - b) A diagonal secundária ($i + j == 4$)
 - c) A soma da linha 4 ($i == 4$)
 - d) A soma da coluna 2 ($j == 2$)



The diagram shows a 5x5 matrix with the following values:

1	2	3	4	5
6	7	8	9	0
1	2	3	4	5
6	7	8	9	0
1	2	3	4	5

Highlighted paths:

- Blue diagonal (Principal):** Cells (1,1), (2,2), (3,3), (4,4), (5,5).
- Orange diagonal (Secondary):** Cells (1,5), (2,4), (3,3), (4,2), (5,1).
- Green horizontal line (Row 4):** Cells (4,1), (4,2), (4,3), (4,4), (4,5).
- Grey vertical line (Column 2):** Cells (1,2), (2,2), (3,2), (4,2), (5,2).