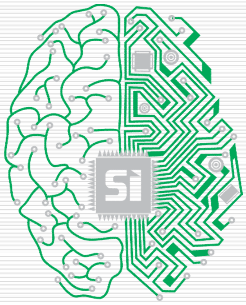
Vetores

Uso de variáveis Indexadas (Vetor)



1

Vetores em Java

Vetores em Java não são tipos básicos e sim, OBJETOS.

Tem a manipulação parecida com a de outras linguagens, salvo conceitos específicos do próprio Java.

2

Vetores em Java

•Uma das características do Java é permitir que variáveis sejam declaradas em qualquer ponto ou escopo.
•Com isso, o Java permite que sejam criados vetores com tamanhos que podem ser pré-determinados pelo próprio usuário.
•Diferente, por exemplo, do Pascal que exige pré-definição

3

Vetores em Java

Criação de um Vetor:

```
<tipo>[] <variavel> = new <tipo>[<n° de posições>];
```

Onde:
<tipo> - qualquer tipo válido (int, double, char,...)
<variavel> - nome da variável vetor
<n° posições> - quantidade de elementos que o vetor conterá.

4

Vetores em Java

Criação de um Vetor:

```
int[] vetor = new int[10];
```

Cria um vetor com 10 índices inteiros.

Lembrete:
Vetores em Java, assim como os Componentes de Delphi, iniciam pelo índice “0”, ou seja, no exemplo acima teremos um vetor que vai do índice “0” até o “9”.

5

Vetores em Java

As mesmas aplicações feitas em qualquer outra linguagem que suporte a estrutura de vetor podem ser realizadas de maneira praticamente equivalente em Java.

6

Vetores em Java

Exemplo:

```
for (int i=0;i<vetor.length;i++){
     System.out.println(i + "=" +
vetor[i]);
}
```

Onde:

vetor.length – retorna o número de índices do vetor. OBS: *length* é uma propriedade e não tem parênteses

7

Vetores em Sub-Rotinas

O fato do Java tratar Vetores como objetos, implica que, com relação a passagem de parâmetros, vetores tem passagem por referência, ou seja, se os valores sofrerem alteração dentro do método, esta alteração se refletirá fora do método.

8

Vetores em Sub-Rotinas

Recebendo um Vetor por parâmetro:

```
public static void <nome>(tipo[]
parametro)
```

Uma boa referência para lembrar é:

```
public static void main(String[]
args)
```

9

Vetores em Sub-Rotinas

Passando um Vetor por parâmetro:

Ler (vetor);

Onde:
Ler – nome do método
vetor – nome da variável Vetor

10

Vetores em Sub-Rotinas

Devolvendo um Vetor com `return` -
O Java permite que vetores sejam retornados por um método. Para tanto o tipo de retorno do método deve seguir o padrão de vetores.

```
public static int[] Ler(){...}
```

E em algum ponto do código deve ser executado um comando `return` com uma variável que seja do tipo vetor de inteiros

11

Visualização de Vetores

Quando se trabalha com vetores, normalmente a quantidade de valores que serão mostrados aumenta em relação a programas que não usam esta estrutura.

Com isso, o comando `showMessageDialog` pode não ser o mais indicado para esta tarefa.

12

Visualização de Vetores

Uma alternativa interessante, mas não necessariamente obrigatória, é utilizarmos um objeto que permita a visualização de várias linhas, sem que para isso seja necessário criarmos variáveis temporárias.

Em `Delphi`, existe um componente similar, chamado `Memo`.

13

Visualização de Vetores

Um componente similar ao `Memo` existe no Java e se chama `JTextArea`.

Para que seja usado é necessário, primeiramente, um `import` adicional:

```
import javax.swing.JTextArea;
```

14

Visualização de Vetores

Para utilizarmos dentro de nosso programa, temos que declarar uma variável (como em qualquer outro tipo...).

Declaração:

```
JTextArea <variavel>=new JTextArea(<linhas>,<colunas>);
```

Exemplo:

```
JTextArea area=new JTextArea(10,20);
```

15

Visualização de Vetores

- Deve-se ter em mente que JTextArea não é um tipo básico e sim um Objeto.
- Os mesmos conceitos que foram vistos quanto a Vetor ser objeto valem também para o JTextArea.

16

Visualização de Vetores

Utilização:
Assim como em `Delphi`, existem comandos específicos para adicionar valores ao `Memo`, o mesmo procedimento existe no `JTextArea`.

```
variavel.append(<String>);
variavel.setText(<String>);
```

17

Visualização de Vetores

A diferença básica entre os dois comandos é que o `append(<String>);` vai adicionando o conteúdo ao já existente e o `variavel.setText(<String>);` substitui todo o conteúdo anterior por aquele passado no parâmetro.

18

Visualização de Vetores

A visualização do conteúdo do JTextArea pode ser feita em formulários onde são adicionados todos os objetos visuais (tal qual o `Delphi`), ou de forma individual.

Como neste curso não é abordada a parte visual do Java, visualizaremos nossos *JTextArea* de maneira isolada.

19

Visualização de Vetores

Esta visualização ocorre de maneira similar a qualquer outra saída de tela que tenhamos feito até agora.

```
JOptionPane.showMessageDialog(null,area);
```

20