

Linguagens de Programação

Java Orientado a Objetos

Prof. Waldeck Lindoso Jr.

Arrays (vetores)

- Na declaração de variáveis, freqüentemente utilizamos um identificador ou um nome e um tipo de dados.
- Para se utilizar uma variável, deve-se chamá-la pelo nome que a identifica.
- Por exemplo, temos três variáveis do tipo int com diferentes identificadores para cada variável

```
int number1;  
int number2;  
int number3;  
number1 = 1;  
number2 = 2;  
number3 = 3;
```



Arrays (vetores)

- Como se vê, inicializar e utilizar variáveis pode torna-se uma tarefa tediosa, especialmente se elas forem utilizadas para o mesmo objetivo.
- Em Java, e em outras linguagens de programação, pode-se utilizar uma variável para armazenar e manipular uma lista de dados com maior eficiência. Este tipo de variável é chamado de array.

	0	1	2
number:	1	2	3

Arrays (vetores)

- Como se vê, inicializar e utilizar variáveis pode torna-se uma tarefa tediosa, especialmente se elas forem utilizadas para o mesmo objetivo.
- Em Java, e em outras linguagens de programação, pode-se utilizar uma variável para armazenar e manipular uma lista de dados com maior eficiência. Este tipo de variável é chamado de array.

	0	1	2
number:	1	2	3

Arrays (vetores)

- Um array armazena múltiplos itens de um mesmo tipo de dado em um bloco contínuo de memória, dividindo-o em certa quantidade de posições.
- Imagine um array como uma variável esticada – que tem um nome que a identifica e que pode conter mais de um valor para esta mesma variável.




Arrays (vetores)

- Array precisa ser declarados como qualquer variável. Ao declarar um array, defina o tipo de dados deste seguido por colchetes [] e pelo nome que o identifica.

Por exemplo: `int[] ages;`

ou colocando os colchetes depois do identificador.

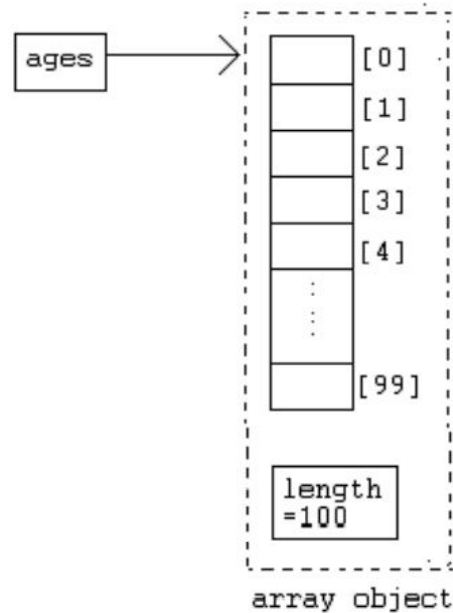
Por exemplo: `int ages[];`



Arrays (vetores)

- Depois da declaração, precisamos criar o array e especificar seu tamanho. Este processo é chamado de construção (a palavra, em orientação a objetos, para a criação de objetos). Para se construir um objeto, precisamos utilizar um construtor.

```
// declaração  
int ages[];  
// construindo  
ages = new int[100];
```



Arrays (vetores)

- Em vez de utilizar uma nova linha de instrução para construir um array, também é possível automaticamente declarar, construir e adicionar um valor uma única vez.

```
/*  
 * Criando um array de valores lógicos em uma variável results.  
 * Este array contém 4 elementos que são inicializados com os  
 * valores {true, false, true, false}  
 */  
boolean results[] = { true, false, true, false };  
  
/*  
 * Criando um array de 4 variáveis double inicializados com os  
 * valores {100, 90, 80, 75};  
 */  
double grades = {100, 90, 80, 75};  
  
/*  
 * Criando um array de Strings com identificador days e também  
 * já inicializado. Este array contém 7 elementos  
 */  
String days[] = {"Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"};
```


Arrays (vetores)

- Uma vez que tenha sido inicializado, o tamanho de um array não pode ser modificado, pois é armazenado em um bloco contínuo de memória. Para acessar um elemento do array, ou parte de um array, utiliza-se um número inteiro chamado de índice.



Arrays (vetores)

- Um índice é atribuído para cada membro de um array, permitindo ao programa e ao programador acessar os valores individualmente quando necessário.
- Os números dos índices são sempre inteiros. Eles começam com zero e progridem seqüencialmente por todas as posições até o fim do array.
- Lembre-se que os elementos dentro do array possuem índice de:
 - 0 “zero” até tamanhoDoArray-1.

Por exemplo, dado o array ages que declaramos anteriormente, temos:

```
int[] ages;  
ages = new int[10];  
// atribuir 10 ao primeiro elemento do array  
ages[0] = 10;  
// imprimir o último elemento do array  
System.out.print(ages[9]);
```

Arrays (vetores)

- Lembre-se que o array, uma vez declarado e construído, terá o valor de cada membro inicializado automaticamente. Conforme a seguinte tabela:

Tipo primitivo	Inicializado com
boolean	false
byte, short e int	0
char	\u0000'
long	0L
float	0.0F
double	0.0

Arrays (vetores)

- Entretanto, tipos de dados por referência, como as Strings, não serão inicializados caracteres em branco ou com uma string vazia " ", serão inicializados com o valor **null**. Deste modo, o ideal é preencher os elementos do arrays de forma explícita antes de utilizar-los.
- A manipulação de objetos nulos pode causar a desagradável surpresa de uma exceção do tipo **NullPointerException**, por exemplo, ao tentar executar algum método da classe String, conforme o exemplo a seguir:

```
9 public class aula01 {  
10     public static void main(String[] args) {  
11         String[] nulls = new String[2];  
12         System.out.print(nulls[0]); // Linha correta, mostra null  
13         System.out.print(nulls[1].trim()); // Causa erro  
14     }  
15 }
```

```
NullPointerException in thread "main" java.lang.NullPointerException  
    at br.com.aula01.main(aula01.java:16)
```

Arrays (vetores)

- O código abaixo utiliza uma declaração for para mostrar todos os elementos de um array.

```
9 public class aula01 {  
10     public static void main(String[] args) {  
11         int[] ages = new int[100];  
12         for (int i = 0; i < 100; i++) {  
13             ages[i] = i;  
14             System.out.println(ages[i]);  
15         }  
16     }  
17 }
```

Tamanho de Array

- Para se obter o número de elementos de um array, pode-se utilizar o atributo `length`. O atributo `length` de um array retorna seu tamanho, ou seja, a quantidade de elementos.
- É utilizado como no código abaixo:
 - `nomeArray.length`

Por exemplo, dado o código anterior, podemos reescrevê-lo como:

```
int[] ages = new int[100];  
for (int i = 0; i < ages.length; i++) {  
    System.out.print(ages[i]);  
}
```

Arrays Multidimensionais

- Arrays multidimensionais são implementados como arrays dentro de arrays. São declarados ao atribuir um novo conjunto de colchetes depois do nome do array. Por exemplo:

```
// array de int de 10 x 8 elementos
int [][] duasDim = new int[10][8];
// array de char de 8 x 16 x 24
char [][][] tresDim = new char[8][16][24];
// array de String de 4 linhas x 2 colunas
String [][] carros = {{"uno", "preto"},
                      {"palio", "vermelho"},
                      {"gol", "azul"},
                      {"corsa", "verde"}}};
```

Arrays Multidimensionais

- Acessar um elemento em um array multidimensional é semelhante a acessar elementos em um array de uma dimensão.
- Por exemplo, para acessar o primeiro elemento da primeira linha do array `carros`, escreve-se:

```
// array de String de 4 linhas x 2 colunas
String [][] carros = {{"uno", "preto"},
                      {"palio", "vermelho"},
                      {"gol", "azul"},
                      {"corsa", "verde"}};
```

```
System.out.print(carros[0][0]);
```



Arrays Multidimensionais

- Isso mostrará a String "uno" na saída padrão. Caso queira mostrar todos os elementos deste array, escreve-se:

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



Arrays Multidimensionais

- Isso mostrará a String "uno" na saída padrão. Caso queira mostrar todos os elementos deste array, escreve-se:

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



Exercícios - Arrays

Dias da semana

- Criar um array de Strings inicializado com os nomes dos sete dias da semana.
 - Por exemplo:

```
String[] diasDaSemana = {"Segunda", "Terça", "Quarta",  
                        "Quinta", "Sexta", "Sábado", "Domingo"};
```

- Usando uma declaração **while**, imprima todo o conteúdo do array. Faça o mesmo para as declarações **do-while** e **for**.



Exercícios - Arrays

Maior número

- Usando a classe Scanner, solicite 10 números ao usuário. Utilize um array para armazenar o valor destes números. Mostre o número de maior valor.



Obrigado!

