

Aula 14: Arrays

Professor(a): Virgínia Fernandes Mota

<http://www.dcc.ufmg.br/~virginiaferm>

ALGORITMOS E ESTRUTURAS DE DADOS - SETOR DE INFORMÁTICA



- Ao término desta aula, você será capaz de:
 - declarar e instanciar arrays;
 - popular e percorrer arrays.

- Dentro de um bloco, podemos declarar diversas variáveis e usá-las:

```
1  int idade1;  
2  int idade2;  
3  int idade3;  
4  int idade4;
```

- Nada bom, certo?
- Para facilitar esse tipo de caso podemos declarar um vetor (array) de inteiros:

```
1  int[] idades;
```

O problema

- O `int[]` é um tipo.
- Um array é sempre um objeto, portanto, a variável `idades` é uma referência.
- Vamos precisar criar um objeto para poder usar o array. Como criamos o objeto-array?

```
1 idades = new int[10];  
2 //para acessar posições  
3 idades[5] = 10;
```

- No Java, os índices do array vão de 0 a $n-1$.
- Se você tentar acessar uma posição fora desse alcance: `Exception ArrayOutOfBoundsException`.
- Importante: Muitas vezes utilizamos outros recursos em vez de arrays, em especial os pacotes de coleções do Java, que veremos mais adiante no curso.

- É comum ouvirmos "array de objetos".
- Correto: O objeto está na memória principal e, no seu array, só ficam guardadas as referências (endereços).

```
1 Conta[] minhasContas;  
2 minhasContas = new Conta[10];
```

- Quantas contas foram criadas aqui?

Arrays de referências

- Por enquanto, eles se referenciam para lugar nenhum (null).
Se você tentar:

```
1 System.out.println(minhasContas[0].saldo);
```

- Erro de execução!!!
- Você deve popular o array antes!

Arrays de referências

```
1  Conta contaNova = new Conta();  
2  contaNova.saldo = 1000.0;  
3  minhasContas[0] = contaNova;  
4  
5  //Ou você pode fazer isso diretamente:  
6  
7  minhasContas[1] = new Conta();  
8  minhasContas[1].saldo = 3200.0;
```

- Um array de tipos primitivos guarda valores, um array de objetos guarda referências.

Percorrendo um array

```
1 public static void main(String args[]) {  
2     int[] idades = new int[10];  
3     for (int i = 0; i < 10; i++) {  
4         idades[i] = i * 10;  
5     }  
6     for (int i = 0; i < 10; i++) {  
7         System.out.println(idades[i]);  
8     }  
9 }
```

- E se o array vier como argumento em um método?

```
1 void imprimeArray(int[] array) {  
2     // não compila!!  
3     for (int i = 0; i < ???; i++) {  
4         System.out.println(array[i]);  
5     }  
6 }
```

Percorrendo um array

- Todo array em Java tem um atributo que se chama **length**!

```
1 void imprimeArray(int[] array) {  
2     for (int i = 0; i < array.length; i++) {  
3         System.out.println(array[i]);  
4     }  
5 }
```

- Observação: A partir do momento que um array foi criado, ele não pode mudar de tamanho.
- Se você precisar de mais espaço, será necessário criar um novo array e, antes de se referir a ele, copie os elementos do array velho.

Percorrendo um array

- No caso de você não ter necessidade de manter uma variável com o índice que indica a posição do elemento no vetor, podemos usar o **enhanced-for**.

```
1  class AlgumaClasse{
2      public static void main(String args[]) {
3          int[] idades = new int[10];
4          for (int i = 0; i < 10; i++) {
5              idades[i] = i * 10;
6          }
7
8          // imprimindo todo o array
9          for (int x : idades) {
10             System.out.println(x);
11         }
12     }
13 }
```

Percorrendo um array

- Não precisamos mais do `length` para percorrer matrizes cujo tamanho não conhecemos:

```
1 class AlgumaClasse {  
2     void imprimeArray(int[] array) {  
3         for (int x : array) {  
4             System.out.println(x);  
5         }  
6     }  
7 }
```

- 1. Volte ao nosso sistema de Funcionario e crie uma classe Empresa dentro do mesmo arquivo .java. A Empresa tem um nome, cnpj e uma referência a um array de Funcionario, além de outros atributos que você julgar necessário.

```
1  class Empresa {  
2      // outros atributos  
3      Funcionario[] empregados;  
4      String cnpj;  
5  }
```

- 2. A Empresa deve ter um método adiciona, que recebe uma referência a Funcionario como argumento e guarda esse funcionário. Algo como:

```
1  void adiciona(Funcionario f) {  
2      // algo tipo:  
3      //   this.empregados[ ??? ] = f;  
4      // mas que posição colocar?  
5  }
```

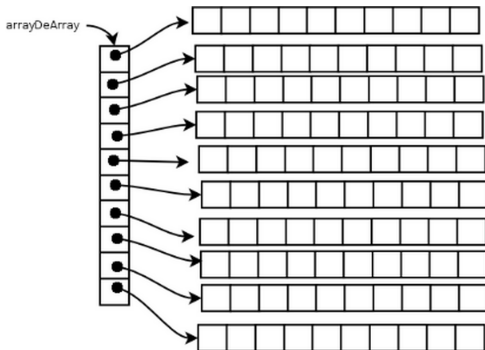
- 3. Crie uma classe TestaEmpresa que possuirá um método main. Dentro dele crie algumas instâncias de Funcionario e passe para a empresa pelo método adiciona.
- 4. Percorra o atributo empregados da sua instância da Empresa e imprima os salários de todos seus funcionários. Para fazer isso, você pode criar um método chamado mostraEmpregados dentro da classe Empresa:

```
1 void mostraEmpregados() {  
2     for (int i = 0; i < this.empregados.length; i++) {  
3         System.out.println("Funcionário na posição: " + i);  
4         // preencher para mostrar outras informacoes do  
           funcionario  
5     }  
6 }
```

- 5. Caso o array já esteja cheia no momento de adicionar um outro funcionário, criar um novo maior e copiar os valores. Isto é, fazer a realocação já que java não tem isso: um array nasce e morre com o mesmo length.

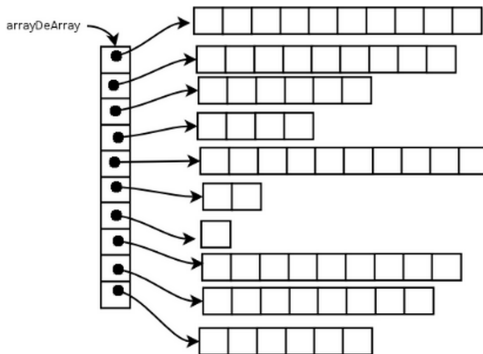
Um pouco mais... de exercícios!

- Arrays podem ter mais de uma dimensão. Isto é, em vez de termos um array de 10 contas, podemos ter um array de 10 por 10 contas e você pode acessar a conta na posição da coluna x e linha y. Na verdade, um array bidimensional em Java é um array de arrays. Pesquise sobre isso.



Um pouco mais... de exercícios!

- Um array bidimensional não precisa ser retangular, isto é, cada linha pode ter um número diferente de colunas. Como? Porque?



Um pouco mais... de exercícios!

- 1. Vamos testar tudo o que vimos até agora com o seguinte programa:
 - **Classe:** Casa **Atributos:** cor, totalDePortas, portas[]
Métodos: void pinta(String s), int quantasPortasEstaoAbertas(), void adicionaPorta(Porta p), int totalDePortas()
- Faça o diagrama de classes desse programa.
- Crie uma casa, pinte-a. Crie três portas e coloque-as na casa através do método adicionaPorta, abra e feche-as como desejar. Utilize o método quantasPortasEstaoAbertas para imprimir o número de portas abertas e o método totalDePortas para imprimir o total de portas em sua casa.

Um pouco mais... de exercícios!

- 2. Escreva uma classe Estatística em Java que contenha métodos que recebam um array de inteiros e calculem:
 - a) a moda dos elementos no array (elemento mais frequente).
 - b) A mediana dos elementos no array (elemento central).
 - c) a média.
- 3. Utilize uma matriz quadrada de tamanho n para construir as n primeiras linhas do **Triângulo de Pascal**.

Modificadores de acesso e atributos de classe