

# Aula VIII - Desenvolvimento de Aplicativo I

## Array (vetores)

### Array List

---

#### Fontes:

<https://distancia.qi.edu.br/mod/book/view.php?id=18698>

<https://www.devmedia.com.br/explorando-a-classe-arraylist-no-java/24298>

[https://www.w3schools.com/java/java\\_arraylist.asp](https://www.w3schools.com/java/java_arraylist.asp)

[Livro de Lógica de Programação - Curso Técnico em Informática - QI Faculdade & Escola Técnica](#)

<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html>

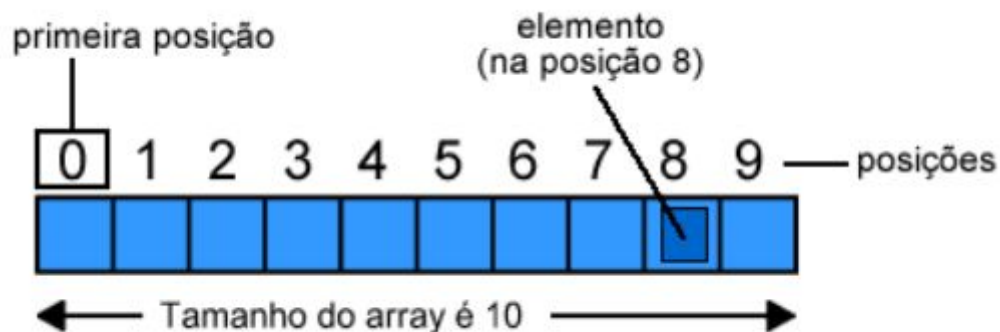
<http://www.javaprogressivo.net>

---

## Array

Um array é um objeto capaz de armazenar **um número fixo de valores de um único tipo**. A quantidade de elementos que armazena é estabelecida quando ele é criado e após a criação seu comprimento de mantém fixo (ORACLE, 2012).

O uso de arrays é indicado quando temos uma certa quantidade de elementos que serão manipulados/armazenados. Por exemplo: as notas de um aluno, as questões de uma prova, as alternativas das questões, um conjunto de números, e assim por diante.



Array de 10 elementos

### Para que serve um Array?

Um array é uma sequência não ordenada de dados. Os elementos residem em um lugar separado na memória, e seu acesso é feito por meio de um índice na primeira posição de cada elemento

## Criando um Array

A declaração de um array utiliza um conjunto de colchetes vazio ao lado do tipo. O tipo pode ser tanto primitivo quanto de referência.

- **Tipos primitivos:** são os tipos de dados básicos da linguagem, com eles podemos criar outros tipos. Exemplos: `int`, `char`, `byte`, `boolean`, etc.
- **Tipos de referência:** são as classes do sistema, tanto as do Java quanto as criadas pelo programador. Exemplos: `String`, `Pessoa`, etc.

## Declarações do Array

### 1ª forma: Declaração e instância em linhas separadas

O exemplo abaixo primeiro declara o **Array**, utilizando o **tipo desejado seguido de colchetes vazios e o nome que se deseja atribuir ao array**. Em outra linha, o comando **new** é responsável pela **criação do objeto na memória**. No exemplo, seria criado um array com capacidade para 5 números inteiros, organizados em índices de 0 até 4. Cria-se um array vazio, pronto para receber elementos.

```
int[] meuArray;  
meuArray = new int[5];
```

### 2ª forma: Declaração e instância na mesma linha

O exemplo abaixo é uma forma abreviada de declarar e instanciar o array também vazio.

```
int[] meuArray = new int[5];
```

### 3ª forma: Declaração com atribuição de valores

O exemplo abaixo demonstra como criar um array já com elementos armazenados.

O primeiro exemplo criaria um array com 5 elementos, conforme a imagem abaixo. O segundo exemplo criaria um array chamado "cores"

```
int[] pares = {0,2,4,6,8};
```

```
String[] cores = {"Azul", "Vermelho", "Amarelo", "Verde"};
```

|   |   |   |   |   |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 |
| 0 | 2 | 4 | 6 | 8 |

|      |          |         |       |
|------|----------|---------|-------|
| 0    | 1        | 2       | 3     |
| Azul | Vermelho | Amarelo | Verde |

### Armazenando valores no array

Após a declaração e instância do array, ele está pronto para armazenar valores. Para isso, é necessário indicar em qual posição queremos armazenar o valor. Esta posição pode ser indicada por um número inteiro definido ou por uma variável do tipo inteiro que contenha um valor de índice válido.

**Vamos utilizar como exemplo o seguinte array:**

```
int[] numeros = new int[6];
```

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
|   |   |   |   |   |   |

Vamos armazenar um número qualquer na primeira posição do array:

```
numeros[0] = 15;
```

|    |   |   |   |   |   |
|----|---|---|---|---|---|
| 0  | 1 | 2 | 3 | 4 | 5 |
| 15 |   |   |   |   |   |

**Ocupando a posição 0**, as demais ficam à disposição para armazenar outros valores. Caso você utilize novamente a posição 0 para armazenar um valor, ele substituirá o valor antigo pelo novo.

**Observe o exemplo:**

```
numeros[0] = 20;
```

|    |   |   |   |   |   |
|----|---|---|---|---|---|
| 0  | 1 | 2 | 3 | 4 | 5 |
| 20 |   |   |   |   |   |

**Preenchendo outras posições:**

```
numeros[1] = 8;  
numeros[4] = 16;
```

|    |   |   |   |    |   |
|----|---|---|---|----|---|
| 0  | 1 | 2 | 3 | 4  | 5 |
| 20 | 8 |   |   | 16 |   |

A posição pode ser indicada por uma variável inteira. Observe o exemplo (considere que o array está vazio):

```

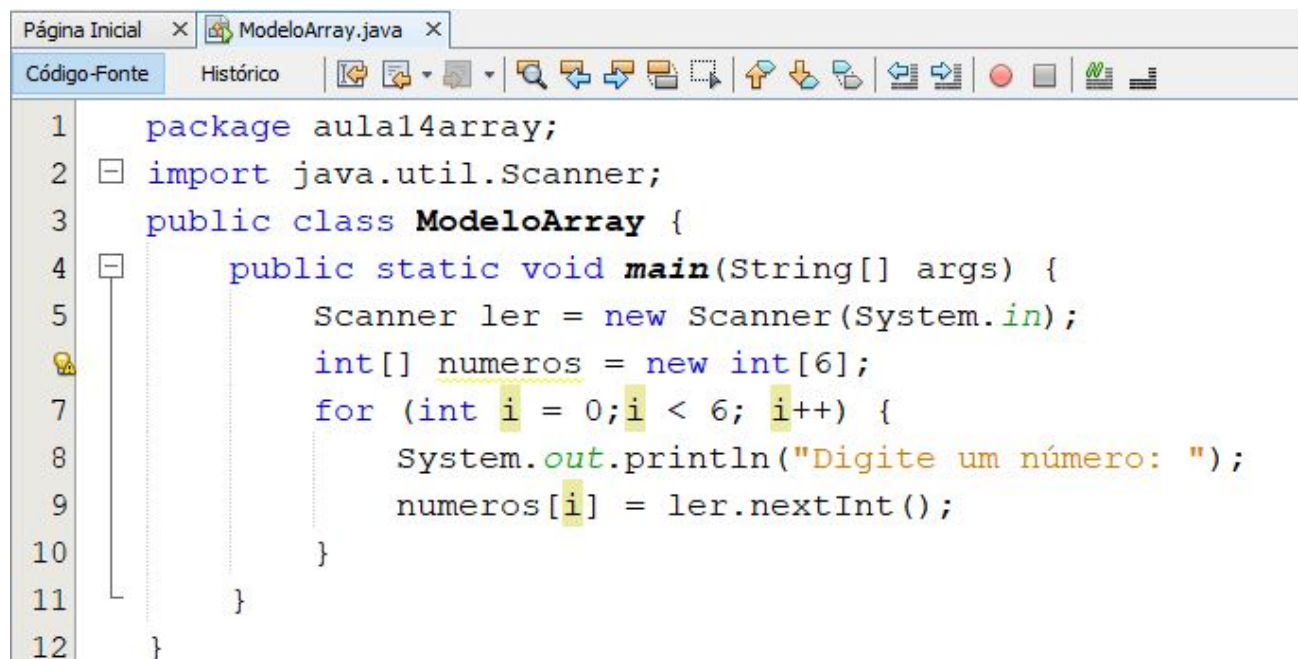
int i = 0; //declara a variável e inicializa com 0
valores[i] = 15; //armazena o valor 15 na posição i (0)
i++; //incrementa o valor de i
valores[i] = 21; //armazena o valor 21 na posição i (1)
i = i + 2; //aumenta 2 no valor de i
valores[i] = 33; //armazena o valor 33 na posição i (3)

```

|    |    |   |    |   |   |
|----|----|---|----|---|---|
| 0  | 1  | 2 | 3  | 4 | 5 |
| 15 | 21 |   | 33 |   |   |

#### Preenchendo um array com valores informados pelo usuário:

Para preencher um array com valores informados pelo usuário podemos utilizar um laço de repetição. Observe o exemplo, que preencherá um array criado na classe Main com valores digitados no terminal:

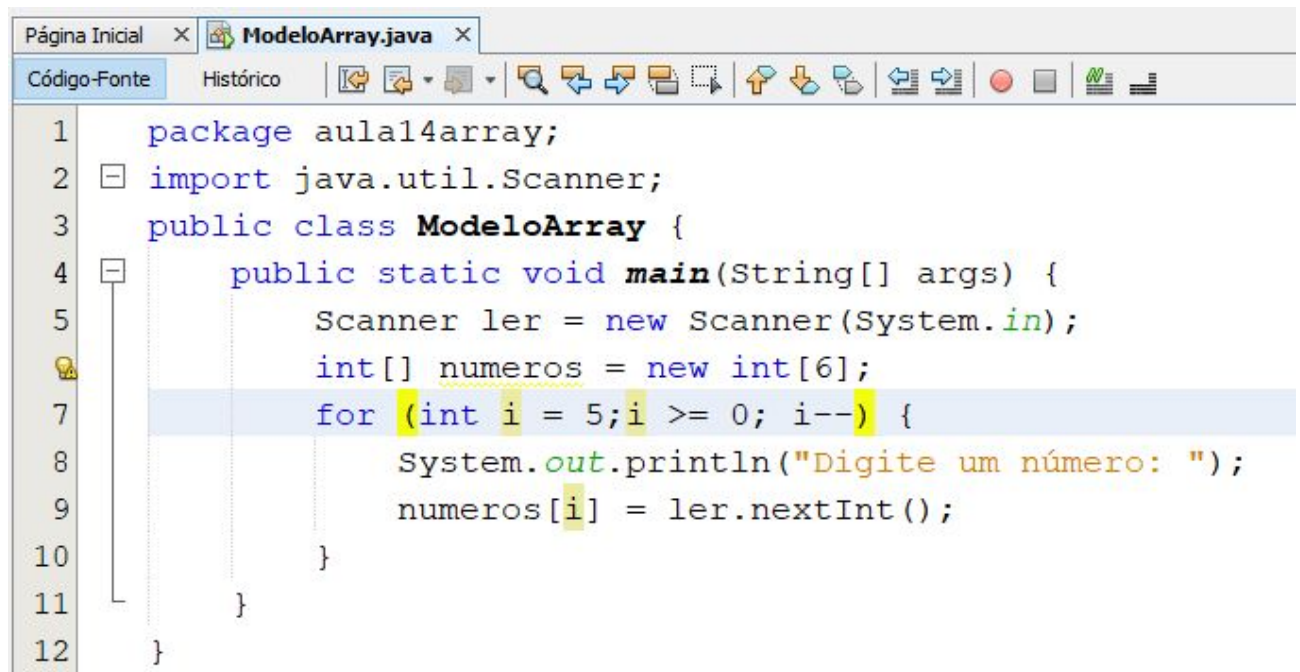


```

1 package aula14array;
2 import java.util.Scanner;
3 public class ModeloArray {
4     public static void main(String[] args) {
5         Scanner ler = new Scanner(System.in);
6         int[] numeros = new int[6];
7         for (int i = 0; i < 6; i++) {
8             System.out.println("Digite um número: ");
9             numeros[i] = ler.nextInt();
10        }
11    }
12 }

```

Observe que na linha 9 foi utilizada a variável "i" para indicar a posição que receberá o elemento digitado pelo usuário. Assim, a cada volta do loop, o número digitado pelo usuário é armazenado numa posição diferente, partindo da primeira (0). Também poderíamos preencher o array de trás para frente, invertendo o for:

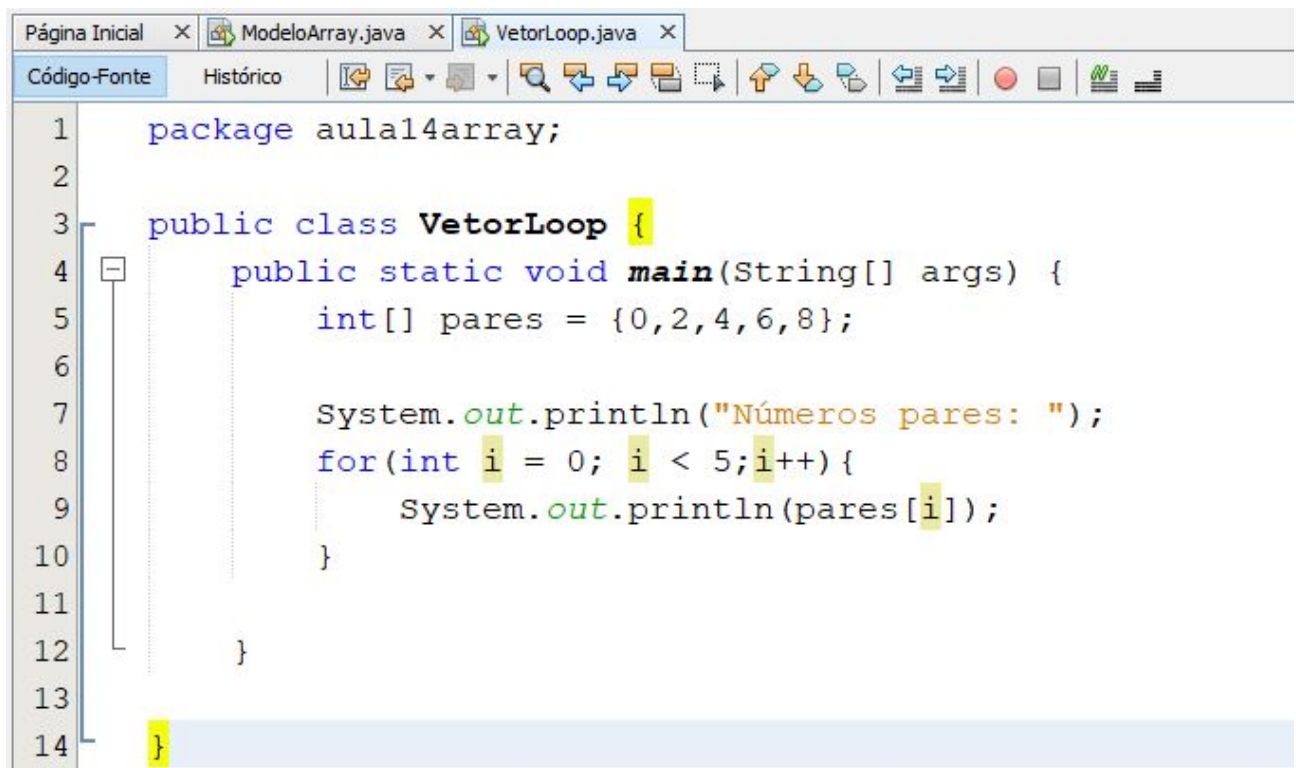


```
1 package aula14array;
2 import java.util.Scanner;
3 public class ModeloArray {
4     public static void main(String[] args) {
5         Scanner ler = new Scanner(System.in);
6         int[] numeros = new int[6];
7         for (int i = 5; i >= 0; i--) {
8             System.out.println("Digite um número: ");
9             numeros[i] = ler.nextInt();
10        }
11    }
12 }
```

### Exibindo valores armazenados no array

Para exibir os valores de um array podemos utilizar um laço de repetição, imprimindo uma posição de cada vez no terminal, ou utilizar o método `toString` da classe `Arrays`.

### Utilizando um loop para exibir o conteúdo de um array



```
1 package aula14array;
2
3 public class VetorLoop {
4     public static void main(String[] args) {
5         int[] pares = {0,2,4,6,8};
6
7         System.out.println("Números pares: ");
8         for(int i = 0; i < 5;i++){
9             System.out.println(pares[i]);
10        }
11    }
12 }
13
14 }
```

## Utilizando a classe Arrays para exibir o conteúdo de um array

A classe Arrays deve ser importada no início da classe. O método **toString** já possui um formato padrão de exibição, que apresenta os elementos entre colchetes separados por ponto e vírgula.

### O que é o método toString?

Classe Object – método toString()

O método toString() retorna uma String que representa o objeto que está chamando o método. Isso é útil para o propósito de log e debug. Vamos descrever o que esse método está fazendo.

Observe o exemplo:



```
Página Inicial x Saída - Aula14Array (run) x ModeloArray.java x VetorLoop.java x
Código-Fonte Histórico
1 package aula14array;
2 import java.util.Arrays;
3 public class VetorLoop {
4     public static void main(String[] args) {
5         int[] pares = {0,2,4,6,8};
6
7         System.out.println("Números pares: ");
8         System.out.println(Arrays.toString(pares));
9     }
10 }
11
12
13
```

## O atributo length

O atributo **length** armazena a quantidade de elementos que um array pode armazenar (considera inclusive as posições vazias). Observe seu uso na linha 10 do exemplo abaixo.

```
Página Inicial x Saída - Aula14Array (run) x ModeloArray.java x VetorLoop.java x VetorTexto.java x VetorComprimento.java x
Código-Fonte Histórico
1
2 package aula14array;
3 import java.util.Arrays;
4
5 public class VetorComprimento {
6     public static void main(String[] args) {
7         int[] pares = {0,2,4,6,8};
8
9         System.out.println("Números pares: ");
10        System.out.println(Arrays.toString(pares));
11        System.out.println("O array contém " + pares.length + " elementos");
12    }
13
14
15 }
```



# Classe ArrayList

Um ArrayList é uma coleção dinâmica capaz de armazenar um número indeterminado de objetos.

A classe ArrayList permite criar um objeto que é capaz de armazenar e gerenciar uma coleção de outros objetos. Um objeto da classe ArrayList é semelhante à estrutura array, porém os arrays são estáticos, nos quais temos que determinar um número de elementos que serão armazenados e esse número de elementos permanece até o final do programa, já a coleção ArrayList é dinâmica, isso significa que a mesma não precisa ter um número determinado para armazenamento e sua estrutura se adapta com cada objeto inserido.

## Vantagens em utilizar a classe ArrayList:

Vantagens em relação à estrutura array:

- Podemos guardar um conjunto de dados especificando o tipo de objeto;
- É dinâmico: não possui tamanho definido;
- Já possui métodos para facilitar o gerenciamento da coleção – inserção, exclusão, classificação e assim por diante.

## Importação da Classe:

```
package aula15vetoreslista;  
import java.util.ArrayList;
```

## Declaração de um ArrayList:

### visibilidade

```
ArrayList <TipoObjeto> nomeObjeto;  
nomeObjeto = new ArrayList<>();
```

### ou visibilidade

```
ArrayList <TipoObjeto> nomeObjeto = new ArrayList<>();
```

**Exemplo:**

```
private ArrayList <Produto> lista;  
lista = new ArrayList<>();
```

## Tipos de Objetos

Podemos armazenar objetos do tipo números inteiros, números reais, textos, valores booleanos e objetos de classes criadas no sistema. Vejamos um exemplo de cada.

| Tipo de Objeto | Sintaxe                                                                                                    |
|----------------|------------------------------------------------------------------------------------------------------------|
| Inteiro        | <pre>private ArrayList &lt;Integer&gt; listaDeNumeros;<br/>listaDeNumeros = new ArrayList&lt;&gt;();</pre> |
| String         | <pre>private ArrayList &lt;String&gt; listaDeNumeros;<br/>listaDeNumeros = new ArrayList&lt;&gt;();</pre>  |
| Byte           | <pre>private ArrayList &lt;Byte&gt; listaDeNumeros;<br/>listaDeNumeros = new ArrayList&lt;&gt;();</pre>    |
| Boolean        | <pre>private ArrayList &lt;Boolean&gt; listaDeNumeros;<br/>listaDeNumeros = new ArrayList&lt;&gt;();</pre> |
| Classes        | <pre>private ArrayList &lt;Pessoas&gt; listaDeNumeros;<br/>listaDeNumeros = new ArrayList&lt;&gt;();</pre> |

**Dica:**

Obs.:

Dependendo da versão do seu JDK, pode haver necessidade de repetir o tipo de dado.

Exemplo:

```
private ArrayList <Funcionario> listaDeFuncionarios;  
listaDeFuncionarios = new ArrayList<Funcionario>();
```

### Métodos da classe ArrayList:

| Método                | Função                                                        |
|-----------------------|---------------------------------------------------------------|
| <b>add(elemento)</b>  | Insere um elemento no ArrayList                               |
| <b>clear()</b>        | Limpa toda a lista do ArrayList                               |
| <b>isEmpty()</b>      | Verifica se a lista está vazia (retorna um boolean)           |
| <b>size()</b>         | Retorna a quantidade do ArrayList                             |
| <b>get(índice)</b>    | Retorna o elemento que está armazenado no índice especificado |
| <b>remove(índice)</b> | Remove o elemento contido no índice especificado              |
| <b>remove(objeto)</b> | Remove o objeto especificado                                  |

| 0      | 1        | 2     | 3    | 4     |
|--------|----------|-------|------|-------|
| Branco | Vermelho | Verde | Azul | Preto |

**Interface Collection:** A coleção em Java é uma estrutura que fornece uma arquitetura para armazenar e manipular o grupo de objetos.

Iterator() e sort()