

# Módulo 1

## Introdução à Programação I



## Lição 7

### Array em Java

*Versão 1.0 - Jan/2007*

**Autor**

Florence Tiu Balagtas

**Equipe**

Joyce Avestro  
 Florence Balagtas  
 Rommel Feria  
 Reginald Hutcherson  
 Rebecca Ong  
 John Paul Petines  
 Sang Shin  
 Raghavan Srinivas  
 Matthew Thompson

**Necessidades para os Exercícios****Sistemas Operacionais Suportados****NetBeans IDE 5.5** para os seguintes sistemas operacionais:

1. Microsoft Windows XP Professional SP2 ou superior
2. Mac OS X 10.4.5 ou superior
3. Red Hat Fedora Core 3
4. Solaris™ 10 Operating System (SPARC® e x86/x64 Platform Edition)

**NetBeans Enterprise Pack**, poderá ser executado nas seguintes plataformas:

1. Microsoft Windows 2000 Professional SP4
2. Solaris™ 8 OS (SPARC e x86/x64 Platform Edition) e Solaris 9 OS (SPARC e x86/x64 Platform Edition)
3. Várias outras distribuições Linux

**Configuração Mínima de Hardware****Nota:** IDE NetBeans com resolução de tela em 1024x768 pixel

Sistema Operacional	Processador	Memória	HD Livre
Microsoft Windows	500 MHz Intel Pentium III workstation ou equivalente	512 MB	850 MB
Linux	500 MHz Intel Pentium III workstation ou equivalente	512 MB	450 MB
Solaris OS (SPARC)	UltraSPARC II 450 MHz	512 MB	450 MB
Solaris OS (x86/x64 Platform Edition)	AMD Opteron 100 Série 1.8 GHz	512 MB	450 MB
Mac OS X	PowerPC G4	512 MB	450 MB

**Configuração Recomendada de Hardware**

Sistema Operacional	Processador	Memória	HD Livre
Microsoft Windows	1.4 GHz Intel Pentium III workstation ou equivalente	1 GB	1 GB
Linux	1.4 GHz Intel Pentium III workstation ou equivalente	1 GB	850 MB
Solaris OS (SPARC)	UltraSPARC IIIi 1 GHz	1 GB	850 MB
Solaris OS (x86/x64 Platform Edition)	AMD Opteron 100 Series 1.8 GHz	1 GB	850 MB
Mac OS X	PowerPC G5	1 GB	850 MB

**Requerimentos de Software**

NetBeans Enterprise Pack 5.5 executando sobre Java 2 Platform Standard Edition Development Kit 5.0 ou superior (JDK 5.0, versão 1.5.0\_01 ou superior), contemplando a Java Runtime Environment, ferramentas de desenvolvimento para compilar, depurar, e executar aplicações escritas em linguagem Java. Sun Java System Application Server Platform Edition 9.

1. Para **Solaris**, **Windows**, e **Linux**, os arquivos da JDK podem ser obtidos para sua plataforma em <http://java.sun.com/j2se/1.5.0/download.html>
2. Para **Mac OS X**, Java 2 Platform Standard Edition (J2SE) 5.0 Release 4, pode ser obtida diretamente da Apple's Developer Connection, no endereço: <http://developer.apple.com/java> (é necessário registrar o download da JDK).

Para mais informações:

<http://www.netbeans.org/community/releases/55/relnotes.html>

## **Colaboradores que auxiliaram no processo de tradução e revisão**

Alexandre Mori	Hugo Leonardo Malheiros Ferreira	Mauro Regis de Sousa Lima
Alexis da Rocha Silva	Ivan Nascimento Fonseca	Namor de Sá e Silva
Aline Sabbatini da Silva Alves	Jacqueline Susann Barbosa	Néres Chaves Rebouças
Allan Wojcik da Silva	Jader de Carvalho Belarmino	Nolyanne Peixoto Brasil Vieira
André Luiz Moreira	João Aurélio Telles da Rocha	Paulo Afonso Corrêa
Andro Márcio Correa Louredo	João Paulo Cirino Silva de Novais	Paulo José Lemos Costa
Antonie de Assis Lima	João Vianney Barrozo Costa	Paulo Oliveira Sampaio Reis
Antonio Jose R. Alves Ramos	José Augusto Martins Nieviadonski	Pedro Antonio Pereira Miranda
Aurélio Soares Neto	José Leonardo Borges de Melo	Pedro Henrique Pereira de Andrade
Bruno da Silva Bonfim	José Ricardo Carneiro	Renato Alves Félix
Bruno dos Santos Miranda	Kleberth Bezerra G. dos Santos	Renato Barbosa da Silva
Bruno Ferreira Rodrigues	Lafaiete de Sá Guimarães	Reydersen Magela dos Reis
Carlos Alberto Vitorino de Almeida	Leandro Silva de Moraes	Ricardo Ferreira Rodrigues
Carlos Alexandre de Sene	Leonardo Leopoldo do Nascimento	Ricardo Ulrich Bomfim
Carlos André Noronha de Sousa	Leonardo Pereira dos Santos	Robson de Oliveira Cunha
Carlos Eduardo Veras Neves	Leonardo Rangel de Melo Filardi	Rodrigo Pereira Machado
Cleber Ferreira de Sousa	Lucas Mauricio Castro e Martins	Rodrigo Rosa Miranda Corrêa
Cleyton Artur Soares Urani	Luciana Rocha de Oliveira	Rodrigo Vaez
Cristiano Borges Ferreira	Luís Carlos André	Ronie Dotzlaw
Cristiano de Siqueira Pires	Luís Octávio Jorge V. Lima	Rosely Moreira de Jesus
Derlon Vandri Aliendres	Luiz Fernandes de Oliveira Junior	Seire Pareja
Fabiano Eduardo de Oliveira	Luiz Victor de Andrade Lima	Sergio Pomerancblum
Fábio Bombonato	Manoel Cotts de Queiroz	Silvio Sznifer
Fernando Antonio Mota Trinta	Marcello Sandi Pinheiro	Suzana da Costa Oliveira
Flávio Alves Gomes	Marcelo Ortolan Pazzetto	Suzana da Costa Oliveira
Francisco das Chagas	Marco Aurélio Martins Bessa	Tásio Vasconcelos da Silveira
Francisco Marcio da Silva	Marcos Vinicius de Toledo	Thiago Magela Rodrigues Dias
Gilson Moreno Costa	Maria Carolina Ferreira da Silva	Tiago Gimenez Ribeiro
Givailson de Souza Neves	Massimiliano Girolodi	Vanderlei Carvalho Rodrigues Pinto
Gustavo Henrique Castellano	Mauricio Azevedo Gamarra	Vanessa dos Santos Almeida
Hebert Julio Gonçalves de Paula	Mauricio da Silva Marinho	Vastí Mendes da Silva Rocha
Heraldo Conceição Domingues	Mauro Cardoso Morton	Wagner Eliezer Roncoletta

## **Auxiliadores especiais**

Revisão Geral do texto para os seguintes Países:

- **Brasil** – Tiago Flach
- **Guiné Bissau** – Alfredo Cá, Bunene Sisse e Buon Olossato Quebi – ONG Asas de Socorro

## **Coordenação do DFJUG**

- **Daniel deOliveira** – JUGLeader responsável pelos acordos de parcerias
- **Luci Campos** - Idealizadora do DFJUG responsável pelo apoio social
- **Fernando Anselmo** - Coordenador responsável pelo processo de tradução e revisão, disponibilização dos materiais e inserção de novos módulos
- **Regina Mariani** - Coordenadora responsável pela parte jurídica
- **Rodrigo Nunes** - Coordenador responsável pela parte multimídia
- **Sérgio Gomes Veloso** - Coordenador responsável pelo ambiente JEDI™ (Moodle)

## **Agradecimento Especial**

**John Paul Petines** – Criador da Iniciativa JEDI™

**Rommel Faria** – Criador da Iniciativa JEDI™

# 1. Objetivos

Nesta lição, abordaremos Array em Java. Primeiro, definiremos o que é array e, então, discutiremos como declará-los e usá-los.

Ao final desta lição, o estudante será capaz de:

- Declarar e criar array
- Acessar elementos de um array
- Determinar o número de elementos de um array
- Declarar e criar array multidimensional

## 2. Introdução a Array

Em lições anteriores, discutimos como declarar diferentes variáveis usando os tipos de dados primitivos. Na declaração de variáveis, frequentemente 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;
```

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**.

	<b>0</b>	<b>1</b>	<b>2</b>
<b>number:</b>	<b>1</b>	<b>2</b>	<b>3</b>

*Figura 1: Exemplo de um array de inteiros*

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.

### 3. Declarando Array

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[];
```

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**. Por exemplo:

```
// declaração
int ages[];

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

ou, pode ser escrito como:

```
// declarar e construir
int ages[] = new int[100];
```

No exemplo, a declaração diz ao compilador Java que o identificador `ages` será usado como um nome de um array contendo inteiros, usado para criar, ou construir, um novo array contendo 100 elementos.

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.

Exemplos:

```
// 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"};
```

Uma vez que tenha sido inicializado, o tamanho de um array não pode ser modificado, pois é armazenado em um bloco contínuo de memória.

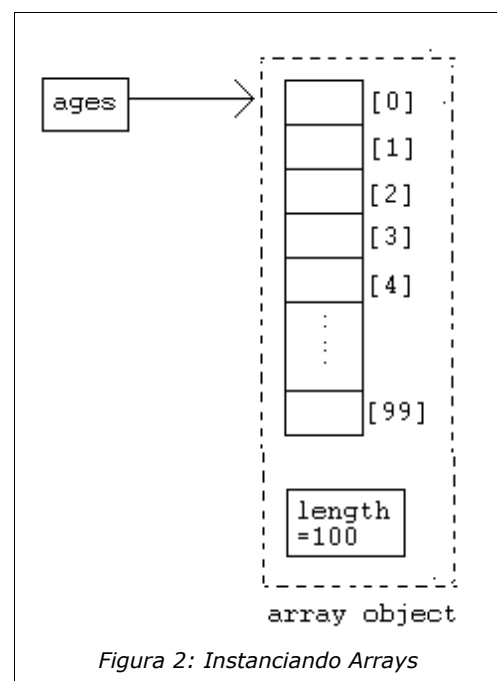


Figura 2: Instanciando Arrays

## 4. Acessando um elemento do Array

Para acessar um elemento do array, ou parte de um array, utiliza-se um número inteiro chamado de **índice**.

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** a **tamanhoDoArray-1**.

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

```
// atribuir 10 ao primeiro elemento do array
ages[0] = 10;

// imprimir o último elemento do array
System.out.print(ages[99]);
```

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

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

Tabela 1: Valor de inicialização automática para os tipos primitivos

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 utilizá-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:

```
public class ArraySample {
    public static void main(String[] args){
        String [] nulls = new String[2];
        System.out.print(nulls[0]); // Linha correta, mostra null
        System.out.print(nulls[1].trim()); // Causa erro
    }
}
```

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

```
public class ArraySample {
    public static void main(String[] args){
        int[] ages = new int[100];
        for (int i = 0; i < 100; i++) {
            System.out.print(ages[i]);
        }
    }
}
```

```
    }  
  }  
}
```

**Dicas de programação:**

1. Normalmente, é melhor inicializar, ou instanciar, um array logo após declará-lo. Por exemplo, a instrução:

```
int []arr = new int[100];
```

é preferível, ao invés de:

```
int [] arr;  
arr = new int[100];
```

2. Os elementos de um array de n elementos tem índices de 0 a n-1. Note que não existe o elemento arr[n]. A tentativa de acesso a este elemento causará uma exceção do tipo **ArrayIndexOutOfBoundsException**, pois o índice deve ser até n-1.
3. Não é possível modificar o tamanho de um array.



## 5. 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:

```
public class ArraySample {  
    public static void main (String[] args) {  
        int[] ages = new int[100];  
        for (int i = 0; i < ages.length; i++) {  
            System.out.print(ages[i]);  
        }  
    }  
}
```

### ***Dicas de programação:***

1. Quando criar laços com **for** para o processamento de um array, utilize o campo **length** como argumento da expressão lógica. Isto irá permitir ao laço ajustar-se, automaticamente para tamanhos de diferentes arrays.
2. Declare o tamanho dos arrays utilizando variáveis do tipo constante para facilitar alterações posteriores. Por exemplo:

```
final int ARRAY_SIZE = 1000; // declarando uma constante  
...  
int[] ages = new int[ARRAY_SIZE];
```

## 6. 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 inteiro de 512 x 128 elementos
int [][] twoD = new int[512][128];

// array de caracteres de 8 x 16 x 24
char [][][] threeD = new char[8][16][24];

// array de String de 4 linhas x 2 colunas
String [][] dogs = {"terry", "brown",
                    {"Kristin", "white"},
                    {"toby", "gray"},
                    {"fido", "black"}};
```

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 **dogs**, escreve-se:

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

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

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

## 7. Exercícios

### 7.1. *Dias da semana*

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

```
String days[] = {"Monday", "Tuesday", "Wednesday", "Thursday",  
                "Friday", "Saturday", "Sunday"};
```

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

### 7.2. *Maior número*

Usando as classes **BufferedReader**, **Scanner** ou **JOptionPane**, solicite 10 números ao usuário. Utilize um array para armazenar o valor destes números. Mostre o número de maior valor.

### 7.3. *Entradas de agenda telefônica*

Dado o seguinte array multidimensional, que contém as entradas da agenda telefônica:

```
String entry = {"Florence", "735-1234", "Manila"},  
               {"Joyce", "983-3333", "Quezon City"},  
               {"Becca", "456-3322", "Manila"};
```

mostre-as conforme o formato abaixo:

```
Name    : Florence  
Tel. #  : 735-1234  
Address: Manila  
  
Name    : Joyce  
Tel. #  : 983-3333  
Address: Quezon City  
  
Name    : Becca  
Tel. #  : 456-3322  
Address: Manila
```

## Parceiros que tornaram JEDI™ possível



### ***Instituto CTS***

Patrocinador do DFJUG.

### ***Sun Microsystems***

Fornecimento de servidor de dados para o armazenamento dos vídeo-aulas.

### ***Java Research and Development Center da Universidade das Filipinas***

Criador da Iniciativa JEDI™.

### ***DFJUG***

Detentor dos direitos do JEDI™ nos países de língua portuguesa.

### ***Banco do Brasil***

Disponibilização de seus *telecentros* para abrigar e difundir a Iniciativa JEDI™.

### ***Politec***

Suporte e apoio financeiro e logístico a todo o processo.

### ***Borland***

Apoio internacional para que possamos alcançar os outros países de língua portuguesa.

### ***Instituto Gaudium/CNBB***

Fornecimento da sua infra-estrutura de hardware de seus servidores para que os milhares de alunos possam acessar o material do curso simultaneamente.