

# Java Starter

[www.t2ti.com](http://www.t2ti.com)

# Curso Java Starter

---

## **Apresentação**

O Curso Java Starter foi projetado com o objetivo de ajudar àquelas pessoas que têm uma base de lógica de programação e desejam entrar no mercado de trabalho sabendo Java,

A estrutura do curso é formada por módulos em PDF e por mini-cursos em vídeo. O aluno deve baixar esse material e estudá-lo. Deve realizar os exercícios propostos. Todas as dúvidas devem ser enviadas para a lista de discussão que está disponível para inscrição na página do Curso Java Starter no site [www.t2ti.com](http://www.t2ti.com). As dúvidas serão respondidas pelos instrutores Albert Eije, Cláudio de Barros e Miguel Kojiio, além dos demais participantes da lista.

Nosso objetivo é que após o estudo do Curso Java Starter o aluno não tenha dificuldades para acompanhar um curso avançado onde poderá aprender a desenvolver aplicativos para Web, utilizando tecnologias como Servlets e JSP e frameworks como Struts e JSF, além do desenvolvimento para dispositivos móveis.

Albert Eije trabalha com informática desde 1993. Durante esse período já trabalhou com várias linguagens de programação: Clipper, PHP, Delphi, C, Java, etc. Atualmente mantém o site [www.alberteije.com](http://www.alberteije.com).

Cláudio de Barros é Tecnólogo em Processamento de Dados.

Miguel Kojiio é bacharel em Sistemas de Informação, profissional certificado Java (SCJP 1.5).

O curso Java Starter surgiu da idéia dos três amigos que trabalham juntos em uma instituição financeira de grande porte.

### Módulo

# 03

## Arrays e entrada de dados

### Introdução

Antes de iniciarmos vale relembrar que o seu aprendizado depende muito da qualidade do seu estudo, com isto nós queremos dizer que: “Os exercícios são parte importante da sedimentação do seu conhecimento”.

Durante o texto as palavras array e vetores serão utilizadas sempre com o mesmo significado (estruturas de dados seqüenciais).

Uma coleção nada mais é do que um conjunto de elementos contidos em uma única estrutura – em Java um objeto – cuja função é oferecer meios de armazenar, disponibilizar, remover, localizar e percorrer o seu conteúdo.

Coleções são estruturas de dados que agrupam elementos que formam um grupo natural como por exemplo: baralho (conjunto de cartas), time (conjunto de jogadores), turma (conjunto de alunos) e etc.

A seguir conceituaremos superficialmente<sup>1</sup> os tipos de coleções mais comuns:

- **Vetor:** É formado por um grupo de elementos acessados através do seu índice;
- **Pilha:** Estrutura de dados onde o último elemento a ser inserido na coleção é o primeiro a ser retirado (Baseado no princípio LIFO, “Last in, first out”);
- **Fila:** Coleção onde a ordem de inserção representa a ordem de saída dos elementos (Baseado no princípio FIFO, “First in, first out”);
- **Árvores:** Estrutura de dados que garante a ordenação dos elementos que a compõe;

<sup>1</sup>Para saber mais sobre estruturas de dados visite: [http://pt.wikipedia.org/wiki/Estrutura\\_de\\_dados](http://pt.wikipedia.org/wiki/Estrutura_de_dados)

## Curso Java Starter

---

Cada uma das estruturas de dados apresentadas possui características que as diferenciam quanto a:

- Eficiência de busca;
- Eficiência de inserção;
- Organização;
- Ordenação;
- Forma de acesso;
- Forma de busca e;
- Forma de inserção.

### Arrays unidimensionais:

A forma mais eficiente de trabalhar com coleções de elementos em Java é através da construção de vetores (arrays). Em Java, arrays são objetos que armazenam múltiplas variáveis do mesmo tipo ou do mesmo sub-tipo (sub-tipo? Não se preocupe por enquanto com isto).

Observe que apesar da sua eficiência, normalmente, para armazenar dados nós utilizamos estruturas de dados mais flexíveis já existentes na linguagem, mais especificamente, no Framework Collections (assunto a ser abordado em mais detalhes durante este curso) ao invés de arrays.

Um array é um objeto que armazena um número pré-definido de elementos, isto é, o seu tamanho é definido no momento da sua construção. Seus elementos são acessados através de índices que iniciam-se sempre por 0 (zero), ou seja, um array de tamanho quatro terá índices 0, 1, 2 e 3.

Em Java existem diversas formas de declarar, construir e inicializar arrays e a melhor forma de utilizar estas estruturas de dados é conhecendo como são realizadas cada uma destas etapas.

A seguir apresentaremos cada uma destas etapas e a forma como elas acontecem na prática:

1. **Declaração:** Etapa em que a referência do array é declarada;
2. **Construção:** Aqui é definido o tamanho e instanciado o array;
3. **Inicialização:** Os elementos são inseridos no array.

## Curso Java Starter

Abaixo temos um exemplo com todas as etapas bem definidas:

```
1. int[] jogoSena; //Declaração
2.
3. jogoSena = new int[6]; //Criação
4.
5. jogoSena[0] = 23; //Inicialização da posição 0
6. jogoSena[1] = 12; //Inicialização da posição 1
7. jogoSena[2] = 55; //Inicialização da posição 2
8. jogoSena[3] = 02; //Inicialização da posição 3
9. jogoSena[4] = 07; //Inicialização da posição 4
10.jogoSena[5] = 19; //Inicialização da posição 5
```

Neste caso estamos criando um vetor de inteiros (int), perceba que a **declaração** da variável ocorre na linha 1, na seqüência temos a **criação** do vetor (linha 3) e por último a **inicialização** de cada uma das suas posições.

Existem formas mais enxutas de efetuarmos as mesmas etapas mostradas no exemplo anterior. Podemos declarar, construir e inicializar em apenas uma linha da seguinte forma:

```
int[] outroJogoSena = {23, 12, 55, 02, 07, 19};
```

No exemplo acima, acontecem quatro coisas em apenas uma linha:

- I. Declaração de uma referência a um array de inteiros chamado outroJogoSena;
- II. Criação de um array com seis posições;
- III. Inicialização das posições com os valores 23, 12, 55, 02, 07 e 19;
- IV. Atribuição do novo objeto (array) a referência outroJogoSena;

O outro atalho que a linguagem Java nos permite é o seguinte:

```
int[] outroJogoSena = new int[]{23, 12, 55, 02, 07, 19};
```

Em ambos os casos todas as etapas continuam sendo executadas (declaração, construção e inicialização).

Certo, mas qual é o tamanho destes arrays? Quando criamos arrays desta maneira a quantidade máxima de elementos que o array irá armazenar será igual a quantidade de elementos com que ele foi inicializado, isto é, em ambos os casos os arrays seriam suficientes para armazenar até 6 elementos. Caso haja necessidade de

## Curso Java Starter

mais espaço um novo array deve ser construído.

Para conhecer o tamanho total de um array basta você acessar o atributo **length**. Este atributo retorna um valor inteiro (int) que indica qual a capacidade máxima de armazenamento deste array.

Lembretes:

- ✓ **Primeira posição (índice):** de qualquer array é sempre 0;
- ✓ **Última posição (índice):** é sempre o seu tamanho - 1 (length - 1).

### Exercício Resolvido

Implemente a lógica para realização de saques em um caixa eletrônico considerando que o mesmo armazena cédulas de R\$100,00, R\$50,00, R\$20,00, R\$10,00, R\$5,00, R\$2,00 e R\$1,00 e devem ser entregues ao cliente o menor número possível de cédulas.

```
public class CaixaEletronico {
    public static void main(String[] args) {
        //Cedulas disponiveis no caixa eletronico
        int[] cedulas = {100, 50, 20, 10, 5, 2, 1};
        //Quantidade total de cedulas entregues ao cliente
        int quantidadeTotal = 0;
        //valor a ser sacado pelo cliente
        int valorReais = 163;
        //Percorrendo todas as cedulas disponiveis no caixa eletronico
        for(int i = 0; i < cedulas.length; i++)
        {
            //Quantidade de cedulas para o valor cedulas[i]
            int quantidadeCedulas = valorReais/cedulas[i];
            //Impressao da quantidade de cedulas
            System.out.println("Quantidadde de cedulas de "+ cedulas[i] +
                ": " +quantidadeCedulas );
            //Resto da divisao
            valorReais %= cedulas[i];
            //Quantidade total de cedulas entregues ao cliente
            quantidadeTotal += quantidadeCedulas;
        }
        //Impressao
        System.out.println("Quantidade total: "+quantidadeTotal);
    }
}
```

Resultado obtido durante a execução deste código:

```
Quantidadde de cedulas de 100: 1
Quantidadde de cedulas de 50: 1
Quantidadde de cedulas de 20: 0
Quantidadde de cedulas de 10: 1
Quantidadde de cedulas de 5: 0
Quantidadde de cedulas de 2: 1
Quantidadde de cedulas de 1: 1
Quantidade total: 5
```

## Exercício Resolvido

Implemente um algoritmo que ordene um array com 10 números inteiros. O algoritmo deve efetuar uma comparação de cada elemento com o seu sucessor e se a ordem não for crescente os elementos devem ter suas posições trocadas, este processo deve se repetir até que o array esteja ordenado.

```
public class OrdenarArray {
    public static void main(String[] args) {
        int[] arrayInteiros = {55, 27, 33, 45, 92, 100, 3, 8, 11, 70};
        boolean estaOrdenado = false;
        while(!estaOrdenado)
        {
            estaOrdenado = true;
            for(int i = 1; i < arrayInteiros.length; i++)
            {
                if(arrayInteiros[i-1] > arrayInteiros[i])
                {
                    estaOrdenado = false;
                    int aux = arrayInteiros[i];
                    arrayInteiros[i] = arrayInteiros[i-1];
                    arrayInteiros[i-1] = aux;
                    System.out.println("Ordem atual: "+
                        arrayInteiros[0]+ " " + arrayInteiros[1]+" "+
                        arrayInteiros[2]+ " " + arrayInteiros[3]+" "+
                        arrayInteiros[4]+ " " + arrayInteiros[5]+" "+
                        arrayInteiros[6]+ " " + arrayInteiros[7]+" "+
                        arrayInteiros[8]+ " " + arrayInteiros[9]);
                }
            }
        }
    }
}
```

Resultado da execução deste código:

```
Ordem atual: 27 55 33 45 92 100 3 8 11 70
Ordem atual: 27 33 55 45 92 100 3 8 11 70
Ordem atual: 27 33 45 55 92 100 3 8 11 70
Ordem atual: 27 33 45 55 92 3 100 8 11 70
Ordem atual: 27 33 45 55 92 3 8 100 11 70
Ordem atual: 27 33 45 55 92 3 8 11 100 70
Ordem atual: 27 33 45 55 92 3 8 11 70 100
Ordem atual: 27 33 45 55 3 92 8 11 70 100
Ordem atual: 27 33 45 55 3 8 92 11 70 100
Ordem atual: 27 33 45 55 3 8 11 92 70 100
Ordem atual: 27 33 45 55 3 8 11 70 92 100
Ordem atual: 27 33 45 3 55 8 11 70 92 100
Ordem atual: 27 33 45 3 8 55 11 70 92 100
Ordem atual: 27 33 45 3 8 11 55 70 92 100
Ordem atual: 27 33 3 45 8 11 55 70 92 100
Ordem atual: 27 33 3 8 45 11 55 70 92 100
Ordem atual: 27 33 3 8 11 45 55 70 92 100
Ordem atual: 27 3 33 8 11 45 55 70 92 100
Ordem atual: 27 3 8 33 11 45 55 70 92 100
Ordem atual: 27 3 8 11 33 45 55 70 92 100
```

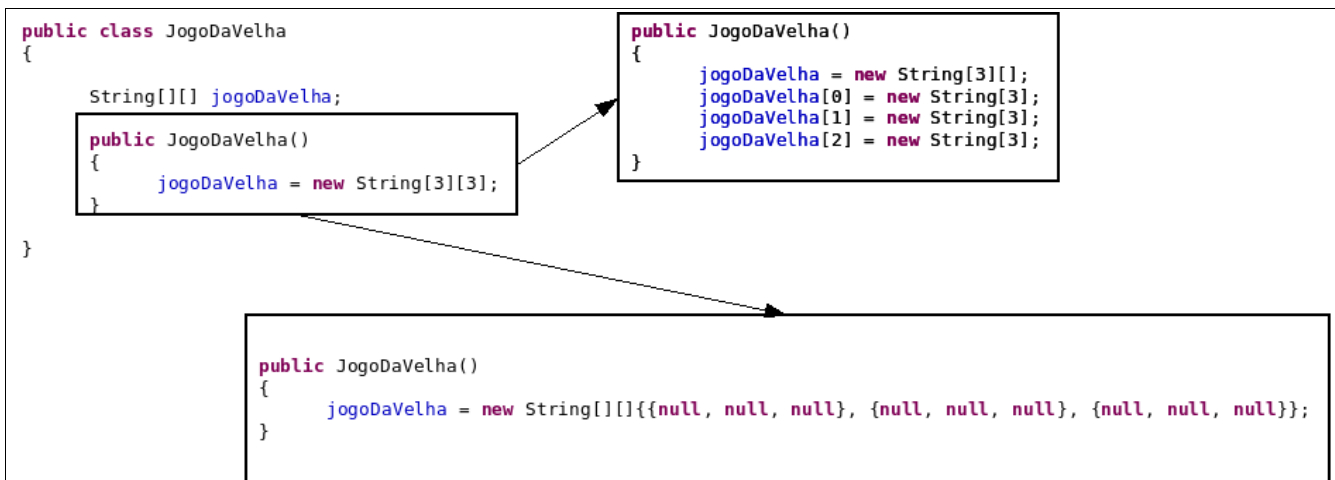
Ordem atual: 3 27 8 11 33 45 55 70 92 100  
Ordem atual: 3 8 27 11 33 45 55 70 92 100  
Ordem atual: 3 8 11 27 33 45 55 70 92 100

## Arrays Multi-dimensionais

Arrays unidimensionais são estruturas de dados bastante simples. Uma estrutura um pouco mais complexa são os arrays multi-dimensionais ou n-dimensionais.

A função destes arrays é a mesma dos seus irmãos unidimensionais porém arrays multi-dimensionais permitem a construção de estruturas de dados mais ricas.

No caso de arrays multi-dimensionais a declaração, construção e inicialização é realizada conforme exemplos a seguir, observe que os trechos de códigos são substituíveis entre si, isto é, equivalentes:



Java permite a construção de arrays com qualquer número de dimensões, contudo dificilmente encontram-se arrays com mais de três dimensões.

## Exercício Resolvido

Implemente um programa que construa uma matriz de double 4x4 e para cada célula atribua o valor correspondente a  $i*j$ .



# Curso Java Starter

---

Resolução:

1. A matriz será de double;
2. A classe deverá manipular um array bidimensional;
3. O valor de cada célula corresponderá ao valor dos respectivos índices multiplicados.

```
public class Matriz
{
    public static void main(String[] args)
    {
        double matriz[][] = new double[4][4]; //declaracao e construcao da matriz
        for(int i = 0; i < matriz.length; i++) //percorre a matriz no eixo i
        {
            for(int j = 0; j < matriz[i].length; j++) //percorre a matriz no eixo j
            {
                matriz[i][j] = i*j; //atribui o valor a celula
            }
        }

        for(int i = 0; i < matriz.length; i++) //percorre a matriz no eixo i
        {
            for(int j = 0; j < matriz[i].length; j++) //percorre a matriz no eixo j
            {
                //imprime o resultado
                System.out.println("Valor da posição ["+i+", "+j+"]: "+
                    matriz[i][j]);
            }
        }
    }
}
```

Saída console:

```
Valor da posição [0,0]: 0.0
Valor da posição [0,1]: 0.0
Valor da posição [0,2]: 0.0
Valor da posição [0,3]: 0.0
Valor da posição [1,0]: 0.0
Valor da posição [1,1]: 1.0
Valor da posição [1,2]: 2.0
Valor da posição [1,3]: 3.0
Valor da posição [2,0]: 0.0
Valor da posição [2,1]: 2.0
Valor da posição [2,2]: 4.0
Valor da posição [2,3]: 6.0
Valor da posição [3,0]: 0.0
Valor da posição [3,1]: 3.0
Valor da posição [3,2]: 6.0
Valor da posição [3,3]: 9.0
```

## **Manipulando vetores utilizando a classe Arrays**

A classe Arrays, disponível no pacote java.util, fornece uma grande

## Curso Java Starter

quantidade de métodos utilitários, como por exemplo métodos para ordenação, procura, comparação e etc.. Estes métodos são muito úteis quando manipulamos arrays. A seguir serão apresentados os principais métodos e as respectivas funcionalidades oferecidas.

- **Ordenação:** Realizada utilizando-se o método "sort" cujo parâmetro é o vetor a ser ordenado;
- **Pesquisa:** A localização de um determinado elemento em um array é realizada utilizando-se o método "binarySearch" que retorna a posição que o elemento foi encontrado no array. Caso o elemento não seja encontrado retorna um valor negativo;
- **Preenchimento:** Utilizando-se o método "fill" da classe utilitária Arrays é possível preencher um determinado array com o elemento desejado;
- **Comparação:** Dados dois arrays o método "equals" compara valor a valor e retorna true se os vetores são idênticos em valores e índices.

Agora que conhecemos, superficialmente, a forma como a classe Arrays fornece métodos utilitários vamos ver na prática como algumas destas funcionalidades são utilizadas.

### Exercício Resolvido - Ordenação

Implemente um programa que construa um array de inteiros (int) de tamanho 10.000 com valores atribuídos da seguinte forma, cada posição do array conterá o resultado da operação 10.000 – índice da posição, isto é, a posição 0 terá o valor 10.000, a posição 1 terá o valor 9.999 e assim por diante. Ordene utilizando o algoritmo de ordenação implementado no exercício resolvido pag. 6 e na seqüência utilizando a classe utilitária Arrays. Compare os tempos de ordenação.

```
import java.util.Arrays;

public class OrdenacaoArray {

    public static void main(String[] args)
    {
        //Criacao do array de 10.000 posicoes
        int[] arrayInteiros = new int[10000];
        //Atribuindo os valores a cada posicao
        for(int i = 10000; i > 0; i--)
        {
            arrayInteiros[arrayInteiros.length - i] = i;
        }
    }
}
```

```
}
//inicio do algoritmo de ordenacao - implementado
boolean estaOrdenado = false;
//armazena o tempo de inicio da ordenacao
long inicio = System.currentTimeMillis();
while(!estaOrdenado)
{
    estaOrdenado = true;
    for(int i = 1; i < arrayInteiros.length; i++)
    {
        if(arrayInteiros[i-1] > arrayInteiros[i])
        {
            estaOrdenado = false;
            int aux = arrayInteiros[i];
            arrayInteiros[i] = arrayInteiros[i-1];
            arrayInteiros[i-1] = aux;
        }
    }
}
//armazena o tempo fim da ordenacao
long fim = System.currentTimeMillis();
//Imprime o tempo total de ordenacao
System.out.println("Tempo ordenar 1: "+(fim-inicio)+" ms");
//Atribuindo os valores a cada posicao
for(int i = 10000; i > 0; i--)
{
    arrayInteiros[arrayInteiros.length - i] = i;
}
//armazena o tempo de inicio da ordenacao
inicio = System.currentTimeMillis();
//Ordena utilizando a classe Arrays
Arrays.sort(arrayInteiros);
//armazena o tempo fim da ordenacao
fim = System.currentTimeMillis();
//Imprime o tempo total de ordenacao
System.out.println("Tempo ordenar 2: "+(fim-inicio)+" ms");
}
}
```

## Resultado da execução:

```
Tempo ordenar 1: 1013 ms
Tempo ordenar 2: 9 ms
```

O resultado desta execução mostra que o método de ordenação implementado pela classe Arrays foi, aproximadamente, 100 vezes mais eficiente que a implementação feita no exercício referenciado.

## **Exercício Resolvido - Pesquisa**

Implemente um programa que construa um array de inteiros (int) de tamanho 1.000.000 com valores atribuídos de acordo com o seu índice, procure o valor 555.000, entre os valores armazenados no vetor. Compare o tempo gasto utilizando o método "binarySearch" da classe Arrays e uma procura simples percorrendo todos os elementos do array até encontrar o valor desejado.

```
import java.util.Arrays;
public class PesquisarArray {
    public static void main(String[] args) {
        //Criação do array de 1.000.000 de posições
        int[] numerosInteiros = new int[1000000];
    }
}
```

# Curso Java Starter

```
//Definicao do numero a ser pesquisado
int numeroPesquisado = 555000;
//Preenchimento do Array
for(int i = 0; i < 1000000; i++)
    numerosInteiros[i] = i;
//Inicio da contagem do tempo
long inicio = System.currentTimeMillis();
//Percorrendo o array em busca do numero
for(int i = 0; i < 1000000; i++)
{
    if(numerosInteiros[i] == numeroPesquisado)
        break;
}
//fim da contagem do tempo
long fim = System.currentTimeMillis();
//Imprime o tempo total de pesquisa
System.out.println("Pesquisa 1: "+(fim-inicio)+" ms");
//Inicia a contagem do tempo
inicio = System.currentTimeMillis();
//Faz a pesquisa utilizando a classe Arrays
Arrays.binarySearch(numerosInteiros, numeroPesquisado);
//Fim da contagem do tempo
fim = System.currentTimeMillis();
//Imprime o tempo total de pesquisa
System.out.println("Pesquisa 2: "+(fim-inicio)+" ms");
}
}
```

Possível saída no console:

```
Pesquisa 1: 94 ms
Pesquisa 2: 1 ms
```

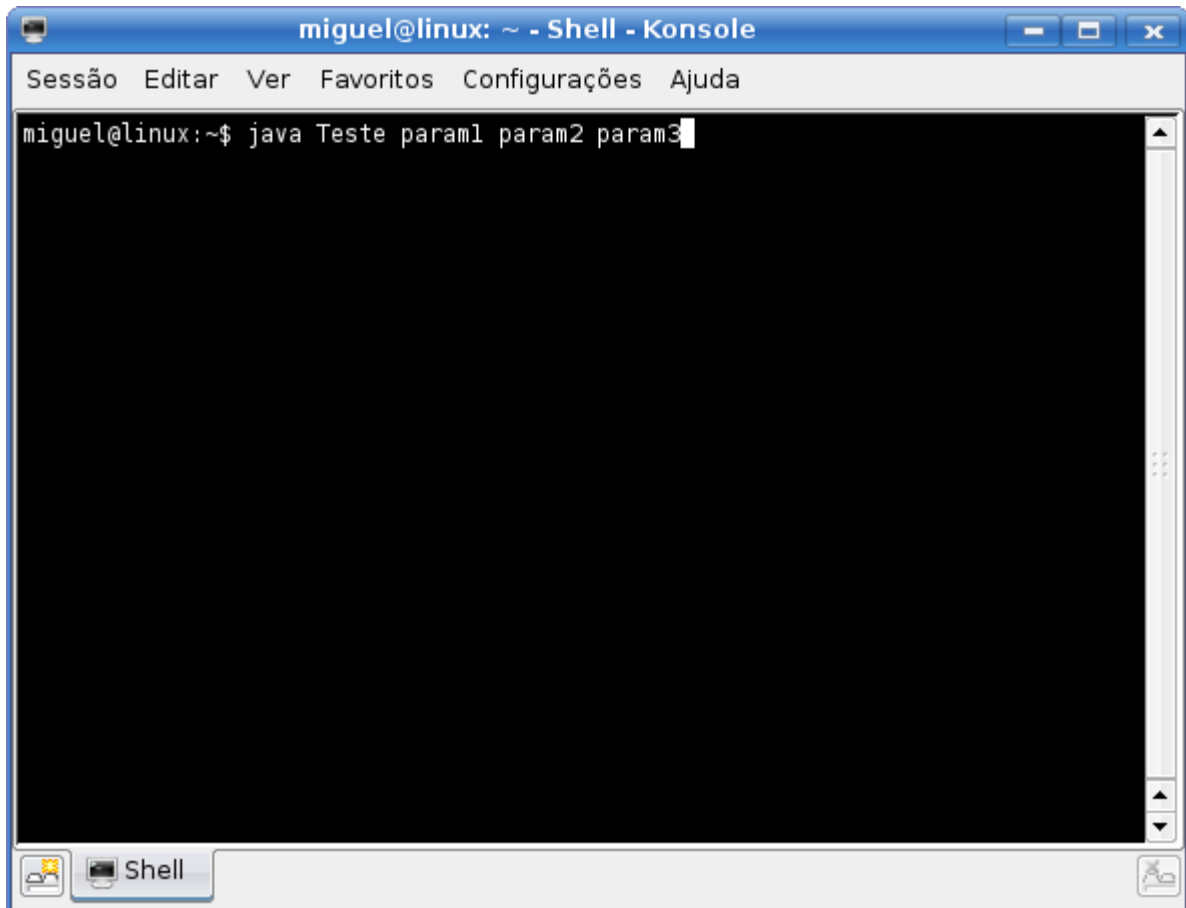
Como era esperado, a pesquisa utilizando a classe Arrays foi muito mais eficiente que a pesquisa percorrendo todos os elementos.

## Entrada de dados – linha de comando

Da mesma forma que outras linguagens de programação, em Java é possível passar argumentos através da linha de comando. Nestes casos nós invocamos o interpretador e na seqüencia passamos os parâmetros para nossa aplicação.

No exemplo a seguir a classe Teste está recebendo 3 parâmetros (param1, param2 e param3) durante a execução.

# Curso Java Starter



Estes parâmetros são recebidos através do método main. Observe abaixo a assinatura do método, os parâmetros são agrupados em um vetor de String, ou seja, uma das portas de entrada de informação externa para programas Java é o próprio método main.

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

É importante compreender que todos os parâmetros são recebidos pelo método main como String, para serem tratados como números eles devem ser convertidos, esta conversão será abordada no Módulo 7.

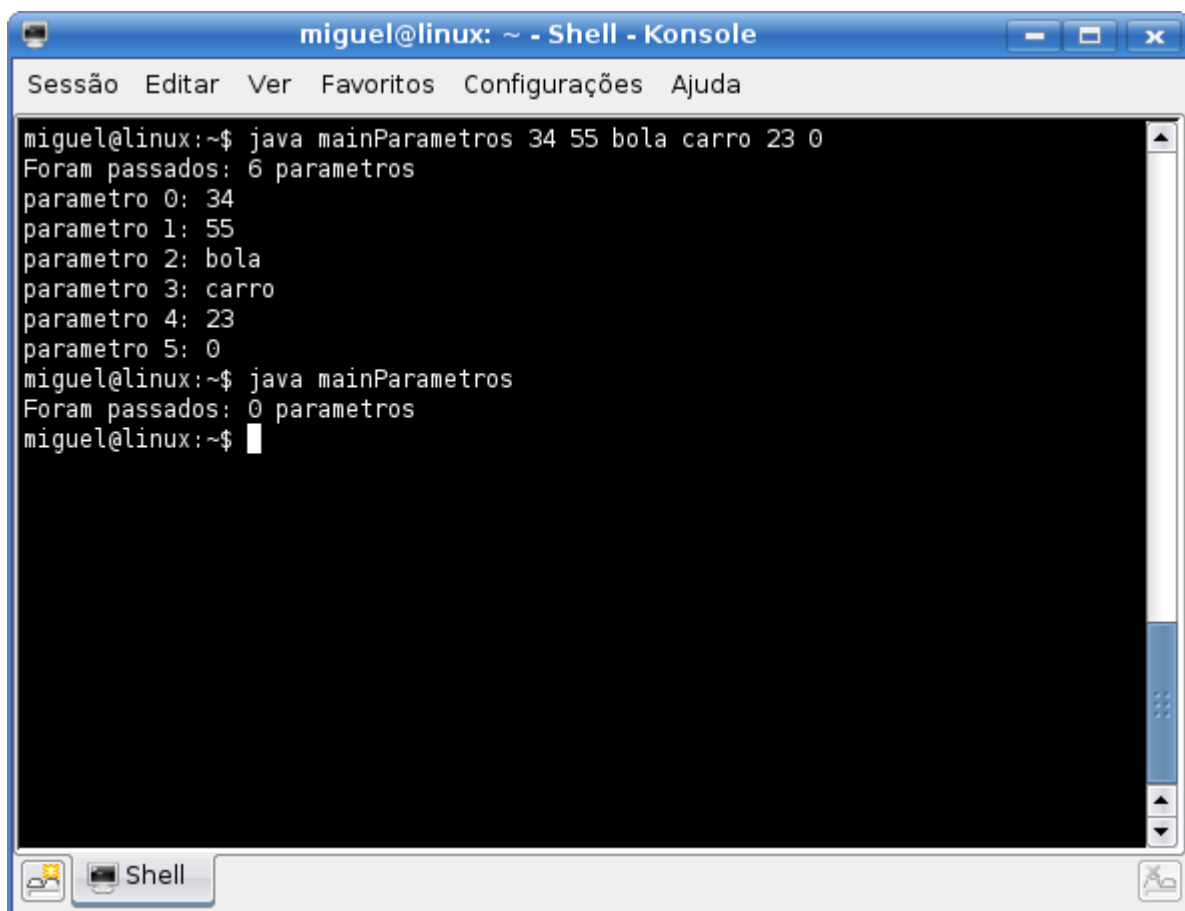
## **Exercício Resolvido**

Implemente um programa Java que imprima e conte quantos parâmetros foram passados durante a sua interpretação, execute este programa passando 6 parâmetros quaisquer e , após, execute novamente sem passar parâmetros.

# Curso Java Starter

```
public class mainParametros {
    public static void main(String[] parametros) {
        if(parametros != null)
        {
            System.out.println("Foram passados: "+parametros.length+" parametros");
            for(int i = 0; i < parametros.length; i++)
                System.out.println("parametro "+i+": "+parametros[i]);
        }
    }
}
```

Abaixo temos a imagem com a execução deste programa, primeiro foram passados 6 parâmetros e na seqüência nenhum.



```
miguel@linux: ~ - Shell - Konsole
Sessão Editar Ver Favoritos Configurações Ajuda
miguel@linux:~$ java mainParametros 34 55 bola carro 23 0
Foram passados: 6 parametros
parametro 0: 34
parametro 1: 55
parametro 2: bola
parametro 3: carro
parametro 4: 23
parametro 5: 0
miguel@linux:~$ java mainParametros
Foram passados: 0 parametros
miguel@linux:~$
```

Agora nós já sabemos como enviar parâmetros através do prompt de duas formas, uma através do método main e outra, apresentada no Módulo 4, através do uso de um Scanner. Mas nós podemos fazer estas mesmas funções de forma um pouco mais elaborada utilizando interfaces gráficas. A próxima seção irá descrever como isto pode ser realizado.

### Entrada de dados – interface gráfica

Aqui nós iremos utilizar a biblioteca Swing que é responsável pela geração dos artefatos gráficos em Java mas nós não iremos nos aprofundar nesta biblioteca pois este assunto será abordado em um módulo dedicado apenas a isto.

A entrada de dados utilizando prompt pode, dependendo da situação, deixar o seu programa menos “palatável” para usuários iniciantes. Para evitar esta situação podemos efetuar a entrada de dados utilizando a classe **JOptionPane**.

A classe **JOptionPane** torna simples a implementação de diálogos que solicitam a entrada de dados ou mesmo apenas informam alguma coisa. Os principais métodos desta classe estão relacionados a seguir.

Método	Descrição
showConfirmDialog	Utilizado para confirmações, diálogos cuja resposta seja do tipo sim, não ou cancela.
showInputDialog	Utilizado para entrada de dados pelo usuário.
showMessageDialog	Informa ao usuário alguma coisa
showOptionDialog	Uma mistura dos três outros métodos, ou seja, pode ser utilizado para informar, para entrada de dados e ainda confirmações.

Vamos ao exemplo prático para melhor compreensão, neste exemplo serão utilizados os métodos **showInputDialog** e **showMessageDialog**.

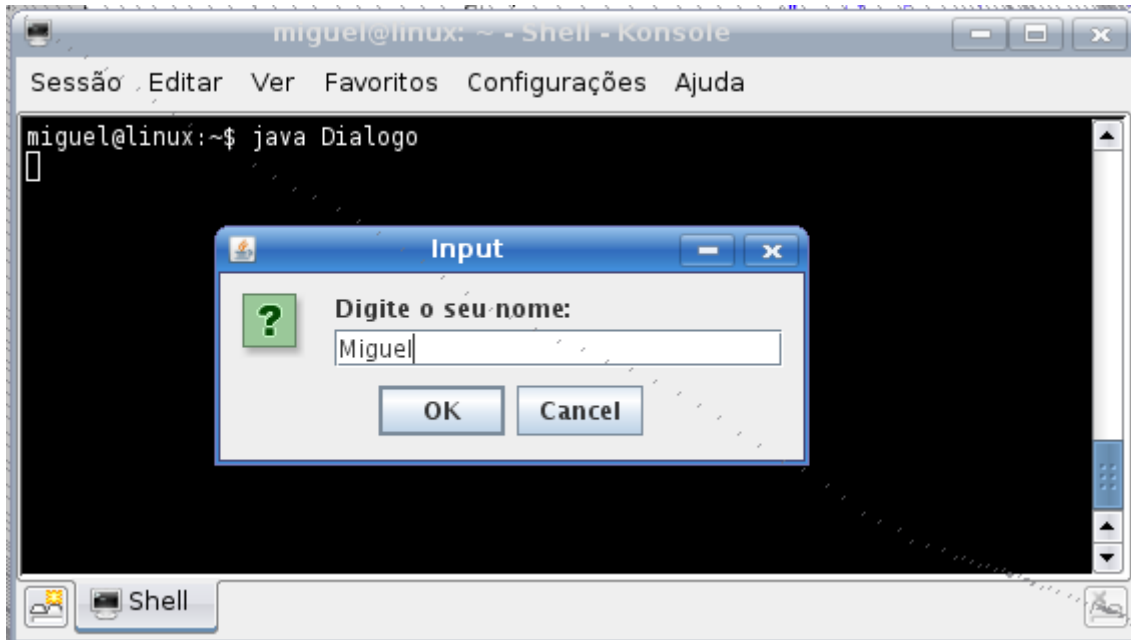
```
//Importando classe do Swing
import javax.swing.JOptionPane;

public class Dialogo {
    public static void main(String[] args) {
        String nome;
        //Este comando mostra um dialogo que solicita entrada de dados
        nome = JOptionPane.showInputDialog("Digite o seu nome: ");
        String mensagem = nome+" está fazendo o curso Java Iniciante";
        //Este comando mostra um dialogo que apenas exhibe a mensagem
        JOptionPane.showMessageDialog(null, mensagem);
    }
}
```

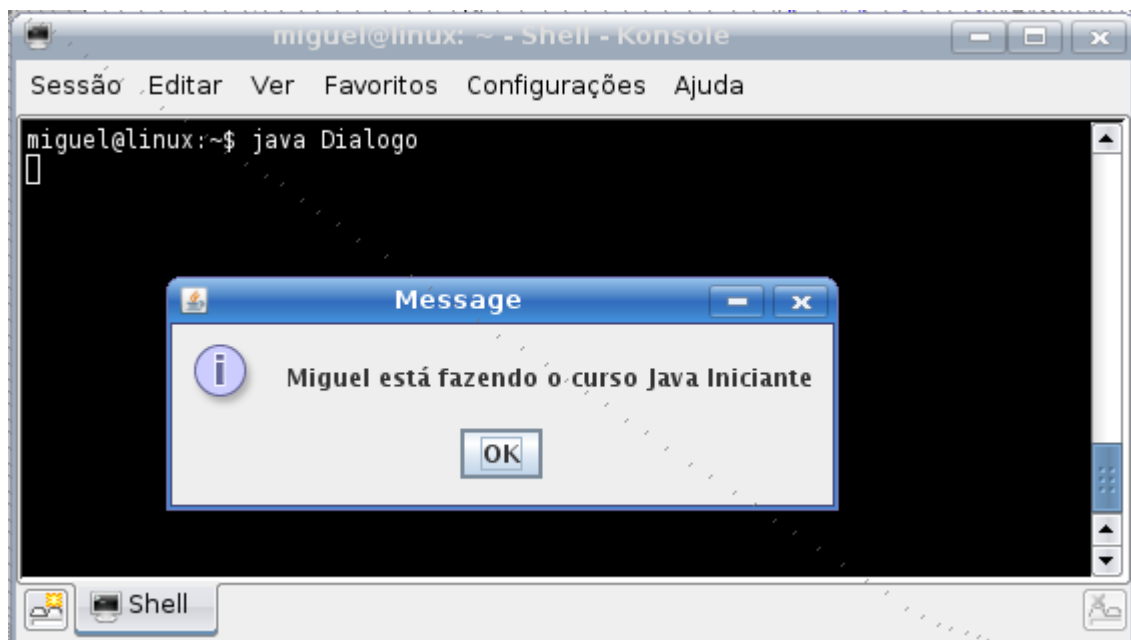
A seguir temos as telas demonstrando a execução deste programa no prompt.

## Curso Java Starter

Entrada de dados no diálogo.



Exibição da mensagem de conclusão do programa.





### **Exercícios**

Aprenda com quem também está aprendendo, veja e compartilhe as suas respostas no nosso [Fórum](#):

[Exercícios – Módulo 03 – Vetores \(Arrays\) e Entrada de Dados](#)

- 1) Crie um programa que percorra um array de 10 posições e imprima o seu conteúdo.
- 2) Escreva um programa que some todos os valores contidos em um array de inteiros e calcule a média.
- 3) Desenvolva um programa que dado um array de 10 números inteiros multiplique o primeiro elemento pelo seguinte, o resultado deve então ser multiplicado pelo próximo elemento até que todos os elementos sejam percorridos. Imprima o valor final.
- 4) Faça um programa que inverta as posições de um array com 10 elementos.
- 5) Dados dois vetores quaisquer de booleanos compare-os e informe se ambos são idênticos, para serem idênticos os vetores devem possuir o mesmo tamanho e os mesmos elementos em cada posição.
- 6) Faça um programa que calcule o produto escalar de dois vetores de double, isto é, multiplique cada elemento pelo seu respectivo no outro vetor e some os resultados.
- 7) Utilizando a classe Scanner, já apresentada em módulos anteriores, faça um programa que permita o usuário efetuar a entrada de 10 números quaisquer. Ordene estes números utilizando a classe Arrays e imprima o resultado.
- 8) Faça um programa que receba um parâmetro qualquer através do método main e informe ao usuário através de uma caixa de diálogo qual foi o parâmetro passado.
- 9) Crie um programa que mostre um diálogo, informando que não existem parâmetros, caso não seja passado nenhum parâmetro através da linha de comando.
- 10) Utilizando a classe Scanner faça um programa que solicite a entrada de 10 números inteiros, logo após solicite a entrada de mais 5 números inteiros compare os números obtidos e verifique se o segundo vetor está contido dentro

do primeiro.

- 11) Utilizando a classe Scanner faça um programa que solicite a entrada de 10 números inteiros, verifique a frequência de ocorrência de cada número no vetor.
- 12) Crie um programa que receba como parâmetro de entrada 16 valores, coloque-os em uma matriz 4x4 e faça a transposição dos valores.
- 13) Desenvolva um programa que receba 10 parâmetros por linha de comando, inverta todas as posições e imprima o resultado.
- 14) Desenvolva um programa, utilizando a classe Scanner, que obtenha 10 valores de entrada e verifique se eles formam uma progressão aritmética.
- 15) Desenvolva um programa, utilizando a classe Scanner, que obtenha 10 valores de entrada e identifique o maior valor, o menor e a diferença entre os valores.
- 16) Faça um algoritmo para ordenar um vetor da seguinte forma, o algoritmo identifica o maior valor e o coloca na última posição livre de um outro vetor. Na próxima iteração este valor é descartado, repetindo até não sobra mais valores.