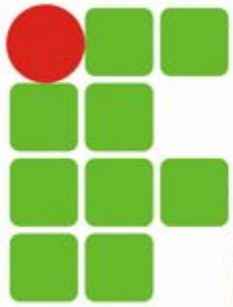


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
FLUMINENSE

# ARRAYS

**Disciplina**
**PROGRAMAÇÃO ORIENTADA A OBJETOS**

**Prof. Roberta B Tôrres**

# ARRAYS

ARRAYS são estruturas de dados consistindo em itens de dados do mesmo tipo relacionados.

Java trata array como um objeto.

O array pode conter tipos primitivos ou tipos referências (inclusive outro array, claro, é um objeto também);

Os elementos são acessados através dos índices, inicia com 0 (zero).

# ARRAYS - Utilidade

Suponha a criação de vários objetos CONTA, conforme ilustrado abaixo:

```
double saldoDaConta1 = conta1.getSaldo();
double saldoDaConta2 = conta2.getSaldo();
double saldoDaConta3 = conta3.getSaldo();
double saldoDaConta4 = conta4.getSaldo();
```

Imagine 1000 contas, logo declarar 1000 objetos, o que se torna um problema.

**SOLUÇÃO**: Para facilitar esse tipo de caso basta declarar um **vetor (array)** de doubles: **double[] saldosDasContas;**

**double[]** é um tipo. Uma array é sempre um objeto, portanto, a variável **saldosDasContas** é uma referência.
Vamos precisar criar um objeto para poder usar o Array. Como criamos o objeto-array?

# ARRAYS - Declaração

Para declarar um vetor, por tratar-se de um objeto, é necessário chamar o construtor (operador “new”) e passar a quantidade de elementos do array. Ao definir um tamanho esse não poderá mudar.

**saldosDasContas= new double[10];**

O código acima cria um array de double de 10 posições. Podemos assim acessar as posições do array:

**saldosDasContas[5] = conta5.getSaldo();**

# ARRAYS - Declaração

**ATENÇÃO**: Em Java, um array é um objeto. Assim, é um erro comum tentar declarar um array, em Java, definindo seu tamanho na declaração. Então, tenha atenção.

**Por exemplo:**

- float salarios[10]; << erro >>

**Correto:**

- float salarios[] = new salarios[10];

Os colchetes para demarcar a criação do array, podem se posicionar das seguintes formas:

```
int[] ages; ou int ages[];
```

```java
import java.util.Scanner;
// Detalhes sobre a API Scanner - Consulte o link abaixo
// https://www.devmedia.com.br/como-funciona-a-classe-scanner-do-java/28448
//forma simples de fazer entrada de dados via Console

public class Exemplo01 {
    private static Scanner in;

    public static void main(String[] args)
    {
        int num[] = new int[5];
        in = new Scanner(System.in);

        for(int i=0; i < num.length; i++)
        {   System.out.print("Informe um numero:");
            num[i] = in.nextInt();
        }

        System.out.print("Numeros:");
        for(int i= num.length-1; i >=0; i--)
        {   System.out.print(num[i] + " - ");   }

        System.out.print("Numeros:");
        //comando FOR aprimorado - for(parametro : objeto)

        for(int i : num )
        {   System.out.print(i + " - ");    }
    }
}
```

No Tópico Exemplos de Arrays, execute o arquivo ***Exemplo01.java***.

```java
import java.util.Scanner;

public class GameCartas {
    private static Scanner in;

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        String[] naipes = {"Ouro", "Copas", "Espada", "Paus"};
        String[] baralho = {"1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "Valente", "Dama", "Rei"};
        in = new Scanner(System.in);

        System.out.print("Quantas cartas deseja mostrar? ");
        int qtde_cartas = in.nextInt();

        for(int i=0; i < qtde_cartas; i++)
        {   // Sorteia um naipe e uma carta do baralho
            String naipe = naipes[(int) (Math.random() * 4)];
            String carta = baralho[(int) (Math.random()*baralho.length)];

            System.out.println("\n Carta Sorteada = "+naipe+ " - " + carta );
        }

        System.out.print("\n *** FIM DO GAME ***");
    }
}
```

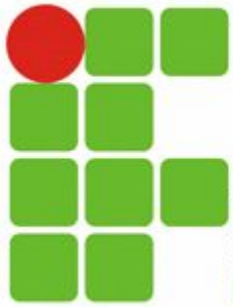
No Tópico Exemplos de Arrays, execute o arquivo **GameCartas.*java***.

Exemplo do Livro Java Como Programar 8ª Edição - página 198.

```
1  // Figura 7.8: StudentPoll.java
2  // Programa de análise de enquete.
3
4  public class StudentPoll
5  {
6     public static void main( String[] args )
7     {
8        // array de respostas da enquete
9        int[] responses = { 1, 2, 6, 4, 8, 5, 9, 7, 8, 10, 1, 6, 3, 8, 6,
10          10, 3, 8, 2, 7, 6, 5, 7, 6, 8, 6, 7, 5, 6, 6, 5, 6, 7, 5, 6,
11          4, 8, 6, 8, 10 };
12       int[] frequency = new int[ 11 ]; // array de contadores de frequência
13
14       // para cada resposta, seleciona elemento da resposta e
15       // usa esse valor como índice de frequência para determinar o elemento a incrementar
16       for ( int answer = 0; answer < responses.length; answer++ )
17          ++frequency[ responses[ answer ] ];
18
19       System.out.printf( "%s%10s", "Rating", "Frequency" );
20
21       // gera saída do valor de cada elemento do array
22       for ( int rating = 1; rating < frequency.length; rating++ )
23          System.out.printf( "%d%10d", rating, frequency[ rating ] );
24    } // fim de main
25 } // fim da classe StudentPoll
```

```
Rating Frequency
     1         2
     2         2
     3         2
     4         2
     5         5
     6        11
     7         5
     8         7
     9         1
    10         3
```

Programa de análise de enquete.



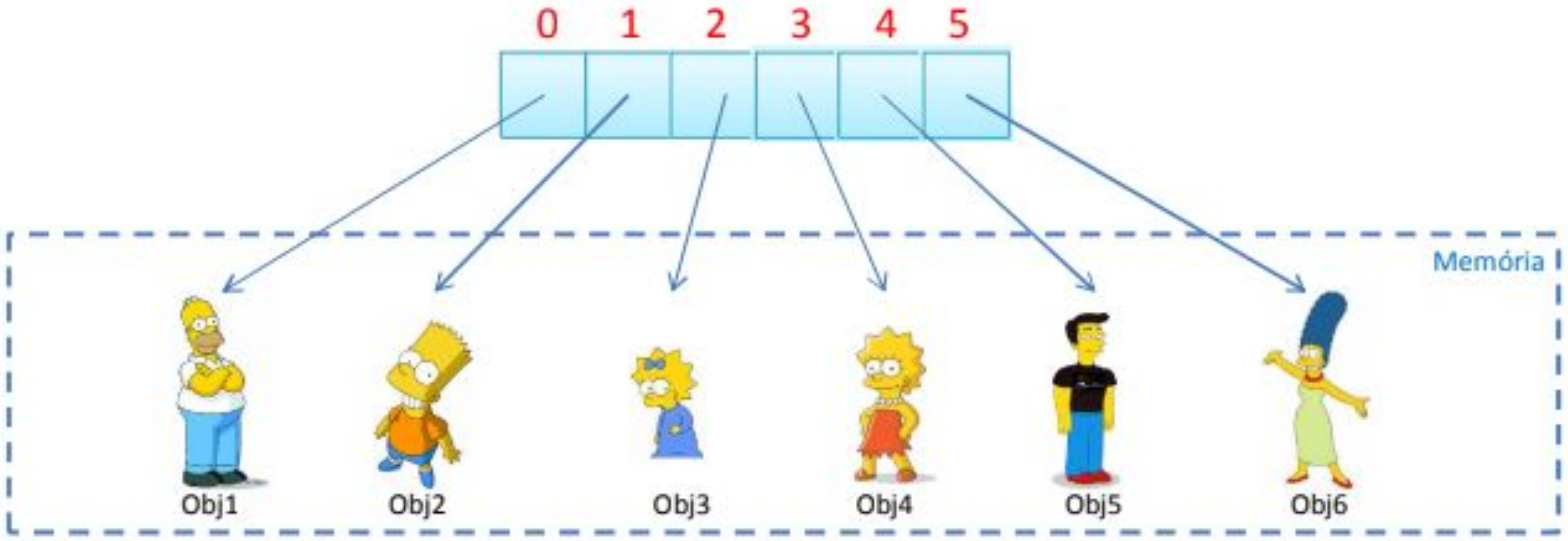
INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
FLUMINENSE

# ARRAYS DE REFERÊNCIA

# Arrays de Referência

É comum ouvir **"array de objetos"**. Quando criamos um array de alguma classe, ela possui referências. O objeto, como sempre, está na memória principal e no array só ficam guardadas as **referências** (endereços).

- Os vetores não guardam os objetos (ou tipos primitivos) e sim apenas uma referência para estes;
- Por exemplo: Um vetor para guardar 06 pessoas.



# Arrays de Referência

No código abaixo, quantas contas foram criadas aqui? Na verdade, **nenhuma**. Foram reservados 10 espaços para guardar uma referência a uma ContaCorrente. Por enquanto, eles se referenciam para lugar nenhum (*null*).

```
ContaCorrente[] minhasContas;
minhasContas = new ContaCorrente[10];
```

Se você tentar: ***System.out.println(minhasContas[0].getSaldo());***

Um erro ocorrerá! Pois, na 1ª posição da array, não há uma referência para uma conta.

# Arrays de Referência

Para “**povoar**” o array ***minhasContas*** faça:

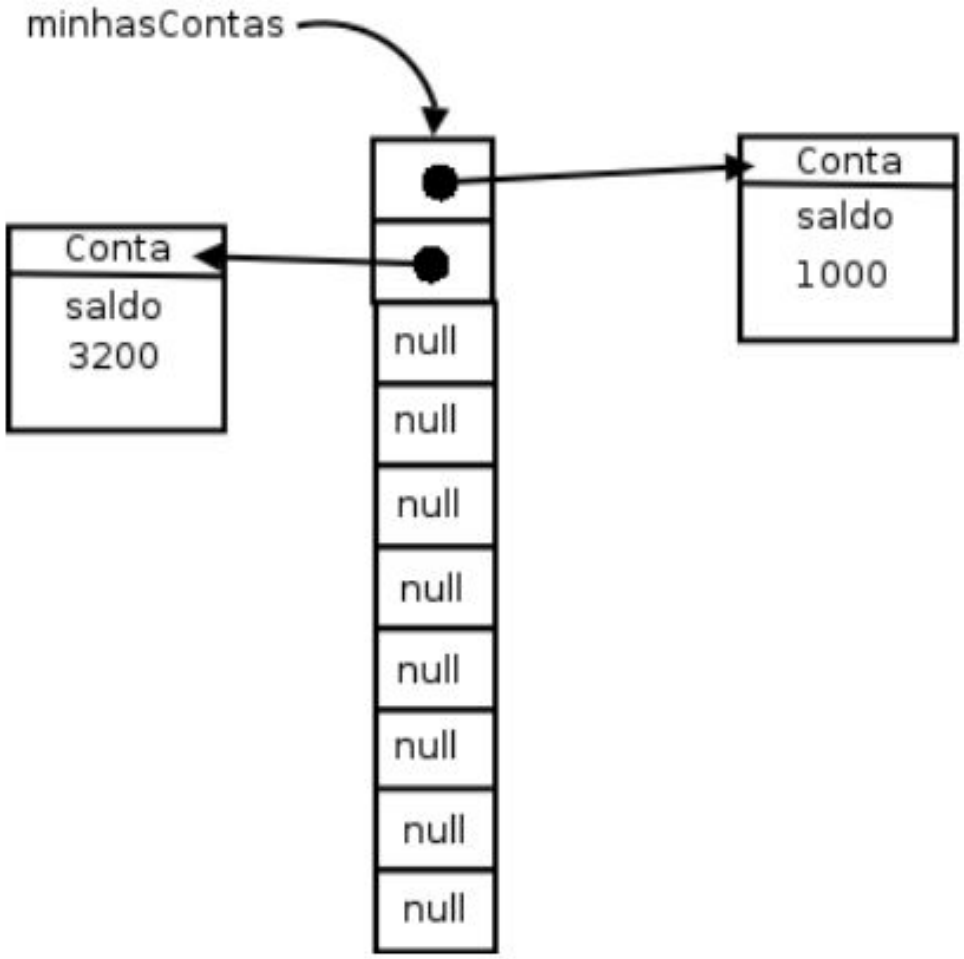
```
ContaCorrente contaNova = new ContaCorrente();
contaNova.deposita(1000.0);
minhasContas[0] = contaNova;
```

Ou pode fazer de forma direta:

```
minhasContas[1] = new ContaCorrente();
minhasContas[1].deposita(3200.0);
```

# Arrays de Referência

Um array de tipos primitivos guarda valores. Um array de objetos guarda referências.



# Arrays de Referência

E se quiser guardar tanto **Conta Corrente** quanto **Conta Poupança**?

Um array de **Conta Corrente** só consegue guardar objetos do mesmo tipo. Se quisermos guardar os dois tipos de conta, devemos criar um array de **Conta**!

```
Conta[] minhasContas = new Conta[10];
minhasContas[0] = new ContaCorrente();
minhasContas[1] = new ContaPoupanca();
```

Perceba que não estamos criando um objeto do tipo Conta, que é abstrata, estamos criando 10 espaços que guardam referências para qualquer tipo de conta.

# Brincando com Array

```
public abstract class Conta {
private float saldo;

public void deposito(float valor)
{   this.saldo = this.saldo + valor; }

public void saque(float valor)
{   this.saldo = this.saldo - valor; }

public float saldo()
{   return this.saldo; }

}
```

No Tópico **Exemplos de Arrays**, faça download do arquivo **Vector_Contas.zip**, incorpore o projeto no Eclipse e execute a classe **Principal.java** para ver este exemplo sendo executado.

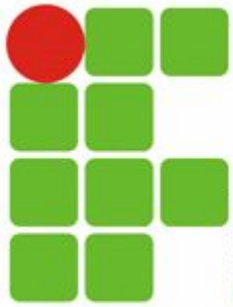
```
public class ContaCorrente extends Conta {

}
```

```
public class ContaPoupanca extends Conta{

}
```

# Brincando com Array

```
import java.util.Scanner;

public class Principal {
    private static Scanner in;

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        Conta[] contas_bancarias = new Conta[5];
        in = new Scanner(System.in);
        int i = 1;
            for(Conta c : contas_bancarias)
            {   System.out.print("Qual conta ( C- corrente | P - poupanca)?");
                String opcao = in.nextLine();
                if (opcao.equalsIgnoreCase("P"))
                {  c = new ContaPoupanca();
                   c.deposito(i * 100);
                }
                else
                {  c = new ContaCorrente();
                   c.deposito(i * 100);
                }

                System.out.println(c.saldo());
                System.out.println(c.getClass());
                System.out.println(c.getClass().getSuperclass());
                i++;

            }
    }
}
```

Classe **Principal.java** que manipula um array de referência para instâncias da classe Conta.



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
FLUMINENSE

# PARÂMETROS DO TIPO ARRAY

# Arrays como Parâmetro

Exemplo de como usar um array como parâmetro:

```
public void imprimeArray(int[] array) {
    for (int i = 0; i < array.length; i++) {
        System.out.println(array[i]);
    }
}
```

**Arrays não podem mudar de tamanho**

A partir do momento que uma array foi criada, ela **não pode** mudar de tamanho.

Se você precisar de mais espaço, será necessário criar uma nova array e, antes de se referir ela, copie os elementos da array velha.

# Arrays como Parâmetro

O método `main` recebe uma **array de Strings** como argumento. Essa array é passada pelo usuário quando ele invoca o programa:

```
$ java Teste argumento1 outro maisoutro
```

E nossa classe:

```
public class Teste {
    public static void main (String[] args) {
        for(String argumento: args) {
            System.out.println(argumento);
        }
    }
}
```

Isso imprimirá:

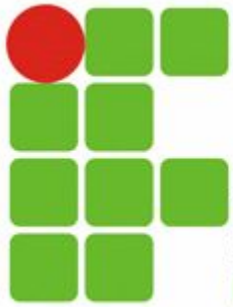
```
argumento1
outro
maisoutro
```

# Arrays como Parâmetro

**CURIOSIDADE**

No Eclipse, para carregar o parâmetro **args** do método ***main()*** faça assim:

- Selecione a classe que contem o método ***main()*** e clique com o botão direito do mouse.
- Selecione a opção ***Run as*** - ***Run Configurations***.
- Na tela apresentada, clique na aba **Arguments**. Na caixa de texto **Program arguments** digite as palavras.
  - O espaço em branco entre as palavras indica que é um novo argumento.
  - Se quiser que uma frase seja considerada como um único argumento, digite a mesma entre aspas.



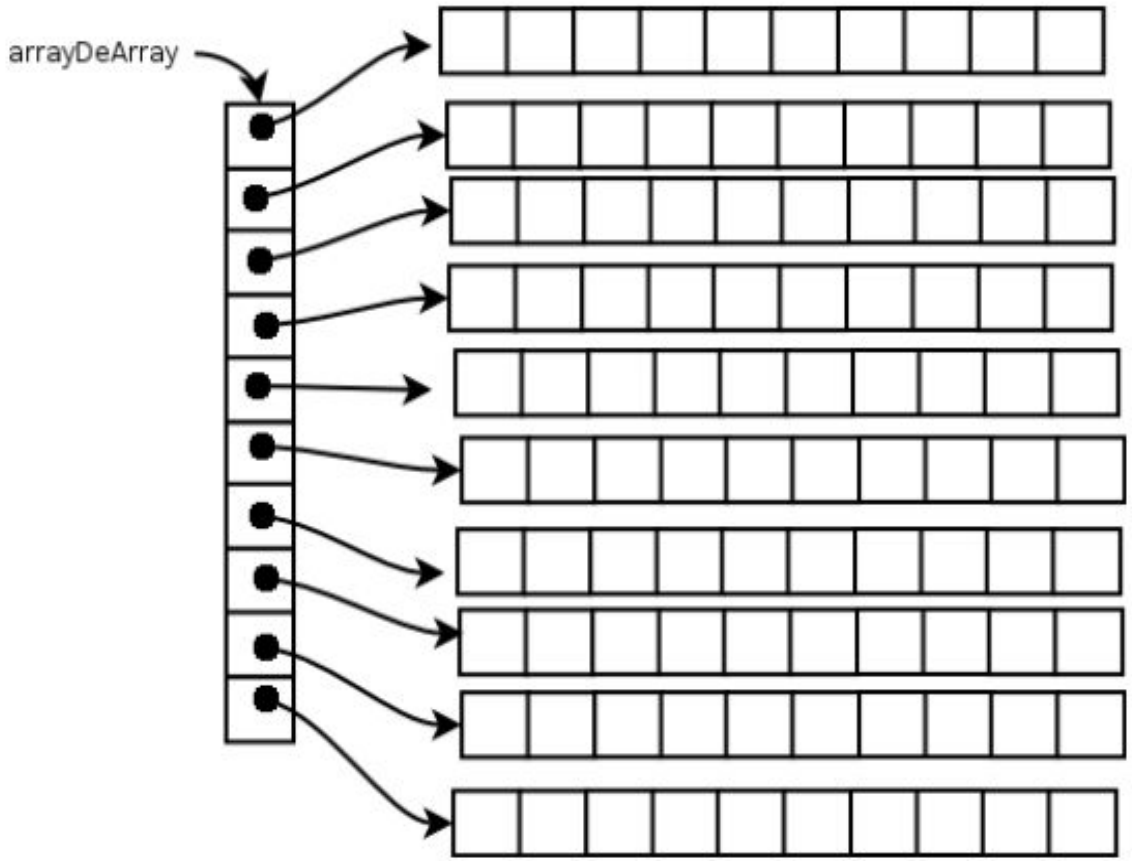
INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
FLUMINENSE

# ARRAYS BIDIMENSIONAIS

# Arrays Bidimensionais

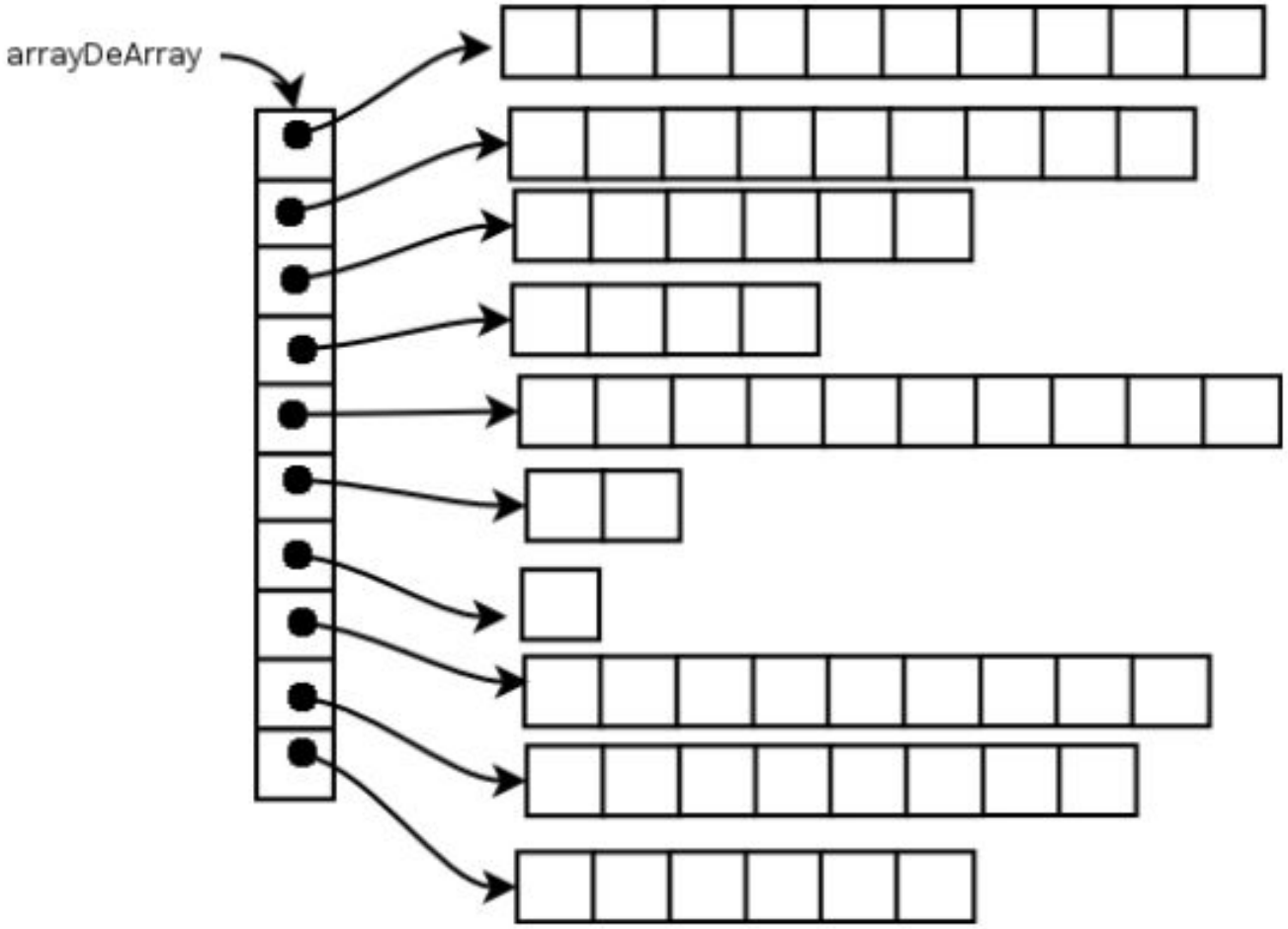
O Java não suporta arrays multidimensionais diretamente.

O array bidimensional em Java é um array unidimensional cujos seus elementos também são arrays unidimensionais.



# Arrays Bidimensionais

Em Java, o array bidimensional não precisa ser retangular, isto é, cada linha pode ter um número diferente de colunas.



# Arrays Bidimensionais

Por exemplo: o array **b** é um array bidimensional em que cada linha tem um número diferente de colunas.

```
int[][] b = new int[ 2 ][ ];   // cria 2 linhas
b[ 0 ] = new int[ 5 ]; // cria 5 colunas para a linha 0
b[ 1 ] = new int[ 3 ]; // cria 3 colunas para a linha 1
```

Lembre também que os programas podem utilizar variáveis para especificar o tamanho do array, já que o comando **new** cria o array em tempo de execução e não em tempo de compilação.

# Exemplo

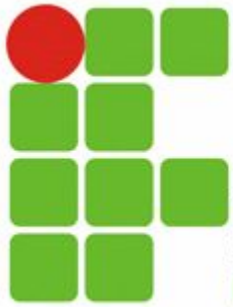
```
public class InitArray
{
   // cria e gera saida de arrays bidimensionais
   public static void main( String[] args )
   {
      int[][] array1 = { { 1, 2, 3 }, { 4, 5, 6 } };
      int[][] array2 = { { 1, 2 }, { 3 }, { 4, 5, 6 } };

      System.out.println( "Values in array1 by row are" );
      outputArray( array1 ); // exibe array1 por linha

      System.out.println( "\nValues in array2 by row are" );
      outputArray( array2 ); // exibe array2 por linha
   } // fim de main

   // gera saida de linhas e colunas de um array bidimensional
   public static void outputArray(int[][] array )
   {
      // faz um loop pelas linhas do array
      for ( int row = 0; row < array.length; row++ )
      {
         // faz um loop pelas colunas da linha atual
         for ( int column = 0; column < array[ row ].length; column++ )
            System.out.printf( "%d  ", array[ row ][ column ] );

         System.out.println(); // inicia nova linha de saida
      } // fim do for externo
   } // fim do método outputArray
} // fim da classe InitArray
```

```
Values in array1 by row are
1  2  3
4  5  6

Values in array2 by row are
1  2
3
4  5  6
```

# Arrays Bidimensionais

```
// 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"}
                   };
```

https://pt.slideshare.net/Portal_do_Estudante_Java/arrays-em-java



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
FLUMINENSE

# Classe Arrays - Métodos (java.util.Arrays)

# Métodos da classe Arrays

A classe **Arrays** possui alguns métodos estáticos úteis, tais como:

- **sort**: ordenação dos elementos do array em ordem crescente.
- **binarySearch**: usa busca binária para localizar um valor específico no array, retornando seu índice. Se não localizar, retorna um índice negativo. Este método deve receber um array já ordenado.

```
import java.util.Arrays;

public class Exemplo05 {

    public static void main(String[] args) {

        int[] vetor = {8, 4, 9, 7, 3};
        Arrays.sort(vetor);

        System.out.println("\n Elementos ordenados: ");
        for (int i : vetor)
            System.out.printf(" - " + i);

        System.out.println("\n Tem numero 8? Índice = " + Arrays.binarySearch(vetor, 8));
        System.out.println("\n Tem numero 10? Índice = " + Arrays.binarySearch(vetor, 10));
    }
}
```