Programação

Aspectos Fundamentais

Por Carla Macedo

atec

Convenção para Nomes

- **Java, como C/C++, distingue letras maiúsculas e minúsculas**
  Exemplo: escola é diferente de Escola
- Nomes de variáveis, classes e métodos devem começar por **letras**, **$** ou _
- Nomes de classes devem começar com maiúscula
  Exemplo: class Bemvindo
- Nomes de variáveis devem começar com minúsculas
  Exemplo: int peso;
- Nomes dos métodos são verbos e devem começar com minúscula e depois usam maiúsculas: Exemplo: alteraPeso
- **O nome do ficheiro.java de ser igual ao nome da classe**
  (obrigatório para classes públicas)

Princípios básicos

- As classes, métodos ou blocos de código **são sempre** limitados por um **abrir** (**{**) e **fechar** (**}**).
- Um comando deve sempre ser finalizado por um **ponto e vírgula** (**;**)
- Dois tipos de comentário : **//**Comentário de uma linha e **/*** Comentário de Várias ***/**
- Nomes de variáveis, classes e métodos devem sempre começar por **letras**, **$** ou _

Comentários

Exemplo0202.java *

```
1  public class Exemplo0202
2  {
3    public static void main ( String args [] )
4    {
5      int x = 10, y = 20; // declaração de variáveis do tipo inteiro
6      double dolar = 2.62;
7      /* As linhas seguintes enviam o conteúdo das
8      variáveis para a tela */
9      System.out.println(x);
10     System.out.println(y);
11     System.out.println(dolar);
12
13     /** Exemplo0202:
14     Essa classe demonstra a utilização de variáveis em uma classe
15     em Java. São declaradas variáveis do tipo int e double.
16     O exemplo também demonstra como imprimir o conteúdo das variáveis
17     */
18   }
19 }
20
```

Palavras Chave

Estas palavras-chave não podem ser usadas como identificadores (variáveis, classes ou métodos) em programas.

abstract	do	implements	package	throw
boolena	double	import	private	throws
break	else	*inner	protected	transient
byte	extends	instanceof	public	try
case	final	int	*rest	*var
*cast	finally	interface	return	void
*catch	float	long	short	volatile
char	for	native	static	while
class	*future	new	super	
*const	generic	null	switch	
continue	*goto	operator	synchronized	
default	if	*outer	this	

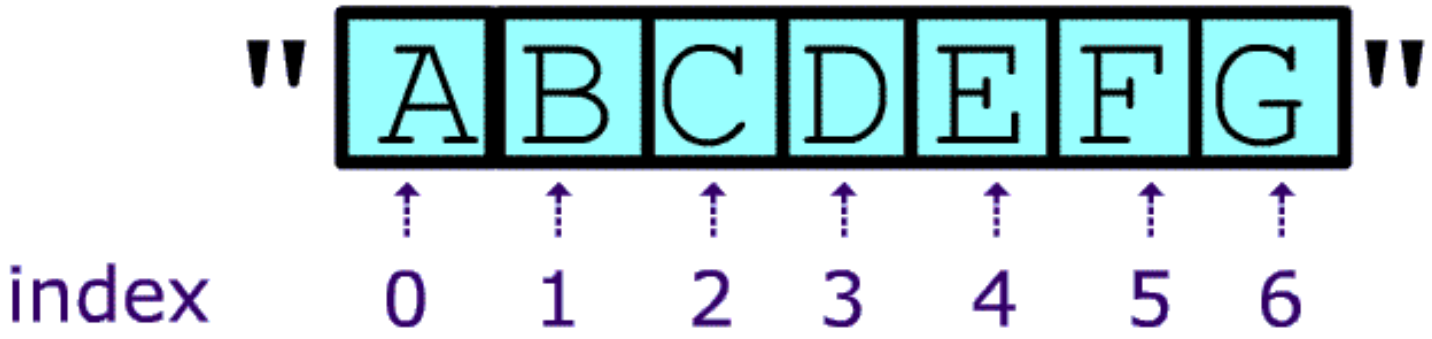
Tipos de Dados

Java possui 8 tipos primitivos onde 6 são tipos numéricos, um tipo *char* para caracteres e um tipo *booleano*.

Nome do Tipo	*Tamanho*	*Variação*
long	8 bytes	-9.223.372.036.854.775.808L até -9.223.372.036.854.775.807L
int	4 bytes	-2.147.483.648 até 2.147.483.647
short	2 bytes	-32.768 até 32.767
byte	1 byte	-128 até 127
double	8 bytes	+/- 1,79759313486231570E+308 (15 dígitos significativos)
float	4 bytes	+/- 3,40282347E+38 (7 dígitos significativos)
char	2 bytes	65.536 caracteres possíveis
boolean	1 bit	verdadeiro ou falso

Tipos de Dados - String

Uma variável do tipo string contem uma sequencia de caracteres. Cada carácter C que compõe a string pode ser referenciado pelo seu índice na string.



Tipos de Dados - String

Métodos String: (**s** variável do tipo String)

- s.charAt(i) - retorna carácter na posição **i** de **s**
- s.length() - retorna o comprimento de **s**
- s.toUpperCase() - retorna **s** em letra grande
- s.toLowerCase() - retorna **s** em letra pequena
- s.equals(“xx") - compara String **s** com “XX”

Nota: existem mais métodos, ver documentação Java.

Tipos de Dados - String

ss.java

```
1  class ss
2  {
3      public static void main(String args[])
4      {
5          String s="INFORmatica";
6
7          System.out.println("String: " + s);
8          System.out.println("String em maiusculas: " + s.toUpperCase());
9          System.out.println("String em maiusculas: " + s.toLowerCase());
10         System.out.println("Comprimento String: " + s.length());
11         System.out.println("Primeio carater String: " + s.charAt(0));
12         System.out.println("Segundo carater String: " + s.charAt(2));
13         System.out.println("Terceiro carater String: " + s.charAt(3));
14     }
15 }
```

Concatenação - String

É uma operação básica para combinar strings, na qual toma uma string **P** e uma string **Q** e as combina numa nova string denotada **P+Q**, que contem todos os caracteres de **P** seguidos por todos os caracteres de **Q**.

String S = “Bom” + “Dia”

S= “BomDIA”

Operadores

Operadores são caracteres especiais usados para instruir o compilador a executar uma determinada operação num operando.

Operador de Atribuição

Operador	Nome	Exemplo

=	Atribuição	int var1 = 0, var2 = 0;
		var2 = var1 + 10;
		var1 = var2 = 50;

Operadores Aritméticos

Operador	Nome	Exemplo	Resultado
+	Adição	23+29	52
-	Subtração	29-23	6
*	Multiplicação	0.5 * salario	
/	Divisão	100/50	2
%	Módulo (resto da divisão)	20%3	2

Operador ++ incrementa de um.

Ex : int var1 = 4; var1++; //var1 = 5

Operador -- decremento de um

Ex : int var1 = 4; var1--; //var1 = 3

Operadores Aritméticos

Exemplo0203.java *

```
1  class Exemplo0203
2  {
3    public static void main (String args[])
4    {
5      // declaração e inicialização de variáveis
6      int x = 10;       float y = 3;
7      // várias operações com as variáveis
8      System.out.println("X = "+ x);
9      System.out.println("Y = "+ y);
10     System.out.println("-X = "+(-x));
11     System.out.println("X/Y = "+(x/y));
12     System.out.println("Resto de X por Y = "+ (x%y)); // resulta 1
13     System.out.println("Inteiro de X por Y = "+ (int)(x/y)); // resulta 3
14     System.out.println("X + 1 = "+ (++x)); // resulta 11
15   }
16 }
17
```

Precedência de Operadores

Operadores	Associatividade
()	esquerda para a direita
! - ++ --	direita para a esquerda
* / %	esquerda para a direita
+ -	esquerda para a direita
<< >>	esquerda para a direita
< <= > >=	esquerda para a direita
== !=	esquerda para a direita
&	esquerda para a direita
^	esquerda para a direita
|	esquerda para a direita
&&	esquerda para a direita
||	esquerda para a direita
?:	direita para a esquerda
= += -= *= /= %= &= ^= |= <<= >>=	direita para a esquerda

Operadores relacionais

Operador	Nome	Exemplo	Resultado
= =	Igual	x = = 10	
!=	Diferente	3 != 2	true
<	Menor	10 < 10	false
>	Maior	10 > 6	true
>=	Maior ou igual	3 >= 3	true
<=	Menor ou igual	7 <= 6	false

Operadores Lógicos

Operador	Nome	Exemplo	Resultado
&&	AND	(0 < 2) && (10 > 5)	true
||	OR	(10 >11) || (10 < 12)	true
!	NOT	!(1= = 4)	true
^	XOR	(1!= 0) ^ (2< 3)	false

Método Main

- Uma aplicação em Java é caracterizada por possuir o método **main().**
- A declaração do método deve ser :
  public static void main(String[] args)
- O método main é um método especial pois representa o ponto de entrada para a execução de um programa em Java. Quando um programa é executado, o interpretador chamará primeiramente o método main da classe. É ele quem controla o fluxo de execução do programa e executa qualquer outro método necessário.
- Nem toda classe terá um método main. Uma classe que não possua um método **main** não pode ser "executada" pois não representa um programa em Java. Ela será utilizada como classe utilitária para a construção de outras classes ou mesmo de um programa.

Estrutura básica de um programa

Os programas em java são sempre feitos através de classes.

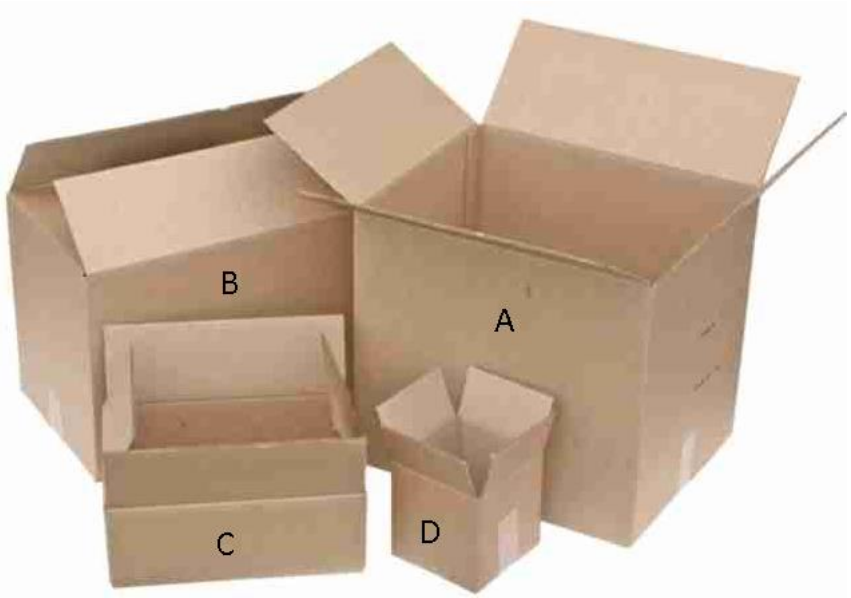
```
<modificador de acesso> class <nome da classe>
{
        <Declaração das Variáveis de Instância (Atributos)>
        <Declaração de Métodos>

        public static void main( String args[] )
        {
                //corpo principal do programa
        }
}
```

Variáveis

As variáveis servem para guardar informação.

Permitem que o programa efectue cálculos, e guarde os resultados para futuras operações.
Imagine que uma variável é uma caixa que contem informação.
Quando queremos saber a informação que está dentro da caixa, abrimos a caixa e lemos essa informação.
No fim, colocamos a informação na caixa e deixamos lá ficar, até essa informação ser novamente necessária.

De modo a facilitar a identificação das caixas, atribuímos um nome a cada.
O tipo da caixa define o tamanho da caixa e o tipo que informação que vai guardar.



Declaração de Variáveis

Uma variável ou constante é um tipo de identificador cujo nome, escolhido pelo programador, é associado a um valor pertencente a um certo tipo de dados. As variáveis devem ser sempre definidas com valores iniciais.

Exemplo0201.java

```
1  public class Exemplo0201
2  {
3    public static void main ( String args [] )
4    {
5      int x = 10, y = 20;
6      double dolar = 2.62;
7      System.out.println(x);
8      System.out.println(y);
9      System.out.println(dolar);
10   }
11 }
12
```

Declaração de Constantes

- As constantes em Java são definidas utilizando a palavra reservada *final*.

Exemplo0202.java *

```
1  public class Exemplo0202
2  {
3    public static void main ( String args [] )
4    {
5      int c = 10;      // declaração de variáveis do tipo inteiro
6      final double iva = 21;  // declaracao das constantes
7
8      System.out.println("Custo sem IVA: " + c);
9      System.out.println("Custo do IVA : " + (c*iva/100));
10     System.out.println("Cutso Total  :" + ((c*iva/100)+c));
11     // tente alterar o valor do iva = 23;
12
13   }
14 }
```

Passagem de Parâmetros

Uma aplicação em Java pode receber valores a partir da linha de comando. Quando um parâmetro é recebido pelo programa, ele pode ser manipulado dentro do programa.

soma.java *

```
1  class soma
2  {
3    public static void main (String args[])
4    {
5      // imprime a soma dos 2 primeiros argumentos ???
6      System.out.println((args[0])+(args[1]));
7    }
8  }
9
10
11
12
```

```
C:\java>java soma 3 5
```

Conversão de tipos

- Os parâmetros são passados no tipo string, de forma a converter, utilize a seguinte tabela:

Supondo a variável x	Converter em	y recebe o valor convertido
int x = 10	Float	Float y = **(float)** x
int x = 10	double	double y = **(double)** x
float x = 10.5	Int	int y = **(int)** x
String x = "10"	Int	int y = **Integer.parseInt**(x)
String x = "20.54"	Float	Float y = **Float.parseFloat**(x)
String x = "20.54"	double	double y = **Double.parseDouble**(x)
String x = "Java"	Vetor de bytes	Byte b[] = **x.getBytes**()
int x = 10	String	String y = **String.valueOf**(x)
float x = 10.35	String	String y = **String.valueOf**(x)
double x = 254.34	String	String y = **String.valueOf**(x)
byte x[] - (x é um vetor de bytes)	String	String y = **new String**(x)

Conversão de tipos

soma2.java

```
1  class soma2
2  {
3    public static void main (String args[])
4    {
5      Integer v1, v2;
6      v1 = Integer.parseInt(args[0]);
7      v2 = Integer.parseInt(args[1]);
8      System.out.println(v1+v2);
9    }
10 }
11
```

Ou (sem utilizar variáveis)

soma2.java

```
1  class soma2
2  {
3    public static void main (String args[])
4    {
5      System.out.println(Integer.parseInt(args[0])+Integer.parseInt(args[1]));
6    }
7  }
```

Entrada de Dados pelo Teclado

De forma a poder receber dados do teclado, é necessário importar o pacote de classes **Scanner** que pertence ao pacote **java.util**,

A instrução **import** indica que um determinado pacote é carregado no momento da compilação.

O asterisco indica que são carregadas todas as classes do pacote (**import java.util.*)**

Entrada de Dados pelo Teclado

atriangulo.java

```
1  //import java.util.Scanner;
2  import java.util.*;
3
4  class atriangulo
5  {
6  public static void main(String args[])
7      {
8          float a, b;
9          // kbd objecto criado com a class Scanner
10         Scanner kbd = new Scanner(System.in);
11
12
13         System.out.print("Base do triangulo: ");
14         b=kbd.nextFloat();
15
16         System.out.print("Altura do triangulo: ");
17         a=kbd.nextFloat();
18
19         System.out.println("Area do triangulo: " + (a*b/2));
20     }
21 }
```

Entrada de Dados pelo Teclado

Classe **Scanner**:

- nextByte() – ler dados tipo **Byte**
- nextInt() – ler dados tipo **Int**
- nextShort() – ler dados tipo **Short**
- nextLong() – ler dados tipo **Long**
- nextFloat() – ler dados tipo **Float**
- nextDouble() – ler dados tipo **Double**
- nextBoolean() – ler dados tipo **Boolean**
- nextLine() – ler dados tipo **String**

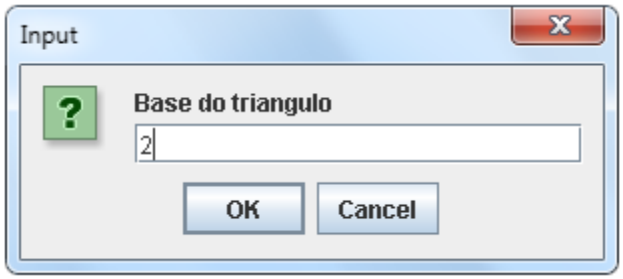
Entrada de Dados pelo Teclado

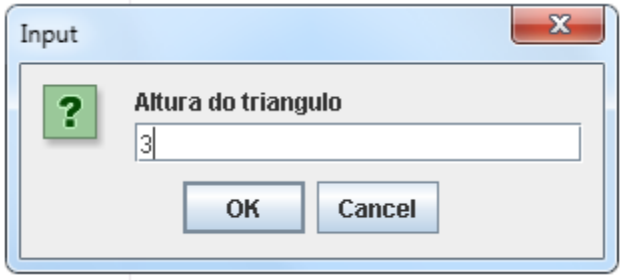
```java
import java.util.Scanner;

class lerdados
{
public static void main(String args[])
    {
        int i;
        float f;
        String s;

        Scanner kbd = new Scanner(System.in);


        System.out.print("Insira uma variavel do tipo String: ");
        s=kbd.nextLine();
        System.out.println(s);

        System.out.print("Insira uma variavel do tipo Int: ");
        i=kbd.nextInt();
        System.out.println(i);

        System.out.print("Insira uma variavel do tipo Float: ");
        f=kbd.nextFloat();
        System.out.println(f);
    }
}
```

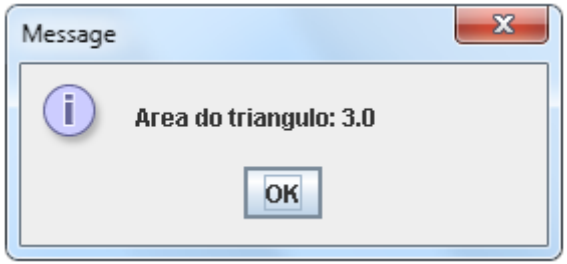
Classe JOptionPane

atriangulow.java

```
1  import javax.swing.JOptionPane;
2
3  class atriangulow
4  {
5      public static void main(String args[])
6      {
7          float base,altura;
8          base = Float.parseFloat(JOptionPane.showInputDialog(null,"Base do triangulo"));
9          altura = Float.parseFloat(JOptionPane.showInputDialog(null,"Altura do triangulo"));
10
11         JOptionPane.showMessageDialog(null,"Area do triangulo: " + ((base*altura)/2));
12     }
13 }
```







Exercícios

- **J1** – crie uma classe que receba como parâmetros a base e a altura e retorne a área de um triangulo.

```
C:\tmp>java atriangulo 3 5
Area do triangulo: 7.5
```

- **J2** – crie uma classe que receba como 3 valores e retorne a media desses valores.

```
C:\tmp>java media 1 2 4
Media: 2.3333333
```

Comandos Condicionais

- São usados em todas as linguagens de programação e possibilitam que a execução de um programa seja desviada com certas condições.

if-else **switch-case**

Estrutura *if-else*

- Permite a selecção de entre dois caminhos distintos para a execução, dependendo do resultado (verdadeiro ou falso) de uma expressão lógica.
- Se a condição for **verdadeira**, são executadas as instruções que estiverem entre os comandos **if/else*.***
- Se a condição for **falsa**, são executadas as instruções que estiverem após a instrução **else**.

Estrutura *if-else*

- Em java é obrigatório a condição aparecer sempre entre parênteses.

```
if (<Condicao>)
     {
          <Instruções para condição verdadeira>
     }
else
     {
          <Instruções para condição falsa>
     }
```

Estrutura *if-else*

maiormenor.java *

```
 1  import java.util.Scanner;
 2
 3  class maiormenor
 4  {
 5  public static void main(String args[])
 6      {
 7          int a, b;
 8          Scanner kbd = new Scanner(System.in);
 9
10          System.out.print("Digite o primeiro numero: ");
11          a=kbd.nextInt();
12          System.out.print("Digite o segundo numero: ");
13          b=kbd.nextInt();
14          if (a>b)
15              {
16                  System.out.println(a + " maior que " + b);
17              }
18          else
19              {
20                  System.out.println(a + " menor que " + b);
21              }
22      }
23  }
```

Exercícios

- **J3** - crie uma classe que receba um numero inteiro e diga se o numero é **par ou impar**.
- **J4** - crie uma classe que receba um numero inteiro e diga se o numero é **= 0, >0 ou <0.**
- **J5** - crie uma classe que receba o nome e idade de duas pessoas e imprima o **nome da pessoa mais nova, e ano de nascimento.**

Classe Calendar

dnascimento.java *

```
1  import java.util.Calendar;
2  import java.util.Scanner;
3
4  class dnascimento
5  {
6      public static void main(String args[])
7      {
8          int ano_nascimento, ano_actual;
9          Scanner kbd = new Scanner(System.in);
10         Calendar data = Calendar.getInstance(); // cria e inicializa o objecto data
11
12         System.out.print("Ano de nascimento: ");
13         ano_nascimento=kbd.nextInt();
14         ano_actual=data.get(Calendar.YEAR);
15
16         System.out.println("Tem " + (ano_actual-ano_nascimento) + " anos.");
17     }
18 }
```

Estrutura *switch-case*

- Permite efectuar o desvio da execução do programa de acordo com certas condições.
- Possibilita uma forma mais adequada e eficiente de trabalhar com uma grande quantidade de condições, construindo uma estrutura de controle com escola múltipla.

Estrutura *switch-case*

- Em java é obrigatório a condição aparecer sempre entre parênteses.

```
switch (<expressão>)
{
     case 1: intruções; break;
     case 1: intruções; break;
     case 1: intruções; break;
     default: intruções;
}
```

SÒ FUNCIONA COM VARAVEIS **int** OU **char**

Exemplo *switch-case (int)*

```
import java.util.Scanner;

class digito
{
public static void main(String args[])
    {
        int d;
        Scanner kbd = new Scanner(System.in);

        System.out.print("Digite numero: ");
        d=kbd.nextInt();
        switch (d)
        {
            case 0: System.out.println("Zero");break;
            case 1: System.out.println("Um");break;
            case 2: System.out.println("Dois");break;
            case 3: System.out.println("Tres");break;
            case 4: System.out.println("Quatro");break;
            case 5: System.out.println("Cinco");break;
            case 6: System.out.println("Seis");break;
            case 7: System.out.println("Sete");break;
            case 8: System.out.println("Oito");break;
            case 9: System.out.println("Nove");break;
            default : System.out.println("Digito desconhecido.");break;
        }
    }
}
```

Exemplo *switch-case (char)*

alfabeto.java *

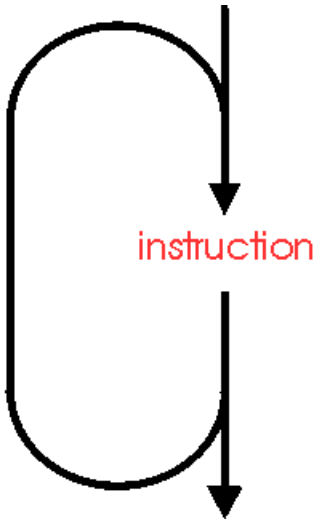
```
1  import java.util.Scanner;
2
3  class alfabeto
4  {
5  public static void main(String args[])
6      {
7                  char c;
8                  Scanner kbd = new Scanner(System.in);
9
10                 System.out.print("Digite uma letra: ");
11                 c=kbd.nextLine().charAt(0);  // c assume o valor do 1º caracter lido
12
13                 switch(c)
14                 {
15                     case 'A':
16                     case 'a': System.out.println("Alfa");break;
17                     case 'b':
18                     case 'B': System.out.println("Beta");break;
19                     case 'c':
20                     case 'C': System.out.println("Charlie");break;
21                     case 'd':
22                     case 'D': System.out.println("Delta");break;
23
24                     default:  System.out.println("Letra descolnhecica");
25
26                 }
27     }
28 }
```

Exercícios

- **J6** - crie uma classe que receba um numero e retorne o nome do mês equivalente.
- **J7** - crie uma classe que receba um caracter e avalie se este é uma vogal ou consoante.
- **J8** - crie uma classe que simule uma maquina de calcular, com as seguintes operações: +, -, /,x.

Ciclos

Os ciclos possibilitam a repetição da execução de um bloco de instruções num programa. Eles determinam que um certo bloco seja executado repetidamente ate que uma condição especifica ocorra.

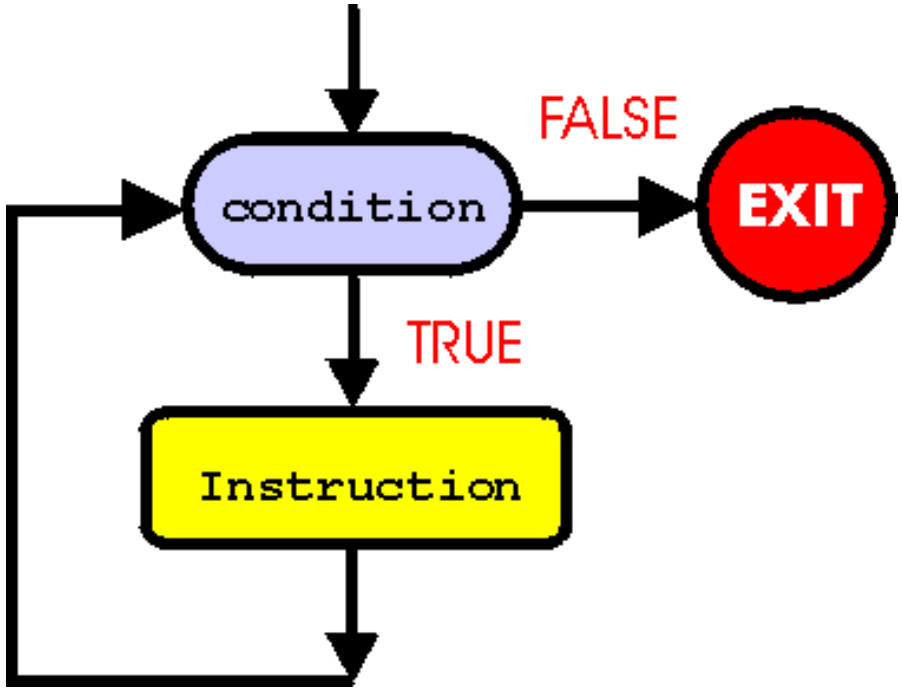


Ciclo - while

Implementa um ciclo para executar um bloco de comandos sucessivas vezes. A expressão de comparação é avaliada antes que o ciclo seja novamente executado.

Sintaxe :

```
while ( boolean_expr ) {

  comandos;

}
```



Ciclo – while - Exemplo

ciclow.java *

```
 1  class ciclow
 2  {
 3  public static void main(String args[])
 4       {
 5            int n=0;
 6            while (n!=10)
 7                 {
 8                      System.out.println(n);
 9                      n++;
10                 }
11       }
12  }
13
14
15
```

Ciclo – while - Exercícios

J9 - crie uma classe que mostre os primeiros 100 números pares, por ordem decrescente.

J10 – crie uma classe que leia dois números, e mostre o intervalo entre eles.

J11 - crie uma classe que leia vários números (termina com 0), calcula a soma e a media.

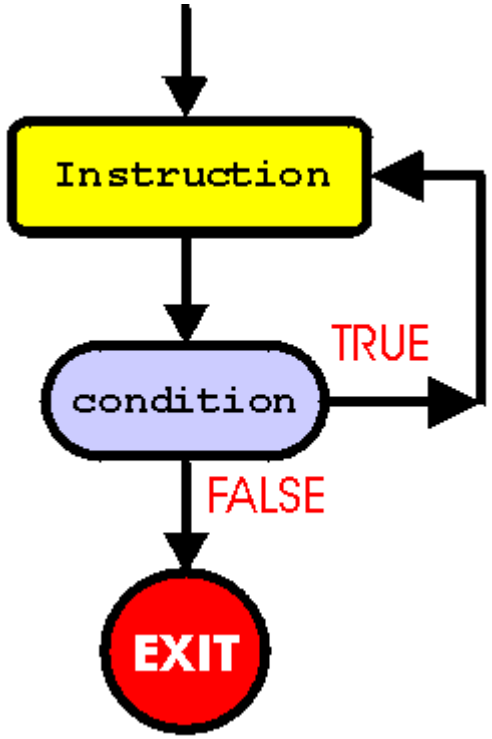
Ciclo do while

Utilizado quando se quer que o corpo do ciclo seja necessariamente executado pelo menos uma vez.

A expressão de comparação é avaliada depois do corpo do ciclo ser executado.

Sintaxe:

```
do {
        comandos;
} while ( boolean_expr );
```



Ciclo do while - Exemplo

ciclodw.java *

```
1  class ciclodw
2  {
3  public static void main(String args[])
4       {
5            int contador=10;
6            do
7                 {
8                      System.out.println(contador + "!");
9                      contador--;
10                }
11           while (contador>0);
12           System.out.println("Fogo!!!!");
13      }
14 }
```

Ciclo do while - Exercícios

J12 - crie uma classe que leia vários números (termina com 0), e mostre o maior e o menor.

J13 - crie uma classe retorne o numero de dígitos de um numero inteiro.

J14 - crie uma classe que simule o login de um utilizador (ver método l.equals("lo"))

Ciclo for

Este tipo de ciclo é usado quando se sabe o numero de vezes que o ciclo é repetido. Tem uma parte inicial com a inicialização das variáveis, seguida por uma expressão de comparação e depois a parte final com o incremento ou decremento das variáveis do ciclo.

Sintaxe :

```
for  ( inicialização; condição; iteração ) {
        comandos;
}
```

Ciclo for - Exemplo

Mostra tabela ASCII

```
public class ascii
{

    public static void main(String args[])
    {
        //System.out.println((int)'A');
        //System.out.println((char)65);
        for (int i=0;i<255;i++)
            System.out.println(i+":"+(char)i);

    }
}
```

Ciclo for - Exemplo

É possível declarar variáveis na inicialização do for.

ciclof.java

```
1  class ciclof
2  {
3  public static void main(String args[])
4       {
5            for (int n=10;n>=0;n--)
6                 {
7                      System.out.println(n);
8                 }
9       }
10 }
```

Ciclo for - Exemplo

É possível ter ciclos dentro de ciclos

ciclof.java

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

Ciclo for - Exercícios

J15 - crie uma classe que solicite um nome e o reescreva de trás para frente.

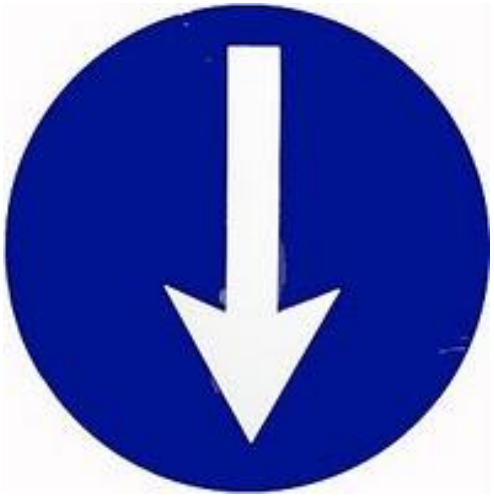
J16 - crie uma classe que solicite uma palavra, e retorne o número de vogais e consoantes dessa palavra. "*s.toUpperCase()*"

J17 – crie uma classe que retorne x^n

Controlo de Ciclos (salto)

Os ciclos são executados dependendo da avaliação das condições booleanas.

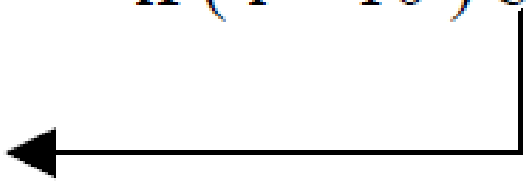
No entanto é possível alterar o percurso normal de um ciclo, sem que o resto do ciclo seja executado, com os comandos: break e continue



Comando break

O comando **break** é utilizado para interromper um ciclo (for, do-while, while) ou um switch.
O ciclo termina independentemente do valor da condição, e o 1º comando depois do ciclo é executado.

```
int i = 0;
while (true) {
        System.out.println(i);
        i++;
        if ( i > 10 ) break;
}
...
```



Comando continue

Quando este comando é encontrado (somente dentro de ciclos), a execução pára naquele ponto e o programa volta para o início do ciclo, para dar início a uma nova iteração.

```
for (int i = -10; i<10; i++)
{
        if ( i = = 0 )
                continue;
        System.out.println(1/i); //Não deve ser executado o cálculo 1/0 !!
}
```

Abandona a iteração corrente e salta para a próxima iteração

Escopo das variáveis

Em Java, podemos declarar variáveis em qualquer parte de uma classe. Essa variável vai ser visível de um determinado ponto a outro.

```
//aqui a variável i não existe
int i = 5;
// a partir daqui ela existe
```

Escopo das variáveis

O **escopo da variável** é o nome dado ao intervalo de código em que aquela variável existe e que é possível acede-la.

Quando abrimos um novo bloco { }, as variáveis declaradas ali dentro **só tem validade até o fim desse bloco**.

```
//aqui a variável i não existe
int i = 5;
// a partir daqui ela existe
while (condicao)
{
    // o i ainda vale aqui
    int j = 7;
    // o j passa a existir
}
// aqui o j não existe mais, mas o i continua a valer
```

Escopo das variáveis

```
public static void main(String args[])
{
    for (int i=0;i<10;i++)
        {
            System.out.println("ola");
        }
    System.out.print(i); // ATENÇÃO
}
```

A variável i é apenas visível dentro do ciclo { }

A variável deve ser criada for do ciclo.

```
public static void main(String args[])
{
    int i;
    for (i=0;i<10;i++)
        {
            System.out.println("ola");
        }
    System.out.print(i);
}
```

Diminuir casas decimais

float total;

System.out.printf(“total: %.2f”,total);

System.out.printf(“total: %.2f”, +total);

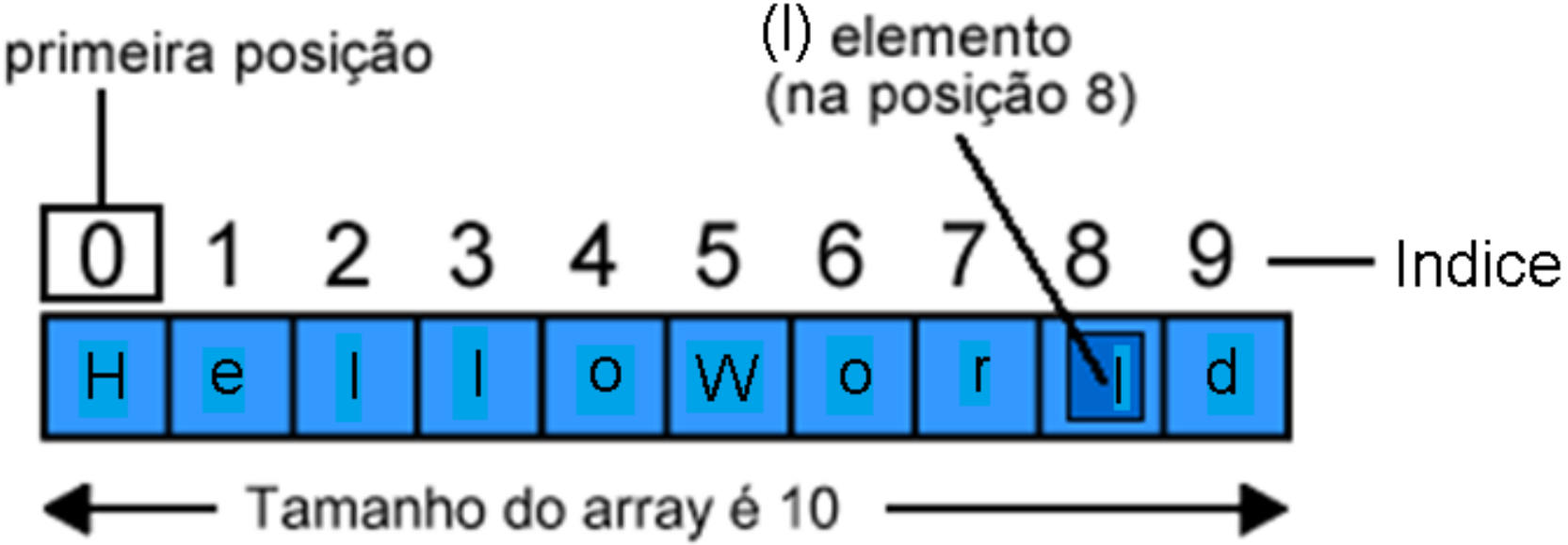
System.out.println()

System.out.print()

System.out.printf()

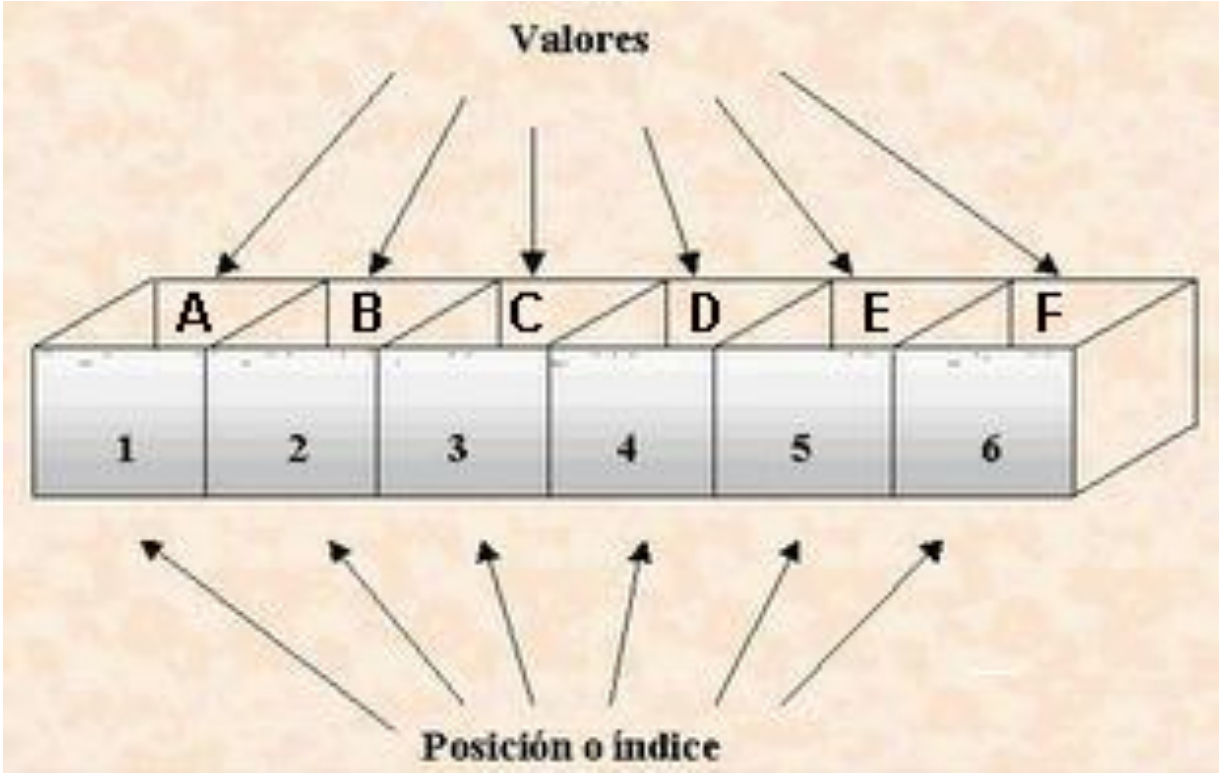
Vectores (array)

- Um **array** é um conjunto com um numero fixo de valores todos do mesmo tipo.
- O comprimento do array é definido quando o array é criado.
- Cada elemento do array é acedido pelo seu índice numérico, a começar em zero.

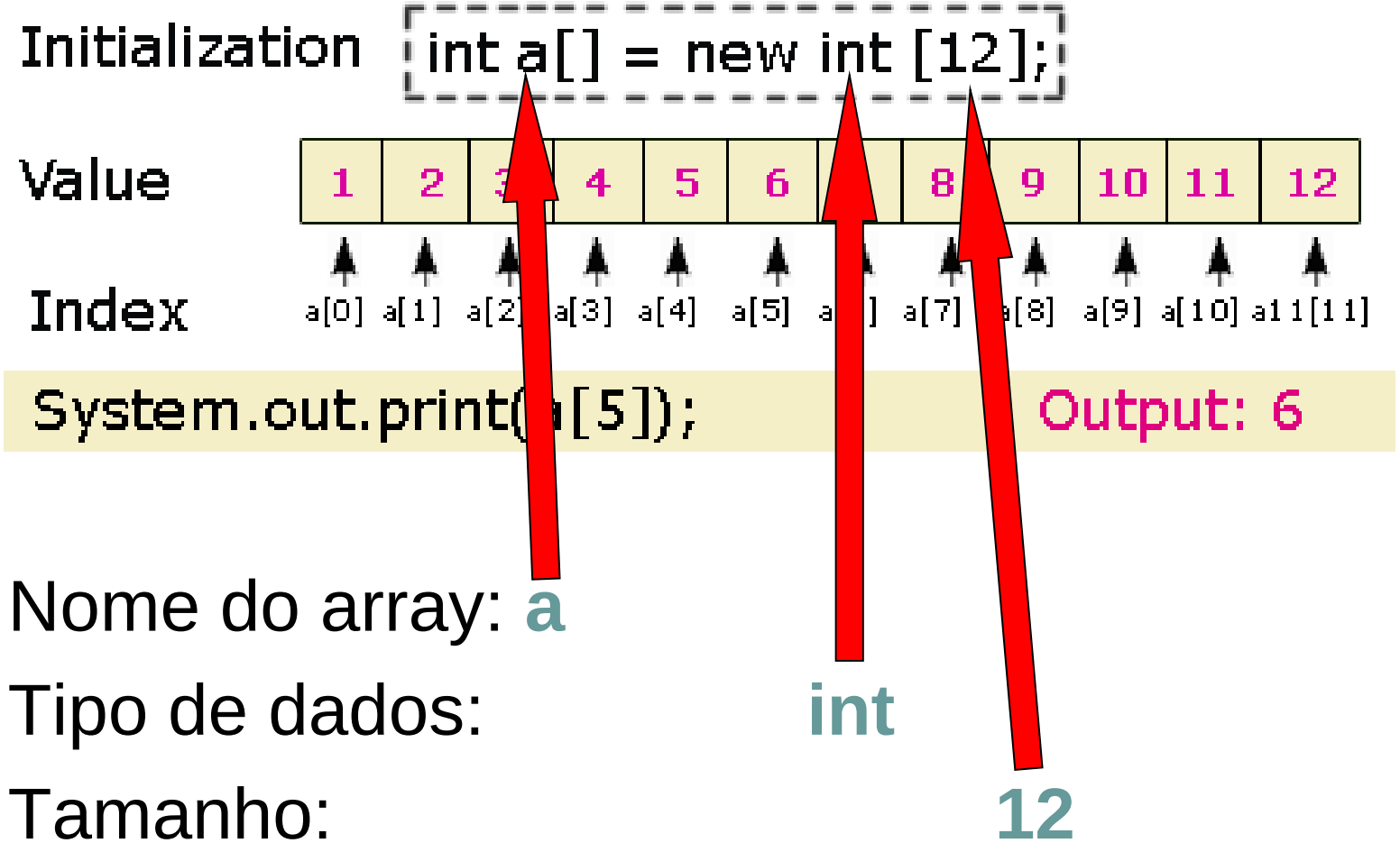


Arrays Unidimensionais

- São arrays apenas com um dimensão, tem apenas um índice para aceder ao seu conteúdo.



Arrays Unidimensionais



Nome do array: **a**

Tipo de dados: **int**

Tamanho: **12**

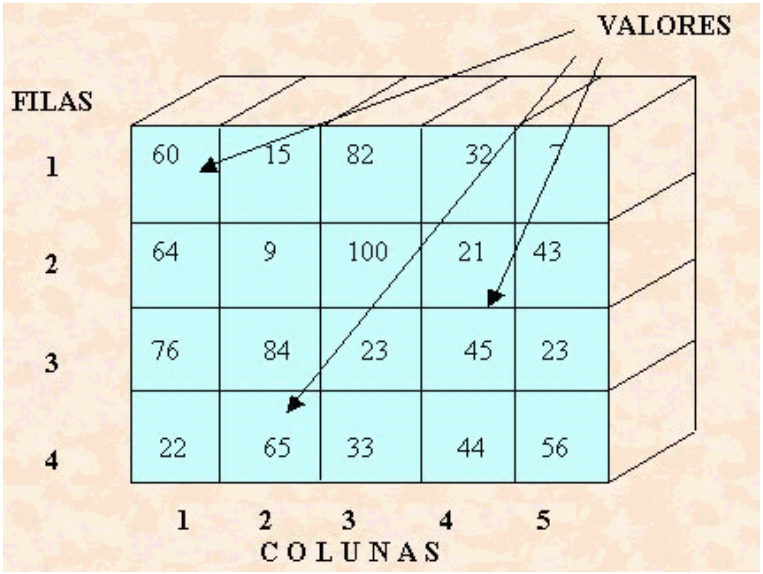
Arrays Unidimensionais Exemplo

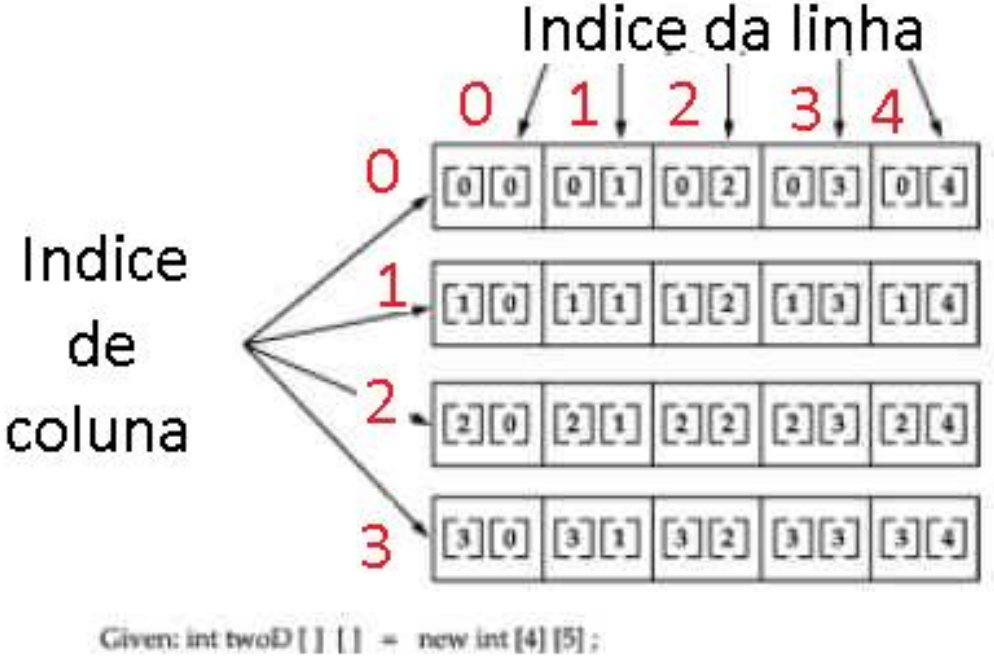
array1.java *

```
1  public class array1
2  {
3      public static void main(String args[])
4      {
5          int a[] = new int[3]; // Declaração de um array de inteiros
6
7          a[0]=0;
8          a[1]=1;
9          a[2]=2;
10
11         System.out.println("a[0]=" + a[0]);
12         System.out.println("a[1]=" + a[1]);
13         System.out.println("a[2]=" + a[2]);
14         System.out.println("Comprimento array a:"+a.length);
15
16         String mes[]={"Janeiro","Fevereiro","Março",
17                       "Abril","Maio","Junho",
18                       "Julho","Agosto","Setembro",
19                       "Outubro","Novembro","Dezembro"};
20
21         System.out.println();
22         for (int i=0;i<12;i++)
23             System.out.println(mes[i]);
24         System.out.println("Comprimento array mes:"+mes.length);
25     }
26 }
```

Arrays bidimensionais

- Este tipo de arrays permitem a criação de vectores com 2 índices, sob a forma de matrizes.





Given: int twoD [ ] [ ] = new int [4] [5] ;

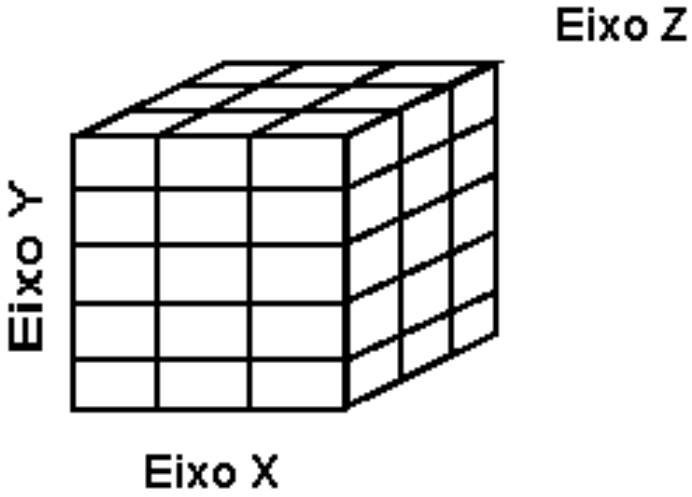
Arrays Multidimensionais Exemplo

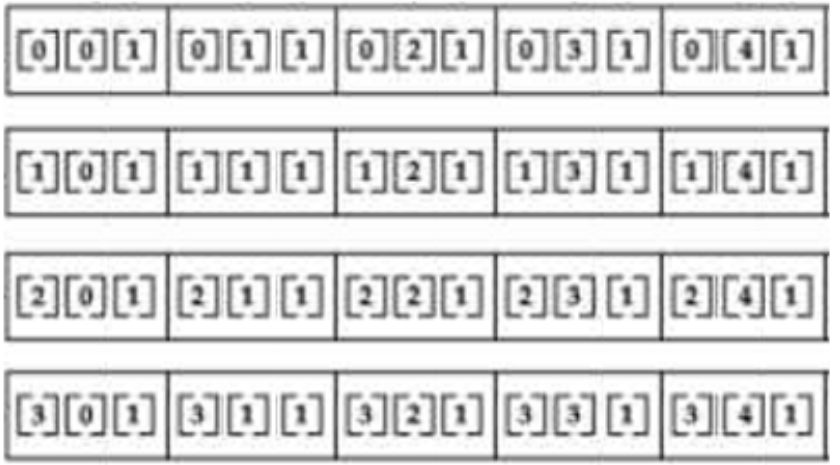
array2.java *

```
1  public class array2
2  {
3      public static void main(String args[])
4      {
5          int a[][] = new int[4][5];
6
7          a[0][0]=1;
8          a[0][1]=2;
9          a[0][2]=3;
10         a[0][3]=4;
11         a[0][4]=5;
12
13         System.out.println("a[0][0]=" + a[0][0]);
14         System.out.println("a[0][1]=" + a[0][1]);
15         System.out.println("a[0][2]=" + a[0][2]);
16         System.out.println("a[0][3]=" + a[0][3]);
17         System.out.println("a[0][4]=" + a[0][4]);
18
19
20     }
21 }
```

Arrays multidimensionais

- Este tipo de arrays permitem a criação de vectores com pelo menos 3 índices, sob a forma figuras em 3D.





Arrays - Exercícios

J18 - crie uma classe que leia um vector com n números inteiros e depois os mostre.

J19 - crie uma classe que leia um vector com n números inteiros e os mostre por ordem inversa.

J20 – crie uma classe utilizando um vector que guarda o numero de dias do mes {31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}, onde é pedido o número do mês, e é retornado o número de dias desse mês, com validação do mes.

Arrays - Exercícios

J21 - crie uma classe que gere um vector com n números inteiros (até 100), mostre o maior, menor e média dos números gerados. *(int)(Math.random()*100)+1*

J22 - crie uma classe que gere um vector com n números inteiros sem repetições.

J23 - crie uma classe que leia um vector com n números e o mostre ordenado por ordem crescente.

J24 - *crie uma classe que gere chaves ordenadas por ordem crescente para o euromilhoes.*

(5 números de 1 a 50 e 2 estrelas entre 1 e 9)

FIM

ORACLE®

Sun microsystems