

|| (ou lógico) e | (ou binário)

Condição 1	Operador	Condição2	Resultado
true		true	true
false		true	true
true		false	true
false		false	false



A diferença básica entre os operadores || e |, é que, semelhante ao operador &&, o || também suporta a avaliação parcial.

\wedge (ou exclusivo binário)



Condição 1	Operador	Condição2	Resultado
true	\wedge	true	false
false	\wedge	true	true
true	\wedge	false	true
false	\wedge	false	false

! (Negação)



Condição	Operador	Resultado
true	!	false
false	!	true

Operadores de Incremento e Decremento

int varInt = 1;

varInt ++ ++
++ ++ **varInt**

varInt -- --
-- -- **varInt**

Precedência de Operadores



1º Calcula o parêntese em maior nível

$$((10\%4)*5)+(4/2)+88-10$$

$$(2*5)+(4/2)+88-10$$

$$10+2+88-10$$

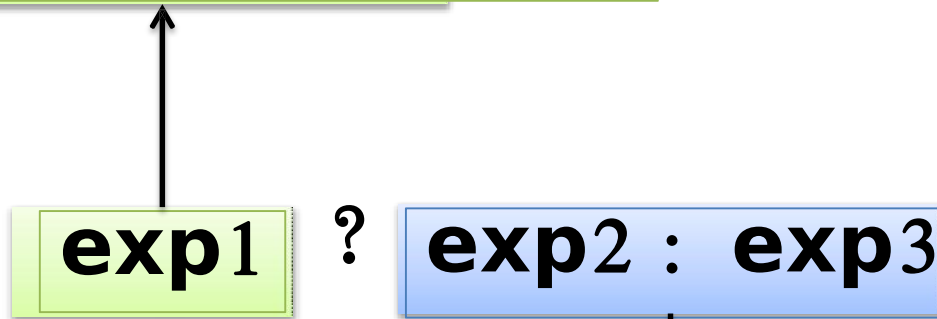
$$90$$

2º Calcula os valores dos parênteses restantes

3º Calcula as operações de adição e subtração

Operador Condicional (?:)

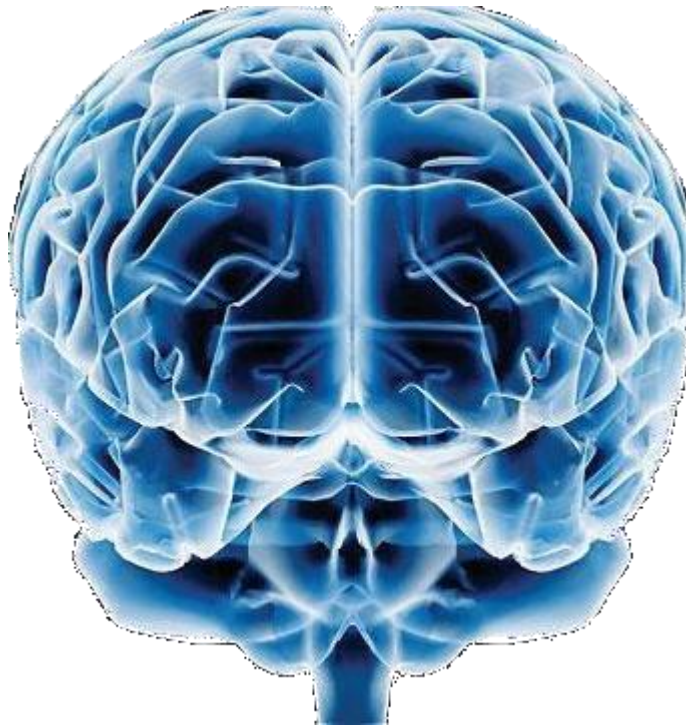
Expressão Booleana
retorna **true** ou **false**



Se o valor de **exp1** for **verdadeiro**, então o resultado será **exp2**, caso contrário, **exp3**

Java Orientado a Objetos

Estruturas de Controle

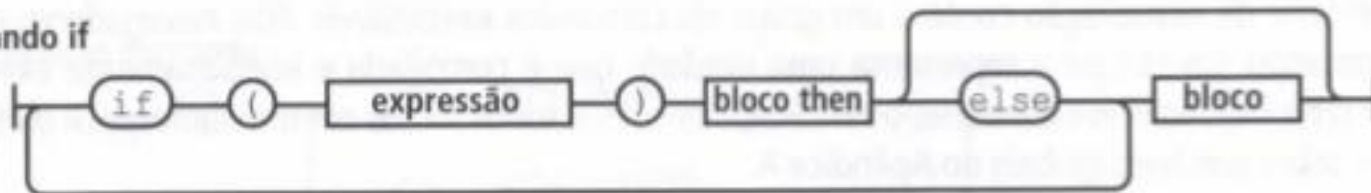


Estrutura de decisão (if)

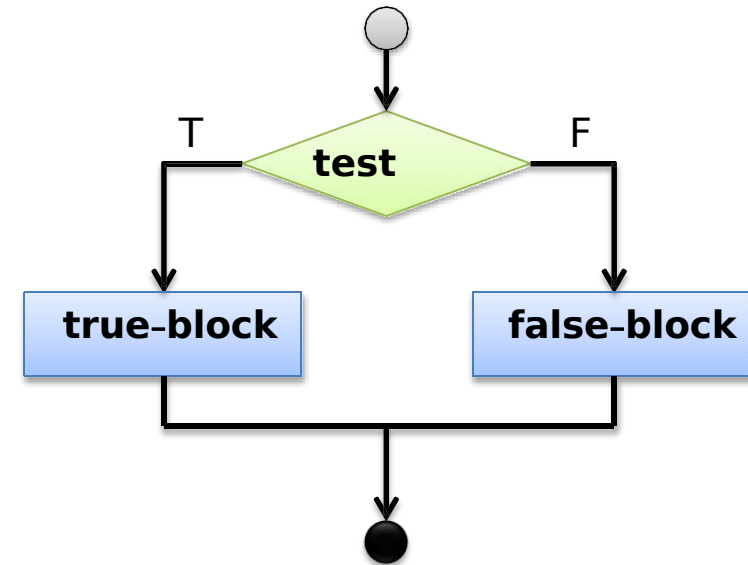
```
if (expressão-condicional) {  
    ... executar código  
    ... executar código  
    ... executar código  
} else {  
    ... executar código  
    ... executar código  
    ... executar código  
}
```



comando if



Estrutura de decisão (if-else)



Bloco de código da expressão
true true-block

```
int nota = 68;  
if (nota > 60) {  
    System.out.println("Parabéns!");  
    System.out.println("Você passou!");  
} else {  
    System.out.println("Lamento !");  
    System.out.println("Reprovado");  
}
```

Bloco de código da expressão
false false-block

Expressão
Booleana
test

Estrutura de decisão (if-else-if)

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

```
    int mesDoAno = 13;
```

```
    if (mesDoAno == 12 || mesDoAno == 1 || mesDoAno == 2) {  
        System.out.println("Verão");
```

```
    } else if (mesDoAno == 3 || mesDoAno == 4 || mesDoAno == 5) {  
        System.out.println("Outono");
```

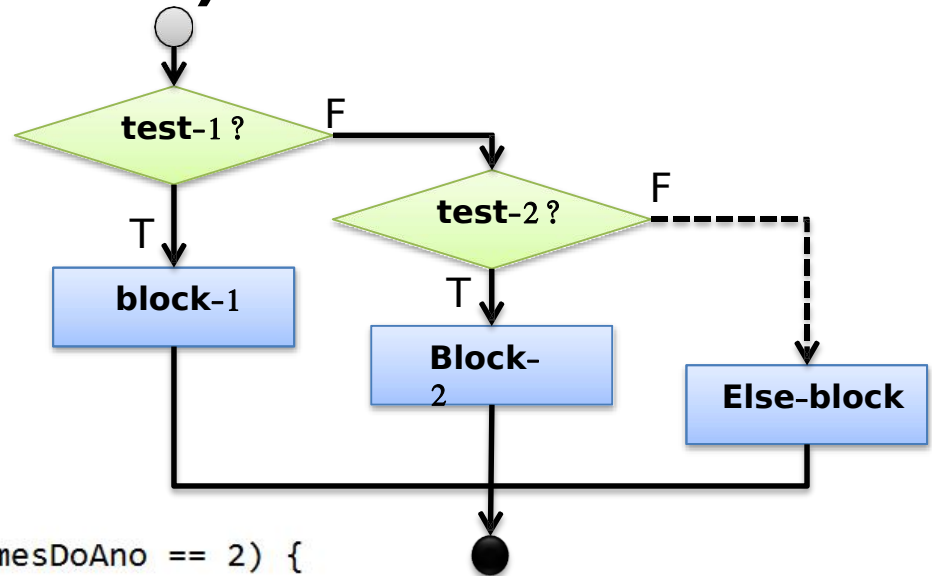
```
    } else if (mesDoAno == 6 || mesDoAno == 7 || mesDoAno == 8) {  
        System.out.println("Inverno");
```

```
    } else if (mesDoAno == 9 || mesDoAno == 10 || mesDoAno == 1) {  
        System.out.println("Primavera");
```

```
    } else {  
        System.out.println("Mês não é válido " + mesDoAno);
```

```
    }
```

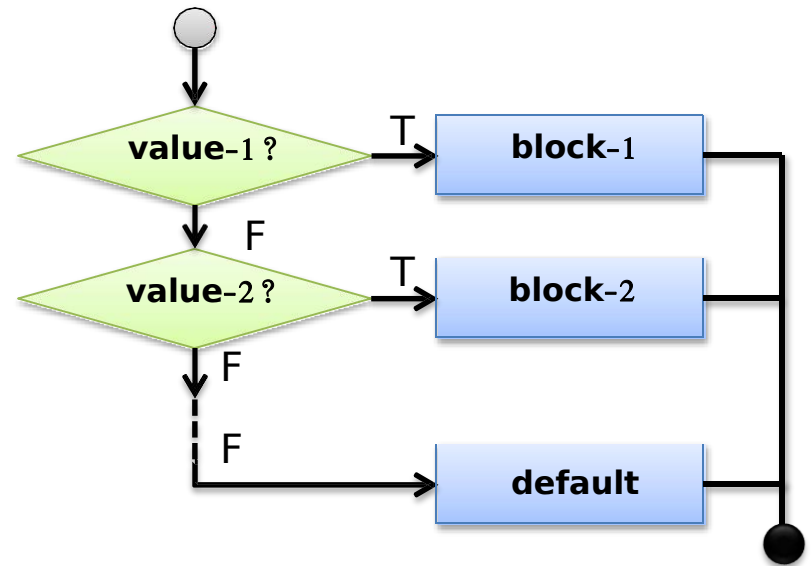
```
}
```



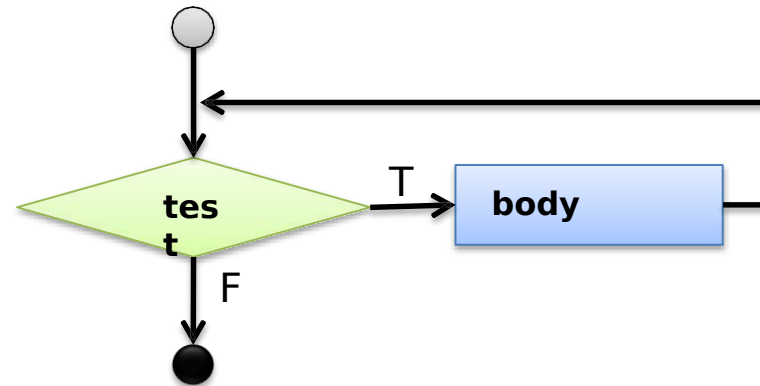
Estrutura de decisão (switch)

```
int mesDoAno = 13;

switch (mesDoAno) {
    case 12:
    case 1:
    case 2:
        System.out.println("Verão");
        break;
    case 3:
    case 4:
    case 5:
        System.out.println("Outono");
        break;
    case 6:
    case 7:
    case 8:
        System.out.println("Inverno");
        break;
    case 9:
    case 10:
    case 11:
        System.out.println("Primavera");
        break;
    default:
        System.out.println("Mês não é válido " + mesDoAno);
        break;
}
```



Estrutura de repetição (while)



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

```
    int contador = 0;
```

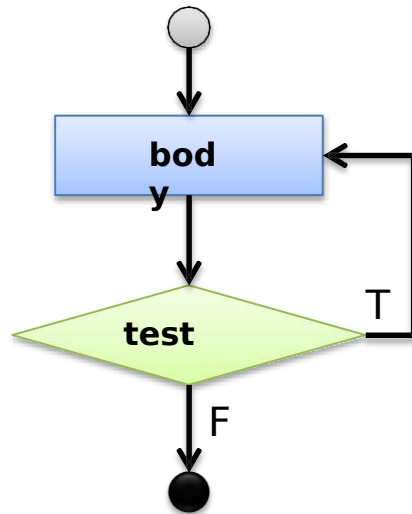
```
    while (contador < 10) {  
        System.out.println(contador);  
        contador++;  
    }
```

```
}
```

Expressão
Booleana
test

Corpo da estrutura de
repetição body

Estrutura de repetição (do-while)

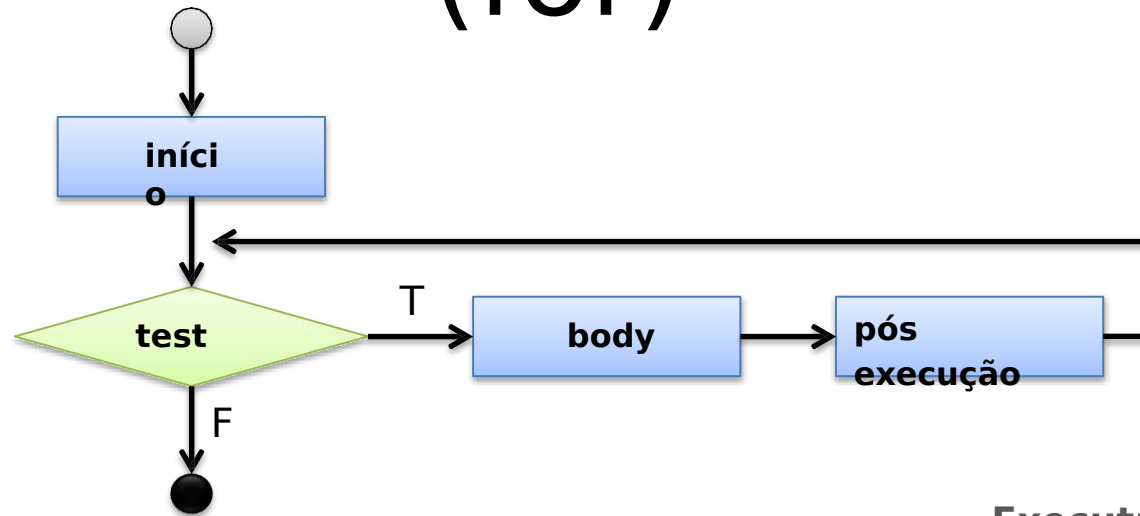


```
public static void main(String[] args) {  
    int contador = 0;  
    do {  
        System.out.println(contador);  
        contador++;  
    } while (contador < 10);  
}
```

Expressão
Booleana
t

Corpo da estrutura de
repetição bod
y

Estrutura de repetição (for)



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

```
    for(int contador = 0; contador < 10; contador++){  
        System.out.println(contador);
```

```
    }
```

```
}
```

Corpo da repetição
for body

Expressão
Booleana
t

Executa ao final do
for pós
execução

Declaração break

Interrompe a
repetição
infinita

```
public static void main(String[] args) {  
    int contador = 0;  
    while(true){ //laço infinito  
        if(contador == 10){  
            System.out.println("break - (while-true)");  
            break;  
        }  
        System.out.println(contador);  
        contador ++;  
    }  
}
```



Declaração continue

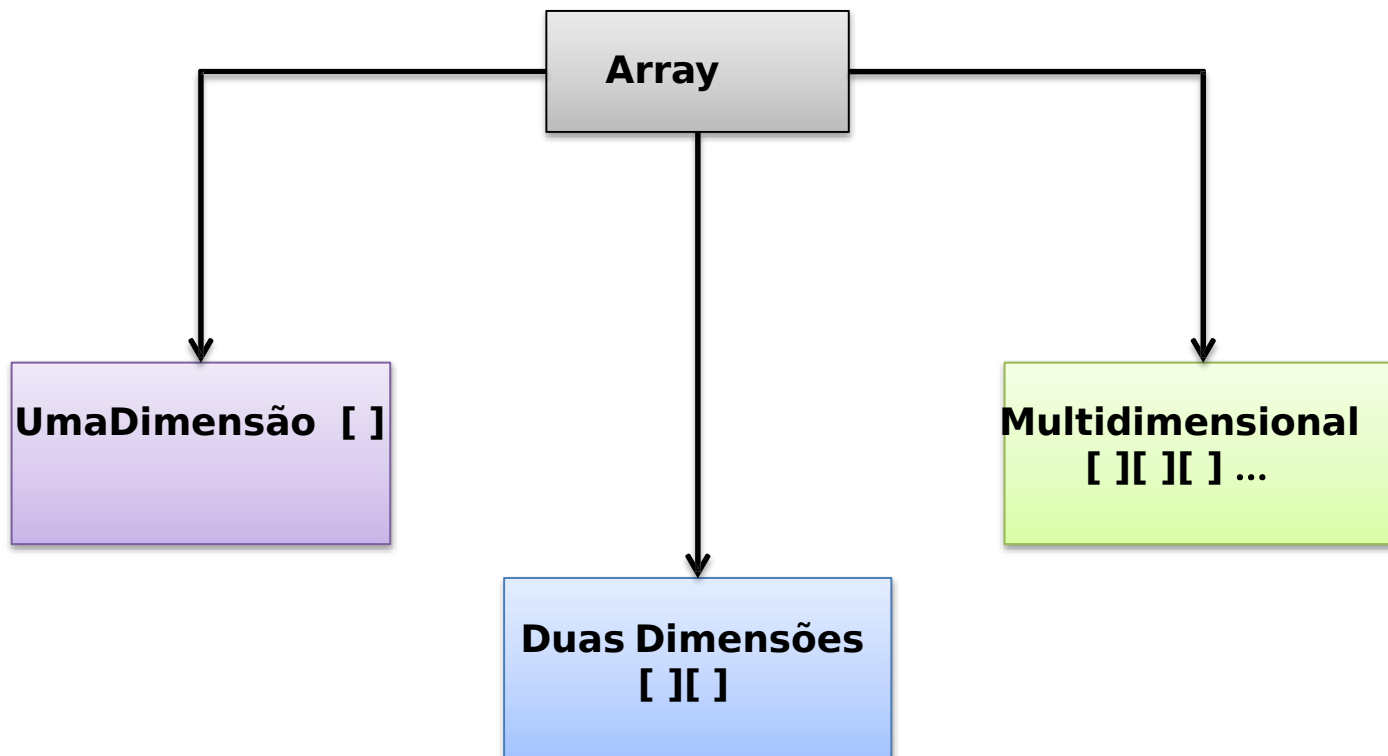
**Continua
Repetição**

Java Orientado a Objetos

Array



Array ?



Declarando Array

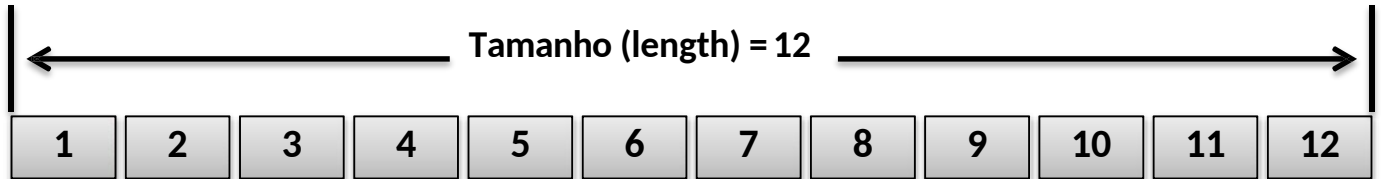
Inicialização

```
int a[] = new int[12];
```

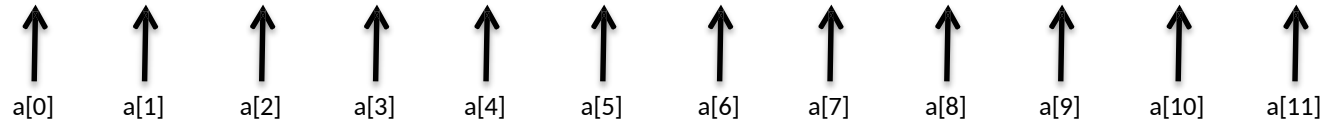
||

```
int []a = {1,2,3,4,5,6,7,8,9,10,11,12};
```

Valores



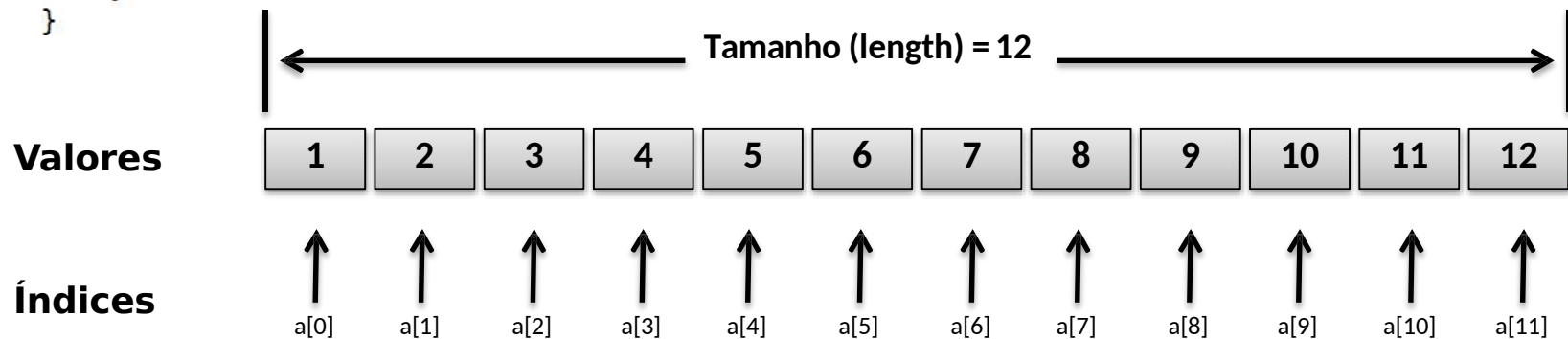
Índices



→ **“Pode guardar somente papel”**
(Um único tipo de dado, anteriormente definido)

Acessando um elemento do Array

```
public static void main(String[] args) {  
    int[] a = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };  
    System.out.println(a[0]); // acessando primeiro elemento do array  
    System.out.println(a[6]); // acessando elemento de indice 6  
    System.out.println(a[a.length - 1]); // acessando ultimo elemento do array  
    for (int i = 0; i < a.length; i++) { // percorre e imprime todos elementos do array  
        System.out.println(i);  
    }  
}
```



Arrays

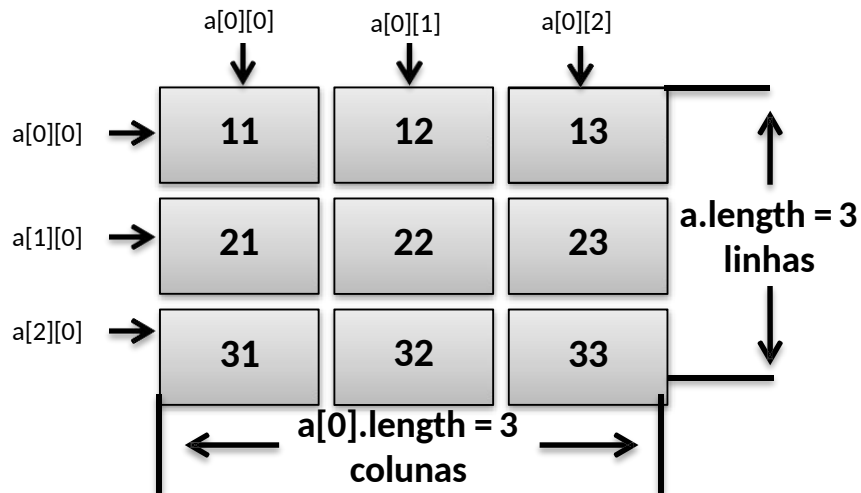
Multidimensionais

Inicializaçã

o

```
int a[][] = new int[3][3];
```

```
int [][]a = {{11,12,13},{21,22,23},{31,32,33}};
```



“Pode guardar somente papel”
(Um único tipo de dado, anteriormente definido)

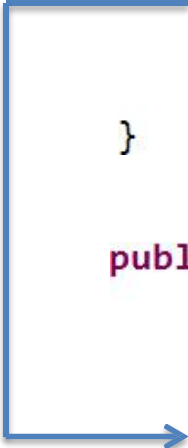
Acessando um elemento de um Array Multidimensional

```
public static void main(String[] args) {  
  
    int[][] matriz = { { 11, 12, 13 }, { 21, 22, 23 }, { 31, 32, 33 } }; //constroi matriz  
  
    System.out.println(matriz.length); // imprime numero de linhas  
  
    System.out.println(matriz[0].length); // imprime numero de colunas  
  
    System.out.println(matriz[0][0]); // acessa elemento na linha [0] e coluna [0]  
  
    System.out.println(matriz[2][2]); // acessa elemento na linha [2] e coluna [2]  
  
    for (int linha = 0; linha < matriz.length; linha++) { // percorre todas linhas  
        for (int coluna = 0; coluna < matriz[linha].length; coluna++) { // percorre todas colunas  
            System.out.print(matriz[linha][coluna] + " "); // imprime matriz  
        }  
        System.out.println("\n");  
    }  
}
```

		a[0][0]	a[0][1]	a[0][2]
		↓	↓	↓
a[0][0] →		11	12	13
a[1][0] →		21	22	23
a[2][0] →		31	32	33

Percorrendo Arrays com Enhanced- for

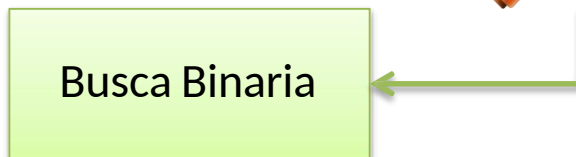
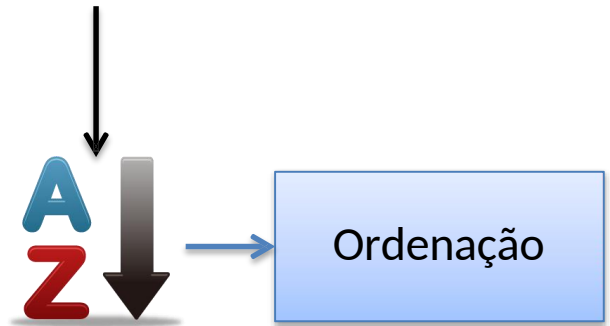
```
public static void main(String[] args) {  
    int[] a = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };  
    for (int i = 0; i < a.length; i++) { // percorre array  
        System.out.println(i); // imprime todos elementos do array  
    }  
}  
  
public static void main(String[] args) {  
    int[] a = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };  
    for (int i : a) { // utilizando Enhanced-for  
        System.out.println(i); // imprime todos elementos do array  
    }  
}
```

A blue line with an arrow at the end originates from the 'for' statement of the first code block and points to the 'for' statement of the second code block, illustrating the transition from a traditional loop to an enhanced for loop.

Manipulando Arrays com `java.util.Arrays`



```
public static void main(String[] args) {  
    int[] a = { 8, 4, 1, 9, 2, 5, 12, 3, 6, 10, 7, 11 };  
    System.out.println(Arrays.toString(a));  
    Arrays.sort(a);  
    System.out.println(Arrays.toString(a));  
    System.out.println("a[" + Arrays.binarySearch(a, 6) + "]");  
}
```



Java Orientado a Objetos Bases da programação Java OO

