



AULA 2 – ENTRADA DE DADOS, DECISÃO E REPETIÇÃO

PROGRAMAÇÃO ORIENTADA A OBJETOS

Prof. Ms. Nathan Cirillo e Silva
Prof. Ms. Peter Jád Júnior
Prof. Ms. Télvio Orru

Universidade Paulista UNIP

O que vimos na última aula...

- Características da linguagem Java;
- Estrutura de um **programa mínimo**;
- Stream (dutos) para a **saída de dados**;
- **Compilação e Execução** via terminal (`javac | java`);
- Criação de um programa pela **IDE Eclipse**.



Programa Mínimo

Deve haver uma classe com o método principal:

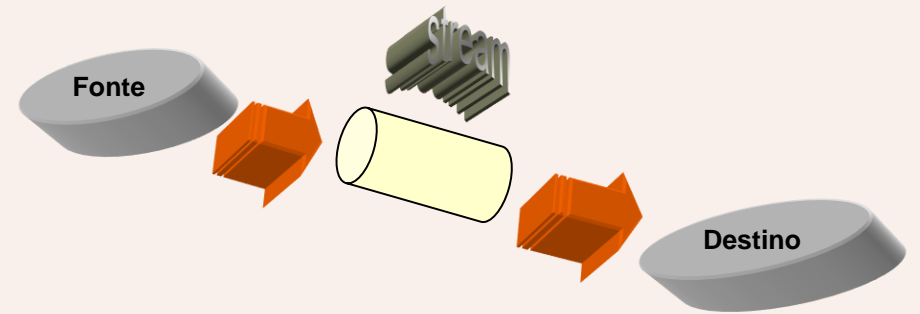
```
public static void main(String[]args)
```



```
public class Exemplo → Declaração da classe
{
    public static void main(String []args) → Método principal
    {
        System.out.println("Olá Pessoal!"); → Código
    }
}
```

Saída de Dados

- A saída padrão de dados (default) é o próprio console;
- É acessado via classe `java.lang.System` que oferece o objeto `out`;
- `out` é uma stream de dados que leva dados da aplicação para o console (i.e. “imprime na tela”).



System.out

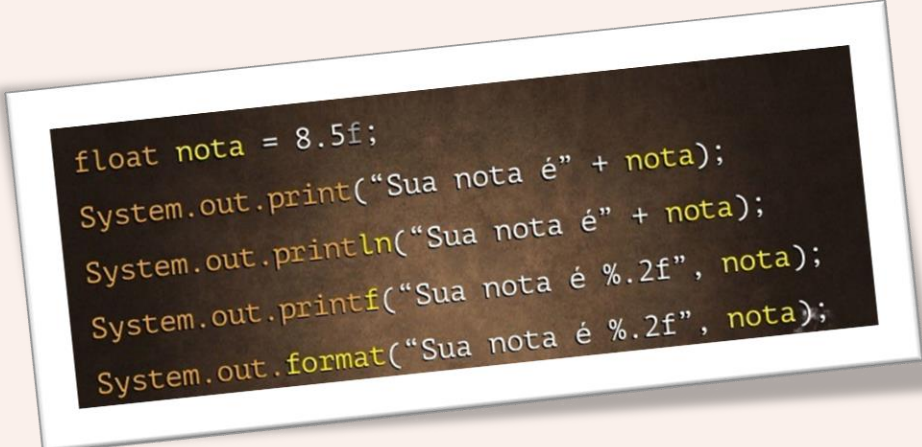
Objeto do tipo `java.io.PrintStream`;

Métodos importantes:

`print(argumento);`

`println(argumento);`

`printf("formato", argumentos);` // ling C



```
float nota = 8.5f;  
System.out.print("Sua nota é" + nota);  
System.out.println("Sua nota é" + nota);  
System.out.printf("Sua nota é %.2f", nota);  
System.out.format("Sua nota é %.2f", nota);
```

Onde argumento pode ser: *inteiro, String, real, char etc.*

Tipos Primitivos

- Inteiros: byte, short, int e long;
- Ponto flutuante: float e double;
- Caractere: char;
- Lógico: boolean.



*O tipo inteiro preferencial é **int**, enquanto o real preferencial é **double**.*

Declaração de Variáveis

Para declarar uma variável em Java é usada a sintaxe:

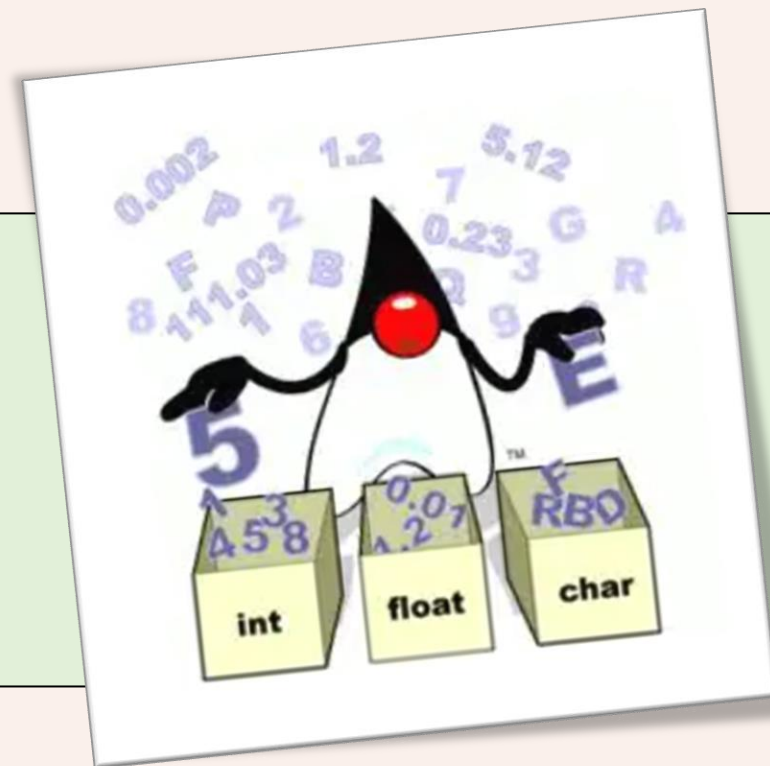
```
tipo var1 [, var2] [, var3] [, varN];
```

Veja alguns exemplos:

```
int i;
```

```
float total, preco;
```

double valorMedio.



Comentários

// comentário de uma linha

/* comentário
de múltiplas linhas */

/** comentário de documentação
* que também pode
* ter múltiplas linhas
*/

Arrays no Java

No Java os arrays são objetos:

- É necessário efetuar a alocação dinâmica dos arranjos antes de seu uso;
- São automaticamente "zerados" ao serem criados;
- Tem uma propriedade length que indica seu tamanho;
- Proibido índices inválidos (não inteiros, menores que zero ou maiores que length - 1).

Declaração de Arrays

Sintaxe:

<tipo> nome[];

Exemplo:

int v[];	// tipo primitivo
double x[];	// tipo primitivo
String nome[];	// String
Object coisas[];	// tipo Objeto

Alocação de Arrays

Dado um arranjo já declarado:

```
nome = new <tipo> [ num_elementos];
```

Exemplos:

```
v = new int [10];
```

```
d = new double [20];
```

```
nome = new String [45];
```

```
coisas = new Object [100];
```

Declaração e Alocação Simultâneas

Também é possível fazer:

```
int v[ ] = new int [10];
```

```
double x[ ] = new double [20];
```

```
String nome[ ] = new String [45];
```

```
Object coisas[ ] = new Object [100];
```

Declaração e Inicialização Simultâneas

Além disso também é possível fazer:

```
int v[ ] = {5, 13, -2, 2034, -192, 0, 10};
```

```
double x[ ] = {1.4}; // só um também pode
```

```
String nome[ ] = { "Pedro", "Lucas", "Matheus" };
```

```
Object coisas[ ] = { new Object(),  
                     new Object(),  
                     objetoExistente  
};
```

Nestes casos o compilador efetua a alocação e atribuição necessária para os elementos dados.

Atribuição de Valor

Considere um array declarado como:
`double t[] = new double[30];`

Por padrão, o array estará preenchido com valores zeros

Em cada posição válida podemos atribuir um valor:

`t[0] = 123.45;`

`t[1]= 75.31;`

`:`

`t[28]= 0.00001;`

`t[29]= 13;`

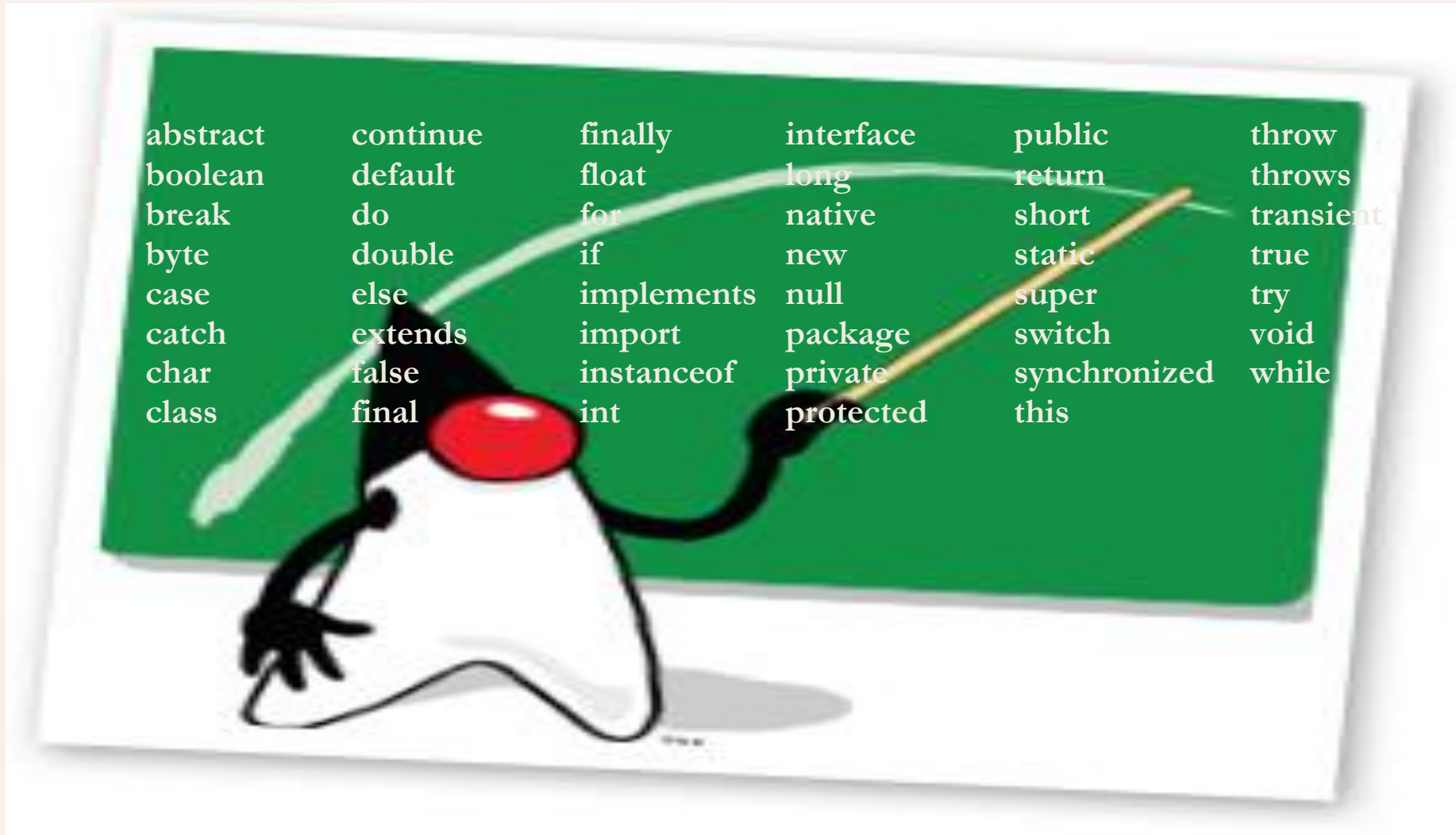
Uso do Valor

A combinação do nome do array com um índice, como: `t[1]`, `v[33]`, `lista[8]` equivale ao uso de uma variável individual, ou seja, o valor contido naquela posição é usado no local onde se indexa um array!

Exemplo de Uso do Array

```
public class Arranjo1 {  
    public static void main(String a[]) {  
        int tam =10;  
        // declara e aloca array  
        int v[ ] = new int [tam];  
        // inicia array com inteiros  
        for(int i=0; i<v.length; i++) {  
            // atribui a própria posição como conteúdo  
            v[i] = i;  
        }  
        // exibe array  
        for(int i=0; i<v.length; i++) {  
            System.out.println("v[" + i + "] = " + v[i]);  
        }  
    }  
}
```


Palavras Reservadas



Entrada de Dados

- A entrada de dados é feita pelo objeto `System.in` que é do tipo `java.io.InputStream`;
- Leitura orientada a byte com necessidade de tratamentos específicos;
- Para contornar isso devemos usar em conjunto a classe `java.util.Scanner`;
- É uma “nova possibilidade” eficiente de leitura de dados à partir do Java 5.

java.util.Scanner

- Faz a entrada eficiente de dados para os tipos primitivos existentes;
- Usa para isso métodos como: `next()` e o `nextLine()` (leem **palavra** e **linha**);
- Existem métodos de entrada específicos para cada tipo de dado a ser lido;
- **Ex.:** `nextByte()`; `nextInt()`; `nextLong()`; `nextFloat()`; `nextDouble()`; e `nextShort()`.

Uso do Objeto Scanner

Necessário criar um objeto desse tipo:

```
Scanner sc = new Scanner(System.in);
```



Lendo uma String	Lendo um Inteiro	Lendo um Real
String txt = sc .next();	int num = sc .nextInt();	float num = sc .nextFloat();
String txt = sc .nextLine();	long num = sc .nextLong();	double num = sc .nextDouble();

Operadores

Aritméticos:

- +, -, *, / (aritmética simples)
- % (resto da divisão inteira)
- - e + (sinal)
- ++ (incremento)
- -- (decremento)

Atribuição:

- =

```
public class Aritmetica {  
    static public void main (String args[]) {  
        int a = 5, b = 2; // Decl de 2 variaveis  
        // Exemplos de operacoes sobre variaveis  
        System.out.println("a = " + a);  
        System.out.println("-b = " + (-b));  
        System.out.println("a + b = " + (a + b));  
        System.out.println("a * b = " + (a * b));  
        System.out.println("a / b = " + (a / b));  
        System.out.println("a % b = " + (a % b));  
        System.out.println("a++ = " + (a++));  
        System.out.println("--b = " + (--b));  
    }  
}
```

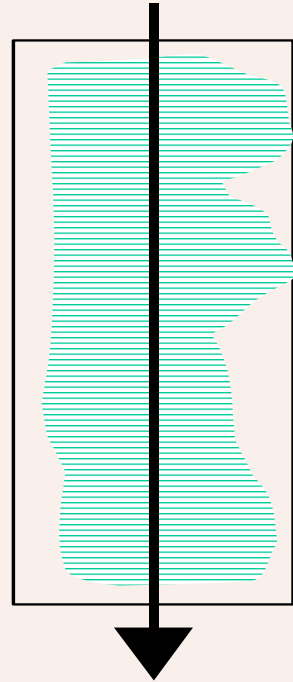
Operadores

Relacionais:

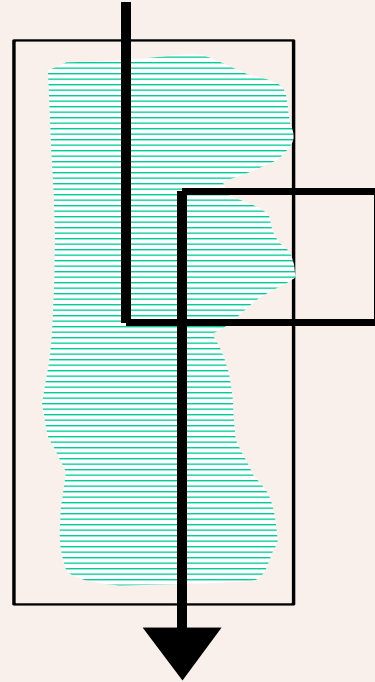
- > (maior)
- < (menor)
- >= (maior ou igual)
- <= (menor ou igual)
- == (igual)
- != (diferente)

```
public class Relacional {  
    static public void main (String args[]) {  
        int a = 15; int b = 12;  
        System.out.println("a == b -> " + (a == b));  
        System.out.println("a != b -> " + (a != b));  
        System.out.println("a < b -> " + (a < b));  
        System.out.println("a > b -> " + (a > b));  
        System.out.println("a <= b -> " + (a <= b));  
        System.out.println("a >= b -> " + (a >= b));  
    }  
}
```

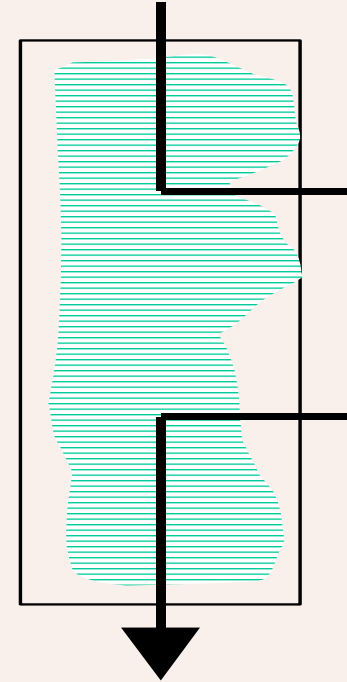
Fluxo de Execução



Fluxo
Sequencial de
Execução



Fluxo
Repetitivo de
Execução



Desvio do
Fluxo de
Execução

Diretivas

Isoladas:

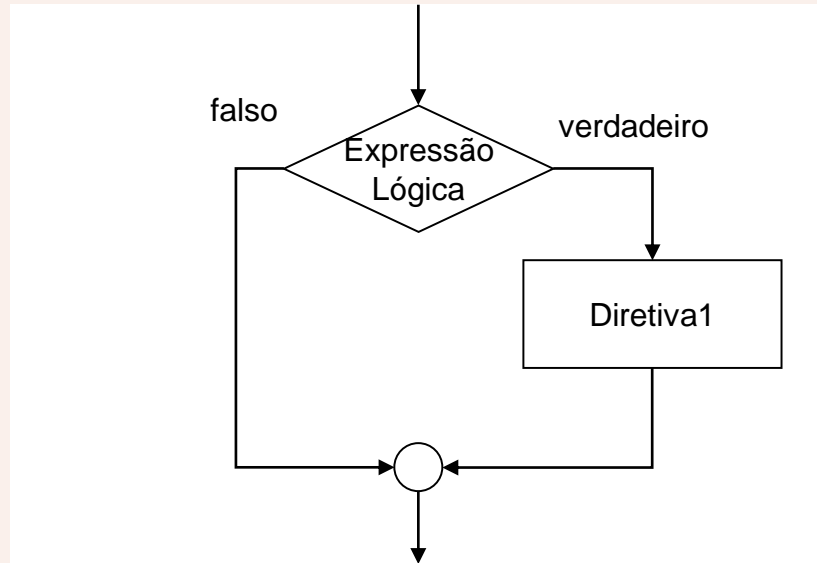
```
diretiva1;  
diretiva2;  
diretiva3;  
:  
diretivaN;
```

Bloco:

```
{  
    diretiva1;  
    diretiva2;  
    diretiva3;  
    :  
    diretivaN;  
}
```

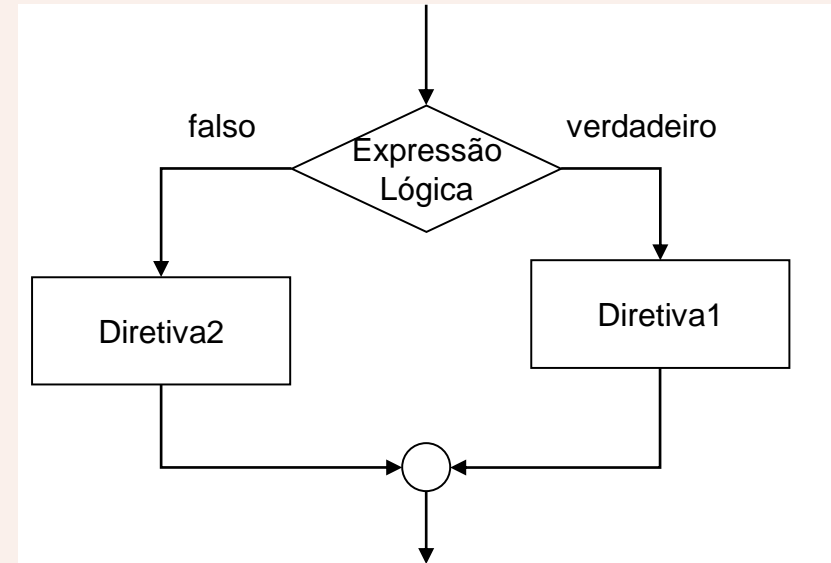

Desvio de Fluxo

Uma condição:



```
if (condição) {  
    diretiva1;  
}
```

Duas ou mais condições:



```
if (condição) {  
    diretiva1;  
} else {  
    diretiva2;  
}
```

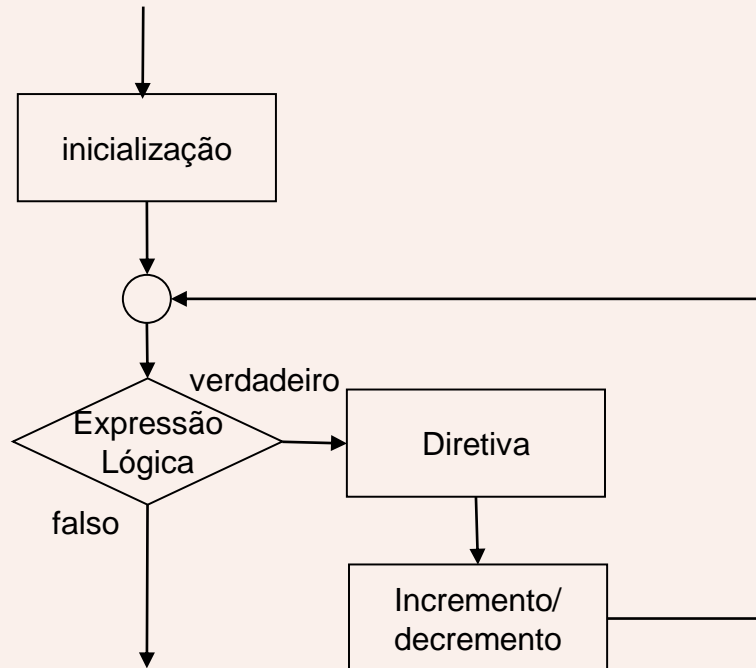
Exemplo de Desvio

```
public class ExemploIf {  
    public static void main (String args[]) {  
        if (args.length > 0) {  
            for (int j=0;j<Integer.parseInt(args[0]);j++) {  
                System.out.print(" " + j + " ");  
            }  
            System.out.println("\nFim da Contagem");  
        }  
        System.out.println("Fim do Programa");  
    }  
}
```

Repetição For

```
for (inicialização; condição; incremento/decremento){  
    diretiva;  
}
```

SINTAXE



```
public class ExemploFor {  
    public static void main (String args[]) {  
        int j;  
        for (j=0; j<10; j++) {  
            System.out.println(j);  
        }  
    }  
}
```

Repetição While

```
public class ExemploWhile {  
    public static void main (String args[]) {  
        int j = 10;  
        while (j > 0) {  
            System.out.println("j="+j);  
            j--;  
        }  
    }  
}
```

Repetição Do...While

```
public class ExemploDoWhile {  
  
    public static void main (String args[]) {  
        int j = 10;  
        do {  
            System.out.println("j="+j);  
            j--;  
        } while (j > 0);  
    }  
}
```

Dúvidas?

ANY
QUESTIONS

