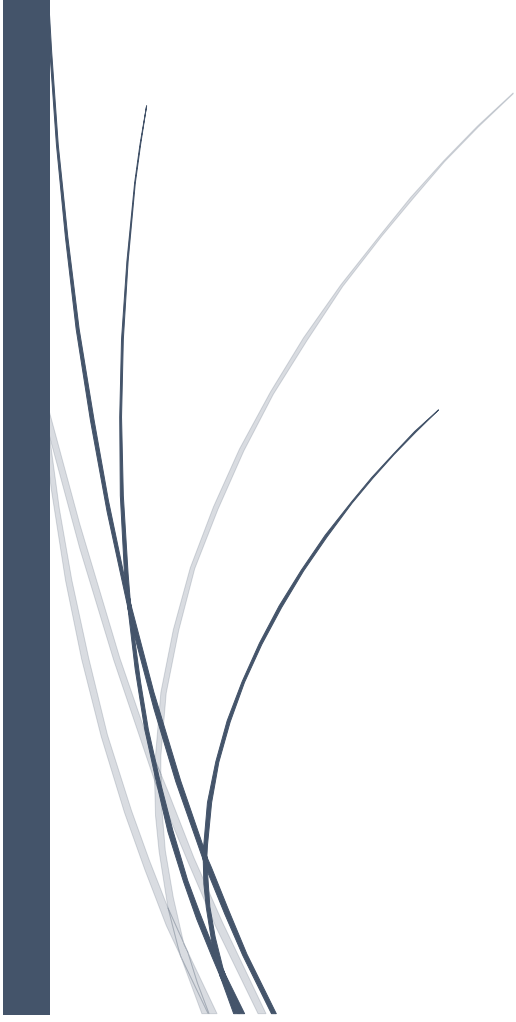
2013

Portugol

Equivalências de estruturas entre Portugol e Java



Decode Team
INSTITUTO POLITÉCNICO DE TOMAR

Índice

Nota Geral:

Devido à especificação da linguagem, a tradução só é possível depois de ser executado o fluxograma.

Algumas notas sobre Java:

- É case sensitive.
- Usa o ponto e virgula (;) para terminar uma linha de código.
- As funções podem ser definidas antes ou depois no main.
- O código dever ser guardado num ficheiro com o mesmo nome da classe e com extensão *.java*.
- A linguagem permite fazer a inclusão de bibliotecas através da instrução import. As bibliotecas devem ser incluídas imediatamente antes da definição da classe (ver *estrutura início*).
- A primeira função a ser codificada deve ser o início.

Estrutura Início

Início:


```
public class Programa {

    public static void main(String[] args) {

    Resto do programa


}
```

Nota: O nome da classe (*Programa*), é um nome que identifica o algoritmo que está a ser resolvido. Todo o código dever ser guardado no ficheiro Programa.java.

Estrutura Fim

Fim:

}

Variáveis

Equivalência entre TIPOS de variáveis

TIPO	Portugol	Java
Inteiro	Inteiro	Long
Real	Real	double
Texto	Texto	String
Caracter	Caracter	char
Lógico	Logico	boolean

Tabela 1 - Tipos de variáveis

Definição e atribuição de variáveis

variavel <- expressao

Se a variável não estiver definida em memória

Passo 1: Avaliar a expressão (VALOR).

Passo 2: Calcular Tipo do VALOR.

Passo 3: Declarar a variável: TIPO variavel = expressao;

Se a variável estiver definida em memória

variavel = expressao;

Alguns exemplos de definição e atribuição de variáveis

Existem duas formas de definir variáveis e proceder à sua atribuição.

- Long

  1 – Definir e atribuir variável no mesmo passo:

  Long variavel =valorL;

  2 – Definir e atribuir variável em passos separados:

  Long variavel;

  Variável=valorL;

  Nota 1: Pode ser definido como *Long* ou *long.*

  Nota 2: *valor* é um número inteiro.

  Nota 3: O tipo de dados long termina com L ou l.

- Double

  1 – Definir e atribuir variável no mesmo passo:

  Double variavel =valor;

  2 – Definir e atribuir variável em passos separados:

  Double variavel;

  variavel=valor;

  Nota 1: Pode ser definido como *Double* ou *double*

  Nota 2: *valor* é um número decimal. Ex: 5.3.

- String

  1 – Definir e atribuir variável no mesmo passo:

  String variavel="valor";

  2 – Definir e atribuir variável em passos separados:

  String variavel;
  variavel=”valor”;

  Nota 1: tem que ser definido com letra maiúscula.
  Nota 2: têm de ser usadas aspas.

- char

  1 – Definir e atribuir variável no mesmo passo:

char variavel =’X’;

2 – Definir e atribuir variável em passos separados:

Char variavel;

variavel=’X’;

Nota 1: tem que ser definido com letra minúscula

Nota 2: *X* é um caracter e deve estar dentro de pelicas.

- Boolean

1 – Definir e atribuir variável no mesmo passo:

Boolean variavel =false;

2 – Definir e atribuir variável em passos separados:

Boolean variavel;

variavel=false;

Nota 1: Pode ser definido como *Boolean* ou *boolean.*

Nota 2: Este tipo de dados pode assumir o valor *true* ou *false.*

Estruturas input/output

Input – Ler

Tipo	Java
Real	Double
Texto	Line
Lógico	Boolean
INT	Long
Char	Char

Tabela 2 - Tipo de variáveis para leitura

Se for a primeira vez a ler do teclado

Passo 1: É necessário é fazer o import da biblioteca java.util.Scanner:

```
import java.util.Scanner;
```

Passo 2: Depois da definição da classe (ver *estrutura início*), definir uma variável para ler o teclado:

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

Se a variável não estiver definida em memória

Passo 1: Identificar o tipo (TIPO) de dados que foi lido.

Passo 2: Definir a variável:

```
TIPO variavel;
```

Passo 3: variavel=scanner.nextTipo();

Se a variável já estiver definida em memória

Passo 1: Realizar apenas o ***Passo 3*** do ponto anterior.

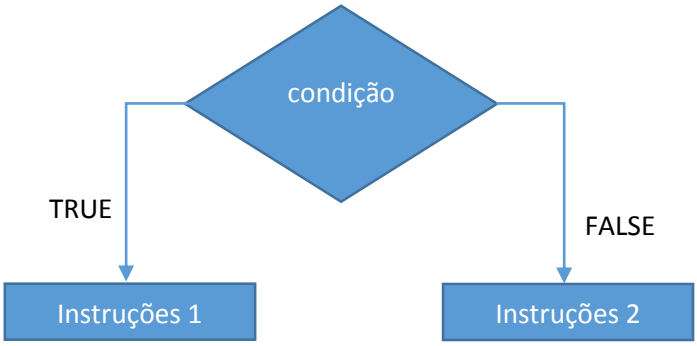
Output – Escrever

Para escrever no ecrã:

System.out.print((expressão) + “”);

Estruturas de Decisão

Condição *“if”* e *“if else”*



Para TRUE, escrever:

If (*condição*) {

Instruções 1

Para FALSE:

Se Instruções 2 for igual a ● (conector) não fazer nada.

Senão, escrever:

} else {

Instruções 2

Exemplos práticos

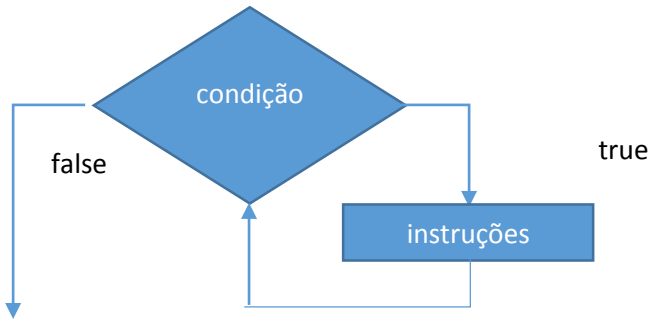
Condição "if"

```
if (n%2==0) {
        System.out.println("Par");
}
```

Condição "if else"

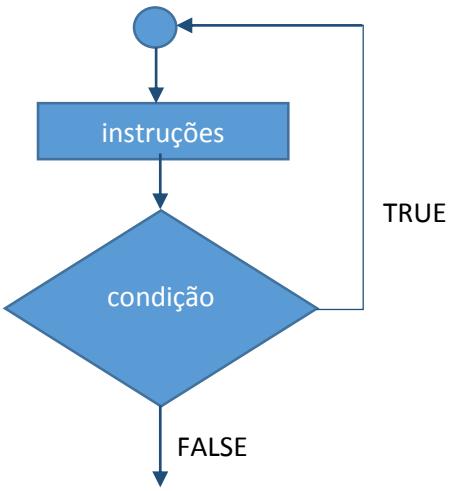
```
if (n % 2 == 0) {
        System.out.println("Par");
} else {
        System.out.println("Impar");
}
```

Condição *"while"*




```
While (condição){
        Instruções
}
```

Condição *"do while"*



Instruções

}while(*condição*);

Exemplos práticos

Condição "while"

```
while(i<=10){
        System.out.print(i);
        i++;
}
```

Condição "do while"

```
do{
        i=scanner.nextInt();
}while(i<0);
```

Estrutura Conector

Conector

Se for uma condição "*do while*" escrever:

do {

Senão, escrever:

}

Funções

Definir funções

Nota: Depois da função ser executada pelo menos uma vez (ver *Algumas notas sobre Java*), o tipo de retorno das função RETURN_TIPO e o TIPOx dos parametros pode ser identificado:

```
public static RETURN_TIPO exemplo( TIPO1 a , TIPO2 b, . . .. )
{
```

Definir função *Exemplo* sem parâmetros de entrada

```
public static TIPO NOME ()   {
```

Definir função *Exemplo* com parâmetros de entrada

```
public static TIPO NOME (PARAMETRO)   {
```

TIPO – Executa a função e calcula o tipo de retorno.
Consultar *tabela 1* no ponto *Equivalência entre TIPOS de variáveis*.

NOME – Nome dado à função.

PARAMETRO – Variável utilizada pela função para auxiliar o cálculo.

Chamada de funções


```
variavel = NOME(PARAMETRO);
```

Estrutura de retorno

Return

return expressao;

Operadores

Aritméticos

Nome	Portugol	Java
Adição	a + b	a + b
Subtração	a – b	a - b
Divisão	a / b	a / b
Multiplicação	a * b	a * b
Resto da divisão inteira	a % b	a % b
Potenciação	a ^ b	Math.pow(base,expoente);
Concatenação de texto	,	+

Tabela 3 - Equivalência de operadores aritméticos

Lógicos

Nome	Portugol	Java
Disjunção	a E b	a && b
Conjunção	a OU b	a || b
Conjunção Exclusiva	a XOU b	a ^ b
Negação	NAO a	! a

Tabela 4 - Equivalência de operadores lógicos

Relacionais

Nome	Portugol	Java
Igual	a = b	a == b
Diferente	a =/= b	a != b
Maior	a > b	a > b
Maior ou igual	a >= b	a >= b
Menor	a < b	a < b
Menor ou igual	a <= b	a <= b

Tabela 5 - Equivalência de operadores relacionais

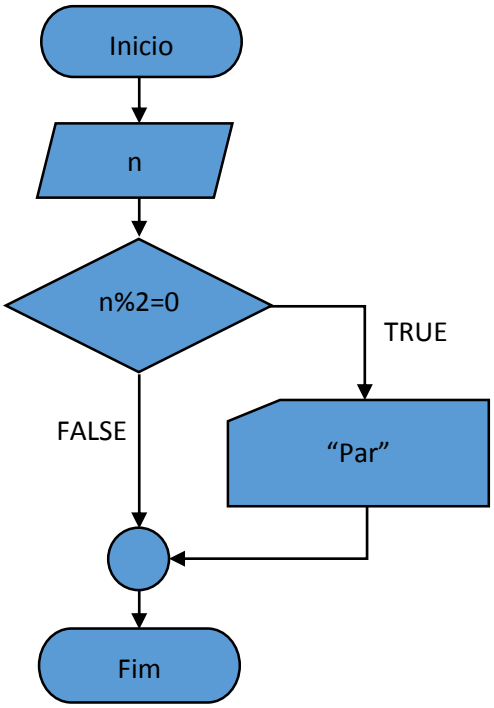
ANEXO

Para uma compreensão mais abrangente do uso das estruturas, ficam alguns exemplos mais extensivos, com o uso de várias estruturas em algoritmos completos.

Algoritmo com o uso da condição "if"

Problema: Verificar se um número introduzido pelo utilizador é par.

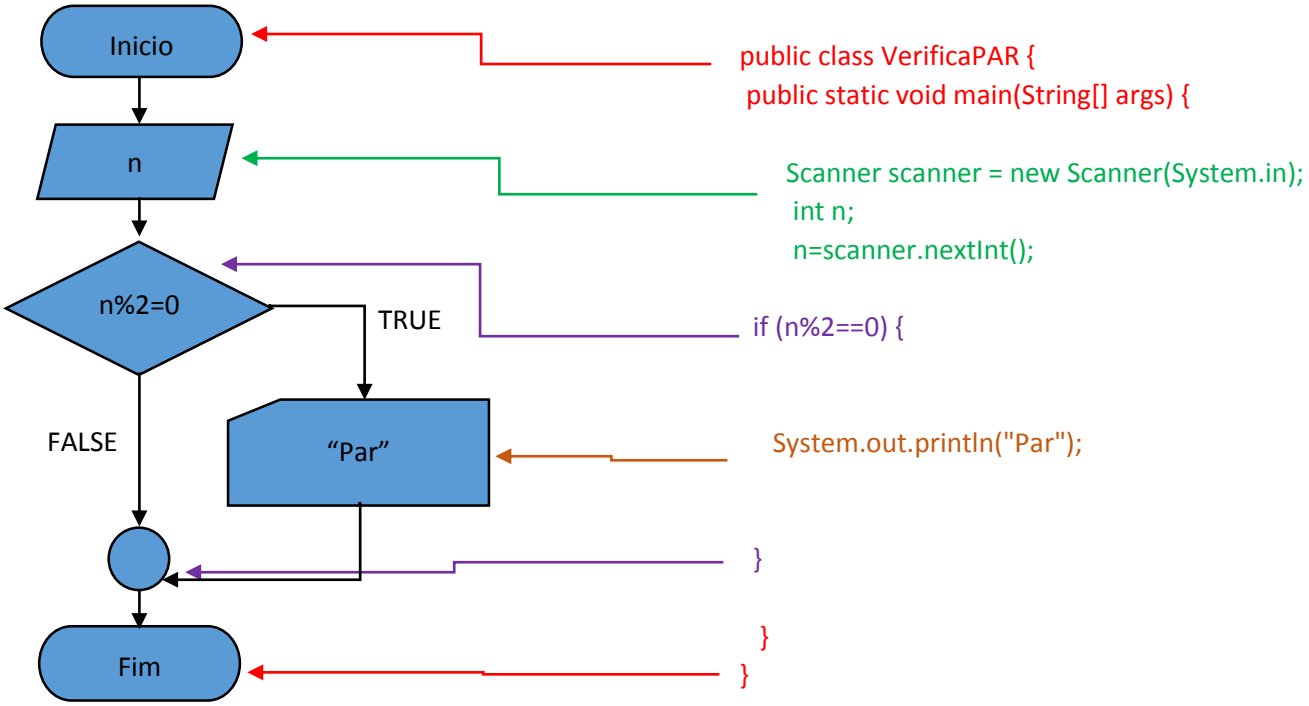
Fluxograma



Código

```
import java.util.Scanner;

public class VerificaPAR {

   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      int n;
      n=scanner.nextInt();
      if (n%2==0) {
         System.out.println("Par");
      }
   }
}
```

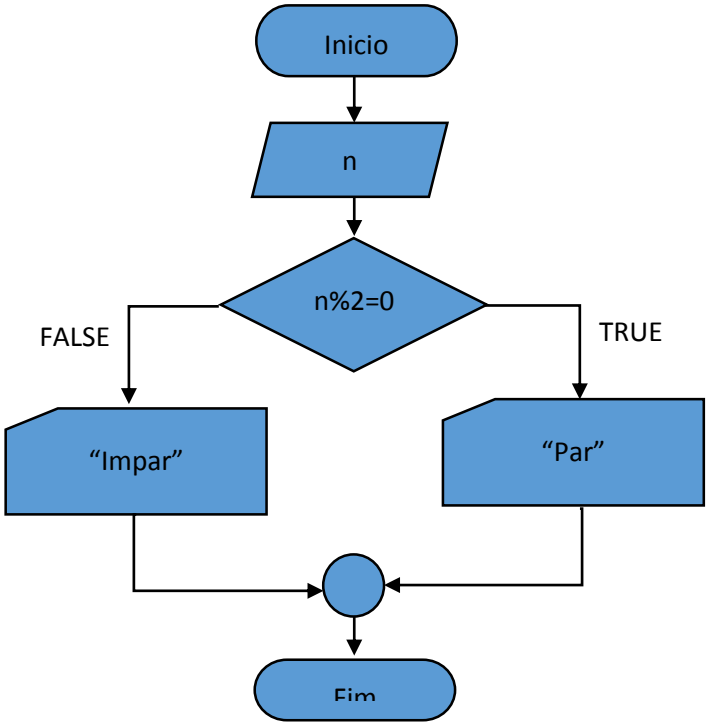
Esquema detalhado



Algoritmo com o uso da condição "if else"

Problema: Verificar se um número introduzido pelo utilizador é par ou ímpar.

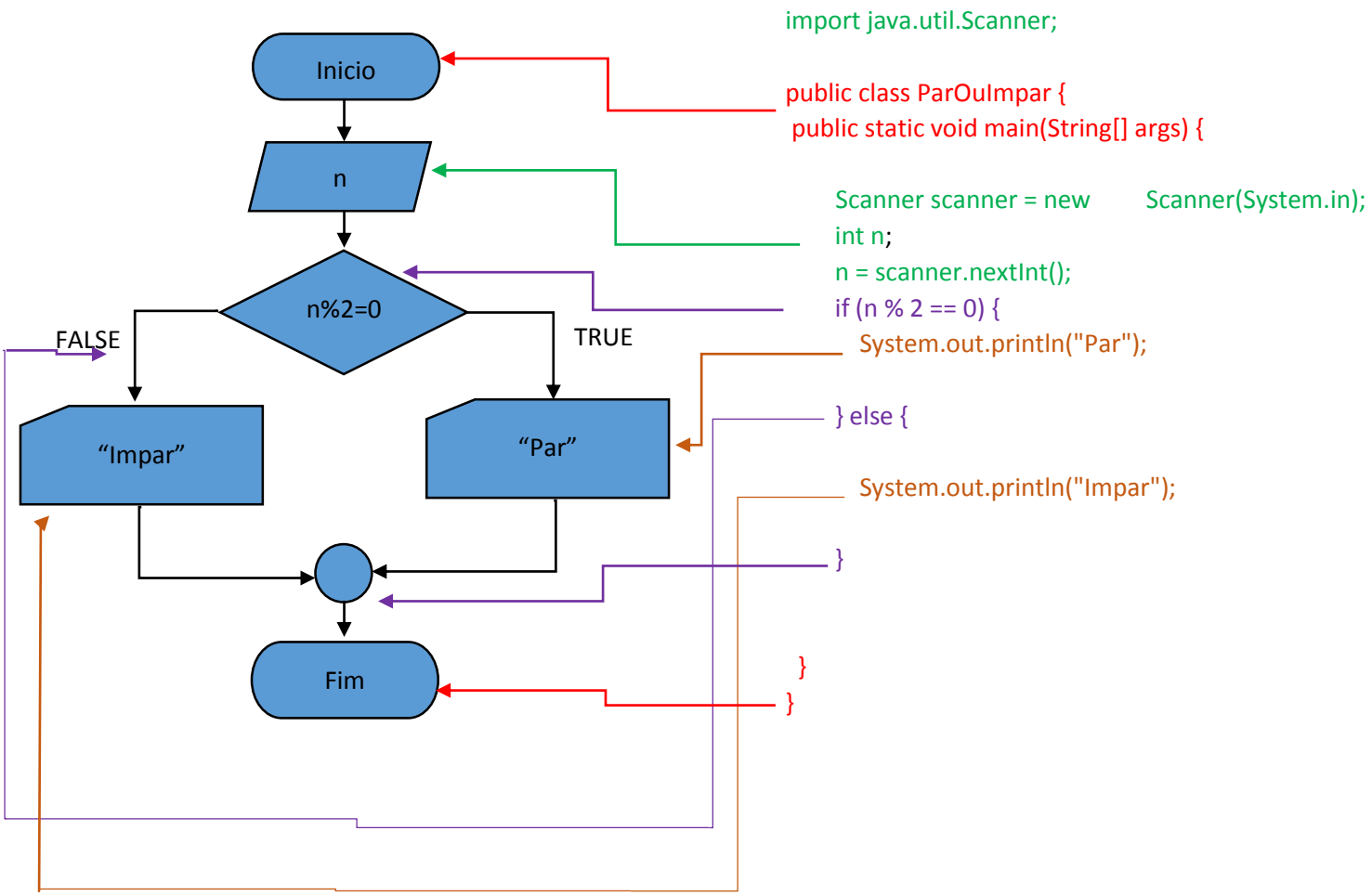
Fluxograma



Código:

```
import java.util.Scanner;

public class ParOuImpar {

   public static void main(String[] args) {

      Scanner scanner = new Scanner(System.in);
      int n;
      n = scanner.nextInt();
      if (n % 2 == 0) {
         System.out.println("Par");
      } else {
         System.out.println("Impar");
      }

   }

}
```

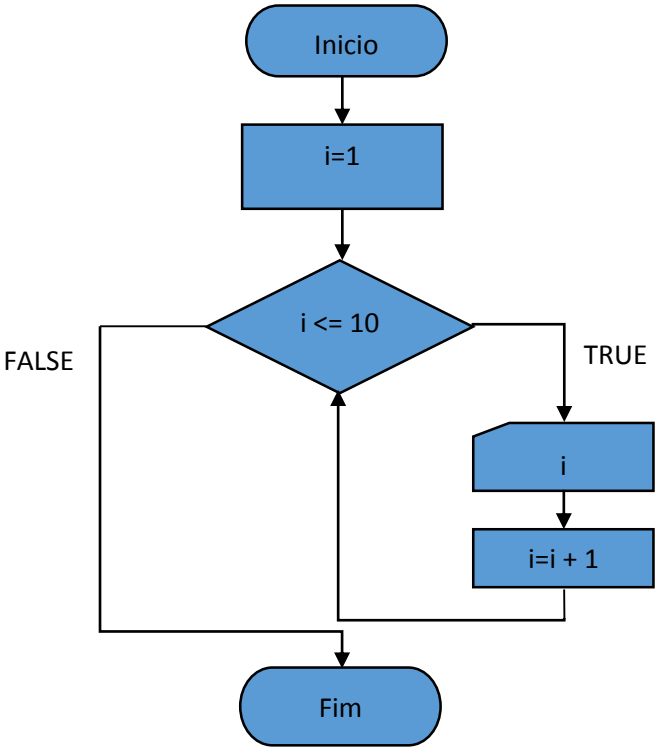
Esquema detalhado



Algoritmo com o uso da condição “while”

Problema: Escrever um número de 1 a 10.

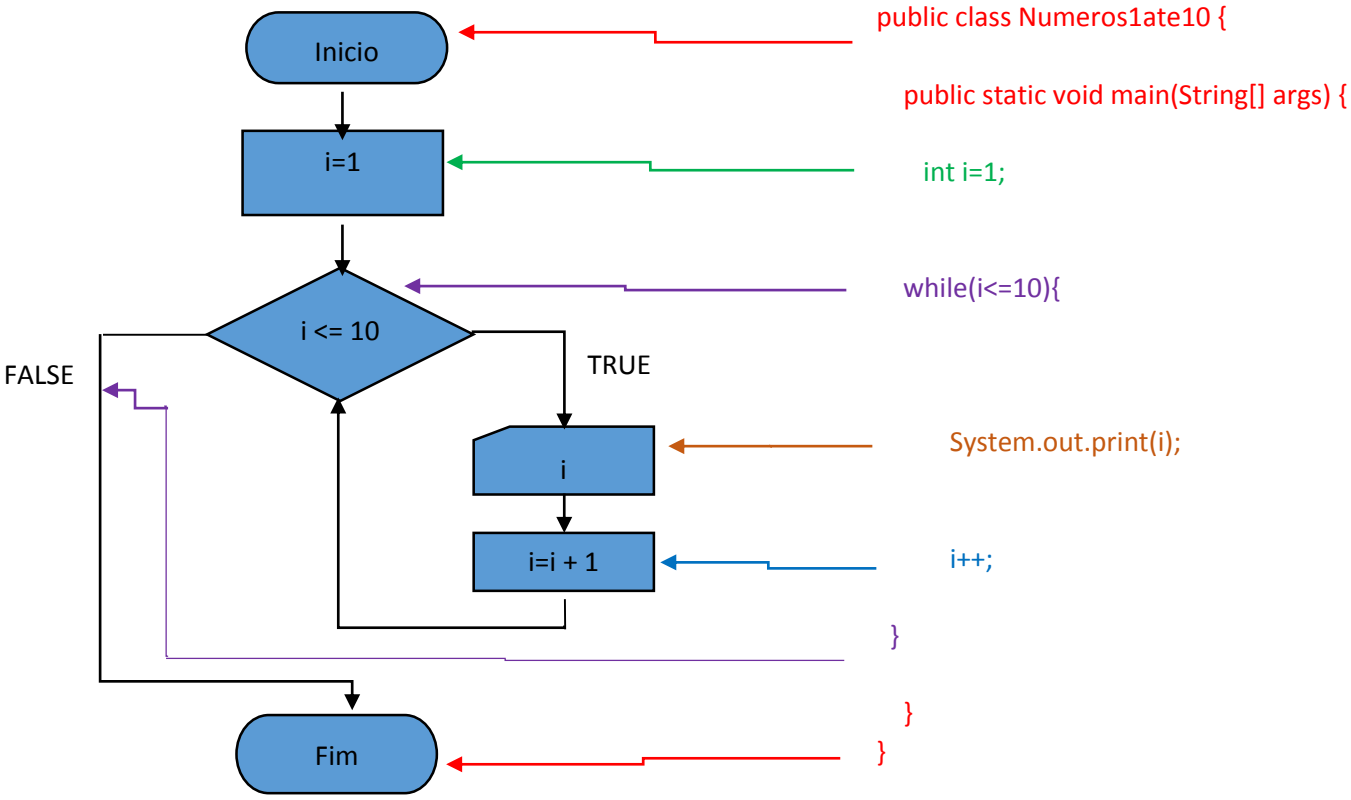
Fluxograma



Código

```
public class Numeros1ate10 {

   public static void main(String[] args) {

      int i=1;
      while(i<=10){
         System.out.print(i);
         i++;
      }

   }

}
```

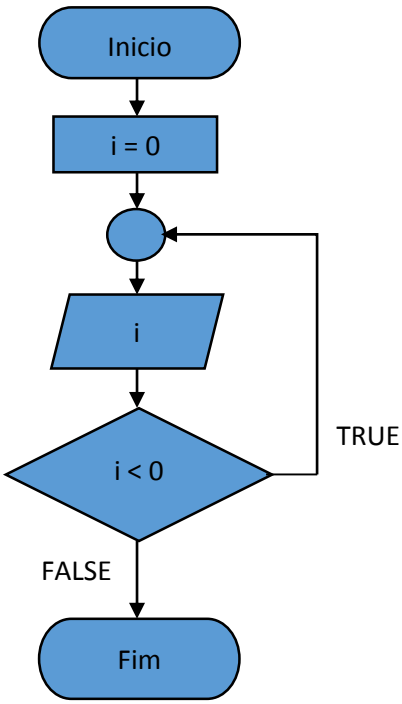
Esquema detalhado



Algoritmo com o uso da condição " do while"

Problema: Pedir um número positivo.

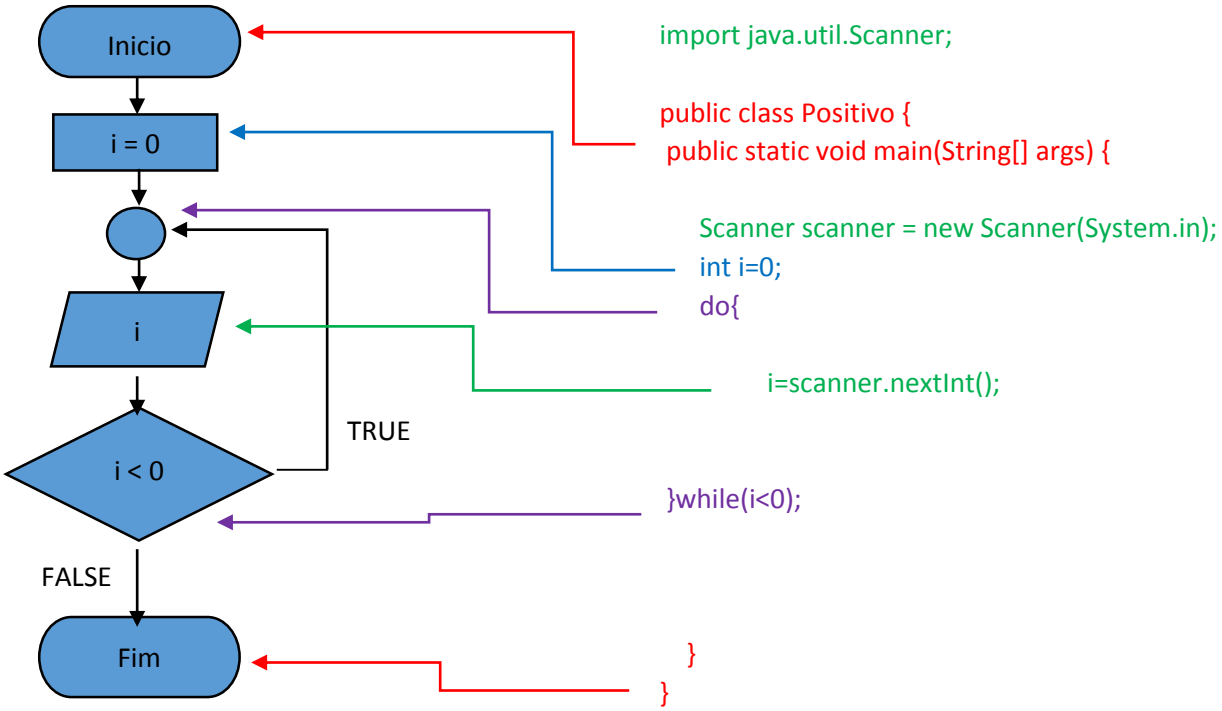
Fluxograma



Código

```
import java.util.Scanner;

public class Positivo {

   public static void main(String[] args) {

     Scanner scanner = new Scanner(System.in);
     int i=0;
     do{
        i=scanner.nextInt();
     }while(i<0);

   }

}
```

Esquema detalhado

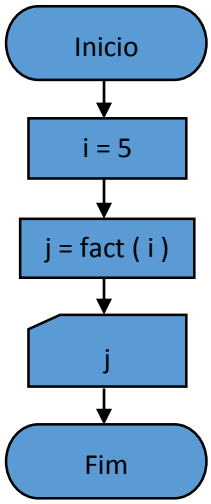


Algoritmo com o uso de uma função

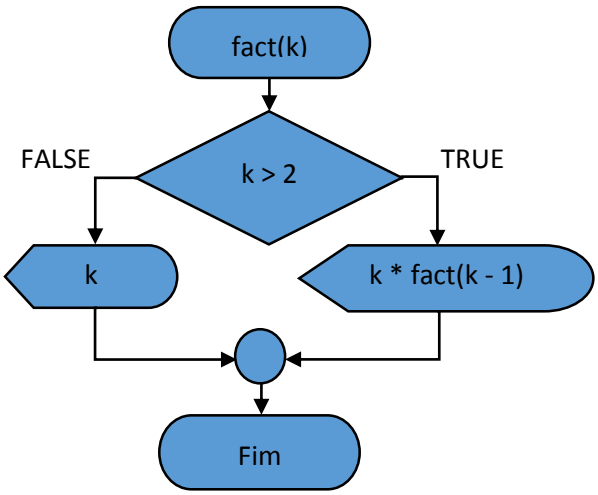
Problema: Factorial de um número.

Fluxogramas

Código principal



Função fact(k)

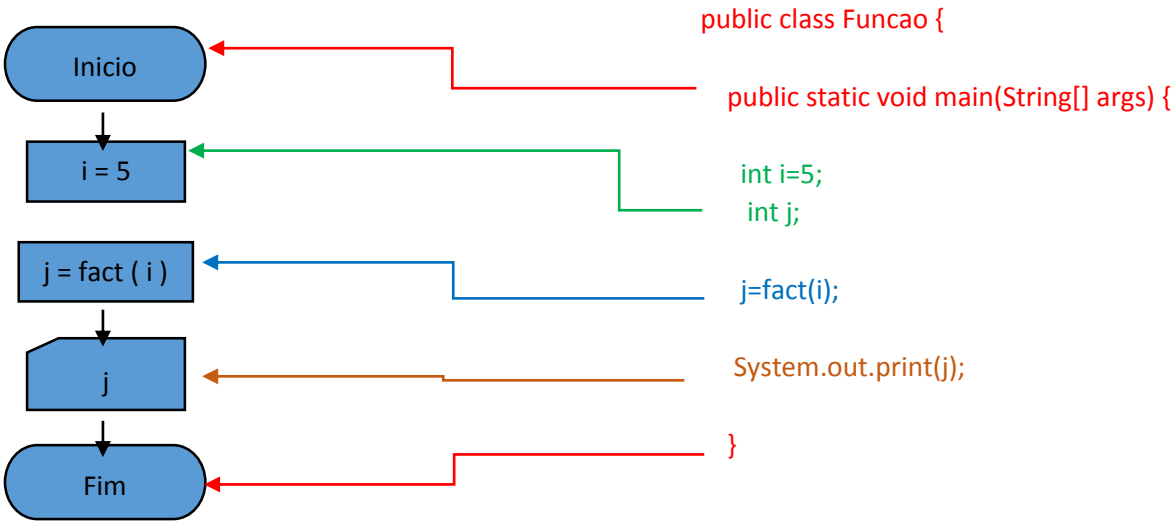


Código

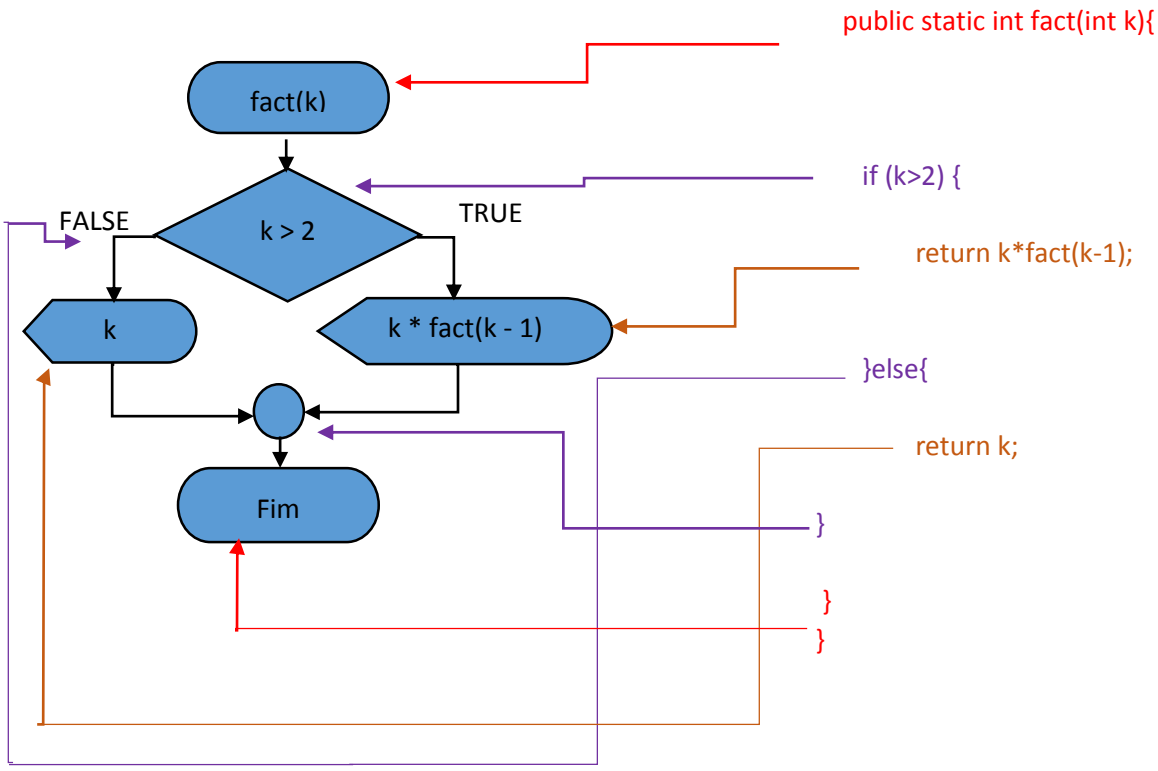
```
public class Funcao {

   public static void main(String[] args) {

    int i=5;
    int j;
    j=fact(i);
    System.out.print(j);

   }

   public static int fact(int k){

     if (k>2) {
       return k*fact(k-1);
     }else{
       return k;
     }
   }

}
```

Esquema detalhado

Função Main



Função fact


public static int fact(int k){
fact(k)
if (k>2) {
FALSE
k > 2
TRUE
return k*fact(k-1);
k
k * fact(k - 1)
}else{
return k;
Fim
}
}
}