

# **Capítulo 04:**

# **Estruturas de Controle**

Leonardo Moura Leitão

Marcelo Gonçalves Pinheiro Quirino

# Agenda

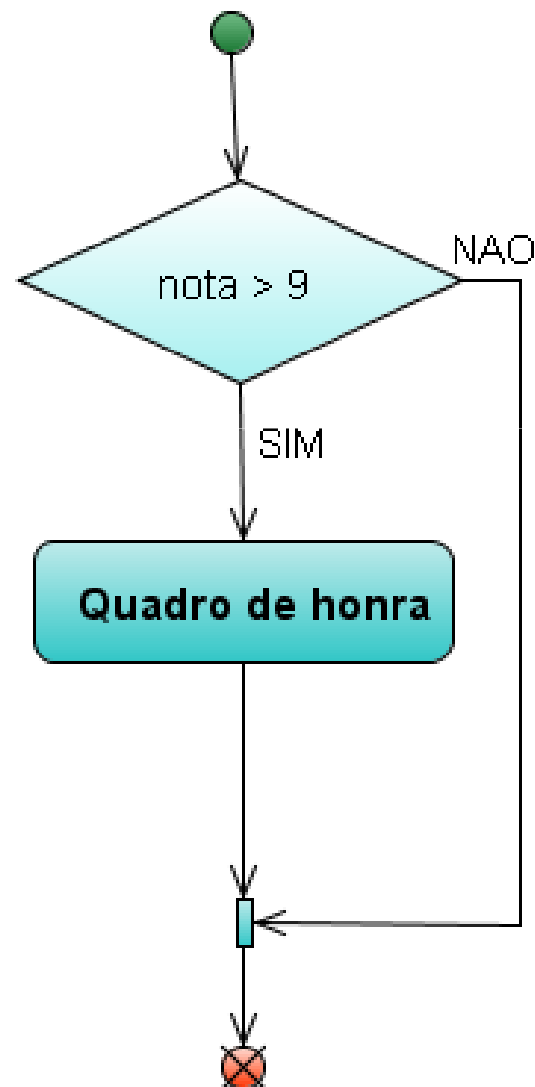
- Estruturas de controle
  - Se
  - Se/Senão
  - Se/Senão Se
  - Enquanto/Faça
  - Faça/Até
  - Para (N iterações)
  - Seleções Múltiplas

# Agenda

- Estruturas de transferência
  - break
  - continue
  - return

# Estruturas de Controle

- Se



# Estruturas de Controle

- Se

```
if ( condição )  
{  
  
}
```

```
if ( ok ) facaAlgo();
```

```
if ( ok )  
    facaAlgo();
```

```
if ( ( x > y ) && teste() )  
{  
  
}
```

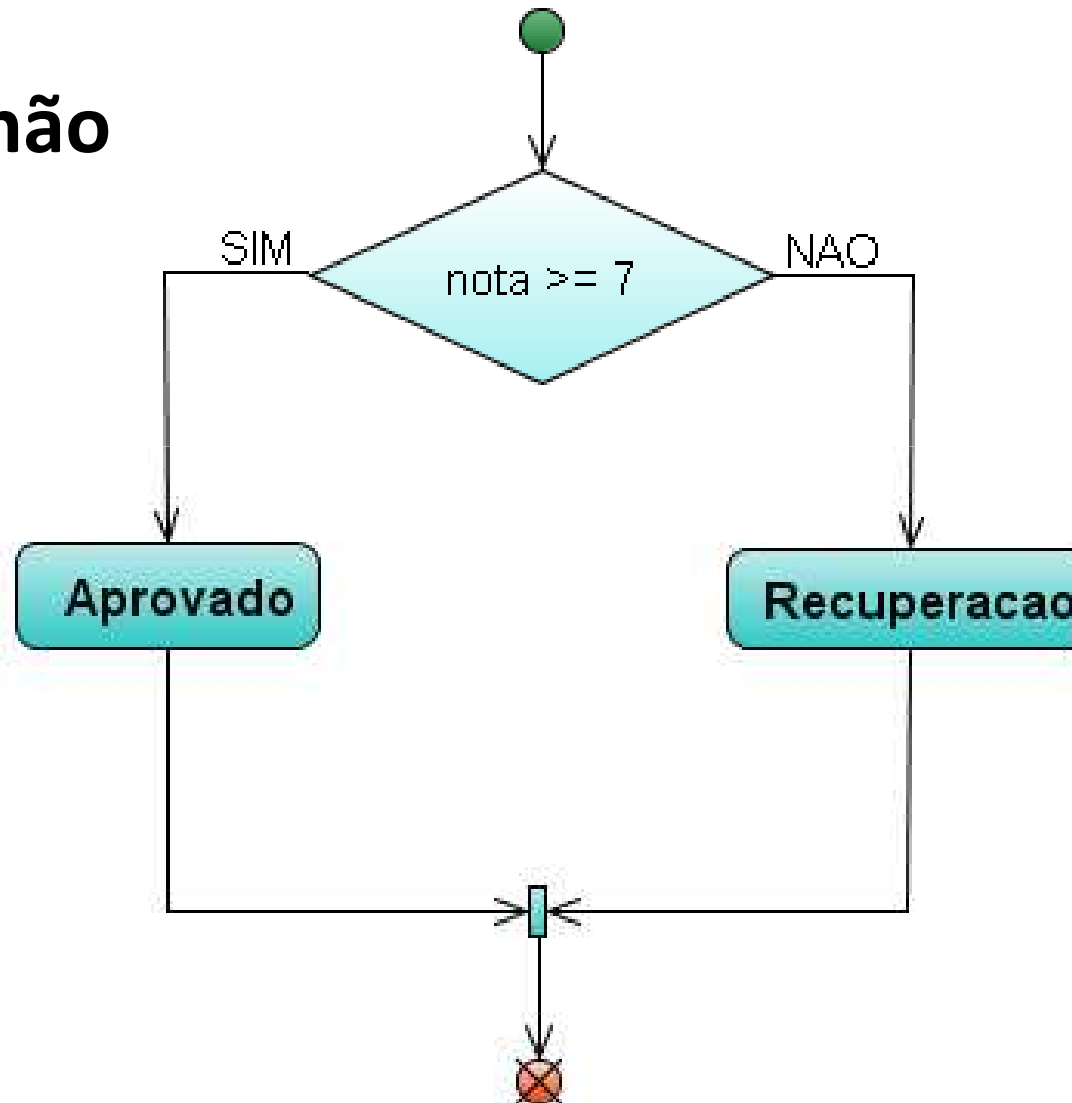
```
if ( ok );  
    facaAlgo();
```



Fora do laço

# Estrutura de controle

- Se/Senão



# Estruturas de Controle

- Se/Senão

```
if ( condição )  
{  
    ...  
}  
else  
{  
    ...  
}
```

```
if ( ok ) facaAlgo();  
else facaOutraCoisa();
```

```
if ( ok )  
    facaAlgo();  
else  
    facaOutraCoisa();
```

# Estruturas de Controle

- Se/Senão

```
if ( x > y )  
    if ( ok )  
        executar( 1 );  
    else  
        executar( 2 );
```

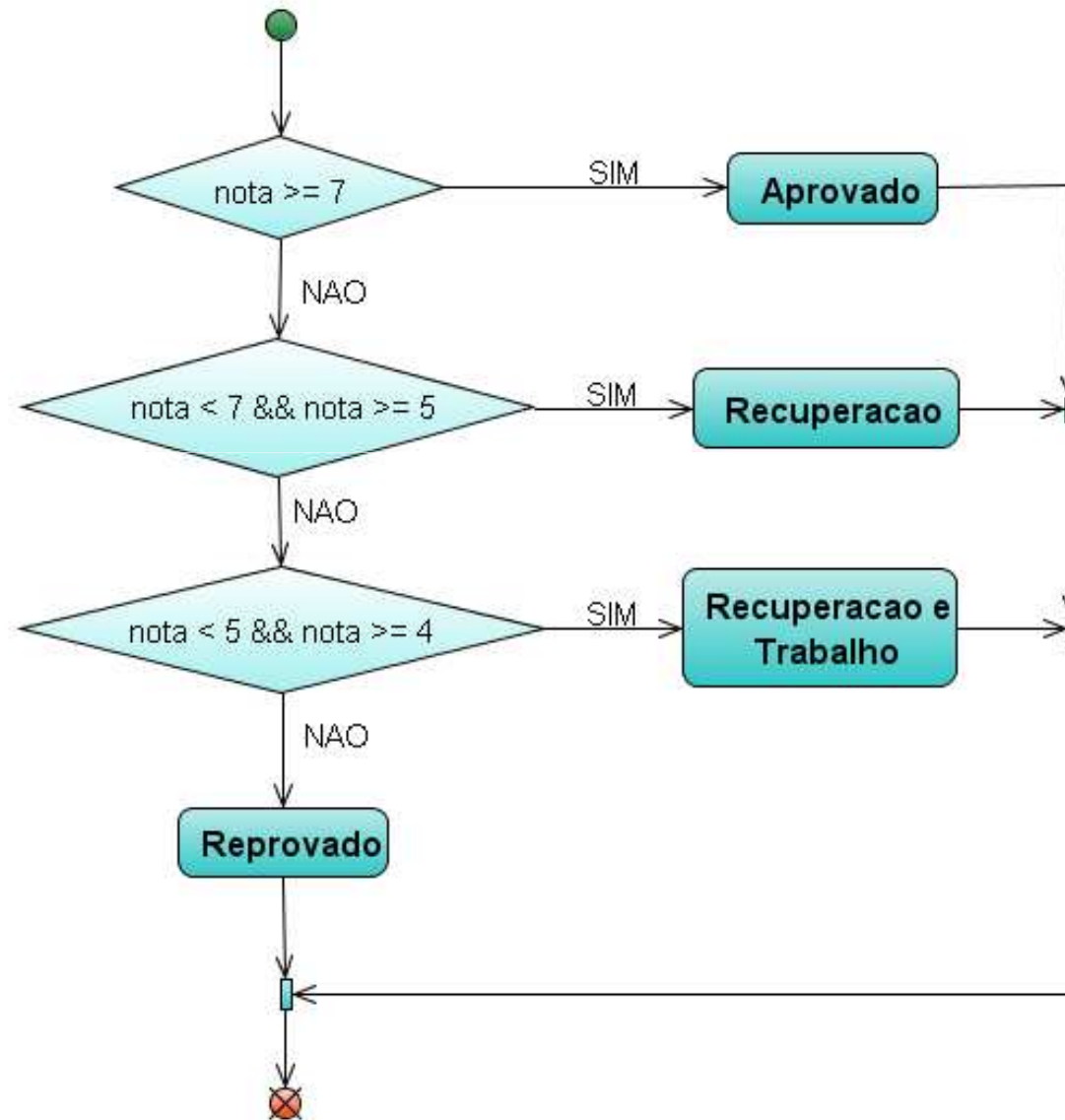
```
if ( x > y )  
    if ( ok )  
        executar( 1 );  
    else  
        executar( 2 );  
else  
    executar( 3 );
```

```
if ( x > y )  
{  
    if ( ok )  
        executar( 1 );  
}  
else  
    executar( 2 );
```



# Estrutura de controle

- Se/Senão Se



# Estruturas de Controle

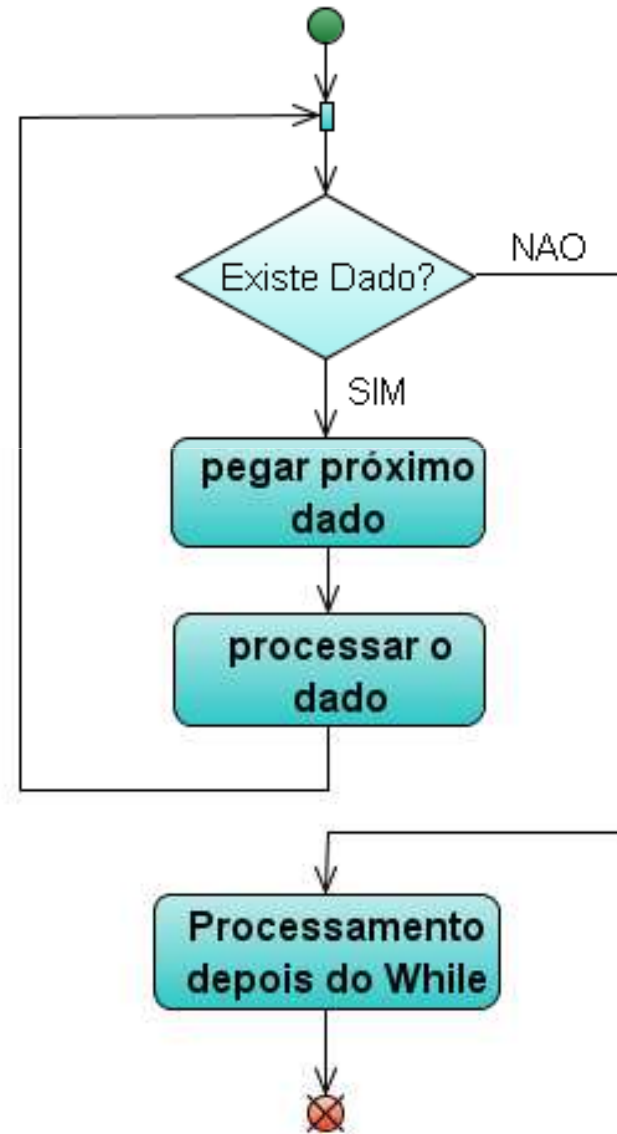
- Se/Senão Se

```
if ( condição )  
{  
  
}  
else if ( condição2 )  
{  
  
}  
else  
{  
  
}
```

```
if ( ok )  
    executar( 1 );  
else if ( ok2 )  
    executar( 2 );  
else  
    executar( 3 );
```

# Estrutura de controle

- Enquanto/Faça



# Estruturas de Controle

- Enquanto/Faça

```
while ( condição )  
{  
  
}
```

```
while ( ok )  
{  
    executar( 1 );  
    facaAlgo();  
    executar( 2 );  
}
```

```
while ( ok )  
    facaAlgo();
```

```
while ( ok );  
    facaAlgo();
```



Fora do laço

# Estrutura de controle

- Faça/Até



# Estruturas de Controle

- Faça/Até

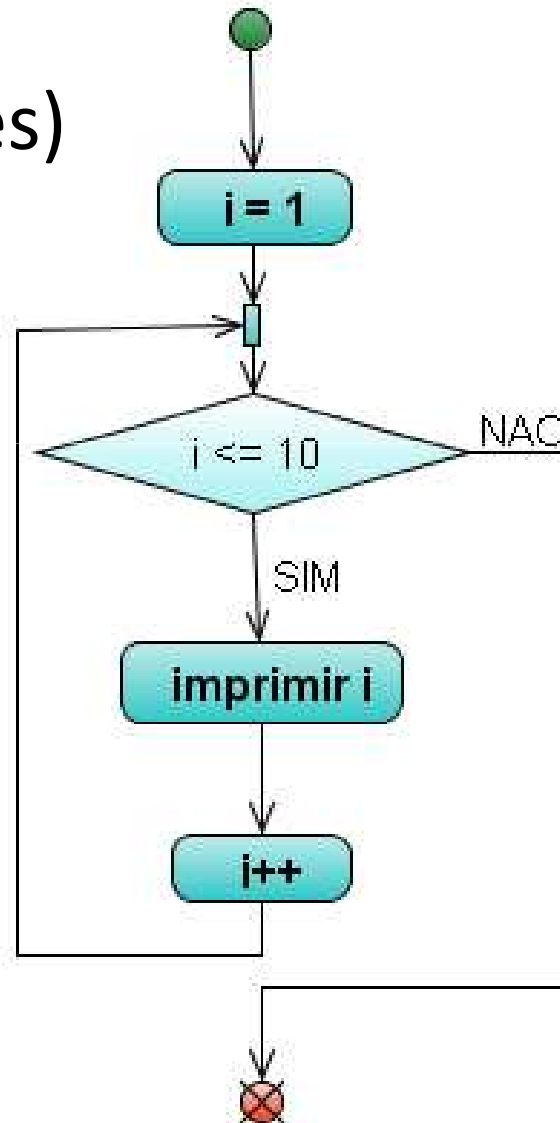
```
do  
{  
  
}  
while ( condição );
```

```
do  
{  
    executar( 1 );  
    facaAlgo();  
    executar( 2 );  
}  
while ( x > y );
```

```
do  
    facaAlgo();  
while ( teste() );
```

# Estrutura de controle

- Para (N iterações)



# Estruturas de Controle

- Para (N iterações)

```
for( inicializações;  
    condição;  
    incrementos )  
{  
  
}
```

```
for ( int i = 0; i < 10; i++ )  
{  
    facaAlgo();  
}
```

```
for ( ;; )  
    facaAlgo();
```

Laço infinito

```
for ( int i = 0; i < 10; i++ );  
    facaAlgo();
```

Fora do laço



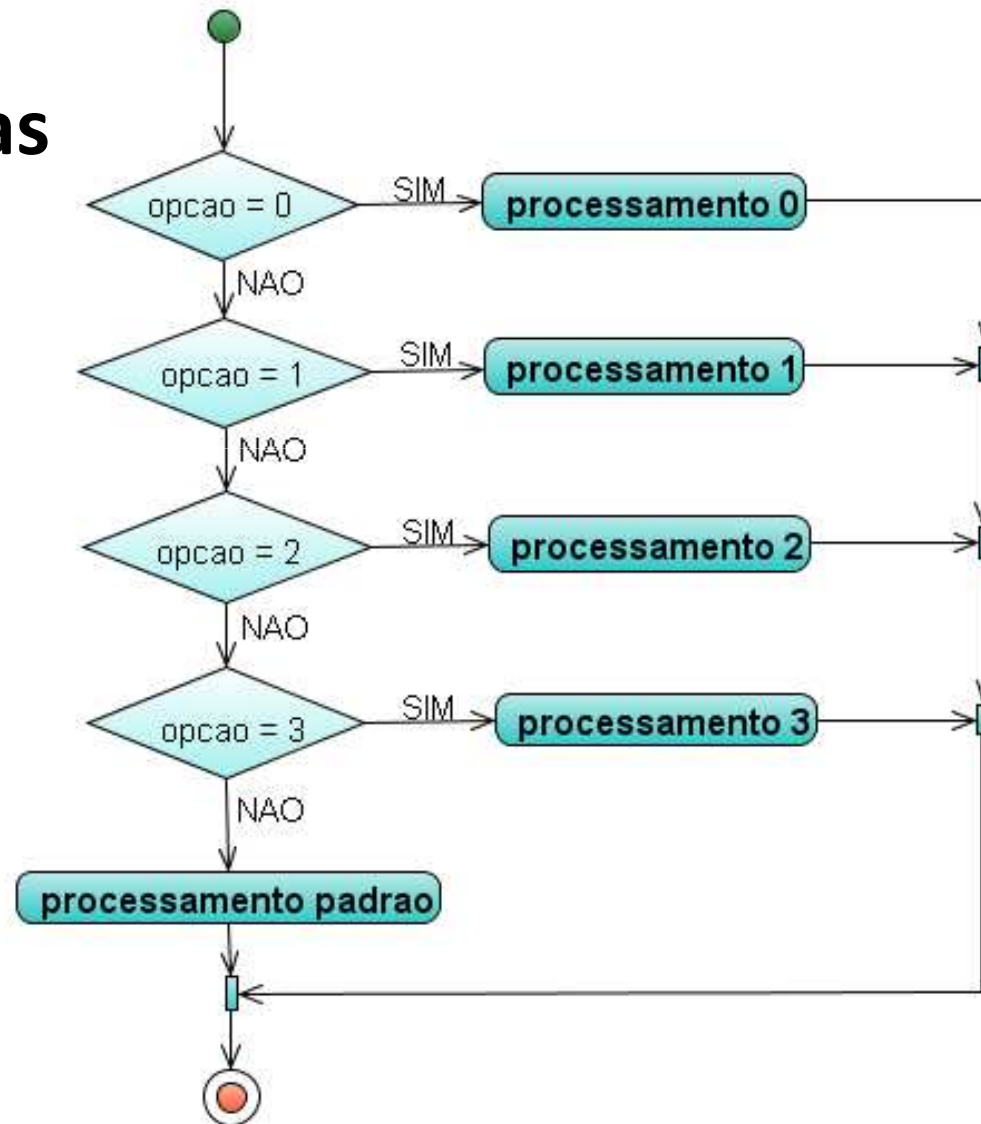
# Estruturas de Controle

- Para (N iterações)

```
int c = 0;
for ( int a = 0, b = 1;
      a < 10 && b < 11 && c < 30;
      a++, ++b, c += 3 )
{
    facaAlgo( );
    facaOutraCoisa( );
}
```

# Estrutura de controle

- Seleções Múltiplas



# Estruturas de Controle

- Seleções Múltiplas

```
switch ( chave )
{
    case valor1:
    {
        ...
        break;
    }
    case valor2:
    {
        ...
        break;
    }
    default:
    {
        ...
        break;
    }
}
```

```
switch ( x )
{
    case 2:
        System.out.println("2");
        break;
    case 1:
        System.out.println("1");
        break;
    default:
        System.out.println("P");
        break;
    case 3:
        System.out.println("3");
        break;
}
```

Resultado:

valor = 1 --> 1

valor = 2 --> 2

valor = 3 --> 3

valor = 4 --> P

# Estruturas de Controle

- Seleções Múltiplas

```
switch ( x )
{
    case 0: case 2: case 4: case 6: case 8:
        System.out.println( "Par" );
        break;
    case 1:
    case 3:
    case 5:
    case 7:
    case 9:
        System.out.println( "Ímpar" );
        break;
}
```

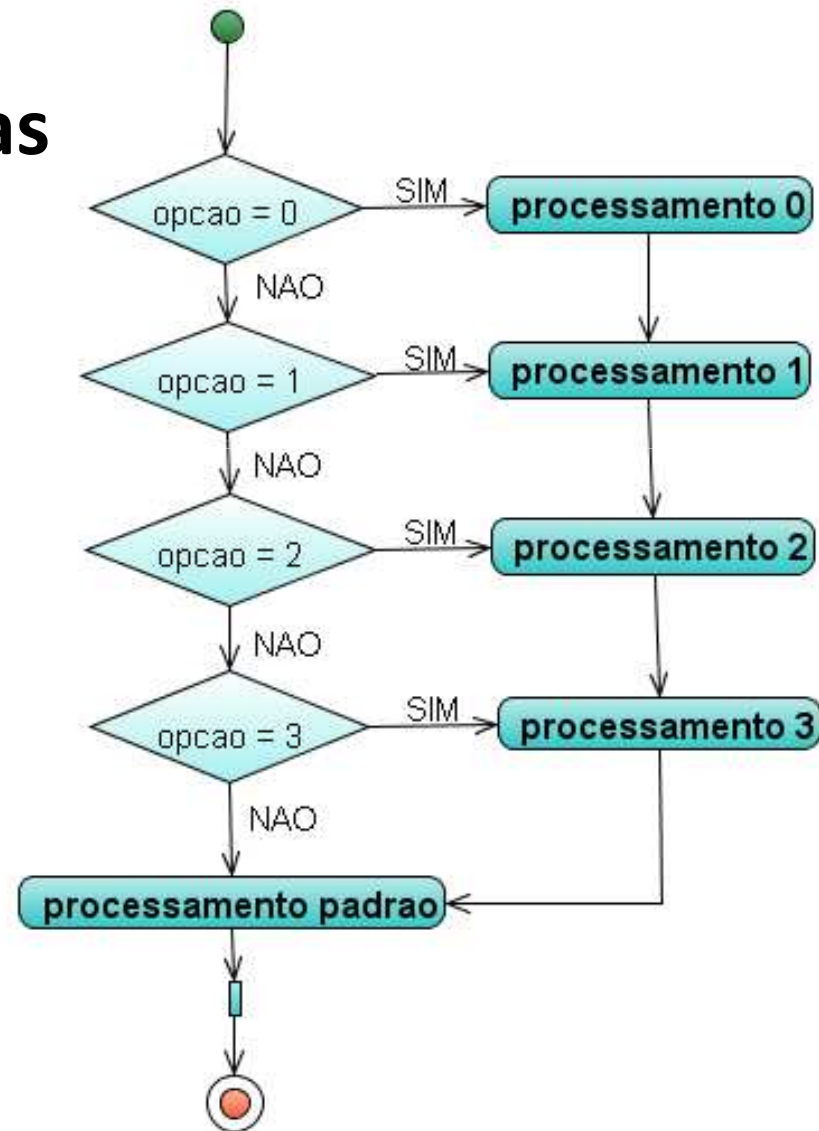
Resultado:

valor = 0, 2, 4, 6, 8--> Par

valor = 1, 3, 5, 7, 9--> Ímpar

# Estrutura de controle

- Seleções Múltiplas



# Estruturas de Controle

- Seleções Múltiplas

```
switch ( chave )
{
    case valor1:
    {
    }
    case valor2:
    {
    }
    default:
    {
    }
}
```

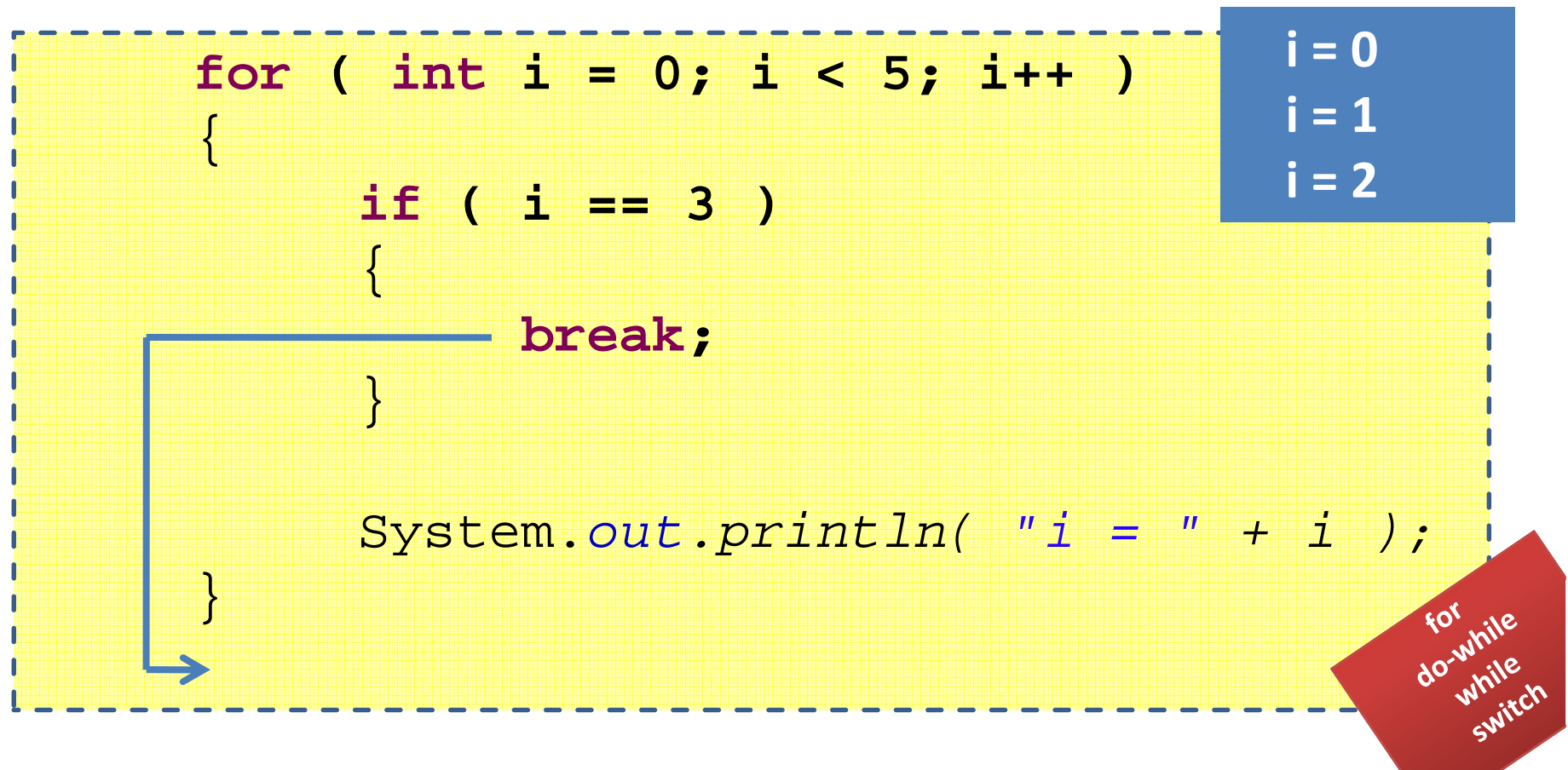
```
switch ( x )
{
    case 2:
        System.out.println("2");
    case 1:
        System.out.println("1");
    default:
        System.out.println("P");
    case 3:
        System.out.println("3");
}
```

Resultado:

valor = 1 --> 1P3  
valor = 2 --> 21P3  
valor = 3 --> 3  
valor = 4 --> P3

# Estruturas de transferência

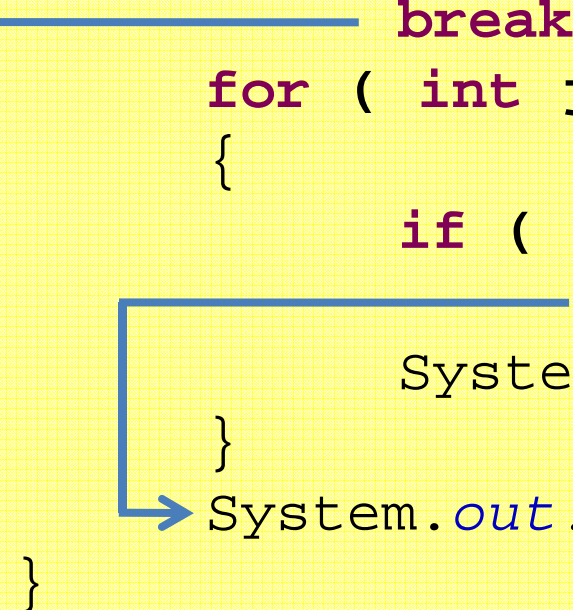
- **break** – transfere o fluxo para fora do contexto corrente.



# Estruturas de transferência

- break

```
for ( int i = 0; i < 4; i++ )  
{  
    if ( i == 2 )  
        break;  
    for ( int j = 0; j < 4; j++ )  
    {  
        if ( j == 2 )  
            break;  
        System.out.println("j = " + j);  
    }  
    System.out.println("i = " + i);  
}
```



j = 0  
j = 1  
i = 0  
j = 0  
j = 1  
i = 1

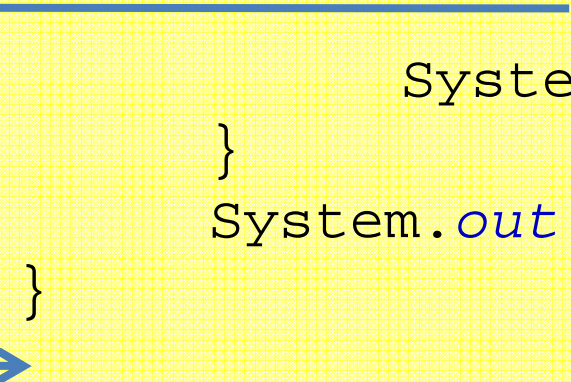


# Estruturas de transferência

- break rotulado

j = 0  
j = 1

```
ponto1:
for ( int i = 0; i < 4; i++ )
{
    for ( int j = 0; j < 4; j++ )
    {
        if ( j == 2 )
            break ponto1;
        System.out.println("j = " + j);
    }
    System.out.println("i = " + i);
}
```



# Estruturas de transferência

- **continue** – interrompe prematuramente o fluxo, avançando para a próxima iteração.

```
for ( int i = 0; i < 5; i++ )  
{  
    if ( i == 3 )  
    {  
        continue;  
    }  
    System.out.println( "i = " + i );  
}
```

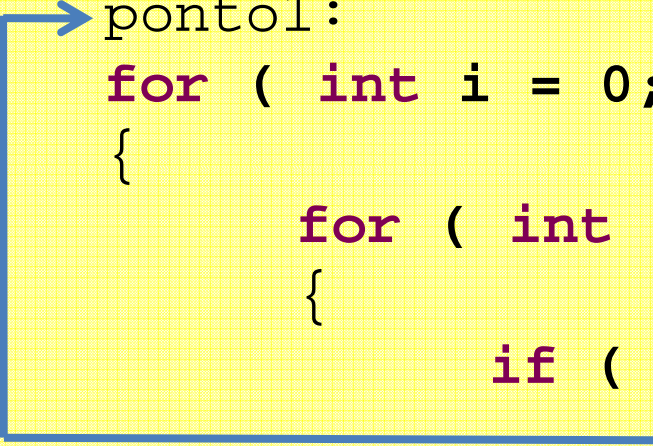
i = 0  
i = 1  
i = 2  
i = 4

for  
do-while  
while  
switch

# Estruturas de transferência

- continue rotulado

```
ponto1:
for ( int i = 0; i < 3; i++ )
{
    for ( int j = 0; j < 4; j++ )
    {
        if ( j == 2 )
            continue ponto1;
        System.out.println("j = " + j);
    }
    System.out.println("i = " + i);
}
```




j = 0  
j = 1  
j = 0  
j = 1  
j = 0  
j = 1

# Estruturas de transferência

- **return** - transfere o controle para o método que o chamou.

```
private void calc ()  
{  
    for ( int i = 0; i < 5; i++ )  
    {  
        if ( i == 3 )  
            return;  
    }  
}  
  
...  
calc();
```



A blue line originates from the `return;` statement inside the `calc()` method, extends to the right, then turns down and left, ending with an arrowhead pointing to the `calc();` call in the code below the method definition. This illustrates how the `return` statement transfers control back to the caller.