Linguagem de Programação II

Estruturas de controle e repetição

Universidade do Estado do Rio de Janeiro-UERJ
Instituto de Matemática e Estatística-IME
Ciência da Computação
Professor: Alexandre Sztajnberg

Estruturas de repetição e controle

- Laços ou *loops*
  - Executar, repetidamente, uma instrução ou conjunto de instruções enquanto certa condição estiver sendo satisfeita
    - `for`
    - `while` / `do-while`
- Estruturas de controle
  - Executar seletivamente ou condicionalmente uma instrução ou conjuntos de instrução mediante um critério de seleção
    - `if-else`
    - `switch-case`

Estruturas de repetição: `for`

- Executar um número específico de vezes um mesmo bloco de código, enquanto uma condição é satisfeita (pode ser o número de vezes)

```
for(inicialização; condição; incremento/decremento){
    <bloco de instruções>
}
```

- **inicialização**: declaração ou inicialização de uma ou mais variáveis. Executada apenas uma vez, antes do início do loop
- **condição de execução**: condição para que o laço seja repetido. Recomendável utilizar a variável/as variáveis declaradas na inicialização. Avaliada a cada passada do loop. A execução é encerrada quando a condição de execução for avaliada como falsa
- **incremento/decremento**: utilizada para fazer operações sobre as variáveis que controlam a repetição (adicionar, subtrair, multiplicar, entre outras) declaradas na `inicialização`. Mas outras variáveis podem ser envolvidas. Não é obrigatório. Executada a cada passada do loop

Estruturas de repetição: `for`

```
public class TesteFor
{
   public static void main(String[] args)
   {
     int n=Integer.parseInt(args[0]);
     int i, soma;
         //(inicialização; condição; incremento)
     for(i=0, soma=0; i<n; i++)
     {
        System.out.print("Soma = "+soma+"+"+i);
        soma+=i;
        System.out.println("= "+soma+"\n");
      }
   }
```

Sendo n = 10..

ESTADO DAS VARIÁVEIS

iteração	i	soma(antes)	soma(depois)
1	0		
2			
3			
4			
5			
6			
7			
8			
9			
10			

Estruturas de repetição: `for`

```
public class TesteFor
{
   public static void main(String[] args)
   {
     int n=Integer.parseInt(args[0]);
     int i, soma;
         //(inicialização; condição; incremento)
     for(i=0, soma=0; i<n; i++)
     {
        System.out.print("Soma = "+soma+"+"+i);
        soma+=i;
        System.out.println("= "+soma+"\n");
      }
   }
```

ESTADO DAS VARIÁVEIS

iteração	i	soma(antes)	soma(depois)
1	0	0	
2			
3			
4			
5			
6			
7			
8			
9			
10			

Estruturas de repetição: `for`

```
public class TesteFor
{
   public static void main(String[] args)
   {
     int n=Integer.parseInt(args[0]);
     int i, soma;
         //(inicialização; condição; incremento)
     for(i=0, soma=0; i<n; i++)
     {
        System.out.print("Soma = "+soma+"+"+i);
        soma+=i;
        System.out.println("= "+soma+"\n");
      }
   }
```

ESTADO DAS VARIÁVEIS

iteração	i	soma(antes)	soma(depois)
1	0	0	0
2			
3			
4			
5			
6			
7			
8			
9			
10			

Estruturas de repetição: `for`

```
public class TesteFor
{
   public static void main(String[] args)
   {
     int n=Integer.parseInt(args[0]);
     int i, soma;
         //(inicialização; condição; incremento)
     for(i=0, soma=0; i<n; i++)
     {
        System.out.print("Soma = "+soma+"+"+i);
        soma+=i;
        System.out.println("= "+soma+"\n");
      }
   }
```

ESTADO DAS VARIÁVEIS

iteração	i	soma(antes)	soma(depois)
1	0	0	0
2	1		
3			
4			
5			
6			
7			
8			
9			
10			

Estruturas de repetição: `for`

```
public class TesteFor
{
   public static void main(String[] args)
   {
     int n=Integer.parseInt(args[0]);
     int i, soma;
         //(inicialização; condição; incremento)
     for(i=0, soma=0; i<n; i++)
     {
        System.out.print("Soma = "+soma+"+"+i);
        soma+=i;
        System.out.println("= "+soma+"\n");
      }
   }
```

ESTADO DAS VARIÁVEIS

iteração	i	soma(antes)	soma(depois)
1	0	0	0
2	1	0	
3			
4			
5			
6			
7			
8			
9			
10			

Estruturas de repetição: `for`

```
public class TesteFor
{
   public static void main(String[] args)
   {
     int n=Integer.parseInt(args[0]);
     int i, soma;
         //(inicialização; condição; incremento)
     for(i=0, soma=0; i<n; i++)
     {
        System.out.print("Soma = "+soma+"+"+i);
        soma+=i;
        System.out.println("= "+soma+"\n");
      }
   }
```

ESTADO DAS VARIÁVEIS

iteração	i	soma(antes)	soma(depois)
1	0	0	0
2	1	0	1
3			
4			
5			
6			
7			
8			
9			
10			

Estruturas de repetição: `for`

```
public class TesteFor
{
   public static void main(String[] args)
   {
     int n=Integer.parseInt(args[0]);
     int i, soma;
         //(inicialização; condição; incremento)
     for(i=0, soma=0; i<n; i++)
     {
        System.out.print("Soma = "+soma+"+"+i);
        soma+=i;
        System.out.println("= "+soma+"\n");
      }
   }
```

ESTADO DAS VARIÁVEIS

iteração	i	soma(antes)	soma(depois)
1	0	0	0
2	1	0	1
3	2		
4			
5			
6			
7			
8			
9			
10			

Estruturas de repetição: `for`

```
public class TesteFor
{
   public static void main(String[] args)
   {
     int n=Integer.parseInt(args[0]);
     int i, soma;
         //(inicialização; condição; incremento)
     for(i=0, soma=0; i<n; i++)
     {
        System.out.print("Soma = "+soma+"+"+i);
        soma+=i;
        System.out.println("= "+soma+"\n");
      }
   }
```

ESTADO DAS VARIÁVEIS

iteração	i	soma(antes)	soma(depois)
1	0	0	0
2	1	0	1
3	2	1	
4			
5			
6			
7			
8			
9			
10			

Estruturas de repetição: `for`

```
public class TesteFor
{
   public static void main(String[] args)
   {
     int n=Integer.parseInt(args[0]);
     int i, soma;
         //(inicialização; condição; incremento)
     for(i=0, soma=0; i<n; i++)
     {
        System.out.print("Soma = "+soma+"+"+i);
        soma+=i;
        System.out.println("= "+soma+"\n");
      }
   }
```

ESTADO DAS VARIÁVEIS

iteração	i	soma(antes)	soma(depois)
1	0	0	0
2	1	0	1
3	2	1	3
4			
5			
6			
7			
8			
9			
10			

Estruturas de repetição: `for`

```
public class TesteFor
{
   public static void main(String[] args)
   {
     int n=Integer.parseInt(args[0]);
     int i, soma;
         //(inicialização; condição; incremento)
     for(i=0, soma=0; i<n; i++)
     {
        System.out.print("Soma = "+soma+"+"+i);
        soma+=i;
        System.out.println("= "+soma+"\n");
      }
   }
```

ESTADO DAS VARIÁVEIS

iteração	i	soma(antes)	soma(depois)
1	0	0	0
2	1	0	1
3	2	1	3
4	3		
5			
6			
7			
8			
9			
10			

Estruturas de repetição: `for`

```
public class TesteFor
{
   public static void main(String[] args)
   {
     int n=Integer.parseInt(args[0]);
     int i, soma;
         //(inicialização; condição; incremento)
     for(i=0, soma=0; i<n; i++)
     {
        System.out.print("Soma = "+soma+"+"+i);
        soma+=i;
        System.out.println("= "+soma+"\n");
      }
   }
```

ESTADO DAS VARIÁVEIS

iteração	i	soma(antes)	soma(depois)
1	0	0	0
2	1	0	1
3	2	1	3
4	3	3	
5			
6			
7			
8			
9			
10			

Estruturas de repetição: `for`

```
public class TesteFor
{
   public static void main(String[] args)
   {
     int n=Integer.parseInt(args[0]);
     int i, soma;
         //(inicialização; condição; incremento)
     for(i=0, soma=0; i<n; i++)
     {
        System.out.print("Soma = "+soma+"+"+i);
        soma+=i;
        System.out.println("= "+soma+"\n");
      }
   }
```

ESTADO DAS VARIÁVEIS

iteração	i	soma(antes)	soma(depois)
1	0	0	0
2	1	0	1
3	2	1	3
4	3	3	6
5			
6			
7			
8			
9			
10			

Estruturas de repetição: `for`

```
public class TesteFor
{
   public static void main(String[] args)
   {
     int n=Integer.parseInt(args[0]);
     int i, soma;
         //(inicialização; condição; incremento)
     for(i=0, soma=0; i<n; i++)
     {
        System.out.print("Soma = "+soma+"+"+i);
        soma+=i;
        System.out.println("= "+soma+"\n");
      }
   }
```

ESTADO DAS VARIÁVEIS

iteração	i	soma(antes)	soma(depois)
1	0	0	0
2	1	0	1
3	2	1	3
4	3	3	6
5	4	6	10
6			
7			
8			
9			
10			

Estruturas de repetição: `for`

```
public class TesteFor
{
   public static void main(String[] args)
   {
     int n=Integer.parseInt(args[0]);
     int i, soma;
         //(inicialização; condição; incremento)
     for(i=0, soma=0; i<n; i++)
     {
        System.out.print("Soma = "+soma+"+"+i);
        soma+=i;
        System.out.println("= "+soma+"\n");
      }
   }
```

ESTADO DAS VARIÁVEIS

iteração	i	soma(antes)	soma(depois)
1	0	0	0
2	1	0	1
3	2	1	3
4	3	3	6
5	4	6	10
6	5	10	15
7			
8			
9			
10			

Estruturas de repetição: `for`

```
public class TesteFor
{
   public static void main(String[] args)
   {
     int n=Integer.parseInt(args[0]);
     int i, soma;
         //(inicialização; condição; incremento)
     for(i=0, soma=0; i<n; i++)
     {
        System.out.print("Soma = "+soma+"+"+i);
        soma+=i;
        System.out.println("= "+soma+"\n");
      }
   }
```

ESTADO DAS VARIÁVEIS

iteração	i	soma(antes)	soma(depois)
1	0	0	0
2	1	0	1
3	2	1	3
4	3	3	6
5	4	6	10
6	5	10	15
7	6	15	21
8			
9			
10			

Estruturas de repetição: `for`

```
public class TesteFor
{
   public static void main(String[] args)
   {
     int n=Integer.parseInt(args[0]);
     int i, soma;
         //(inicialização; condição; incremento)
     for(i=0, soma=0; i<n; i++)
     {
        System.out.print("Soma = "+soma+"+"+i);
        soma+=i;
        System.out.println("= "+soma+"\n");
      }
   }
```

ESTADO DAS VARIÁVEIS

iteração	i	soma(antes)	soma(depois)
1	0	0	0
2	1	0	1
3	2	1	3
4	3	3	6
5	4	6	10
6	5	10	15
7	6	15	21
8	7	21	28
9			
10			

Estruturas de repetição: `for`

```
public class TesteFor
{
   public static void main(String[] args)
   {
     int n=Integer.parseInt(args[0]);
     int i, soma;
         //(inicialização; condição; incremento)
     for(i=0, soma=0; i<n; i++)
     {
        System.out.print("Soma = "+soma+"+"+i);
        soma+=i;
        System.out.println("= "+soma+"\n");
      }
   }
```

ESTADO DAS VARIÁVEIS

iteração	i	soma(antes)	soma(depois)
1	0	0	0
2	1	0	1
3	2	1	3
4	3	3	6
5	4	6	10
6	5	10	15
7	6	15	21
8	7	21	28
9	8	28	36
10			

Estruturas de repetição: `for`

```
public class TesteFor
{
   public static void main(String[] args)
   {
     int n=Integer.parseInt(args[0]);
     int i, soma;
         //(inicialização; condição; incremento)
     for(i=0, soma=0; i<n; i++)
     {
        System.out.print("Soma = "+soma+"+"+i);
        soma+=i;
        System.out.println("= "+soma+"\n");
      }
   }
```

ESTADO DAS VARIÁVEIS

iteração	i	soma(antes)	soma(depois)
1	0	0	0
2	1	0	1
3	2	1	3
4	3	3	6
5	4	6	10
6	5	10	15
7	6	15	21
8	7	21	28
9	8	28	36
10	9	36	45

TesteFor.java

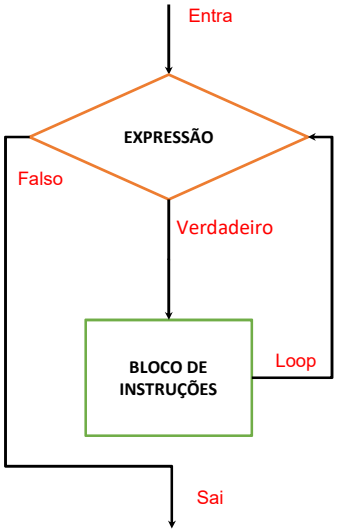
Estruturas de repetição: `for`

```
Soma = 0+0= 0
Soma = 0+1= 1
Soma = 1+2= 3
Soma = 3+3= 6
Soma = 6+4= 10
Soma = 10+5= 15
Soma = 15+6= 21
Soma = 21+7= 28
Soma = 28+8= 36
Soma = 36+9= 45
```

Estruturas de repetição: `while`

- Nessa estrutura, a expressão é avaliada logo no início (antes) da execução do laço
  - Se a expressão for falsa ao ser avaliada pela primeira vez, o laço não será executado.

```
while (expressão){
    //corpo do loop
}
```



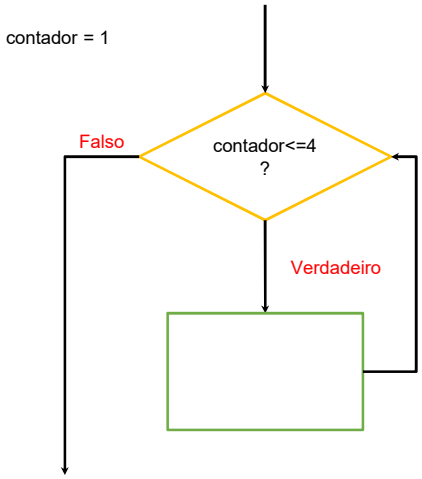
Estruturas de repetição: `while`

```
public class  TesteWhile{

   public static void main(String[] args){
      int contador;// Declara variável de controle
      contador = 1;// Inicializa variável de controle
      while (contador <= 4)// Testa a expressão
      {
         // Repete instruções
         System.out.println("Contador = " + contador);
         contador++; // Atualiza variável de controle
      }
      System.out.println("Terminado");
   }
}
```

contador = 1

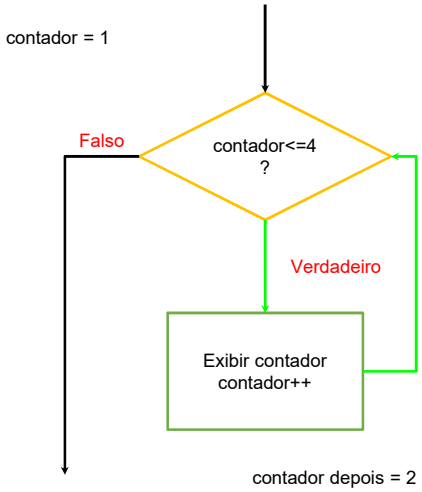
Estruturas de repetição: `while`

```
public class  TesteWhile{

   public static void main(String[] args){
      int contador;// Declara variável de controle
      contador = 1;// Inicializa variável de controle
      while (contador <= 4)// Testa a expressão
      {
         // Repete instruções
         System.out.println("Contador = " + contador);
         contador++;     // Atualiza variável de controle
      }
      System.out.println("Terminado");
   }
}
```



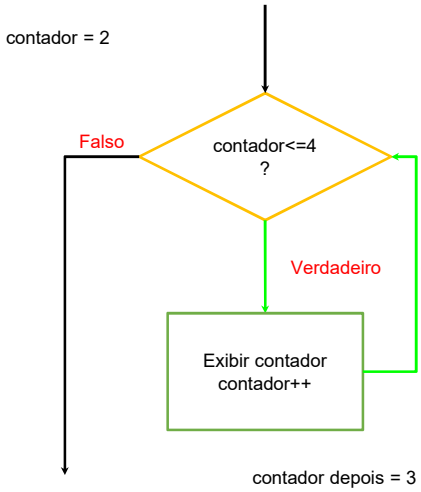
Estruturas de repetição: `while`

```
public class  TesteWhile{

   public static void main(String[] args){
      int contador;// Declara variável de controle
      contador = 1;// Inicializa variável de controle
      while (contador <= 4)// Testa a expressão
      {
         // Repete instruções
         System.out.println("Contador = " + contador);
         contador++;     // Atualiza variável de controle
      }
      System.out.println("Terminado");
   }
}
```



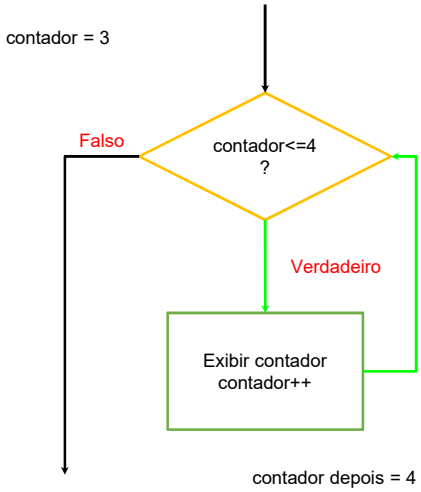
Estruturas de repetição: `while`

```
public class  TesteWhile{

   public static void main(String[] args){
      int contador;// Declara variável de controle
      contador = 1;// Inicializa variável de controle
      while (contador <= 4)// Testa a expressão
      {
         // Repete instruções
         System.out.println("Contador = " + contador);
         contador++;     // Atualiza variável de controle
      }
      System.out.println("Terminado");
   }
}
```



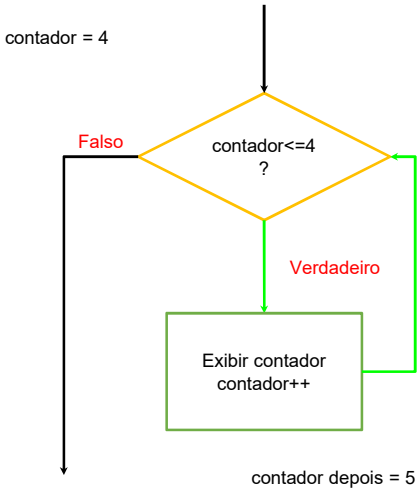
Estruturas de repetição: `while`

```
public class  TesteWhile{

   public static void main(String[] args){
      int contador;// Declara variável de controle
      contador = 1;// Inicializa variável de controle
      while (contador <= 4)// Testa a expressão
      {
         // Repete instruções
         System.out.println("Contador = " + contador);
         contador++;     // Atualiza variável de controle
      }
      System.out.println("Terminado");
   }
}
```



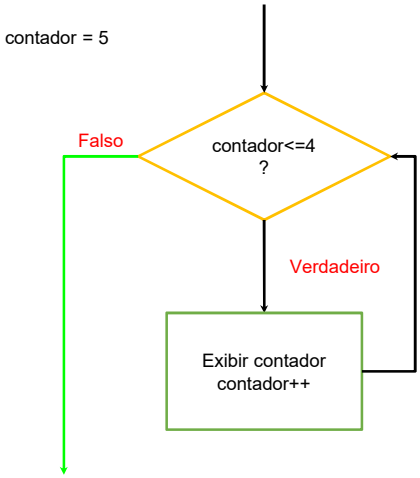
Estruturas de repetição: `while`

```
public class  TesteWhile{

   public static void main(String[] args){
      int contador;// Declara variável de controle
      contador = 1;// Inicializa variável de controle
      while (contador <= 4)// Testa a expressão
      {
         // Repete instruções
         System.out.println("Contador = " + contador);
         contador++;     // Atualiza variável de controle
      }
      System.out.println("Terminado");
   }
}
```



Estruturas de repetição: `while`

```
public class  TesteWhile{

   public static void main(String[] args){
      int contador;// Declara variável de controle
      contador = 1;// Inicializa variável de controle
      while (contador <= 4)// Testa a expressão
      {
         // Repete instruções
         System.out.println("Contador = " + contador);
         contador++;     // Atualiza variável de controle
            }
      System.out.println("Terminado");
   }
}
```



TesteWhile.java

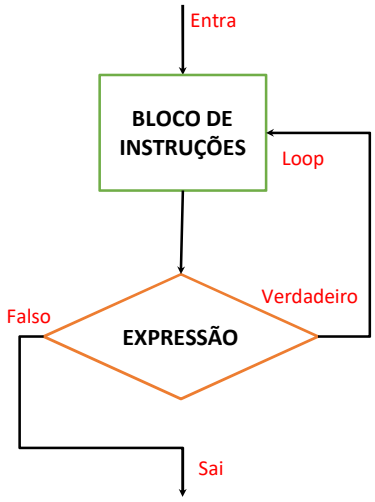
Estruturas de repetição: `while`

```
Contador = 1
Contador = 2
Contador = 3
Contador = 4
Terminado
```

Estruturas de repetição: `do-while`

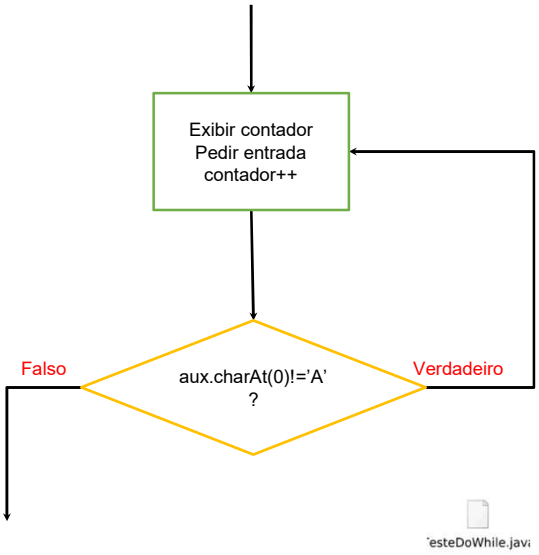
- A expressão é avaliada no final da execução do laço
  - Será executado pelo menos **uma** vez.

```
do{
 //corpo do loop
}while (expressão)
```



Estruturas de repetição: do-while

```
import java.io.*;
public class TesteDoWhile {
    public static void main(String[] args)throws
IOException{
        BufferedReader in;
        in = new BufferedReader
                              (new
InputStreamReader(System.in));
        String aux;
        int contador=0;
        System.out.println("Quantidade de vezes que uma
tecla diferente de A foi pressionada:");
        do{
            System.out.println("Contador = "+contador);
            System.out.print("Digite uma letra e ENTER:
");
            aux = in.readLine();
            contador++;
        }while (aux.charAt(0)!='A');//Criterio de Parada
    }
}
```



esteDoWhile.java

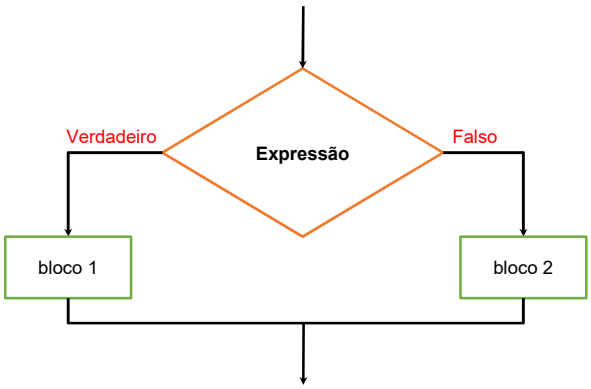
Estruturas de repetição: do-while

```
Quantidade de vezes que uma tecla diferente de A foi pressionada:
Contador = 0
Digite uma letra e ENTER: u
Contador = 1
Digite uma letra e ENTER: o
Contador = 2
Digite uma letra e ENTER: p
Contador = 3
Digite uma letra e ENTER: ;
Contador = 4
Digite uma letra e ENTER: a
Contador = 5
Digite uma letra e ENTER: A
```

Estruturas de controle: `if-else`

- Estrutura que fornece a seleção em dois ou mais sentidos, onde somente um será executado.
- Formatos:

```
if(Expressão){
   bloco 1
}
else{
   bloco 2
}
```



Estruturas de controle: `if-else`

```
if (Expressão1){
      bloco 1
}
else if(Expressão2){
      bloco 2
}
      ...

else if(ExpressãoN){
      bloco N
}
else{
      bloco N+1
}
```

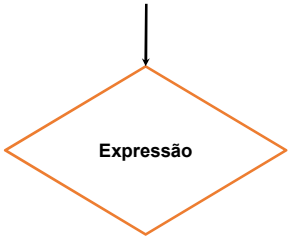
- Essa estrutura é chamada de `if-else` aninhados.
- Neste caso, cada expressão é avaliada em sequência, até que uma expressão verdadeira seja encontrada.
- Apenas o bloco de instruções que segue a expressão verdadeira é executado.
- Se nenhuma expressão for verdadeira, o bloco seguinte ao `else` final é executado. Deste modo, o `else` final e o bloco final são opcionais.

Estruturas de controle: `if-else`

```
public class TesteIf{
    public static void main(String[] args){
        int numero = Integer.parseInt(args[0]);
        if (numero > 0)
                System.out.println("Positivo");
        else
            if (numero < 0)
                System.out.println("Negativo");
            else
                System.out.println("Zero");
    }
}
```

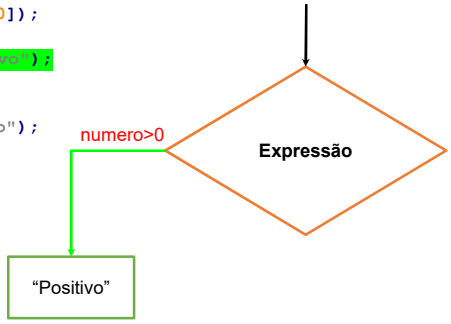

Estruturas de controle: `if-else`

```
public class TesteIf{
    public static void main(String[] args){
        int numero = Integer.parseInt(args[0]);
        if (numero > 0)
                System.out.println("Positivo");
        else
            if (numero < 0)
                System.out.println("Negativo");
            else
                System.out.println("Zero");
    }
}
```



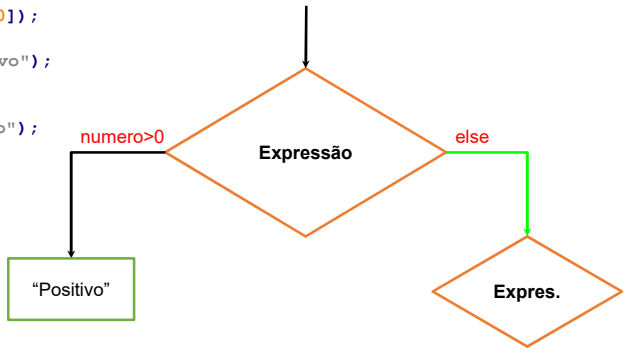
Estruturas de controle: `if-else`

```
public class TesteIf{
    public static void main(String[] args){
        int numero = Integer.parseInt(args[0]);
        if (numero > 0)
                System.out.println("Positivo");
        else
            if (numero < 0)
                System.out.println("Negativo");
            else
                System.out.println("Zero");
    }
}
```



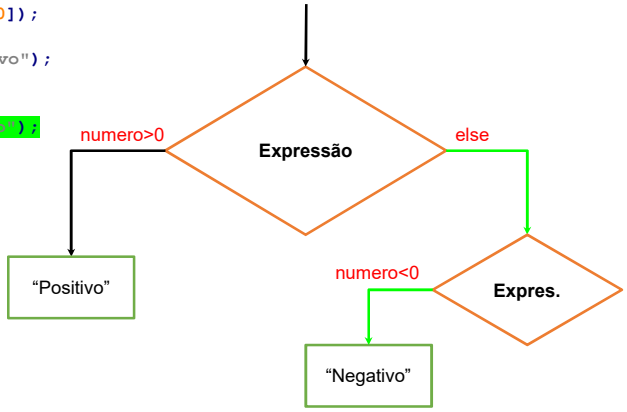
Estruturas de controle: `if-else`

```
public class TesteIf{
    public static void main(String[] args){
        int numero = Integer.parseInt(args[0]);
        if (numero > 0)
                System.out.println("Positivo");
        else
            if (numero < 0)
                System.out.println("Negativo");
            else
                System.out.println("Zero");
    }
}
```



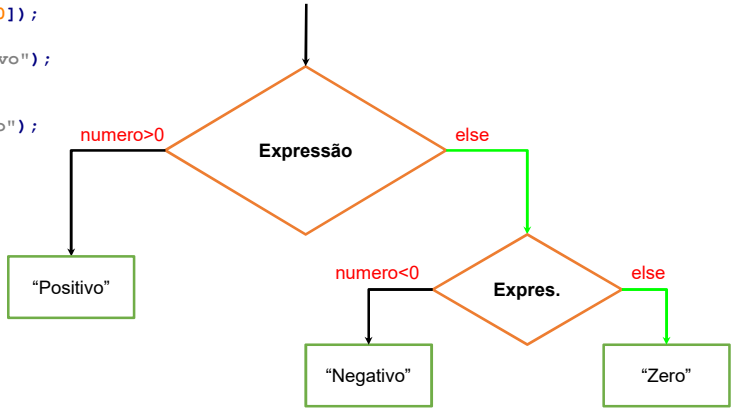
Estruturas de controle: `if-else`

```
public class TesteIf{
    public static void main(String[] args){
        int numero = Integer.parseInt(args[0]);
        if (numero > 0)
                System.out.println("Positivo");
        else
            if (numero < 0)
                System.out.println("Negativo");
            else
                System.out.println("Zero");
    }
}
```



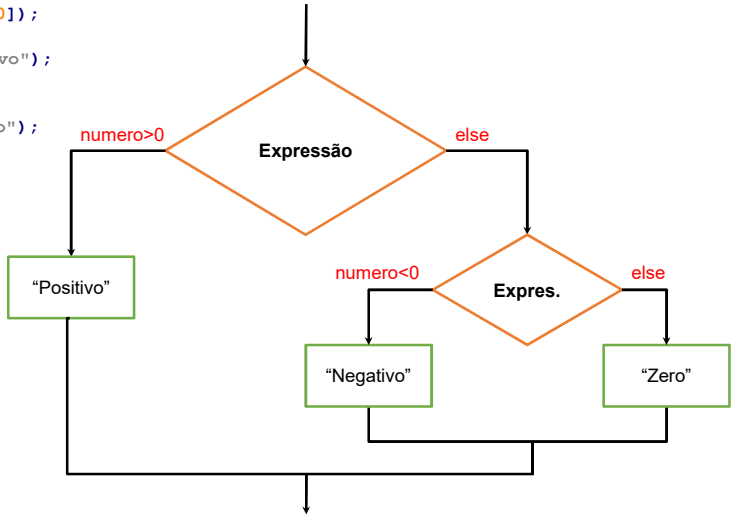
Estruturas de repetição: `if-else`

```
public class TesteIf{
    public static void main(String[] args){
        int numero = Integer.parseInt(args[0]);
        if (numero > 0)
                System.out.println("Positivo");
        else
            if (numero < 0)
                System.out.println("Negativo");
            else
                System.out.println("Zero");
    }
}
```



Estruturas de controle: `if-else`

```
public class TesteIf{
    public static void main(String[] args){
        int numero = Integer.parseInt(args[0]);
        if (numero > 0)
                System.out.println("Positivo");
        else
            if (numero < 0)
                System.out.println("Negativo");
            else
                System.out.println("Zero");
    }
}
```



Estruturas de controle: `if-else`

- O exemplo anterior está usando a estrutura `if-else` sem as "`{ }`". Em sua ausência, um `else` sempre forma par com o `if` precedente mais próximo que não tenha um `else` emparelhado com ele.
- Exemplo:

```
double average=100.0;
if (average >= 60.0)
    if (average < 70.0) // forma par com o else
        System.out.println("Marginal PASS");
else
    System.out.println("FAIL");
```

- A saída é "FAIL". O compilador ignora a indentação e forma par com o segundo `if`

TesteIf2.java

Estruturas de controle: `if-else`

- Versão correta:

```
double average = 100.0;
if (average >= 60.0)
{
   if(average < 70.0)
      System.out.println("Marginal PASS");
}
else
   System.out.println("FAIL");
```

Estruturas de controle: `switch-case`

- Equivale a um conjunto de `if's` encadeados, porém mais estruturado
- O ponto de início de execução é sempre um `case`, e a execução só para quando um `break` é encontrado
- Se o valor da expressão ordinal não for encontrado nos `cases`, a diretiva `default` é executada (está é opcional)

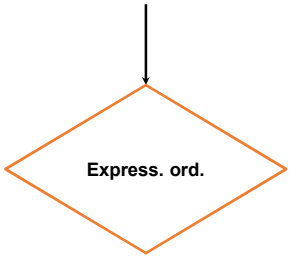
```
switch(expressao_ordinal){
  ordinal_1: bloco1;
             break;
  ordinal_2: bloco2;
             break;
  default: diretiva_default;
}
```

Estruturas de controle: `switch-case`

```
public class TesteSwitch{
    public static void main(String[] args){
        switch (args[0].charAt(0)){
            case 'a':
            case 'A':System.out.println("Vogal A");
                     break;
            case 'e':
            case 'E':System.out.println("Vogal E");
                     break;
            case 'i':
            case 'I':System.out.println("Vogal I");
                     break;
            case 'o':
            case 'O':System.out.println("Vogal O");
                     break;
            case 'u':
            case 'U':System.out.println("Vogal U");
                     break;
            default:System.out.println("A letra não é vogal.");
        }
    }
}
```

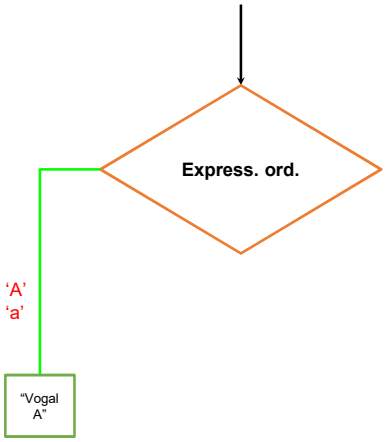

Estruturas de controle: `switch-case`

```
public class TesteSwitch{
    public static void main(String[] args){
        switch (args[0].charAt(0)){
            case 'a':
            case 'A':System.out.println("Vogal A");
                     break;
            case 'e':
            case 'E':System.out.println("Vogal E");
                     break;
            case 'i':
            case 'I':System.out.println("Vogal I");
                     break;
            case 'o':
            case 'O':System.out.println("Vogal O");
                     break;
            case 'u':
            case 'U':System.out.println("Vogal U");
                     break;
            default:System.out.println("A letra não é vogal.");
        }
    }
}
```



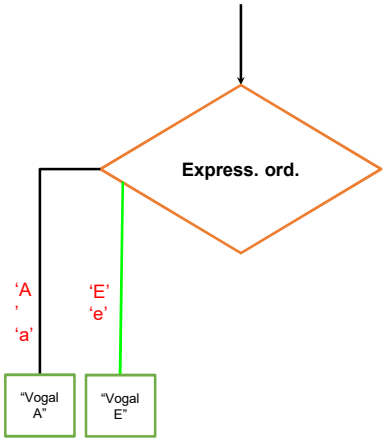
Estruturas de controle: `switch-case`

```
public class TesteSwitch{
    public static void main(String[] args){
        switch (args[0].charAt(0)){
            case 'a':
            case 'A':System.out.println("Vogal A");
                     break;
            case 'e':
            case 'E':System.out.println("Vogal E");
                     break;
            case 'i':
            case 'I':System.out.println("Vogal I");
                     break;
            case 'o':
            case 'O':System.out.println("Vogal O");
                     break;
            case 'u':
            case 'U':System.out.println("Vogal U");
                     break;
            default:System.out.println("A letra não é vogal.");
        }
    }
}
```



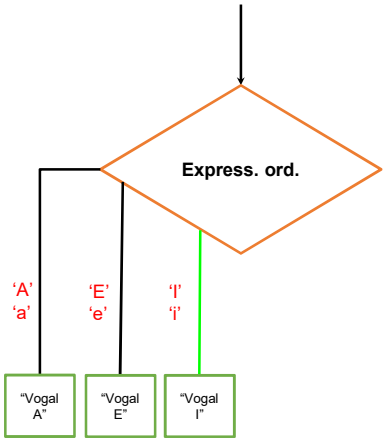
Estruturas de controle: `switch-case`

```
public class TesteSwitch{
    public static void main(String[] args){
        switch (args[0].charAt(0)){
            case 'a':
            case 'A':System.out.println("Vogal A");
                     break;
            case 'e':
            case 'E':System.out.println("Vogal E");
                     break;
            case 'i':
            case 'I':System.out.println("Vogal I");
                     break;
            case 'o':
            case 'O':System.out.println("Vogal O");
                     break;
            case 'u':
            case 'U':System.out.println("Vogal U");
                     break;
            default:System.out.println("A letra não é vogal.");
        }
    }
}
```



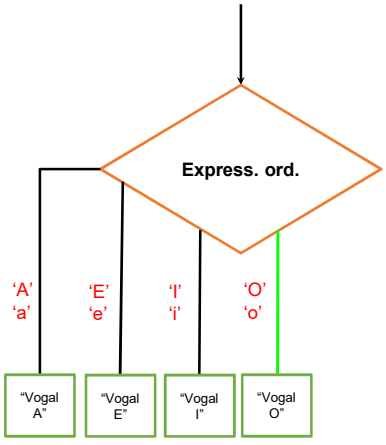
Estruturas de controle: `switch-case`

```
public class TesteSwitch{
    public static void main(String[] args){
        switch (args[0].charAt(0)){
            case 'a':
            case 'A':System.out.println("Vogal A");
                     break;
            case 'e':
            case 'E':System.out.println("Vogal E");
                     break;
            case 'i':
            case 'I':System.out.println("Vogal I");
                     break;
            case 'o':
            case 'O':System.out.println("Vogal O");
                     break;
            case 'u':
            case 'U':System.out.println("Vogal U");
                     break;
            default:System.out.println("A letra não é vogal.");
        }
    }
}
```



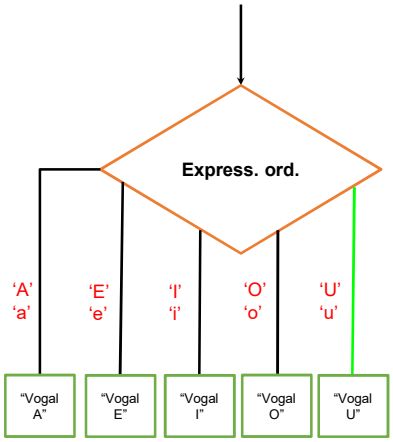
Estruturas de controle: `switch-case`

```
public class TesteSwitch{
    public static void main(String[] args){
        switch (args[0].charAt(0)){
            case 'a':
            case 'A':System.out.println("Vogal A");
                     break;
            case 'e':
            case 'E':System.out.println("Vogal E");
                     break;
            case 'i':
            case 'I':System.out.println("Vogal I");
                     break;
            case 'o':
            case 'O':System.out.println("Vogal O");
                     break;
            case 'u':
            case 'U':System.out.println("Vogal U");
                     break;
            default:System.out.println("A letra não é vogal.");
        }
    }
}
```



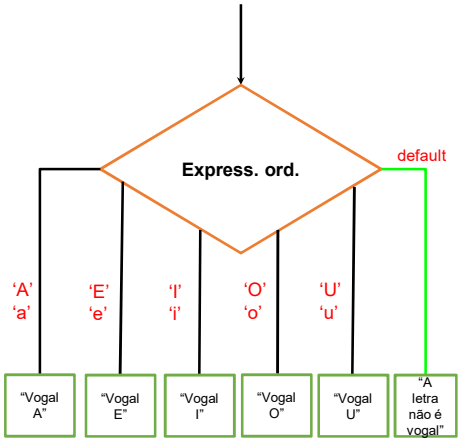
Estruturas de controle: `switch-case`

```
public class TesteSwitch{
    public static void main(String[] args){
        switch (args[0].charAt(0)){
            case 'a':
            case 'A':System.out.println("Vogal A");
                     break;
            case 'e':
            case 'E':System.out.println("Vogal E");
                     break;
            case 'i':
            case 'I':System.out.println("Vogal I");
                     break;
            case 'o':
            case 'O':System.out.println("Vogal O");
                     break;
            case 'u':
            case 'U':System.out.println("Vogal U");
                     break;
            default:System.out.println("A letra não é vogal.");
        }
    }
}
```



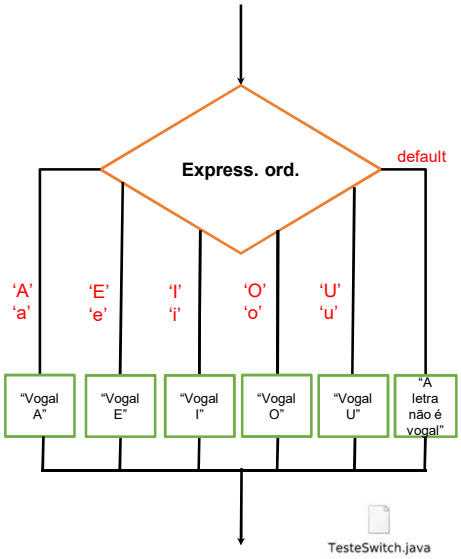
Estruturas de controle: `switch-case`

```
public class TesteSwitch{
    public static void main(String[] args){
        switch (args[0].charAt(0)){
            case 'a':
            case 'A':System.out.println("Vogal A");
                     break;
            case 'e':
            case 'E':System.out.println("Vogal E");
                     break;
            case 'i':
            case 'I':System.out.println("Vogal I");
                     break;
            case 'o':
            case 'O':System.out.println("Vogal O");
                     break;
            case 'u':
            case 'U':System.out.println("Vogal U");
                     break;
            default:System.out.println("A letra não é vogal.");
        }
    }
}
```



Estruturas de controle: `switch-case`

```
public class TesteSwitch{
    public static void main(String[] args){
        switch (args[0].charAt(0)){
            case 'a':
            case 'A':System.out.println("Vogal A");
                     break;
            case 'e':
            case 'E':System.out.println("Vogal E");
                     break;
            case 'i':
            case 'I':System.out.println("Vogal I");
                     break;
            case 'o':
            case 'O':System.out.println("Vogal O");
                     break;
            case 'u':
            case 'U':System.out.println("Vogal U");
                     break;
            default:System.out.println("A letra não é vogal.");
        }
    }
}
```



Exercícios

- Faça um programa que não permita a criação de mais de um objeto `pessoa` na sua execução.
- Faça pelo menos um programa para cada estrutura apresentada. Seja criativo!
- Continue avançando no tutorial da *Oracle*.