

Aula 09 – Laços (cont.)

Norton Trevisan Roman

16 de abril de 2018

Do ... While

- Existe ainda um outro tipo de laço: do...while

```
do {  
    comando1;  
    comando2;  
    ...  
    comandon;  
} while (condição);
```

Do ... While

- Existe ainda um outro tipo de laço: do...while

```
do {  
    comando1;  
    comando2;  
    ...  
    comandon;  
} while (condição);
```

```
public static void main(String[] args) {  
    double area = 50;  
    int tipo = ALVENARIA;  
  
    System.out.println("Área\tValor");  
    do {  
        System.out.println(area+"\t"+  
                             valorPiscina(area,tipo));  
        area = area + 50;  
    } while (area<=200);  
}
```

Do ... While

- O while faz o teste antes de rodar o laço pela primeira vez
- O do...while faz o teste depois de rodar o laço pela primeira vez
 - Rodando novamente apenas se o teste for positivo

```
while (condição) {  
    comando1;  
    comando2;  
    ...  
    comandon;  
}
```

```
do {  
    comando1;  
    comando2;  
    ...  
    comandon;  
} while (condição);
```

Do ... While

- E quando usar um ou o outro?

```
while (condição) {  
    comando1;  
    comando2;  
    ...  
    comandon;  
}
```

```
do {  
    comando1;  
    comando2;  
    ...  
    comandon;  
} while (condição);
```

Do ... While

- E quando usar um ou o outro?
- Depende de quando o teste deve ser feito:
 - Se antes ou depois do corpo do laço rodar uma vez

```
while (condição) {  
    comando1;  
    comando2;  
    ...  
    comandon;  
}
```

```
do {  
    comando1;  
    comando2;  
    ...  
    comandon;  
} while (condição);
```

Do ... While

- E qual dentre o *while* e *do...while* é melhor?

Do ... While

- E qual dentre o *while* e *do...while* é melhor?
- São totalmente equivalentes

Do ... While

- E qual dentre o *while* e *do...while* é melhor?

- São totalmente equivalentes

- Todo *while* pode ser escrito como *do... while*

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

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

Do ... While

- E qual dentre o *while* e *do...while* é melhor?

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

- E vice versa:

```
condição = true;  
while (condição) {  
    comando;  
    recalcula_condição;  
}
```

Do ... While

- O que ditará qual deles será usado é a conveniência para o programador

```
while (condição) {  
    comando1;  
    comando2;  
    ...  
    comandon;  
}
```

```
do {  
    comando1;  
    comando2;  
    ...  
    comandon;  
} while (condição);
```

Do ... While

- O que ditará qual deles será usado é a conveniência para o programador
- Será escolhido, naturalmente, aquele que exigir a escrita de menos código

```
while (condição) {  
    comando1;  
    comando2;  
    ...  
    comandon;  
}
```

```
do {  
    comando1;  
    comando2;  
    ...  
    comandon;  
} while (condição);
```

Do ... While

- Em nosso código, esses trechos são equivalentes:

```
public static void main(String[]  
                        args) {  
    double area = 50;  
    int tipo = ALVENARIA;  
  
    System.out.println("Área\t  
                        Valor");  
  
    while (area <= 200) {  
        System.out.println(area+"\t"  
            + valorPiscina(area,tipo));  
        area = area+50;  
    }  
}
```

```
public static void main(String[]  
                        args) {  
    double area = 50;  
    int tipo = ALVENARIA;  
  
    System.out.println("Área\t  
                        Valor");  
  
    if (area <= 200) {  
        do {  
            System.out.println(area+  
                                "\t"  
                                + valorPiscina(area,tipo));  
            area = area + 50;  
        } while (area<=200);  
    }  
}
```

Do ... While

- Nesse caso, contudo, o **if** sempre será verdadeiro

```
public static void main(String[]  
                        args) {  
    double area = 50;  
    int tipo = ALVENARIA;  
  
    System.out.println("Área\t  
                        Valor");  
  
    while (area <= 200) {  
        System.out.println(area+"\t"  
            + valorPiscina(area,tipo));  
        area = area+50;  
    }  
}
```

```
public static void main(String[]  
                        args) {  
    double area = 50;  
    int tipo = ALVENARIA;  
  
    System.out.println("Área\t  
                        Valor");  
  
    if (area <= 200) {  
        do {  
            System.out.println(area+  
                                "\t"  
                                + valorPiscina(area,tipo));  
            area = area + 50;  
        } while (area<=200);  
    }  
}
```

Do ... While

- Então podemos removê-lo

```
public static void main(String[]  
                        args) {  
    double area = 50;  
    int tipo = ALVENARIA;  
  
    System.out.println("Área\t  
                        Valor");  
    while (area <= 200) {  
        System.out.println(area+"\t"  
            + valorPiscina(area,tipo));  
        area = area+50;  
    }  
}
```

```
public static void main(String[]  
                        args) {  
    double area = 50;  
    int tipo = ALVENARIA;  
  
    System.out.println("Área\t  
                        Valor");  
    do {  
        System.out.println(area+"\t"  
            + valorPiscina(area,tipo));  
        area = area + 50;  
    } while (area<=200);  
}
```

Do ... While

Da mesma forma, esses trechos são equivalentes:

```
public static void main(String[]
                        args) {
    double area = 100;
    int tipo = ALVENARIA;

    System.out.println("Material\t
                        Valor");
    while (tipo <= PLASTICO) {
        System.out.println(tipo+"\t
                        \t"+valorPiscina(area,
                        tipo));
        tipo = tipo+1;
    }
}
```

```
public static void main(String[]
                        args) {
    double area = 50;
    int tipo = ALVENARIA;

    System.out.println("Material\t
                        Valor");
    if (tipo <= PLASTICO) {
        do {
            System.out.println(tipo+
                                "\t\t"+valorPiscina(area,
                                tipo));
            tipo = tipo + 1;
        } while (tipo <= PLASTICO);
    }
}
```


Do ... While

Também aqui o if era redundante

```
public static void main(String[]
                        args) {
    double area = 100;
    int tipo = ALVENARIA;

    System.out.println("Material\t
                        Valor");
    while (tipo <= PLASTICO) {
        System.out.println(tipo+"\t
                        \t"+valorPiscina(area,
                        tipo));
        tipo = tipo+1;
    }
}
```

```
public static void main(String[]
                        args) {
    double area = 50;
    int tipo = ALVENARIA;

    System.out.println("Material\t
                        Valor");
    do {
        System.out.println(tipo+"\t
                        \t"+valorPiscina(area,
                        tipo));
        tipo = tipo + 1;
    } while (tipo <= PLASTICO);
}
```

Do ... While

- E, finalmente, assim como no while...

```
public static void main(String[] args) {  
    double area = 50;  
    int tipo;  
  
    System.out.println("Área\tMaterial\t  
                        Valor");  
  
    while (area <= 200) {  
        tipo = ALVENARIA;  
        while (tipo <= PLASTICO) {  
            System.out.println(area+"\t"+tipo+  
                                "\t\t"+valorPiscina(area,tipo));  
            tipo = tipo+1;  
        }  
        area = area+50;  
    }  
}
```

Do ... While

- Também o do...while pode estar aninhado

```
public static void main(String[] args) {
    double area = 50;
    int tipo;

    System.out.println("Área\tMaterial\t
                        Valor");

    do {
        tipo = ALVENARIA;
        do {
            System.out.println(area+"\t"+tipo+
                               "\t\t"+valorPiscina(area,tipo));
            tipo = tipo+1;
        } while (tipo <= PLASTICO);
        area = area+50;
    } while (area <= 200);
}
```

Em Suma...

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

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

Em Suma...

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

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

- Primeiro testa a condição

Em Suma...

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

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

- Primeiro testa a condição
- Se ela for verdadeira, executa o laço (corpo do while)

Em Suma...

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

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

- Primeiro testa a condição
- Se ela for verdadeira, executa o laço (corpo do while)

- Primeiro executa o laço (corpo do do...while)

Em Suma...

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

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

- Primeiro testa a condição
- Se ela for verdadeira, executa o laço (corpo do while)

- Primeiro executa o laço (corpo do do...while)
- Ao final do corpo, testa então a condição

Em Suma...

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

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

- Ao final do corpo, volta ao início, testando novamente a condição

Em Suma...

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

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

- Ao final do corpo, volta ao início, testando novamente a condição
- Se ela for falsa, passa à próxima instrução após o `while`

Em Suma...

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

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

- Ao final do corpo, volta ao início, testando novamente a condição
- Se ela for falsa, passa à próxima instrução após o `while`

- Se a condição for verdadeira, executa o laço novamente

Em Suma...

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

- Ao final do corpo, volta ao início, testando novamente a condição
- Se ela for falsa, passa à próxima instrução após o `while`

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

- Se a condição for verdadeira, executa o laço novamente
- Se ela for falsa, passa à próxima instrução após o `do...while`

- Os while vistos tinham características em comum em suas variáveis de controle:

```
public static void main(String[]  
                           args) {  
    double area = 100;  
    int tipo = ALVENARIA;  
  
    System.out.println("Material\t  
                           Valor");  
    while (tipo <= PLASTICO) {  
        System.out.println(tipo+"\t\t"+  
                           valorPiscina(area,tipo));  
        tipo = tipo+1;  
    }  
}
```

- Os while vistos tinham características em comum em suas variáveis de controle:
- Ambos variavam em passos constantes (de 1 em 1 ou de 50 em 50)

```
public static void main(String[]  
                        args) {  
    double area = 100;  
    int tipo = ALVENARIA;  
  
    System.out.println("Material\t  
                        Valor");  
    while (tipo <= PLASTICO) {  
        System.out.println(tipo+"\t\t"+  
                            valorPiscina(area,tipo));  
        tipo = tipo+1;  
    }  
}
```

- Não haveria um modo de deixar esse código mais enxuto?

```
public static void main(String[]  
                           args) {  
  
    double area = 100;  
    int tipo = ALVENARIA;  
  
    System.out.println("Material\t  
                        Valor");  
    while (tipo <= PLASTICO) {  
        System.out.println(tipo+"\t\t"+  
                            valorPiscina(area,tipo));  
        tipo = tipo+1;  
    }  
}
```

For

- Não haveria um modo de deixar esse código mais enxuto?
- Um modo de dizer “para tipo começando em 0, variando de 1 em 1, até 3, faça...”

```
public static void main(String[]  
                        args) {  
    double area = 100;  
    int tipo = ALVENARIA;  
  
    System.out.println("Material\t  
                        Valor");  
    while (tipo <= PLASTICO) {  
        System.out.println(tipo+"\t\t"+  
                            valorPiscina(area,tipo));  
        tipo = tipo+1;  
    }  
}
```


- Ou “para area começando em 50, variando de 50 em 50, até 200, faça...”

```
public static void main(String[]  
                        args) {  
  
    double area = 50;  
    int tipo = ALVENARIA;  
  
    System.out.println("Área\tValor");  
    while (area <= 200) {  
        System.out.println(area+"\t"+  
                            valorPiscina(area,tipo));  
        area = area+50;  
    }  
}
```

For

- O laço for:

```
for (inicialização;  
     condição;  
     atualização) {  
    comandos;  
}
```

```
inicialização;  
while (condição) {  
    comandos;  
    atualização;  
}
```

For

- O laço for:

```
for (inicialização;  
     condição;  
     atualização) {  
    comandos;  
}
```

```
inicialização;  
while (condição) {  
    comandos;  
    atualização;  
}
```

- Primeiro, há a **inicialização** das variáveis de controle

For

- O laço for:

```
for (inicialização;  
     condição;  
     atualização) {  
    comandos;  
}
```

```
inicialização;  
while (condição) {  
    comandos;  
    atualização;  
}
```

- Primeiro, há a **inicialização** das variáveis de controle
 - Esse passo é executado uma única vez

For

```
for (inicialização;  
    condição;  
    atualização) {  
    comandos;  
}
```

```
inicialização;  
while (condição) {  
    comandos;  
    atualização;  
}
```

- Em seguida, a **condição** é testada

For

```
for (inicialização;  
    condição;  
    atualização) {  
    comandos;  
}
```

```
inicialização;  
while (condição) {  
    comandos;  
    atualização;  
}
```

- Em seguida, a condição é testada
- Se resultar verdadeira, os comandos do corpo do for são executados

For

```
for (inicialização;  
      condição;  
      atualização) {  
    comandos;  
}
```

```
inicialização;  
while (condição) {  
    comandos;  
    atualização;  
}
```

- Ao final do corpo, é executada a **atualização**

For

```
for (inicialização;  
    condição;  
    atualização) {  
    comandos;  
}
```

```
inicialização;  
while (condição) {  
    comandos;  
    atualização;  
}
```

- Ao final do corpo, é executada a atualização
- Inicia-se o laço novamente, voltando ao teste da **condição**

For

```
for (inicialização;           inicialização;
      condição;               while (condição) {
      atualização) {          comandos;
    comandos;                  atualização;
  }                             }
```

- Ao final do corpo, é executada a atualização
- Inicia-se o laço novamente, voltando ao teste da **condição**
- Se a condição for falsa, o corpo é ignorado

For

```
public static void main(String[]  
                        args) {  
    double area = 100;  
    int tipo = ALVENARIA;  
  
    System.out.println("Material\t  
                        Valor");  
    while (tipo <= PLASTICO) {  
        System.out.println(tipo+"\t\t"+  
                            valorPiscina(area,tipo));  
        tipo = tipo+1;  
    }  
}
```

```
public static void main(String[]  
                        args) {  
    double area = 100;  
  
    System.out.println("Material\t  
                        Valor");  
    for(int tipo = ALVENARIA;  
        tipo <= PLASTICO;  
        tipo = tipo+1) {  
        System.out.println(tipo+"\t\t"+  
                            valorPiscina(area,tipo));  
    }  
}
```

- São totalmente equivalentes: dependem da conveniência do programador

For

- É possível declarar a variável de controle **dentro** do for
- Nesse caso, seu escopo é o corpo do for

```
public static void main(String[]  
                           args) {  
    double area = 100;  
  
    System.out.println("Material\t  
                           Valor");  
    for(int tipo = ALVENARIA;  
        tipo <= PLASTICO;  
        tipo = tipo+1) {  
        System.out.println(tipo+"\t\t"+  
                           valorPiscina(area,tipo));  
    }  
}
```

For

- Nada impede, contudo, que seja declarada antes, para ser visível a mais alguém:

```
public static void main(String[]
                                args) {
    double area = 100;

    System.out.println("Material\t
                                Valor");

    int tipo;
    for(tipo = ALVENARIA;
        tipo <= PLASTICO;
        tipo = tipo+1) {
        System.out.println(tipo+"\t\t"+
                            valorPiscina(area,tipo));
    }
}
```

- Incremento de um em um não é só o que o for é capaz de fazer:

```
public static void main(String[]  
                        args) {  
    double area = 50;  
    int tipo = ALVENARIA;  
  
    System.out.println("Área\tValor");  
    while (area <= 200) {  
        System.out.println(area+"\t"+  
                            valorPiscina(area,tipo));  
        area = area+50;  
    }  
}
```

```
public static void main(String[]  
                        args) {  
    int tipo = ALVENARIA;  
  
    System.out.println("Área\tValor");  
    for(double area = 50; area <= 200;  
        area = area+50) {  
        System.out.println(area+"\t"+  
                            valorPiscina(area,tipo));  
    }  
}
```

- Qualquer expressão algébrica pode ser usada

```
public static void main(String[]  
                           args) {  
    int tipo = ALVENARIA;  
  
    System.out.println("Área\tValor");  
    for(double area = 50; area <= 200;  
        area = area+50) {  
        System.out.println(area+"\t"+  
            valorPiscina(area,tipo));  
    }  
}
```

For

- Qualquer expressão algébrica pode ser usada
- Até mesmo coisas como
$$\text{area} = 2 * \text{area} + \text{Math.pow}(\text{area}, 3)$$

```
public static void main(String[]  
                           args) {  
    int tipo = ALVENARIA;  
  
    System.out.println("Área\tValor");  
    for(double area = 50; area <= 200;  
        area = area+50) {  
        System.out.println(area+"\t"+  
            valorPiscina(area,tipo));  
    }  
}
```

For

- E não é apenas o int que pode ser usado como variável de controle
- Podemos também usar outros tipos –
Cuidado com comparações em ponto flutuante!!!

```
public static void main(String[]  
                           args) {  
    int tipo = ALVENARIA;  
  
    System.out.println("Área\tValor");  
    for(double area = 50; area <= 200;  
        area = area+50) {  
        System.out.println(area+"\t"+  
            valorPiscina(area,tipo));  
    }  
}
```


For

- Da mesma forma, na condição qualquer expressão lógica ou relacional pode ser usada

- Ex:
(area <= 200) ||
(area == 300) }

```
public static void main(String[]  
                           args) {  
    int tipo = ALVENARIA;  
  
    System.out.println("Área\tValor");  
    for(double area = 50; area <= 200;  
        area = area+50) {  
        System.out.println(area+"\t"+  
            valorPiscina(area,tipo));  
    }
```

- Também nada nos impede de fazer um **decremento**

```
public static void main(String[]  
                           args) {  
    int tipo = ALVENARIA;  
  
    System.out.println("Área\tValor");  
    for(double area = 200; area >= 50;  
        area = area-50) {  
        System.out.println(area+"\t"+  
            valorPiscina(area,tipo));  
    }  
}
```

- E o resultado seria apenas a inversão da tabela:

Área	Valor
200.0	300000.0
150.0	225000.0
100.0	150000.0
50.0	75000.0

```
public static void main(String[]  
                           args) {  
    int tipo = ALVENARIA;  
  
    System.out.println("Área\tValor");  
    for(double area = 200; area >= 50;  
        area = area-50) {  
        System.out.println(area+"\t"+  
            valorPiscina(area,tipo));  
    }  
}
```

For Aninhado

- Laços *for* podem também ser aninhados

```
public static void main(String[] args) {  
    double area = 50;  
    int tipo;  
    System.out.println("Área\tMaterial  
                        \tValor");  
    while (area <= 200) {  
        tipo = ALVENARIA;  
        while (tipo <= PLASTICO) {  
            System.out.println(area+"\t"+  
                                tipo+"\t\t"+  
                                valorPiscina(area,tipo));  
            tipo = tipo+1;  
        }  
        area = area+50;  
    }  
}
```

```
public static void main(String[] args) {  
  
    System.out.println("Área\tMaterial  
                        \tValor");  
    for(double area = 50; area <= 200;  
        area = area+50) {  
        for(int tipo = ALVENARIA;  
            tipo <= PLASTICO;  
            tipo = tipo+1) {  
            System.out.println(area+"\t"+  
                                tipo+"\t\t"+  
                                valorPiscina(area,tipo));  
        }  
    }  
}
```

For Aninhado

- Laços *for* podem também ser aninhados

```
public static void main(String[] args) {  
    double area = 50;  
    int tipo;  
    System.out.println("Área\tMaterial  
                        \tValor");  
    while (area <= 200) {  
        tipo = ALVENARIA;  
        while (tipo <= PLASTICO) {  
            System.out.println(area+"\t"+  
                                tipo+"\t\t"+  
                                valorPiscina(area,tipo));  
            tipo = tipo+1;  
        }  
        area = area+50;  
    }  
}
```

```
public static void main(String[] args) {  
  
    System.out.println("Área\tMaterial  
                        \tValor");  
    for(double area = 50; area <= 200;  
        area = area+50) {  
        for(int tipo = ALVENARIA;  
            tipo <= PLASTICO;  
            tipo = tipo+1) {  
            System.out.println(area+"\t"+  
                                tipo+"\t\t"+  
                                valorPiscina(area,tipo));  
        }  
    }  
}
```

- Podem ficar até mais fáceis de serem entendidos

- Embora a **condição** tenha que ser única

```
int a;  
int b;  
for(???; a<b; ???) {  
    System.out.println("a="+a);  
    System.out.println("b="+b);  
}
```

For

- Embora a condição tenha que ser única
- Aceita **múltiplas inicializações**
 - Separadas por vírgula
 - Declaradas fora do for

```
int a;  
int b;  
for(a=1, b=4; a<b; ???) {  
    System.out.println("a="+a);  
    System.out.println("b="+b);  
}
```

For

- Embora a condição tenha que ser única
- Aceita múltiplas inicializações
 - Separadas por vírgula
 - Declaradas fora do for
- E **múltiplas atualizações**
 - Também separadas por vírgula

```
int a;  
int b;  
for(a=1, b=4; a<b; a++,b--) {  
    System.out.println("a="+a);  
    System.out.println("b="+b);  
}
```


For

- Embora a condição tenha que ser única
- Aceita múltiplas inicializações
 - Separadas por vírgula
 - Declaradas fora do for
- E múltiplas atualizações
 - Também separadas por vírgula
- a++? b--?

```
int a;  
int b;  
for(a=1, b=4; a<b; a++,b--) {  
    System.out.println("a="+a);  
    System.out.println("b="+b);  
}
```

Expressões Contraídas

- São “atalhos” →

Expressões contraídas

```
int a;  
int b;  
for(a=1, b=4; a<b; a++,b--) {  
    System.out.println("a="+a);  
    System.out.println("b="+b);  
}
```

Expressões Contraídas

- São “atalhos” →

Expressões contraídas

- Úteis para realizar a operação e armazenar o resultado na mesma variável

```
int a;  
int b;  
for(a=1, b=4; a<b; a++,b--) {  
    System.out.println("a="+a);  
    System.out.println("b="+b);  
}
```

Expressões Contraídas

<i>Expressão</i>	<i>Contraída</i>
<code>x = x + 5</code>	<code>x += 5</code>
<code>x = x - 5</code>	<code>x -= 5</code>
<code>x = x * 5</code>	<code>x *= 5</code>
<code>x = x / 5</code>	<code>x /= 5</code>
<code>x = x % 5</code>	<code>x %= 5</code>

```
int a;  
int b;  
for(a=1, b=4; a<b; a++,b--) {  
    System.out.println("a="+a);  
    System.out.println("b="+b);  
}
```

Expressões Contraídas

<i>Expressão</i>	<i>Contraída</i>
<code>x = x + 5</code>	<code>x += 5</code>
<code>x = x - 5</code>	<code>x -= 5</code>
<code>x = x * 5</code>	<code>x *= 5</code>
<code>x = x / 5</code>	<code>x /= 5</code>
<code>x = x % 5</code>	<code>x %= 5</code>

```
int a;  
int b;  
for(a=1, b=4; a<b; a++,b--) {  
    System.out.println("a="+a);  
    System.out.println("b="+b);  
}
```

- E o ++?

Expressões Contraídas

- Tem duas formas: `x++`
ou `++x`

```
int x = 2;  
int y = 2;  
x++;  
++y;  
System.out.println("x = "  
                    +x+", y = "+y);
```

Saída:

Expressões Contraídas

- Tem duas formas: `x++` ou `++x`

```
int x = 2;  
int y = 2;  
x++;  
++y;  
System.out.println("x = "  
                    +x+", y = "+y);
```

Saída:

x = 3, y = 3

Expressões Contraídas

- Tem duas formas: `x++` ou `++x`
- Usados isoladamente, tanto `++x` quanto `x++` correspondem a `x = x+1`

```
int x = 2;  
int y = 2;  
x++;  
++y;  
System.out.println("x = "  
                    +x+", y = "+y);
```

Saída:

x = 3, y = 3

Expressões Contraídas

- Mas coisas acontecem quando usados em conjunto com outros comandos...

```
int x = 2;
int y = 2;
System.out.println("x = "+
    x++ +", y = "+ ++y);
System.out.println("x = "+
    x +", y = "+ y);
```

Saída:

```
x = 2, y = 3
x = 3, y = 3
```

Expressões Contraídas

- Mas coisas acontecem quando usados em conjunto com outros comandos...
- O que houve?

```
int x = 2;  
int y = 2;  
System.out.println("x = "+  
    x++ +", y = "+ ++y);  
System.out.println("x = "+  
    x +", y = "+ y);
```

Saída:

```
x = 2, y = 3  
x = 3, y = 3
```

Expressões Contraídas

- Mas coisas acontecem quando usados em conjunto com outros comandos...

- O que houve?

- `x++` é um **pós-incremento**

```
int x = 2;
int y = 2;
System.out.println("x = "+
    x++ +", y = "+ ++y);
System.out.println("x = "+
    x +", y = "+ y);
```

Saída:

```
x = 2, y = 3
x = 3, y = 3
```

Expressões Contraídas

- Mas coisas acontecem quando usados em conjunto com outros comandos...

```
int x = 2;
int y = 2;
System.out.println("x = "+
    x++ +", y = "+ ++y);
System.out.println("x = "+
    x +", y = "+ y);
```

- O que houve?

- x++ é um
pós-incremento

Saída:

```
x = 2, y = 3
x = 3, y = 3
```

- Diz que o compilador deve usar o valor que está em x e só então incrementá-lo

Expressões Contraídas

- ++x é um
pré-incremento

```
int x = 2;  
int y = 2;  
System.out.println("x = "+  
                    x++ +", y = "+ ++y);  
System.out.println("x = "+  
                    x +", y = "+ y);
```

Saída:

```
x = 2, y = 3  
x = 3, y = 3
```

Expressões Contraídas

- ++x é um **pré-incremento**
- Diz que o compilador deve primeiro incrementar o valor de x, e só então usá-lo

```
int x = 2;
int y = 2;
System.out.println("x = "+
    x++ +", y = "+ ++y);
System.out.println("x = "+
    x +", y = "+ y);
```

Saída:

```
x = 2, y = 3
x = 3, y = 3
```

Expressões Contraídas

- De forma semelhante ao ++, o -- decrementa, em vez de incrementar
- Também em suas duas formas: x-- e --x

```
int x = 2;  
int y = 2;  
System.out.println("x = "+  
    x-- +", y = "+ --y);  
System.out.println("x = "+  
    x +", y = "+ y);
```

Saída:

Expressões Contraídas

- De forma semelhante ao ++, o -- decrementa, em vez de incrementar
- Também em suas duas formas: x-- e --x

```
int x = 2;
int y = 2;
System.out.println("x = "+
    x-- +", y = "+ --y);
System.out.println("x = "+
    x +", y = "+ y);
```

Saída:

```
x = 2, y = 1
x = 1, y = 1
```


Expressões Contraídas

Mais exemplos:

Código

```
int x = 2;  
int y = x++;  
System.out.println("x = "+ x +", y = "+ y);  
int z = ++x;  
System.out.println("x = "+ x +", z = "+ z);
```

Expressões Contraídas

Mais exemplos:

Código

```
int x = 2;  
int y = x++;  
System.out.println("x = "+ x +", y = "+ y);  
int z = ++x;  
System.out.println("x = "+ x +", z = "+ z);
```

y = x++ fará y conter 2, se x contiver 2 antes do ++

Saída

```
x = 3, y = 2  
x = 4, z = 4
```

z = ++x fará z conter 4, se x contiver 3 antes do ++

Expressões Contraídas

Mais exemplos:

Código

```
int x = 2;  
int y = x++;  
System.out.println("x = " + x + ", y = " + y);  
int z = ++x;  
System.out.println("x = " + x + ", z = " + z);
```

y = x++ fará y conter 2, se x contiver 2 antes do ++

Saída

```
x = 3, y = 2  
x = 4, z = 4
```

z = ++x fará z conter 4, se x contiver 3 antes do ++

Código

```
int x = 1;  
int y = x++ + 4;  
System.out.println("x = " + x + ", y = " + y);  
int z = ++x + 4;  
System.out.println("x = " + x + ", z = " + z);
```

Expressões Contraídas

Mais exemplos:

Código

```
int x = 2;  
int y = x++;  
System.out.println("x = " + x + ", y = " + y);  
int z = ++x;  
System.out.println("x = " + x + ", z = " + z);
```

y = x++ fará y conter 2, se x contiver 2 antes do ++

Saída

```
x = 3, y = 2  
x = 4, z = 4
```

z = ++x fará z conter 4, se x contiver 3 antes do ++

Código

```
int x = 1;  
int y = x++ + 4;  
System.out.println("x = " + x + ", y = " + y);  
int z = ++x + 4;  
System.out.println("x = " + x + ", z = " + z);
```

Saída

```
x = 2, y = 5  
x = 3, z = 7
```

- Considere o código ao lado:

```
public static void main(  
    String[] args) {  
    int x = 1;  
    for (; x<5; x++) {  
        System.out.print(  
            x+" ");  
    }  
}
```

Laços

- Considere o código ao lado:
- O que será impresso?

```
public static void main(  
    String[] args) {  
    int x = 1;  
    for (; x<5; x++) {  
        System.out.print(  
            x+" ");  
    }  
}
```

- Considere o código ao lado:
- O que será impresso?
 - 1 2 3 4

```
public static void main(  
    String[] args) {  
    int x = 1;  
    for (; x<5; x++) {  
        System.out.print(  
            x+" " );  
    }  
}
```

Laços

- Considere o código ao lado:
- O que será impresso?
 - 1 2 3 4
- A **inicialização** em um laço for é opcional

```
public static void main(  
    String[] args) {  
    int x = 1;  
    for (; x<5; x++) {  
        System.out.print(  
            x+" ");  
    }  
}
```


- Considere agora esse código:

```
public static void main(  
    String[] args) {  
    int x = 1;  
    for (; x<5;) {  
        System.out.print(  
            x+" ");  
    }  
}
```

Laços

- Considere agora esse código:
- O que será impresso?

```
public static void main(  
    String[] args) {  
    int x = 1;  
    for (; x<5;) {  
        System.out.print(  
            x+" ");  
    }  
}
```

- Considere agora esse código:
- O que será impresso?
 - 1 1 1 1 1 1 1 1 1 1...

```
public static void main(  
    String[] args) {  
    int x = 1;  
    for (; x<5;) {  
        System.out.print(  
            x+" ");  
    }  
}
```

Laços

- Considere agora esse código:
- O que será impresso?
 - 1 1 1 1 1 1 1 1 1 1...
- **Laço infinito:** a condição de parada nunca é satisfeita

```
public static void main(  
    String[] args) {  
    int x = 1;  
    for (; x<5;) {  
        System.out.print(  
            x+" ");  
    }  
}
```

- Também a **atualização** da variável de controle é opcional

```
public static void main(  
    String[] args) {  
    int x = 1;  
    for (; x<5;) {  
        System.out.print(  
            x+" ");  
    }  
}
```

- E esse código?

```
public static void main(  
    String[] args) {  
    for (int x=1;;x++) {  
        System.out.print(  
            x+" ");  
    }  
}
```

- E esse código?
 - 1 2 3 4 5 6 7 8...

```
public static void main(  
    String[] args) {  
    for (int x=1;;x++) {  
        System.out.print(  
            x+" ");  
    }  
}
```

Laços

- E esse código?
 - 1 2 3 4 5 6 7 8...
- De novo! Ninguém disse ao laço o que testar para parar

```
public static void main(  
    String[] args) {  
    for (int x=1;;x++) {  
        System.out.print(  
            x+" ");  
    }  
}
```


Laços

- E esse código?
 - 1 2 3 4 5 6 7 8...
- De novo! Ninguém disse ao laço o que testar para parar
- A **condição de parada** em um laço for também é opcional

```
public static void main(  
    String[] args) {  
    for (int x=1;;x++) {  
        System.out.print(  
            x+" ");  
    }  
}
```

Em Suma:

- Inicialização, condição e atualização são opcionais
- A condição aceita qualquer expressão que resulte em verdadeiro ou falso (expressões lógicas e relacionais)
- Inicialização e atualização são apenas códigos rodados, respectivamente, antes da primeira iteração e ao fim de cada iteração do laço

<https://www.youtube.com/watch?v=m0rsP1-wt1Me>
<https://www.youtube.com/watch?v=SKoEmpiSXm0>