Programação Orientada à Objetos (POO)

CIÊNCIA DA COMPUTAÇÃO

Prof. Dr. João Paulo Aramuni

UNIVERSIDADE FUMEC

Sumário

* **Controle de Fluxo**
  * Variáveis e escopo
  * Operadores
  * Promoções e Casting
  * Instruções condicionais e de laço

Controle de Fluxo

* **Variáveis e escopo**
  * Java é uma linguagem fortemente tipada.
    * Toda variável precisa ter um tipo.
  * Uma variável é declarada colocando-se o tipo seguido pelo nome do variável.
  * Exemplos:
    * **int** umaVariavelInteira;
    * **double** variavelReal = 14.7; //variável declarada e inicializada
    * **boolean** achou , b; // declaração de duas variáveis do mesmo tipo

Controle de Fluxo

* **Variáveis e escopo**
  * Variáveis podem ser declaradas em qualquer lugar, desde que antes de seu uso.
    * Variáveis podem ser declaradas na primeira expressão de um loop **for**.

```
for (int i = 0; i < 10; i = i + 1) {
    System.out.println("olá!");
}
```

Controle de Fluxo

- **Variáveis e escopo**
  - Escopo de variáveis:
    - Um bloco ou instrução composta é qualquer número de instruções simples Java que são delimitadas por um par de chaves { ... }.
    - Variáveis são sempre locais a um bloco.
    - Não existe o conceito de variáveis globais a um programa.
    - Exemplo:

```
public static void main (String[] args) {
        int  n;         // n é local ao método main
        ...
        {  int  k;
        ...             // k é local a este bloco
        }
        ...             // k não vale aqui
}
```

Controle de Fluxo

* **Variáveis e escopo**
  * Escopo de variáveis:
    * public class AmbitoVariaveis {

```
    public static void main(String[] args) {
      if (true) {
        int y=5;
      }
      System.out.println(y);
    }
}
```

    * Se tentarmos executar este código o compilador nos dará um erro dizendo que a variável não está definida visto que a declaramos em um bloco diferente de onde pretendíamos utiliza-la.

Controle de Fluxo

- **Variáveis e escopo**
  - A atribuição de valores a variáveis é feita pelo operador "=".
    - Exemplo:
      - char c; // declaração
      - c = 'S'; // atribuição
  - Múltiplas variáveis podem ser atribuídas em uma única expressão:
    - Exemplo:
      - int x, y, z;
      - x = y = z = 0;

Controle de Fluxo

* **Variáveis e escopo**
  * Toda variável deve ser explicitamente inicializada.
    * O compilador avisa se isso não ocorrer.
  * Dê bons nomes às variáveis e organize-as de forma lógica de acordo com as boas práticas de programação.

Controle de Fluxo

- **Variáveis e escopo**
  - Constantes:
    - Uma constante é definida usando-se a palavra-chave **final.**
    - Exemplo:

```
public class ExemploConstante {
        public static void main (String[] args) {
            final double PI = 3.14;
            System.out.println("O valor de Pi é "+ PI);
        }
}
```

    - A convenção de Java é sempre usar letras maiúsculas para constantes.

Controle de Fluxo

- **Operadores**
  - **Operadores Aritméticos**
  - \+ adição
  - \- subtração
  - * multiplicação
  - / divisão (inteira, se os operandos forem inteiros e ponto flutuante, caso contrário)
  - % módulo (resto de divisão inteira)

  - Não existe operador de exponenciação. Para isso, use o método **pow** da classe **Math** do pacote **java.lang**.
    - **double** y = **Math.pow**(x,a); // $y = x^a$
  - A classe Math do pacote java.lang possui um grande número de funções matemáticas.

Controle de Fluxo

* **Operadores**
  * **Operadores Aritméticos**

```
public class TesteAritmetico {
    public static void main (String args[]) {
        short x = 6;
        int y = 4;
        double a = 12.6;
        double b = 3.0;
        System.out.println (“x é “ + x + “, y é “ + y );
        System.out.println (“x + y = “ + (x + y) );
        System.out.println (“x - y = “ + (x - y) );
        System.out.println (“x / y = “ + (x / y) );
        System.out.println (“x % y = “ + ( x % y ) );
        System.out.println (“a é “ + a + “, b é “ + b );
        System.out.println (“ a / b = “ + ( a / b ) );
        System.out.println (“ 11.0 / 3 = “ + ( 11.0 / 3 ) );
        System.out.println (“ 11 / 3 = “ + ( 11 / 3 ) );
    }}
```

Controle de Fluxo

* **Operadores**
  * **Operadores de Incremento e Decremento**
  * O operador de incremento adiciona 1 a uma variável numérica e o operador de decremento, subtrai 1.

```
int n = 5;
n++; // faz n = 6
n--; // faz n = 5
```

– Forma pré-fixada: o incremento/decremento é executado antes da avaliação da expressão.

– Forma pós-fixada: o incremento/decremento é executado depois da avaliação da expressão.

```
int m = 5;
int n = 5;
int a = 2 * ++m; // faz a = 12  e  m = 6
int b = 2 * n++; // faz b = 10  e  n = 6
```

Controle de Fluxo

- **Operadores**
  - **Forma Reduzida de Operadores de Atribuição**

  - Forma reduzida de operadores aritméticos binários:

```
x  +=  y   // significa x = x + y;
x  -=  y   // significa x = x – y;
x  *=  y   // significa x = x * y;
x  /=  y   // significa x = x / y;
```

  - Exemplo:

```
int z = 4;
int w = 6;
z += w;   // significa z = z + w;
```

Controle de Fluxo

* **Operadores**
  * **Operadores Relacionais e Lógicos**

  * Operadores Relacionais:

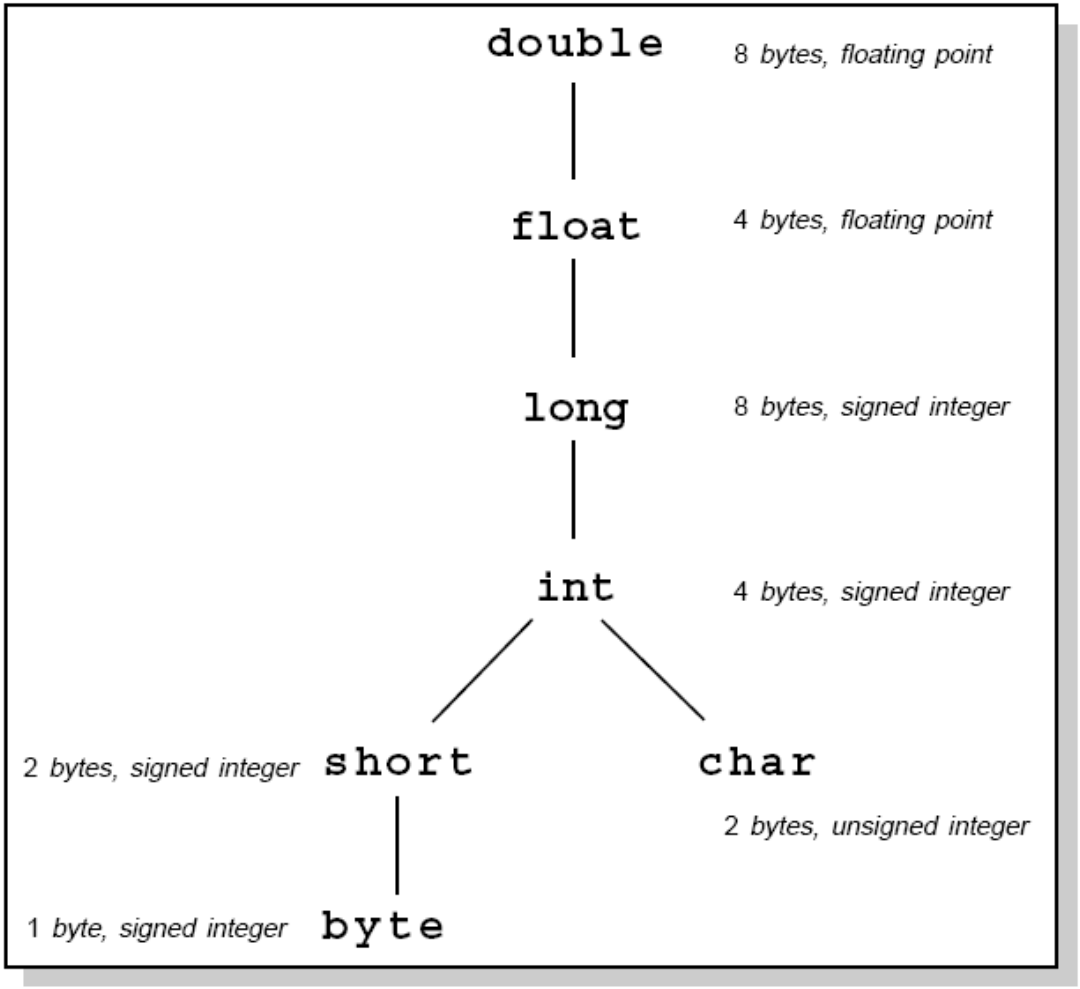
==	Igual	x == 3
!=	Diferente	x != 3
<	Menor que	x < 3
>	Maior que	x > 3
<=	Menor ou igual	x <= 3
>=	Maior ou igual	x >= 3

  * Operadores Lógicos:

&&	Operação lógica AND (E)
||	Operação lógica OR (OU)
!	Operação lógica NOT (Negação)
^	Operação lógica XOR (OU Exclusivo)

Controle de Fluxo

* **Promoções e Casting**



Controle de Fluxo

* **Promoções e Casting**
  * Os tipos byte, short, int, long, float e double são *signed*, ou seja, possuem um intervalo de valores que vai de um número negativo até um número positivo.
  * O tipo char é o único tipo *unsigned* do Java, tendo 2 bytes e um intervalo de valores entre 0 e 65535.

Controle de Fluxo

* **Promoções e Casting**

Um **byte** pode ser convertido em um `short`, `int`, `long`, `float` ou `double`

Um **short** pode ser convertido em um `int`, `long`, `float` ou `double`

Um **char** pode ser convertido em um `int`, `long`, `float` ou `double`

Um **int** pode ser convertido em um `long`, `float` ou `double`

Um **long** pode ser convertido em um `float` ou `double`

Um **float** pode ser convertido em um `double`

Controle de Fluxo

* **Promoções e Casting**
  * As conversões explícitas, chamadas de ‘casting’, são permitidas em todos os tipos (exceto o boolean), mas o programador deve estar ciente que poderá haver perda de bits.

Controle de Fluxo

- **Promoções e Casting**
  - Não há necessidade de conversão explícita se não houver possibilidade de perda. Da esquerda para a direta, as conversões possíveis são:
  - byte --> short --> int --> long --> float --> double
  - char --> int
  - Exemplos:
    - **double** d = 4.5; **float** f = 66.4f;
    - **long** l = 15; **int** i = 2;**char** c = 'H';
    - d = d + f; // correto
    - d = f + l; // correto
    - f = d + l; // incorreto, resultado deve ser double
    - l = 3 * i; // correto
    - i = c; // correto, resultado inteiro correspondente ao
      // código Unicode do caractere 'H'.

Controle de Fluxo

* **Promoções e Casting**
  * Em conversões onde houver possibilidade de perda de informação, as conversões devem ser explícitas (cast).
    * A sintaxe é dada pelo tipo resultante em parênteses, seguido pelo nome da variável.
    * Exemplos:
      * double x = 3.987;
      * int y = (int) x; // y = 3 (truncamento)

      * Para arredondar, use o método Math.round:
      * double x = 3.987;
      * int y = (int) Math.round(x); // y = 4 (arredondamento)
      * // é necessário a conversão explícita pois round retorna um long.

Controle de Fluxo

* **Promoções e Casting**
  * Promoção x Casting
    * A promoção é a elevação dos tipos numéricos existentes na linguagem para um tipo que pode representar uma maior grandeza.
    * As regras do que pode ser promovido estão especificadas na linguagem e indicam sempre que um tipo primitivo de menor capacidade numérica pode ser usado em um lugar onde será necessário um tipo com maior capacidade.
    * A promoção nunca provoca perda de dados. Ela é feita implicitamente pelo compilador.

Controle de Fluxo

* **Promoções e Casting**
  * Promoção x Casting
    * O casting é a mudança de um tipo para outro de forma explícita através do operador de casting (o nome do tipo entre parênteses).
    * Pode haver uma mudança de representação do valor ou não, e nesta conversão pode haver perda de dados. Um erro em tempo de execução ocorrerá se não for possível realizar a conversão ou o compilador pode detectar antes a impossibilidade da operação.
    * O casting pode ou não mudar a representação do valor, o que não ocorre em tipos por referência.

Controle de Fluxo

* **Promoções e Casting**
  * É possível a conversão de tipos através das classes Integer, Double e String.

  * Vejamos um exemplo:

Controle de Fluxo

* **Promoções e Casting**

```
public static void main( String[ ] args ) {
        String v1 = "10";
        int v2  = Integer.parseInt(v1); // String para Inteiro
        String v3 = "5.60";
        double v4 = Double.parseDouble(v3); // String para Real
        String v5 = String.valueOf (v2);  // Inteiro para String
        String v6 = String.valueOf (v4); // Real para String
        System.out.println ("v1 = " + v1 + "\n" +
        "v2 = " + v2 + "\n" +  "v3 = " + v3 + "\n" +
        "v4 = " + v4 + "\n" + "v5 = " + v5 + "\n" +
        "v6 = " + v6  );
}
```

Controle de Fluxo

* **Manipulação de Strings**
  * Exemplos:

    String s = “”; // string vazia

    String saudacao = “Ola”;

  * A concatenação de string é feita pelo operador +.
    * Exemplo: String s = “Curso de ” + “Java”;// s = “Curso de Java”

Controle de Fluxo

* **Manipulação de Strings**
  * Ao concatenar uma string com um valor que não é string, este último é convertido para uma string.
  * Exemplos:

```
final int IDADE = 18;
String censura = “Censurado para menores de “ + IDADE + “ anos”;
int m = 20;
System.out.prinln( “O valor de m e’ “ + m );
```

Controle de Fluxo

* **Operações da Classe String**
  * public String substring(int inicio, int fim)
    * Retorna uma substring.
    * Posição inicial é zero.
    * Especifica-se a posição inicial (inicio) e a primeira posição não incluída (fim).
    * Exemplo:

      String str = “Curso de Java”;

      String j = str.substring(9,13); // j = “Java”
  * public String substring(int inicio)

Controle de Fluxo

- **Operações da Classe String**
  - public int length()
    - Retorna o tamanho da string (número de caracteres).
    - Exemplo (continuação):
      - int tam = str.length(); // tam = 13
  - public char charAt(int indice)
    - Retorna o caractere da posição índice.
    - Exemplo (continuação):
      - char c = str.charAt(9); // J

Controle de Fluxo

* **Operações da Classe String**
  * public boolean equals(Object obj)
    * Retorna true se o conteúdo da string for igual ao conteúdo do objeto (string) obj, e false caso contrário.
    * Exemplo (continuação):
      * boolean b = str.equals(“Curso de Java”); // b = true
      * Obs.: NÃO use o operador == para testar igualdade de strings.
      * == simplesmente compara se duas strings apontam para um mesmo local.

Controle de Fluxo

* **Operações da Classe String**
  * public int indexOf(String substring)
    * Retorna a posição da 1ªocorrência substring na string.
    * Posição inicial da pesquisa é zero.
    * Retorna -1 um se não encontrou nenhuma ocorrência
    * Exemplo:

```
String str = “Curso de Java”;
int j = str.indexOf(“so”); //  j = 3
```

Controle de Fluxo

* **Operações da Classe String**
  * public int indexOf(String substring, int indice)
    * A pesquisa será realizada a partir da posição índice.
    * Exemplo:

```
String str = “Curso de Java”;
int j = str.indexOf(“so”,5); //  j = -1
```

Controle de Fluxo

- **Operações da Classe String**
  - public int compareTo(String str)
    - retorna:
      - um inteiro menor que zero se a string vier antes (ordem alfabética) que str;
      - zero se a string for igual a str;
      - um inteiro maior que zero se a string vier depois (ordem alfabética) que str.
    - Exemplo (continuação):
      - int x = str.compareTo(“Curso de Java”); // x = 0

Controle de Fluxo

* **Operações da Classe String**
  * A classe String possui mais de 50 métodos.
    * Consulte a documentação para maiores detalhes! (equalsIgnoreCase, toLowerCase, toUpperCase(), replaceAll, matches, split, trim ...)

Controle de Fluxo

* **Formatação de Datas**
  * Método **format** para formatar uma Data
  * Classe java.text.SimpleDateFormat

* **Formatação de Datas**
  * Principais símbolos usados na máscara de formatação

Símbolo	Significado	Apresentação	Exemplo
y	Year	(Number)	1996 , 96
M	Month in year	(Text & Number)	July & 07
w	Week in year	(Number)	27
W	Week in month	(Number)	2
d	day in month	(Number)	10
D	day in year	(Number)	189
F	Day of week in month	(Number)	2
E	day in week	(Text)	Tuesday
H	hour in day(0~23)	(Number)	0
m	minute in hour	(Number)	30
s	second in minute	(Number)	55
S	Millisecond	(Number)	978

Controle de Fluxo

* **Formatação de Datas**
  * Exemplo:

```
public static void main(String[ ] args) {
    GregorianCalendar dta = new GregorianCalendar();
    System.out.println("Data : " + new      SimpleDateFormat   (“EEEE
    dd/MM/yyyy          HH:mm:ss,S").format(dta.getTime()));
    // Data : Segunda-Feira 10/02/2009 17:50:29,578
}
```

Controle de Fluxo

* **Formatação de Números**
  * Método **format** para formatar Números
  * Classe java.text.DecimalFormat

* **Formatação de Números**
  * Principais símbolos usados na máscara de formatação

Símbolo	Significado	Apresentação	Exemplo
0	Substituição de Dígito	000 => 23	023
#	Supressão de zero	##0 => 23	23
.	Separar Decimal	##0.00 => 2.35	2,35
,	Separar Milhar	R$#,##0.00 => 1200.5	R$1.200,00
E	Notação Científica	00.##E0 => 1324000000303030	13,24E14
%	Porcentagem	##0.00% => 0.01	1,00%
;	Máscaras Alternadas	#0.00; -#0.00 => -1.5	-1,50
		#0.00; (#0.00) => -9.5	(9,50)

Ex.: System.out.println("Valor Formatado : "
+ new DecimalFormat("R$#,##0.00").format(1200.5));
// Valor Formatado : R$1.200,00

Controle de Fluxo

- **Instruções condicionais e de laço**
  - Comandos if ... Else
    - if (condição) comando;
    - if (condição) { bloco de comandos }
    - if (condição) comando1 ; else comando2 ;
    - if (condição) { bloco1 } else { bloco2}
      - else é opcional; condição entre parênteses.
    - Exemplo 1:
      - **if** (nota >= 60)
        System.out.println(“Aprovado”);
        else
        System.out.println(“Reprovado”);

Controle de Fluxo

- **Instruções condicionais e de laço**
  - Comandos if ... Else
    - Exemplo 2:

```
if (x > 5) {
    if (y > 5)
        System.out.println("x e y maiores do que 5");
}
else  // sem o bloco, este else pertenceria ao segundo if
System.out.println("x menor ou igual a 5");
```

Controle de Fluxo

- **Instruções condicionais e de laço**
  - Operador ternário '?'
    - Para escrever uma operação condicional sem usar if-else.
    - condição ? e1 : e2
    - Avalia e1 se condição for verdadeira ou e2 caso contrário.

    - Exemplo:
      - **System.out.println** (nota >= 70 ? "Aprovado" : "Reprovado") ;

Controle de Fluxo

* **Instruções condicionais e de laço**
  * Operador ternário '?'

    Exemplo :

```
// a linha seguinte
    media = numeroDeNotas == 0 ? 0.0 : soma/numeroDeNotas;
// é equivalente às linhas seguintes
    if( numeroDeNotas  ==  0 ) {
       media = 0.0;
    } else {
       media = soma / numeroDeNotas;
    }
```

Controle de Fluxo

* **Instruções condicionais e de laço**
  * Comando **switch**

```
switch (opção) {
    case valor1 : comandos1;  break;
    case valor2 : comandos2; break;
    ...
    Default :     comandosn;  break;
}
```

- Opção deve ser do tipo char, byte, short, int ou long.

- A execução começa no case que coincide com o valor da seleção realizada, e continua até o break seguinte ou o final do switch. A cláusula default é opcional e será executada se nenhum valor coincidir na seleção.

Controle de Fluxo

- **Instruções condicionais e de laço**
  - Comando **while**
    - while (condição) { bloco }
    - repete execução do bloco enquanto condição for verdadeira.
      - Se a condição inicial é falsa, o bloco não é executado nenhuma vez.
    - Exemplo:

```
int cont = 1;
while (cont <= 10) {
    System.out.println ( “contador = ” + cont );
    cont++;
}
```

Controle de Fluxo

* **Instruções condicionais e de laço**
  * Exemplo:

```
public class teste {
public static void main (String args[])
{
        int total = 0;
        int indice = 0;
        while (indice < 100 ) {
              indice++;
              total = total + indice;   // total += indice;
        }
   System.out.println("A soma dos números de 1 a 100 é igual a: " + total);
}
}
```

Controle de Fluxo

- **Instruções condicionais e de laço**
  - Comando **do ... while**
    - **do** { bloco } **while** (condição)
    - repete execução do bloco enquanto condição for verdadeira.
    - O bloco é executado pelo menos uma vez.
    - Exemplo:

```
int cont = 1;
do {
      System.out.println ( "contador = " + cont );
      cont++;
} while (cont <= 10);
```

Controle de Fluxo

* **Exemplo completo:**

```
// Utilizar a classe Console
public class Menu  {
    private void menu() {
    int opcao ;
    do  {
        System.out.println("Menu Principal”);
        System.out.println("1-Primeira Palavra do Texto”);
        System.out.println("2-Ultima Palavra do Texto”);
        System.out.println(“0-Sair");
        opcao = Console.readInt("Opção: ”);
        switch (opcao) {
                case 1 :
                primeiraPalavraTexto();
                break;
                case 2 :
                ultimaPalavraTexto();
                break;
                case 0 :
                break;
                default  : System.out.println( “Opção Inválida.” );
                break;
      }
} while ( opcao != 0 );
}
```

Controle de Fluxo

* **Instruções condicionais e de laço**
  * Comando **for**
    * **for** (comando; expressão1; expressão2;) { bloco }

    * Equivalente a:

```
{  comando;
while ( expressão1 ) {
bloco;
expressão2;
}}
```

Controle de Fluxo

* **Instruções condicionais e de laço**
  * Comando **for**
    * O comando inicializa um contador (que pode ser declarado aqui), a expressão1 fornece a condição de teste antes de cada passagem pelo laço e a expressão2 determina a alteração do contador.

      * Qualquer expressão é válida nos segmentos do for, mas é aconselhável usar apenas o necessário para inicializar, testar e atualizar o contador.

Controle de Fluxo

* **Exemplo completo:**

```
public class teste {
    public static void main ( String args[] ){
    int total = 0;
    int indice;
    for (indice = 1; indice <= 100; indice++) {
            total += indice;
   }
   System.out.println("A soma dos números de 1 a 100 é igual a: " +
   total);
   }
}
```

Controle de Fluxo

* **Outros exemplos:**

Exemplo1:

```
for ( int cont = 1;  cont <= 10;  cont++ )
        System.out.println ( “contador = ” + cont );
// cont é válida somente dentro do bloco do for
```

Exemplo2:

```
int i;
Int j = 2;
for ( i = 15; i > j; i -= 2 ) {
        System.out.println ( “contador = ” + i );
        j++;
}
// i é válida depois do bloco do for
```

Controle de Fluxo

* **Instruções condicionais e de laço**
  * Comandos para auxiliar no controle de blocos de repetição:
    * **break**
    * **continue**

  * Não é considerado uma boa prática de programação.

Controle de Fluxo

* **Instruções condicionais e de laço**
  * **break** não rotulado:
  * Usado para sair de um laço simples.
  * Exemplo:

```
while (i <= 100) {
    saldo = saldo + deposito + juros;
    if ( saldo > meta )  break;
    i++;
}
// programa sai do laço while se i > 100
// ou se saldo > meta.
```

Controle de Fluxo

* **Instruções condicionais e de laço**
  * **break** rotulado:
  * Usado para sair de laços aninhados.

  * Exemplo:

* **Exemplo 1:**

```
int n;
lerDados: // rótulo identificador do laço while ( ... ) {
While ( ... ) { // 1º laço
...
for ( ... ) { // 2º laço
int n = Console.readInt("n : ");
if (n < 0) // uma condição que normalmente não deveria ocorrer
break lerDados; // sai do laço lerDados (while externo)
...
} // Final 2º laço
} // Final 1º laço
System.out.println( "Valor de n : " + n );
```

* **Exemplo 2:**

```
public class BreakRotulado {
  public static void main(String[] args) {
    int i = 0;
    laco1: // label do laço
    while ( i <= 10 ) { // laço externo
            for ( int j=1; j<=10; j++ ) {    // laço interno
               if ( j==6 && i == 6)  break  laco1; // sair do laço externo
             }
     i++;
     }
    System.out.println( "Saída -> i : " + i ); // Saída -> i : 6
  }
}
```

Controle de Fluxo

* **Instruções condicionais e de laço**
  * Comando **continue**
  * Usado para retornar para o início de um laço.

  * Exemplo:

```
int total = 0;
for ( int i = 0; i < 100; i++ ) {
                    if (i ==49) || (i==59) continue;
    total += i;
}
```

Controle de Fluxo

* **Instruções condicionais e de laço**
  * Laço **for each**
  * Aplicado para o caso de acessar todos os elementos de um array sem ter que lidar com os valores de índices.
  * Vejamos um exemplo:

Controle de Fluxo

* **Instruções condicionais e de laço**

```
int [ ]  a = new int[100];  // array a com 100 elementos

// Opção tradicional
for  (int i = 0 ; i < 100 ; i++)
a [i] = i; // Preenche o array com 0 a 99

// Opção usando laço "for each"  ->
for (NomeClasse  nomeVariavel : nomeLista)
{ comandos; };
for  ( int  elemento : a )
System.out.println (elemento);
```

Obrigado.

joaopauloaramuni@gmail.com
joaopauloaramuni@fumec.br

UNIVERSIDADE FUMEC
50 Anos