Controle de Fluxo e Estrutura de Dados

Sumário

- Estruturas de Dados
  - Array
    - Simples e Multidimensionais
  - ArrayList
- Instruções Condicionais
  - if-else
  - switch
  - for..
  - while
  - do while
  - else Oscilante
  - break continue e break

Estruturas de Dados: Array

- Todo array é um objeto
- Para determinarmos o seu tamanho podemos usar o método **`length`**:

```
int lista [] = new int [10];
for (int j = 0; j < lista.length; j++)
{
   System.out.println(lista[j]);
}
```

Estruturas de Dados: Array

Declaração

```
char s[ ];                  // declaração
s = new char[3];            // Criação
s[0] = 'A';                 // atribuição
s[1] = 'B';                 // atribuição
s[2] = 'C';                 // atribuição
```

Estruturas de Dados: Array

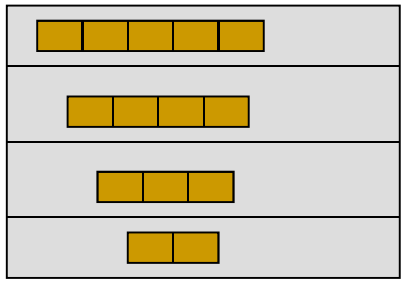
- Declarando, criando e iniciando um array

```
char s[ ]={'A','B','C'};     // declaração,
                             // criação e
                             // incialização
```

Estruturas de Dados: Array

```
import java.util.Scanner;
public class CriaArray {
    public static void main (String[] args) {

        Scanner entrada = new Scanner(System.in);

        // Criação e Definição do tamanho
        String[] nomes = new String[10];
        int[] idades      = new int[10];
        ...
        // Atribuição de valores
        nomes[i] = entrada.next();
        idades[1] = entrada.nextInt();
        ...
    }
}
```

Estruturas de Dados: Array



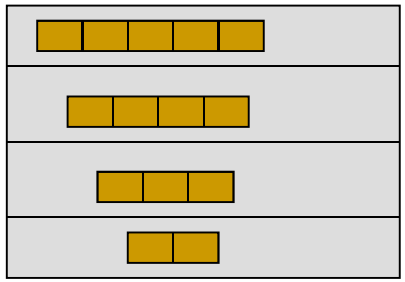
Multi-dimensionais

linhas

colunas

```
int duasDim [ ] [ ] = new int [4] [ ];
duasDim [0] = new int [5];
duasDim [1] = new int [4];
duasDim [2] = new int [3];
duasDim [3] = new int [2];
duasDim [0][0] = 300;
duasDim [1][3] = 600;
```

Estruturas de Dados: Array



Multi-dimensionais

```
int duasDim [ ] [ ] = new int [4] [ ];
duasDim [0] = new int [5];
duasDim [1] = new int [4];
duasDim [2] = new int [3];
duasDim [3] = new int [2];
duasDim [0][0] = 300;
duasDim [1][3] = 600;
```

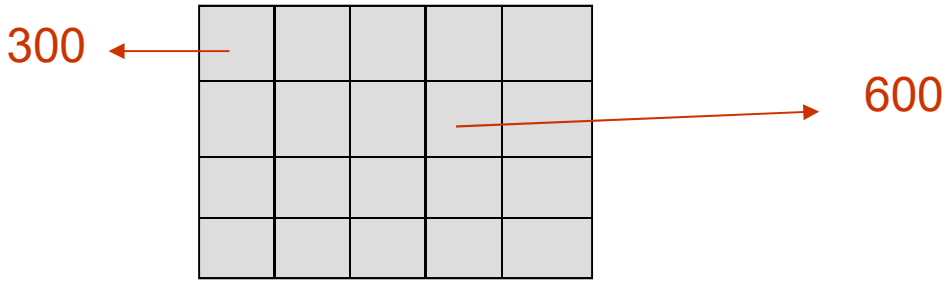
TESTE

```
for (int coluna=0; coluna< 4;coluna++)
        System.out.println("Valor da posicao: " + duasDim[1][coluna]);
```

Estruturas de Dados: Array

Multi-dimensionais, outra forma

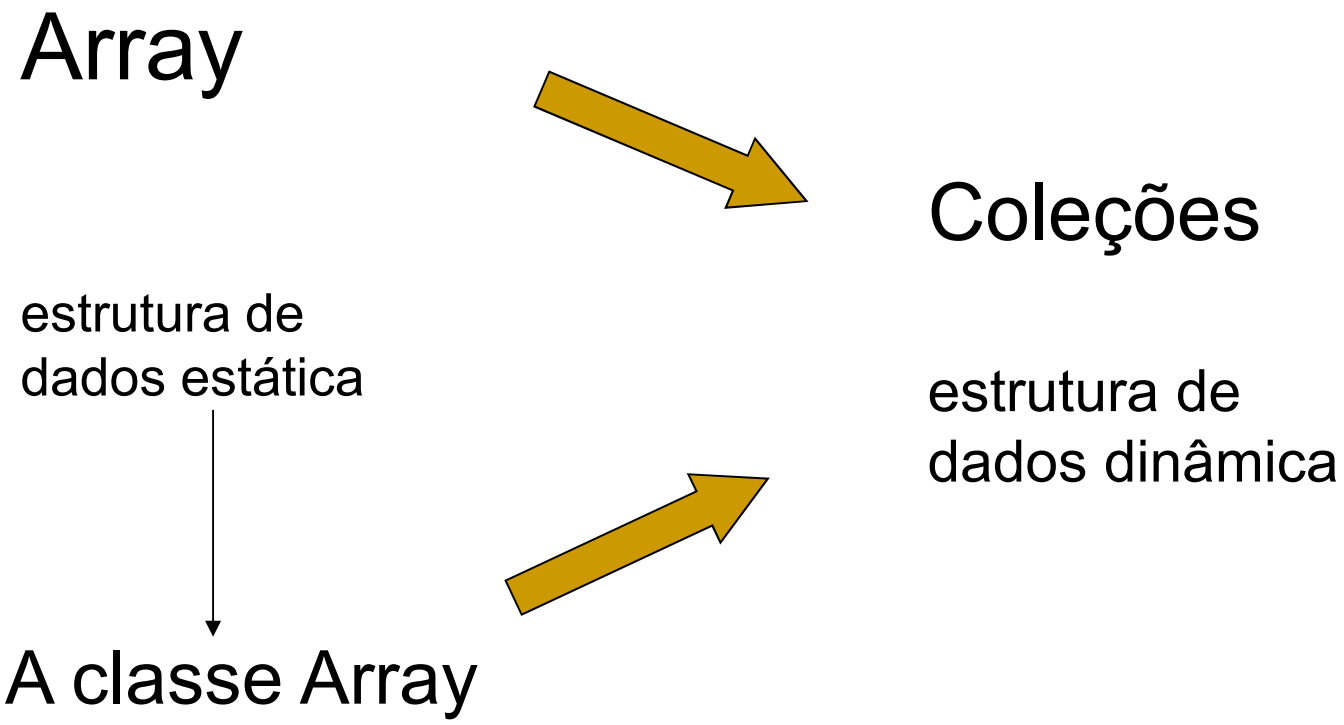
```
int matriz [][] = new int [4][5];
matriz [0][0] = 300;
matriz [1][3] = 600;
```



Estruturas de Dados: Array

```
import java.util.Scanner;
public class CriaMatriz {
    public static void main (String[] args) {

            Scanner entrada = new Scanner(System.in);

            // Criação e Definição do tamanho
            double[][] consumo = new double[5][2];
            ...
            // Atribuição de valores
            consumo[i][0] = entrada.nextDouble();
            consumo[i][1] = entrada.nextDouble();
            ...
    }
}
```

Estruturas de Dados



Classe Arrays

O usuário criava o seu array :

```
...
boolean achou = false;
String [ ] nomes = { “Andre”, “Angelica”, “Camila” , “Jose” , “Maria”};
for (int i=0 ; i<nomes.lenght ; i++)
{
        if (nome[i].equals(“Maria”)) achou = true;
}
...
```

Manipulação de array de “baixo nível” → programação feita pelo programador, sem utilizar os métodos da classe.

A Classe Arrays

Métodos de “alto nível” da classe Arrays:

- binarySearch() pesquisa em array ordenado
- equals() compara arrays
- fill() colocar valores em um array
- sort() ordenar um array

Esses métodos são sobrecarregados para um array de tipo primitivo e arrays do tipo Object

A Classe Arrays

Exemplos:

```
private int arrayA[] = new int[10];
Arrays.fills(arrayA, 7);
```
arrayA = {7,7,7,7,7,7,7,7,7,7};

```
private int arrayB[] = {3,4,6,2,1,5};
Arrays.sort(arrayB);
```
arrayB = {3,4,6,2,1,5};
arrayB = {1,2,3,4,5,6};

```
Arrays.binarySearch(ArrayB, 6);
```
5

```
boolean testando = Arrays.equals(arrayA, arrayB);
```
false

A Classe Arrays

Tente:

Construa um array com 7 posições = {“Janeiro”, “Fevereiro”, “Marco”, “Abril”, “Maio” , “Junho”, “Julho”}. Agora faça:

Ordene seu array

Procure por uma palavra nele;

Agora repita as oeprações acima usando a classe Arrays.

A Classe Arrays

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

public class ArrayEArrays {

public static void main(String[] args) {
  String[] lista = {"Janeiro", "Fevereiro", "Marco", "Abril", "Maio" ,
                    "Junho", "Julho"};
  String aux, palavra;
  Scanner entrada = new Scanner(System.in);
  //-------------------------------------------------------------
  // HARD CODE
  //-------------------------------------------------------------
  System.out.println("HARD CODE");
  System.out.println("Ordenando Array...");
  String elementoMenor = lista[0];
  for (int i=0; i< 6; i++) {
    for (int j=i+1; j< 7; j++)
      if (lista[i].compareToIgnoreCase(lista[j])>0)
      {     elementoMenor=lista[i];
            lista[i]=lista[j];
            lista[j]=elementoMenor;
       }
  }
  for (int i=0 ; i<7; i++){
      System.out.print(lista[i] + ", ");
   }
```

```
   System.out.println("\nEntre com uma palavra");
    palavra = entrada.nextLine();
    System.out.println("Procurando palavra " + palavra +
"...");
   for (int i=0 ; i<7; i++)
       if (palavra.equals(lista[i]))
          System.out.println("Achei a palavra");

// -------------------------------------------------------------
//  UTILIZANDO A CLASSE ARRAYS
// -------------------------------------------------------------

   System.out.println("\nUTILIZANDO A CLASSE
ARRAYS");
   Arrays.sort(lista);

   for (int i=0 ; i<7; i++)
      System.out.print(lista[i] + ", ");

   System.out.println("\nProcurando palavra " + palavra
+ "...");
   if ( Arrays.binarySearch(lista, palavra) >= 0)
       System.out.println("Achei a palavra");

//-------------------------------------------------------------

   }
}
```

Estrutura de Dados: Coleções

As estruturas de coleção Java (ou o Java Collection Framework) oferecem ao programador acesso a estruturas de dados pré-empacotadas e a algoritmos para manipular essas estruturas de dados.

Estrutura de Dados: Coleções

Ao invés de se preocupar com a maneira como as estruturas foram implementadas, o programador simplesmente utiliza as estruturas de dados.

- código mais rápido
- melhora no desempenho
- maximiza a velocidade de execução
- minimiza o consumo de memória
- diminui o tamanho do código

Estrutura de Dados: Coleções

Coleção é uma estrutura de dados – na verdade um objeto – que pode armazenar outros objetos.

Exemplos anteriores a Versão Java2:

→ Vector, Stack e Hashtable

Exemplos da Versão Java2 em diante:

→ Collection, Set, List, Map

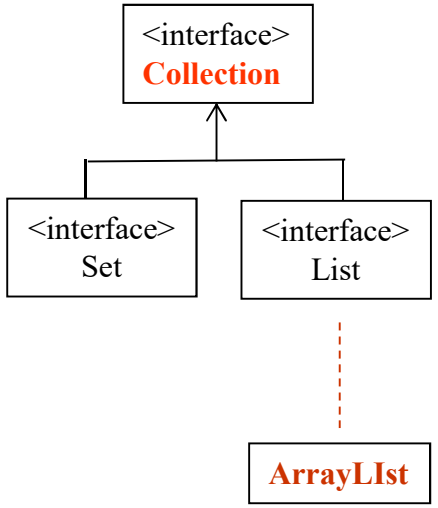
As Collections serão vistas em mais detalhes no próximo módulo, por agora veremos o exemplo somente da classe ***ArrayList***.

ArrayList

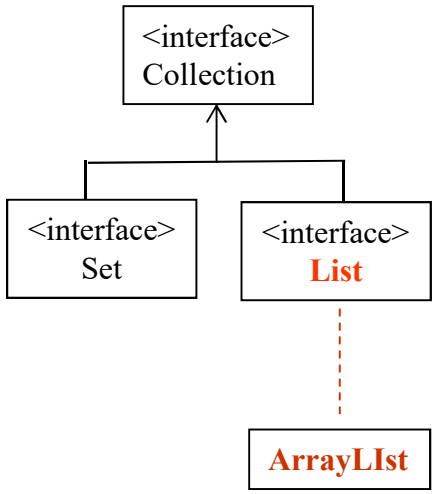
Interface Collection

Métodos:

- adicionar, limpar, comparar e reter elementos
- podem ser convertidas em arrays
- fornece um método que retorna um Iterator



ArrayList

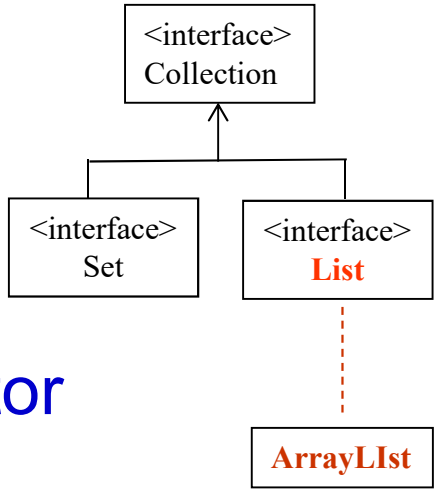


List é uma Collection que pode **conter elementos duplicados**.

A interface é implementada pelas classes:

- ArrayList (funciona como um array redimensionável comportamentos e recursos semelhantes a classe Vector não sincronizada)
- LinkedList funciona como uma lista encadeada (filas simples e duplamente encadeadas, pilhas, árvores)
- Vector

ArrayList



- ArrayList
- LinkedList
- Vector

public class **LinkedList<E>**
extends AbstractSequentialList<E>

implements **List**<E>, Queue<E>, Cloneable, Serializable

public class **ArrayList<E>**
extends AbstractList<E>

implements **List**<E>, RandomAccess, Cloneable, Serializable

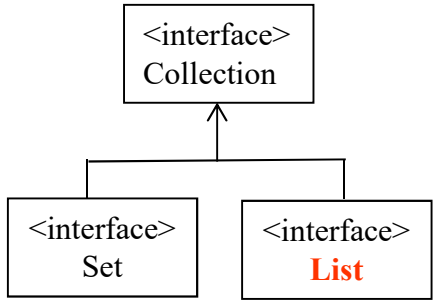
public class **Vector<E>**
extends AbstractList<E>

implements List<E>, RandomAccess, Cloneable, Serializable

ArrayList

```
import java.awt.Color;
import java.util.*;

public class Collecao {

   public static void main( String args[] )
   {String colors[] = { "red", "white", "blue" };

     ArrayList list = new ArrayList();

     list.add( Color.magenta );

     for ( int count = 0; count < colors.length; count++ )
        list.add( colors[ count ] );

    list.add( Color.cyan );

    System.out.println( "\nArrayList: " );
     for ( int count = 0; count < list.size(); count++ )
       System.out.print( list.get( count ) + " " );

     Iterator iterator = list.iterator();

     while ( iterator.hasNext() )

           if ( iterator.next() instanceof String )
              iterator.remove();

      System.out.println( "\n\nArrayList depois de chamar removeStrings: " );

      for ( int count = 0; count < list.size(); count++ )
           System.out.print( list.get( count ) + " " );
   }
}
```

ArrayList



Saída:

ArrayList:
java.awt.Color[r=255,g=0,b=255] red white blue java.awt.Color[r=0,g=255,b=255]

ArrayList depois de chamar removeStrings:
java.awt.Color[r=255,g=0,b=255] java.awt.Color[r=0, g=255, b=255]

ArrayList

- Tente:

Construa um aplicativo que va inserindo em um arrayList os alunos de uma turma. Liste.

Ao final, escolha uma letra e remova todos os nomes que começarem com ela. Liste.

ArrayList

```
public class TestaArrayList {

public static void main( String args[] )
  {
   boolean continua=true;
   Scanner entrada = new Scanner(System.in);
   String nomeLista;
   ArrayList list = new ArrayList();

// Entrada de Dados
   while (continua){
       System.out.println("Entre com o nome do aluno: ");
        nomeLista = entrada.nextLine();
        if (! nomeLista.equals(""))
            list.add(nomeLista);
        else continua = false;
   }
```

```
// Listagem
   for ( int i=0; i < list.size(); i++ )
       System.out.print( list.get(i) + " " );

   // Selecao para remocao
   System.out.println("\nEntre com a letra para remocao: ");
   String letra = entrada.nextLine();

   Iterator iterator = list.iterator();
   while ( iterator.hasNext() ) {
       nomeLista = (String)iterator.next();
       if ( nomeLista.charAt(0) == letra.charAt(0))
           iterator.remove();
   }
  // Nova Listagem
   for ( int i=0; i < list.size(); i++ )
        System.out.print( list.get(i) + " " );
  }
}
```

Controle de Fluxo

- Instruções Condicionais
  - if-else
  - for
  - switch
  - while
  - do while
  - else Oscilante

  - break e continue

Controle de Fluxo

if else

```
if (expressão booleana)
{ instrução ou bloco de comandos }
else
{ instrução ou bloco de comandos }
```

```
if (cont >= 0)
{
    System.out.PrintIn("Erro !!!");
}
else
{
    System.out.println("Ok !");
}
```

Controle de Fluxo

if else

```
// Programa do maior numero

// Utilize a entrada de dados via console

import java.util.Scanner;

public class Maior_Numero {

   public static void main ( String args[] ) {

       int numero1, numero2, maior;
       Scanner entrada;

        entrada = new Scanner(System.in);

        System.out.println(« Digite o primeiro numero: « );
        numero1 = entrada.nextInt();
        System.out.println(«  Digite o segundo numero: « );
        numero2 = entrada.nextInt();
```

Controle de Fluxo

if else

```
// continuação

  if (numero1 > numero2)
     maior=numero1;
  else
     maior=numero2;

    System.out.println(«O maior numero e «    + maior);

  }   // fim método main

}     // fim classe
```

Controle de Fluxo
If´s aninhados

if else

```
// ALTERAR:

if (numero1 == numero2)
   System.out.println(«Os numeros sao iguais »);
else
   {   if (numero1 > numero2)
                maior=numero1;
          else
      maior=numero2;
          System.out.println(« O maior numero e «  + maior);

    }

  }  // fim método main

}   // fim classe
```

Controle de Fluxo

for

```
for (expr_inicial; expr_booleana; expr_increm)
        { bloco de comandos }

         for (int x = 0, int y=0; x <10; x++, y--)
         {
            System.out.println("Valor do X : " + x);
         }
```

Controle de Fluxo

for

```
// Programa Frase invertida

import java.util.Scanner;

public class Palavra {
   public static void main ( String args[] ) {
      int contador=0;
      String arrayString[] = new String[5];
       Scanner entrada = new Scanner(System.in);

      for (contador=0; contador<5; contador++)
       {        System.out.println("Digite uma palavra: ");
                  arrayString[contador] = entrada.nextLine();
         }
       for (contador=4; contador>=0; contador--)
           System.out.println(« Valor: «  + arrayString[contador]);

   }  // fim método main
}             // fim classe
```

Controle de Fluxo

switch

```
switch (expressão short,int,byte ou char)
{
    case expressão2:
        comandos;
        break;
    case expressão3:
        comandos;
        break
    default:
        comandos;
        break;
}
```

Controle de Fluxo

switch

```
int cor=0;
switch (cor)
        {
            case 0:
                setBackground(Color.black);
                break;
            case 2:
                setBackground(Color.red);
                break;
            default:
                setBackground(Color.white);
                break;
        }
```

Controle de Fluxo

switch

```
//  Contagem de votos de uma população de 100 pessoas

import java.util.Scanner;

public class Votacao {
  public static void main ( String args[] ) {
      int voto;
      int lula, alckmin, buarque, luizhelena;
      Scanner entrada = new Scanner(System.in);

      for (int rodada = 0; rodada < 100 ; rodada ++ ) {
            System.out.println("Digite seu voto: ");
            voto = entrada.nextInt();
```

Controle de Fluxo

switch

```
//  continuação
switch (voto ) {
   case 1:
        ++lula;
        break;
   case 2:
        ++alckmin;
        break;
   case 3:
        ++buarque;
        break;
   case 4:
        ++luizahelena;
        break;

}              // fim switch
  }            // fim for
```

Controle de Fluxo

switch

```
            System.out.println(« Votos para Lula: «  + lula);
            System.out.println(« Votos para Alckmin: «  + alckmin);
            System.out.println(« Votos para Cristovam Buarque: «  + buarque);
            System.out.println(« Votos para Luiza Helena: «  + luizahelena);

        }   // fim método main
}           // fim classe
```

Controle de Fluxo

while

```
while (expr_booleana)
        { bloco de comandos }
```

```
int cont = 0;
while (cont < 100)
{
  System.out.println("contando " + cont);
  cont++;
}
```

Controle de Fluxo

while

```
// Programa de média da turma com repetição controlada
import java.util.Scanner;

public class Media {
   public static void main ( String args[] ) {
      int total=0;
      double nota, media;
      Scanner entrada = new Scanner(System.in);

        while ( total < 10 )
      {     System.out.println(« Digite a nota: « );
            nota = entrada.nextDouble();
            media = media + nota;
               total++;
      }  // fim do while

      media = media/10;
      System.out.println(« A média da turma de 10 alunos e «  + media);

   }            // fim metodo main
}               // fim classe
```

Controle de Fluxo

do while

```
do
   { bloco de comandos }
while  { condição }
```

```
int x = 0;
do
{
   x++;
} while (x <10);
```

Controle de Fluxo

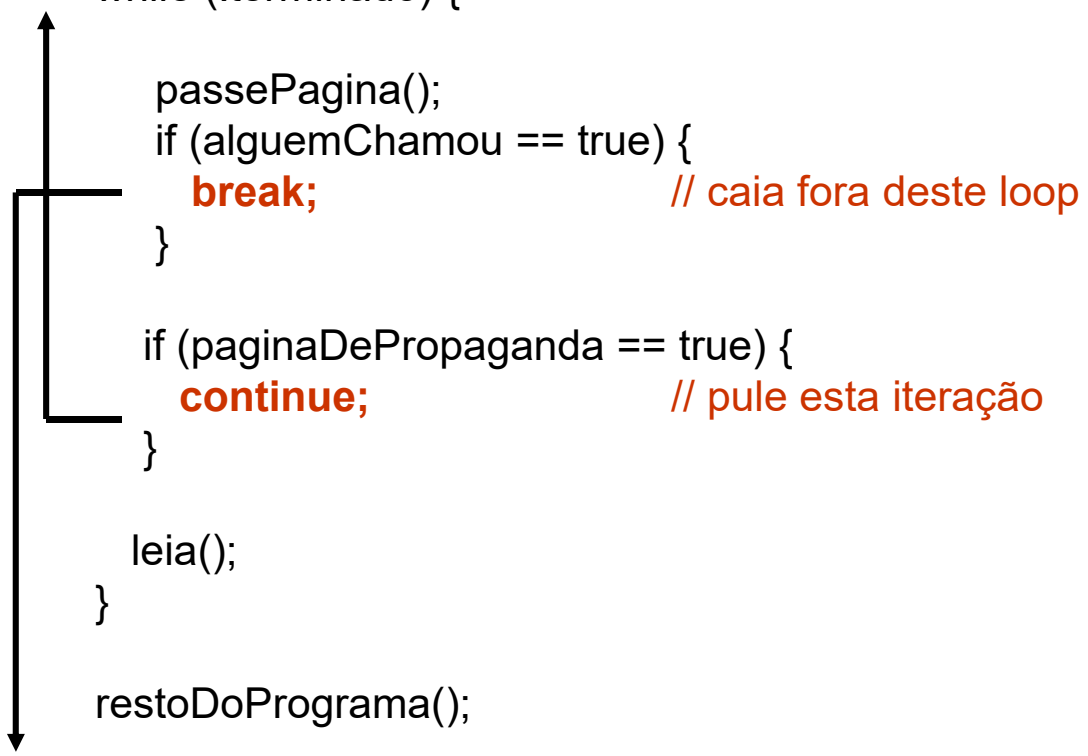
break e continue

break

```
int i = 0;
while( true ) {
      if (++i==10) break;
      System.out.println(i);
}
```

continue

```
int i = 0;
while ( true ) {
   if (++i%2 == 1)
       continue;
   System.out.println(i);
}
```



Controle de Fluxo

break e continue

break

```
int i = 0;
while( true ) {
      if (++i==10) break;
      System.out.println(i);
}
```

Que números serão impressos?

E se trocarmos por i++ ??

1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
	10

continue

```
int i = 0;
while ( true ) {
   if (++i%2 == 1)
      continue;
   System.out.println(i);
}
```

Que números serão impressos?

E se trocarmos por i++ ??

2	1
4	3
6	5
8	7
10	9
12	11
14	13
16	15
18	17
...	...