EN05170 - Programação de Computadores II

Introdução a Programação Orientada a Objetos – Sintaxe Java

Prof. Dr. Lidio Mauro Lima de Campos

limadecampos@gmail.com

Universidade Federal do Pará – UFPA
ICEN
PPGCC

Agenda

- Introdução
- Lendo Entrada com Keyboard
- Casting
- Tipos de Estruturas de Controle
- Exercicios

Lendo Entrada com keyboard

```
package Introdução;
public class examplkeyboard
{public static void main(String[] args)
 {
   float p1,p2;
   System.out.println("Entre com a p1 e a p2:\n");
   p1=Keyboard.readFloat();
   p2=Keyboard.readFloat();

   if(p1>p2)
   {
     System.out.println("A nota maior e p1="+p1); }
   else
   {
     System.out.println("A nota maior e p2="+p2);
   }
   System.out.println("A media das notas e:\n"+(p1+p2)/2);
  }
}
```

Relembrando Tipos primitivos

Categoria	Tipo	Bytes/bits	Faixa de valores
Inteiro	byte (inteiro de oito bits)	1/8	-128 a 127
	short (Inteiro Curto)	2/16	-32768 a 32767
	int (inteiro)	4/32	-2.147.483.648 a 2.147.483.647
	long (inteiro longo)	8/84	-9.223.372.036.854.775.808L a 9.223.372.036.854.775.807L
Real	float (Precisão simples)	4/32	Valores Positivos +1.40129846432481707e-45 a +3.40282346638528860e+38 Valores Negativos -3.40282346638528860e+38 a -1.40129846432481707e-45
	double (Precisão Dupla)	8/64	Valores Positivos +4.94065645841246544e-324 a +1.79769313486231570e+308 Valores negativos -1.79769313486231570e+308 a -4.94065645841246544e-324
Caracter	char	2/16	\u0000 a \UFFFF
Lógico	boolean	1/8	false e true

Casting

- Se existe perda de precisão em uma atribuição é necessário fazer uma conversão explicitamente
- long longValue = 99L;
- int intValue2 = (int) longValue; // Ok!
- int intValue1 = longValue; // ERRO!

- Devido a rigorosa verificação de tipos em Java, nem todos os tipos são compatíveis, e assim nem todas as conversões de tipo são permitidas implicitamente.
  - Por exemplo **boolean** e **int** não são compatíveis
  - int i;
  - float f;
  - i=10;
  - f=i; //atribui um int a um float

Casting

- A conversão de tipos ocorrerá se:
  - Os dois tipos forem compatíveis
  - O tipo de destino for maior que o de origem
    - Ex:

      - byte -> int (possível),
      - long ->double (possível)
      - double->long (não possível)

Conversão de Tipos Incompatíveis

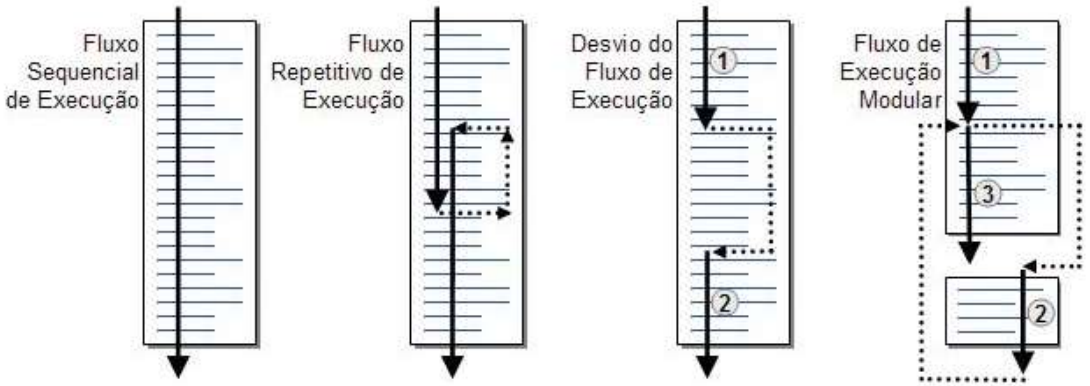
```java
class CastDemo {
  public static void main(String args[])
  {
    double x, y; byte b;
    int i; char ch;
    x = 10.0; y = 3.0;
    i = (int)(x/y); // cast double to int
    System.out.println("Integer outcome of x / y: " + i); // saída : 3
    i = 100;
    b = (byte) i;
    System.out.println("Value of b: " + b); // saída :100
    i = 257;
    b = (byte) i;
    System.out.println("Value of b: " + b); // saída : 1
    b = 88; // ASCII code for X
    ch = (char) b;
    System.out.println("ch: " + ch);  // saída : X
  }
}
```

Tipos de Estruturas de Controle

- **Repetição Simples** – Efetuam repetição de diretivas ou blocos, criando os chamados laços. O número de repetições pode ser predefinido ou determinado pelo programa durante a execução .
  - Corresponde a diretiva **for** no **Java**.
- **Repetição condicionais** – São como estruturas de repetição simples, mas cuja repetição esta associada a avaliação de uma condição. Geralmente são usadas quando não se conhece de antemão o número de repetições.
  - No Java são representadas pelas diretivas **while** e **do while**.
- **Desvio de Fluxo** – Destinadas ao desvio da execução do programa para outra parte, quebrando o fluxo sequencial. O desvio condicional é aquele associado a avaliação de uma expressão, como uma tomada de decisão , e o desvio incondicional ocorre automaticamente.
  - No Java tem-se o **if else e switch** para o **desvio condicional simples** e **break e continue** para o desvio incondicional.

Tipos de Estruturas de Controle

- **Execução Modular** – A divisão de um programa em partes menores facilita o entendimento, a correção ou modificação. No Java isso é feito por meio de construção de classes e métodos, além da divisão do código em pacotes e módulos, a chamada de métodos constitui um desvio incondicional.



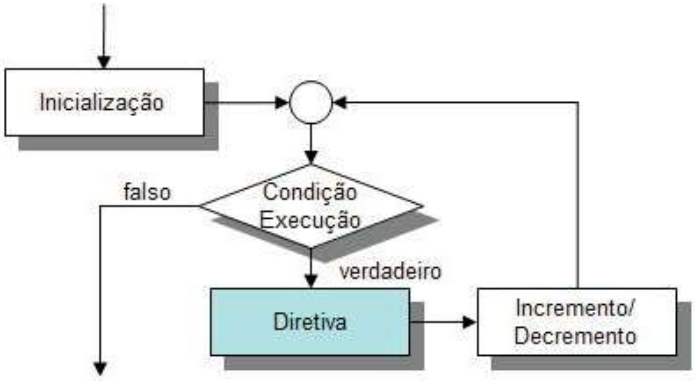
Variações no fluxo de execução de um programa.

- https://www.faladev.com/livros-para-programadores/java-guia-programador-atualizado-java/

Tipos de Estruturas de Controle

- Estruturas de repetição simples, **Diretiva for**

```
package Introdução;
public class For3
{ public static void main (String args[]) {
  int j;
  // efetua contagem de 0 a 10
  for (j=0; j<=10; j++)
  { System.out.println(j);}
  for (j=10; j>=0; j--)
  { System.out.println(j);}
  for (j=0; j<=10; j+=2)
  { System.out.println(j);}
 }
}
```



Comportamento da diretiva for.

Tipos de Estruturas de Controle

- Estruturas de repetição simples, **Diretiva for**

```
package Introdução;
public class For2 {
    public static void main(String[] args)
   {
        for(int cima=1, baixo=10 ; cima<=10 && baixo>=1; cima++, baixo--)
       {
            System.out.printf("%d \t %d \n", cima, baixo);
        }
    }
}
```

Iteração – Laço For

```
class ErroRaiz {
 public static void main(String args[])
 {
  double num, sroot, rerr;
  for(num = 1.0; num < 100.0; num++)
  {
   sroot = Math.sqrt(num);
   System.out.println("Raiz quadrada de: " + num +" é "+ sroot);

   // calcula o erro de arredondamento  rerr = num - (sroot *  sroot);
   System.out.println("O Erro de arredondamento é: " + rerr);  System.out.println();
  }
 }
}
```

Iteração – Laço For

```
class Comma {
  public static void main(String args[])
 {
    int i, j;
     for(i=0, j=10; i <j; i++, j--)  System.out.println("i e j: " +i +" "+j);
  }
}
```

Saída:

```
i e j: 1 10
i e j: 1 9
i e j: 2 8
i e j: 3 7
i e j: 4 6
```

Iteração – Laço For

```
class Empty2 {
 public static void main(String args[])
 {
  int i;
  i = 0; // move inicialização para fora do laço  for(; i < 10; )
  {
    System.out.println("Passo #" + i);
    i++; // incrementa a variável de controle do laço
  }
 }
}
```

Iteração – Laço For

O laço infinito

```
for( ; ; )
{
 //....
}
```

Exercicio

1)Escrever um programa que imprime a sequência e a soma dos termos da mesma

S=1/N+ 2/(N-1)+ 3/(N-2)+...+(N-1)/2+ N/1, para N=100.

```
class Sequencia {
        public static void main(String[] args)
    {
                int i;
                float s= 0f;
                for (i = 1; i <= 100; i++)
                        s += i/(100 - (i-1));
                System.out.println(s);
        }
}
```

Exercicio

2)Fazer um programa para imprimir todos os números pares no intervalo [1,100]

```
class Pares {
        public static void
        main(String[] args)
        {  int i;
         for (i=1; i<=100;i++)
        {
                if (i%2 == 0)
                        System.out.println(i);
        }
}}
```

Exercicio

Exercício 3)Escrever um programa que imprime os termos da sequência e a soma sos termos da mesma

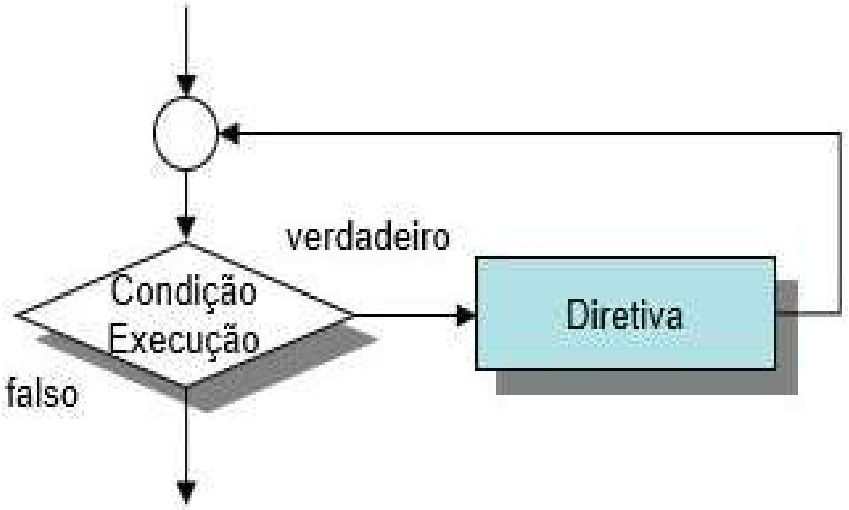
- S=1/+1/2+1/3+1/4+...+1/N, para N=100

```
class Sequencia2{
	public static void main(String[] args){  int i;
	float s = 0f;
	for (i=1; i<=100;i++){
		s+=1/i;  System.out.println(s);
	}
}
}
```

Tipos de Estruturas de Controle

- Estruturas de repetição condicionais
- **Diretiva while –** Na estrutura while, um conjunto de instruções é repetido enquanto o resultado da condição (uma expressão lógica) é avaliado como verdadeiro.
- **Sintaxe: while(<condição>)**

  <diretiva; {/* bloco */} >



Comportamento da diretiva while.

Tipos de Estruturas de Controle

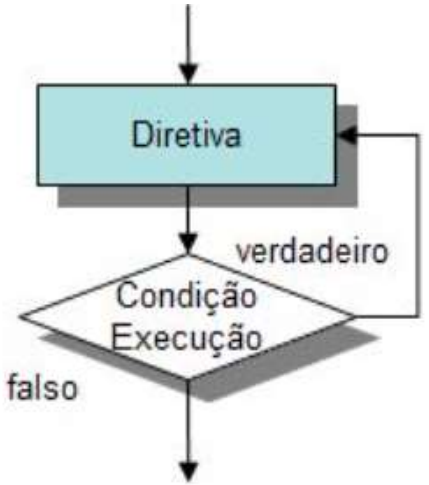
```
// Fragmento 1: contagem progressiva 0 a 10
int x = 0;
while (x<=10) {
  System.out.println(x);
  x = x + 1;
}
// Fragmento 2: contagem regressiva 10 a 0
int k = 10;
while(k>=0) {
  System.out.println(k);
```

```
// Fragmento 3: contagem -5 a +5, de 2 em 2
int z = -5;
while(z<=5) {
  System.out.println(z);
  z = z + 2;
}
```

Tipos de Estruturas de Controle

- Estruturas de repetição condicionais
- **Diretiva do while – é outro laço condicional que repete uma diretiva ou um bloco enquanto a condição é avaliada como verdadeira, mas, diferentemente do while, a instrução associada é executada uma vez antes da avaliação da expressão lógica usada como condição e eventual continuação da repetição.**
- **Sintaxe:**

```
do
{
  diretiva;
}while(<condição>);
```



Comportamento da diretiva do while.

Tipos de Estruturas de Controle

```java
import java.util.Scanner;
public class DoWhile
{
   public static void main (String args[]) {
   Scanner s = new Scanner(System.in);
   System.out.print("Valor inteiro inferior? ");
   int min = s.nextInt();
   System.out.print("Valor inteiro superior? ");
   int max = s.nextInt();
   do {
        System.out.println(min + " < " + max);
        min++; max--;
      } while (min < max);
    System.out.println(min + " e " + max + " Condicao Final!");
  }
}
```

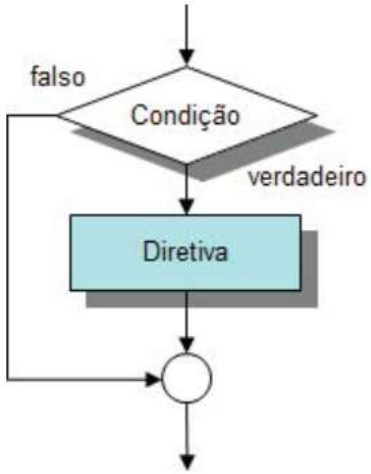
Tipos de Estruturas de Controle

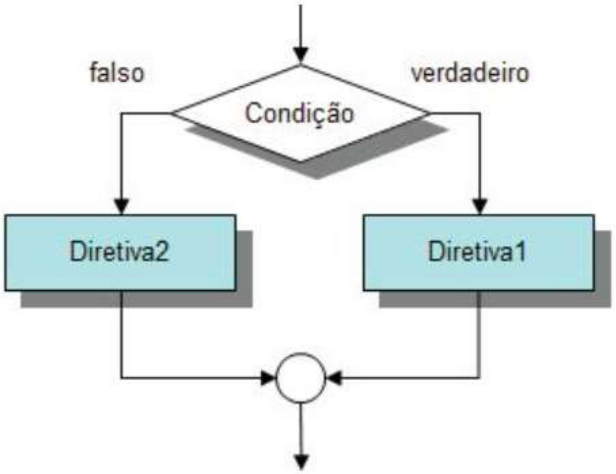
```
public class DowhileKey
{
   public static void main (String args[]) {

   System.out.print("Valor inteiro inferior? ");
   int min = Keyboard.readInt();
   System.out.print("Valor inteiro superior? ");
   int max = Keyboard.readInt();
   do {
        System.out.println(min + " < " + max);
        min++; max--;
      } while (min < max);
    System.out.println(min + " e " + max + " Condicao Final!");
  }
}
```

Tipos de Estruturas de Controle

- if else statement
- O if é uma diretiva de desvio simples do fluxo de execução capaz de selecionar um entre dois caminhos distintos para execução capaz de selecionar um entre dois caminhos distintos para execução, dependendo do resultado falso ou verdadeiro obtido da expressão lógica associada.
- Sua sintaxe é:

```
if ( <condição> )
  < diretiva1; | { /* bloco 1 */ } >
[else
  < diretiva2; | { /* bloco 2 */ } >]
```

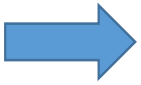




Tipos de Estruturas de Controle

- if else statement

```
int a = 80, b = 50;
if (a > b) // condição1
  if (a > 80) // condição2
    System.out.println("a > b && a > 80"); // diretiva1
else
  System.out.println("a < b"); // diretiva2
```



```
if (condição1)
  if (condição1)
    diretiva1;
  else
    diretiva2;
```

A sintaxe Java determina que uma cláusula else sempre pertence à diretiva if imediatamente anterior, mas o uso de chaves permite modificar a regra como segue:

```
int a = 80, b = 50;
if (a > b) { // condição1
  if (a > 80) // condição2
    System.out.println("a > b && a > 80"); // diretiva1
} else {
  System.out.println("a < b"); // diretiva2
}
```

Exemplo if

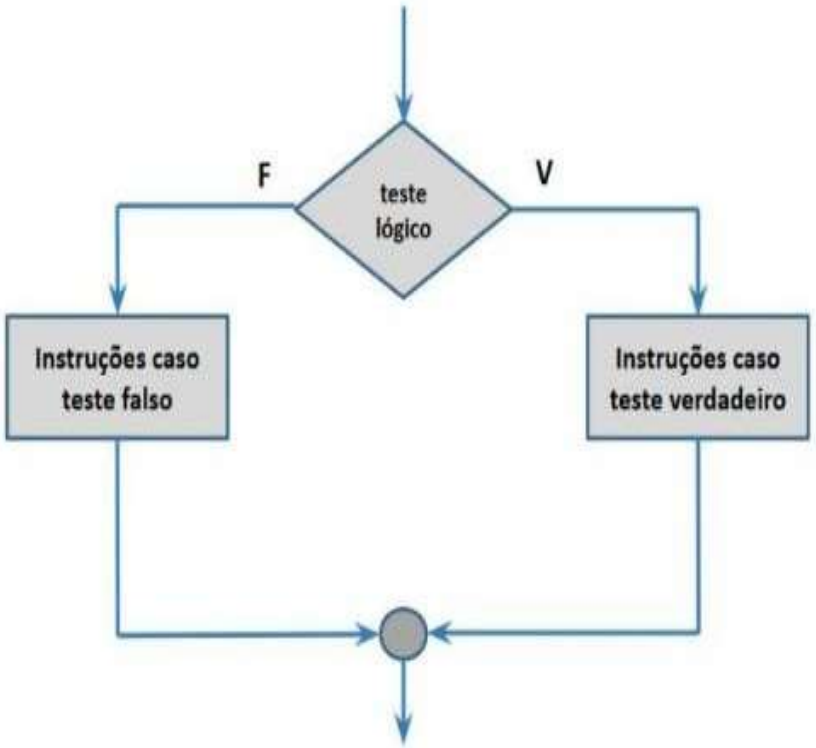
```
public class ExemploIf
{public static void main(String[] args)
 {int resposta = 10;
  if (resposta == 10)
  {
     // Se a variável for igual a 10, a frase abaixo será escrita
     System.out.println("Você acertou!");
  }
      // Se a variável não for igual a 10, nenhuma frase será exibida
 }
}
```

Exemplo if

```
public class Exemploif1
{
 public static void main(String[] args) {
 int resposta = 10;
 if (resposta == 10)
 {
     //Se a variável for igual a 10, a frase abaixo será escrita
      System.out.println("Você acertou!");
 }
 else
 {
   //Caso contrário, a frase abaixo será escrita
    System.out.println("Você errou!");
 }
}
```

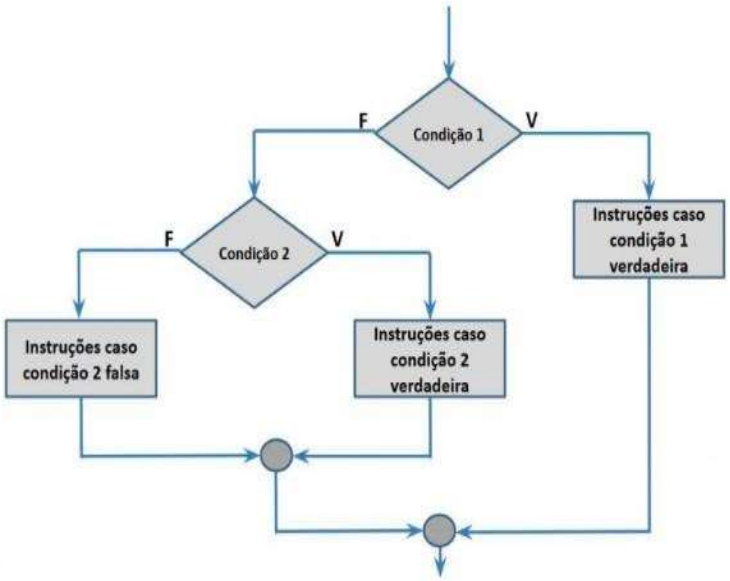
Exemplo if

```
package Introdução;
public class Exemploif2
{public static void main(String args[])
 {
  double livroLinux;
  double livroBancosDados;
  livroLinux = 78.60;
  livroBancosDados = 56.75;
  double total = livroLinux + livroBancosDados;
  System.out.println("O preço total é " + total );

  if (total < 120.00 )
     System.out.println("O preço está bom!");
  else
     System.out.println("Livros muito caros!");
 }
}
```



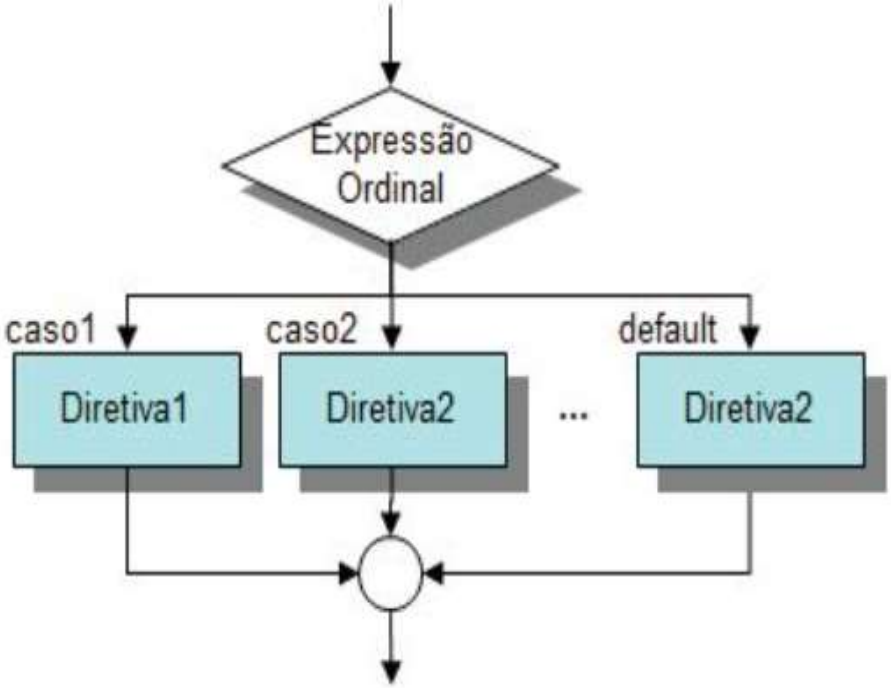
IfAninhados

```
package Introdução;
class IfElseElse
{  public static void main(String args[]) {
   int x;

   for(x=0; x<6; x++)
   {
      if(x==1)
        System.out.println("x e um");
      else if(x==2)
        System.out.println("x e dois");
      else if(x==3)
        System.out.println("x e tres");
      else if(x==4)
        System.out.println("x e quatro");
      else
        System.out.println("x não esta entre 1 e 4");
    }
  }
}
```



Diretiva switch

A diretiva switch é um desvio múltiplo de fluxo, i.e., avalia uma expressão ordinal para escolher um caminho de execução entre vários possíveis . Uma expressão é ordinal quando seu resultado pertence a um conjunto em que são conhecidos os elementos anterior e posterior de qualquer um de seus valores.



Instrução switch

```
switch (expressão)
{
 case constante 1:
    sequencia de instruções;
    break;
 case constante 2:
   sequencia de instruções;
   break;
 case constante 3:
    sequencia de instruções;
    break;
 default:
   sequencia de instruções;
}
```

Instrução switch

```
class SwitchDemo {
 public static void main(String args[])
 { int i;
  for(i=0; i<10; i++)  switch(i) {
    case 0:
     System.out.println("i é zero");  break;
    case 1:
     System.out.println("i é um");  break;
    case 2:
     System.out.println("i é dois");  break;
    case 3:
     System.out.println("i é tres");  break;
    case 4:
     System.out.println("i é quatro");  break;
    default:
     System.out.println("i é cinco ou mais");
   }
 }
}
```

Instrução switch

```
// Variáveis comuns
int mes = 1;
int ndias;
// Fragmento 1: switch como diretiva
switch(mes) { // cases "amontoados"
  case 1: case 3: case 5: case 7:
  case 8: case 10: case 12:
    ndias = 31;
    break;
  case 4: case 6: case 9: case 11:
    ndias = 30;
    break;
default:
    ndias = 28;
}
```

```
// Fragmento 2: switch melhorado
switch(mes) {
  case 1, 3, 5, 7, 8, 10, 12:
    ndias = 31;
  break;
  case 4, 6, 9, 11:
    ndias = 30;
    break;
default:
    ndias = 28;
}
```

Controle de Iteração

- break

- Exemplo do

```
{
Bloco comandos();
 break;
}while(true)
```

```
package Introdução;
public class ExemploBreak {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args)
    {
        // TODO code application logic here
      for(int contador=1;contador<100;contador++)
      {
       System.out.println("Esta e a repetição nr:"+contador);
       if(contador==10)
           break;
      }
    }
}
```

Controle de Iteração

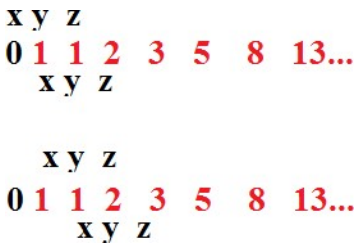
- Continue

```
do
{
 trabalhe();
 continue;
} while(true)
```

```
package Introdução;

public class ControleIteracao {
    public static void main(String[] args)
    {
      // TODO code application logic here
      for(int contador=1;contador<100;contador++)
      {
        if(contador%5!=0)
           continue;

        System.out.println("Contador:"+contador);
      }
    }

}
```

Exercício de Fixação

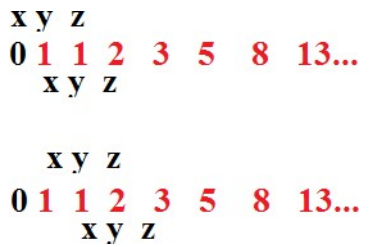
- EX4. A série de Fibonacci é formada pela Sequência 1, 1, 2, 3, 5, 8, 13, ... Escrever um programa que gere a série de Fibonacci até o vigésimo termo.



- EX5. Faça um programa para calcular o fatorial de um número lido do teclado (para ler um número do teclado use a classe Keyboard fornecida no seu material).

Exercício de Fixação

```
public class Fibonacci
{ public static void main(String[] args)
 {  int x= 0; int z ; int y = 1;
    int i; int N=15;
  System.out.println("Para N valendo 15, é temos:");
  System.out.println(x);
  System.out.println(y);
  for (i=1;i<=N;i++)
  { z = x+y;
    x = y;  y = z;
    System.out.println(z);
  }
} }
```



Exercício de Fixação

```
public class Fatorial
{ public static void main(String args[])
 {
  int fat=1;  int num;
  System.out.println("Entre com o valor do número:\n");
  num=Keyboard.readInt();
  for(int i = 1;i <= num; i++)
  {
   fat = fat *  i;
  }
  System.out.println("Fatorial="+ fat);
}
}
```

Exercício de Fixação

- Ex6)Calcular e imprimir a soma dos n primeiros números naturais pares e impares. O usuário deve fornecer quantos números devem ser somados.