Estruturas de Decisão e Repetição

Orientação a Objetos – DCC025

Gleiph Ghiotto Lima de Menezes

gleiph.ghiotto@ufjf.br

Aula de hoje...

- Operadores
  - Aritméticos (usados em contas)
  - Relacionais (usados em comparações numéricas)
  - Lógicos (usados em comparações lógicas)
  - De atribuição (armazenamento de valores em variáveis)
- Estruturas de decisão
  - *If...then*
  - *If...then...else*
  - *Switch...case*

Aula de hoje...

- Estruturas de repetição
  - *while*...*do*
  - *do*...while
  - *for*

- String
  - Manipulação de textos

Operadores aritméticos

Operador	Exemplo	Prioridade
(expr)	(1 + 2) * 3 → 9	1
var++	i++	2
var--	j--	2
++var	++i	3
--var	--j	3
+expr	+15	3
-expr	-(5+3) → -8	3
*	5 * 3→ 15	4
/	5 / 3 → 1	4
%	5 % 3 → 2	4
+	5 + 3 → 8	5
-	5 - 3 → 2	5

Operadores aritméticos

- Operadores com a mesma prioridade (precedência)
  - Analisados da esquerda para a direita
- Aritmética de inteiros
  - Numerador e denominador inteiros
  - Resultado é somente a parte inteira da divisão
- Aritmética em modo misto
  - Numerador ou denominador real
  - Resultado fracionário

Funções matemáticas

- A classe Math
  - Contém constantes (PI e número de Euler)
  - Contém diversas funções matemáticas
  - Não é necessário importar o seu pacote, java.lang, pois está sempre disponível
- Constantes
  - Math.PI = 3.141592653589793
  - Math.E = 2.718281828459045

Funções matemáticas

Método	Descrição	Exemplo
Math.abs(expr)	Valor absoluto	Math.abs(-5.3) → 5.3
Math.round(expr)	Arredonda um número	Math.round(5.3) → 5
Math.ceil(expr)	Arredonda para cima	Math.ceil(5.3) → 6.0
Math.floor(expr)	Arredonda para baixo	Math.floor(5.3) → 5.0
Math.max(expr1, expr2)	Maior de dois números	Math.max(5, 6) → 6
Math.min(expr1, expr2)	Menor de dois números	Math.min(5, 6) → 5
Math.sqrt(expr)	Raiz quadrada	Math.sqrt(4) → 2.0
Math.pow(expr1, expr2)	Potência	Math.pow(2, 3) → 8.0
Math.log10(expr)	Logaritmo na base 10	Math.log10(100) → 2.0
Math.log(expr)	Logaritmo natural (base E)	Math.log(Math.E) → 1.0
Math.exp(expr)	Exponencial (e^{expr})	Math.exp(0) → 1.0

Funções matemáticas

Função	Descrição	Exemplo
Math.sin(expr)	Seno	Math.sin(0) → 0.0
Math.asin(expr)	Arco seno	Math.asin(1) → 1.5707963267948966
Math.cos(expr)	Cosseno	Math.cos(0) → 1.0
Math.acos(expr)	Arco cosseno	Math.acos(-1) → 3.141592653589793
Math.tan(expr)	Tangente	Math.tan(1) → 1.5574077246549023
Math.atan(expr)	Arco tangente	Math.atan(1) → 0.7853981633974483
Math.toDegrees(expr)	Converte radianos para graus	Math.toDegrees(Math.PI) → 180.0
Math.toRadians(expr)	Converte graus para radianos	Math.toRadians(180) → 3.141592653589793

- Funções trigonométricas trabalham com radiano
- Existem algumas outras funções menos usadas

Números aleatórios

- Algumas aplicações necessitam que o computador sorteie um número
  - Método Random.nextDouble()
  - Gera número pseudo aleatório entre 0 e 1
- A partir desse número, é possível gerar números em outros intervalos
  - inicio + (fim - inicio) * Random.nextDouble()

API: https://docs.oracle.com/javase/8/docs/api/java/util/Random.html

Números aleatórios

```java
import java.util.Random;


public class NumerosAleatorios {

    public static void main(String[] args) {
        Random geradorAleatorio = new Random();
        //Número aleatório entre 0 e 1
        double valorAleatorio = geradorAleatorio.nextDouble();
        System.out.println("random = " + valorAleatorio);
        //Número aleatório entre 10 e 20
        double random = 10 + geradorAleatorio.nextDouble()*10;
        System.out.println("random = " + random);

    }
}
```

Operadores relacionais

Operador	Exemplo	Prioridade
expr1 < expr2	5 < 3 → false	1
expr1 <= expr2	5 <= 3 → false	1
expr1 > expr2	5 > 3 → true	1
expr1 >= expr2	5 >= 3 → true	1
expr1 == expr2	5 == 3 → false	2
expr1 != expr2	5 != 3 → true	2

- Prioridade sempre inferior aos operadores aritméticos
- Sempre têm **números como operandos**
- Sempre têm **resultado booleano**

Operadores lógicos

Operador	Exemplo	Prioridade
! expr	!true → false	1
expr1 & expr2	true & false → false	2
expr1 ^ expr2	true ^ true → false	3
expr1 | expr2	true | true → true	4
expr1 && expr2	true && false → false	5
expr1 || expr2	True || false → true	6

- Prioridade sempre inferior aos operadores relacionais
- Exceção para "!", com prioridade superior a *, / e %
- Sempre têm **booleanos como operandos**
- Sempre têm **resultado booleano**

Tabela verdade

a	b	!a	a & b a && b	a ^ b	a | b a || b
true	**true**	false	true	false	true
true	**false**	false	false	true	true
false	**true**	true	false	true	true
false	**false**	true	false	false	false

Ou e E otimizados

- & e &&, assim como | e || têm a mesma tabela verdade, mas
  - & e | sempre avaliam os dois operandos
  - && e || só avaliam o segundo operando se o primeiro não for conclusivo
- Diferença quando o segundo operando altera valores

```
i = 10
Caso 1: (i > 3) | (++i < 2) → true (com i valendo 11)
Caso 2: (i > 3) || (++i < 2) → true (com i valendo 10)
```

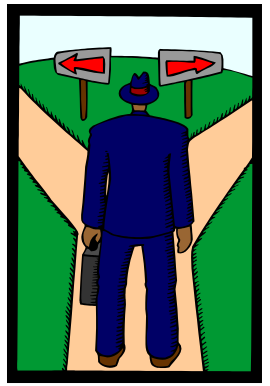
Operadores de atribuição

Operador	Exemplo
var = expr	x = 10 + 5
var += expr	x += 5 → x = x + 5
var -= expr	x -= 5 → x = x - 5
var *= expr	x *= 5 → x = x * 5
var /= expr	x /= 5 → x = x / 5
var %= expr	x %= 5 → x = x % 5
var &= expr	x &= true → x = x & true
var ^= expr	x ^= true → x = x ^ true
var |= expr	x |= true → x = x | true

Decisão

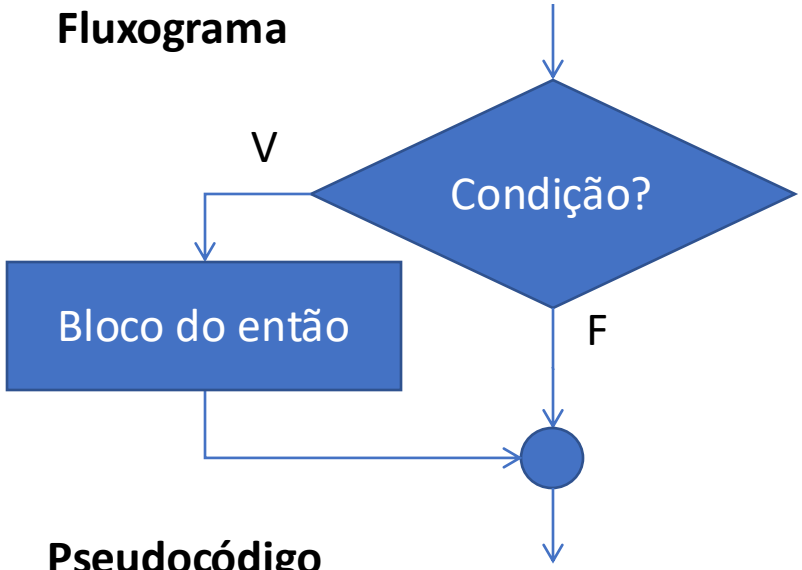
Mecanismos de decisão:

- *If...then*
  - Executa algo somente quando uma condição é verdadeira
- *If...then...else*
  - Bifurca a execução do código em função de uma condição
- *Switch...case*
  - Executa múltiplos trechos de código em função do valor de uma expressão



Decisão do tipo *if...then*

Fluxograma



Pseudocódigo

```
...
Se CONDIÇÃO então
  INSTRUÇÃO 1
  INSTRUÇÃO 2
  ...
  INSTRUÇÃO N
...
```

Java

```
...
if (CONDIÇÃO)
  INSTRUÇÃO;
...
```

Ou

```
...
if (CONDIÇÃO) {
  INSTRUÇÃO 1;
  INSTRUÇÃO 2;
  ...
  INSTRUÇÃO N;
}
...
```

Decisão do tipo *if...then*

- Executa o bloco de instruções somente se a condição for verdadeira
- A condição é uma expressão booleana que pode fazer uso de quaisquer operadores
- A condição deve sempre estar entre parênteses
- Pode omitir { e } caso execute somente uma instrução
  - As variáveis declaradas dentro de um bloco (entre { e }) só valem nesse bloco ou subblocos

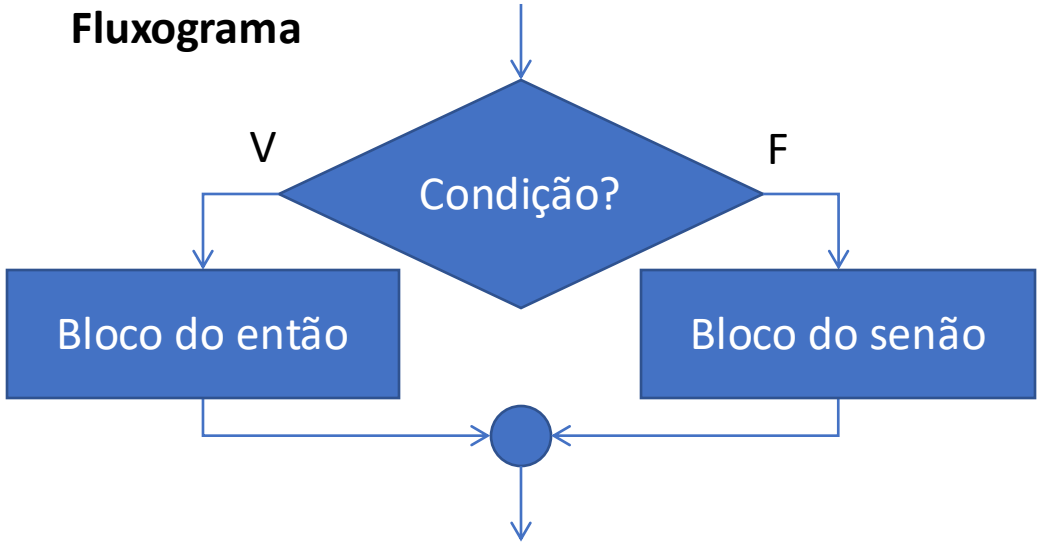
Exemplo de *if...then*

```
import java.util.Scanner;

public class Absoluto {
  public static void main(String[] args) {
    Scanner teclado = new Scanner(System.in);
    System.out.print("Entre com um número: ");
    double numero = teclado.nextDouble();
    if (numero < 0)
      numero = -numero;
    System.out.println("Valor absoluto: " +
numero);
  }
}
```

Programa para informar o valor absoluto de um número

Decisão do tipo *if...then...else*

Fluxograma



Pseudocódigo

```
...
Se CONDIÇÃO então
   INSTRUÇÃO 1
   INSTRUÇÃO 2
   ...
   INSTRUÇÃO N
Senão
   INSTRUÇÃO 1
   INSTRUÇÃO 2
   ...
   INSTRUÇÃO N
...
```

Decisão do tipo *if...then...else*

Java

```
...
if (CONDIÇÃO)
  INSTRUÇÃO;
else
  INSTRUÇÃO;
...
```

Ou

```
...
if (CONDIÇÃO) {
  INSTRUÇÃO 1;
  INSTRUÇÃO 2;
  ...
  INSTRUÇÃO N;
} else {
  INSTRUÇÃO 1;
  INSTRUÇÃO 2;
  ...
  INSTRUÇÃO N;
}
...
```

Decisão do tipo *if*...*then*...*else*

- Executa um ou o outro bloco de instruções em função da condição ser verdadeira ou falsa
- Valem as mesmas regras para *if*...*then*
- Qualquer combinação de instrução individual ou em bloco é aceita no *then* e no *else*
- Podem ser aninhados com outras estruturas do tipo *if*...*then*...*else*

Exemplo de *if...then...else*

```
import java.util.Scanner;
public class Paridade {
    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);
        System.out.print("Entre com um número: ");
        int numero = teclado.nextInt();
        if (numero % 2 == 0)
            System.out.println("O número é par!");
        else
            System.out.println("O número é impar!");
    }
}
```

Programa para informar se um número é par ou impar

Exemplo de *if* aninhado

```
import java.util.Scanner;
public class DiasMes {
  public static void main(String[] args) {
    Scanner teclado = new Scanner(System.in);
    System.out.print("Entre com um mês (1 a 12): ");
    byte mes = teclado.nextByte();
    if ((mes==1)||(mes==3)||(mes==5)||(mes==7)||(mes==8)||
  (mes==10)||(mes==12))
      System.out.println("Esse mês tem 31 dias!");
    else if ((mes==4)||(mes==6)||(mes==9)||(mes==11))
```

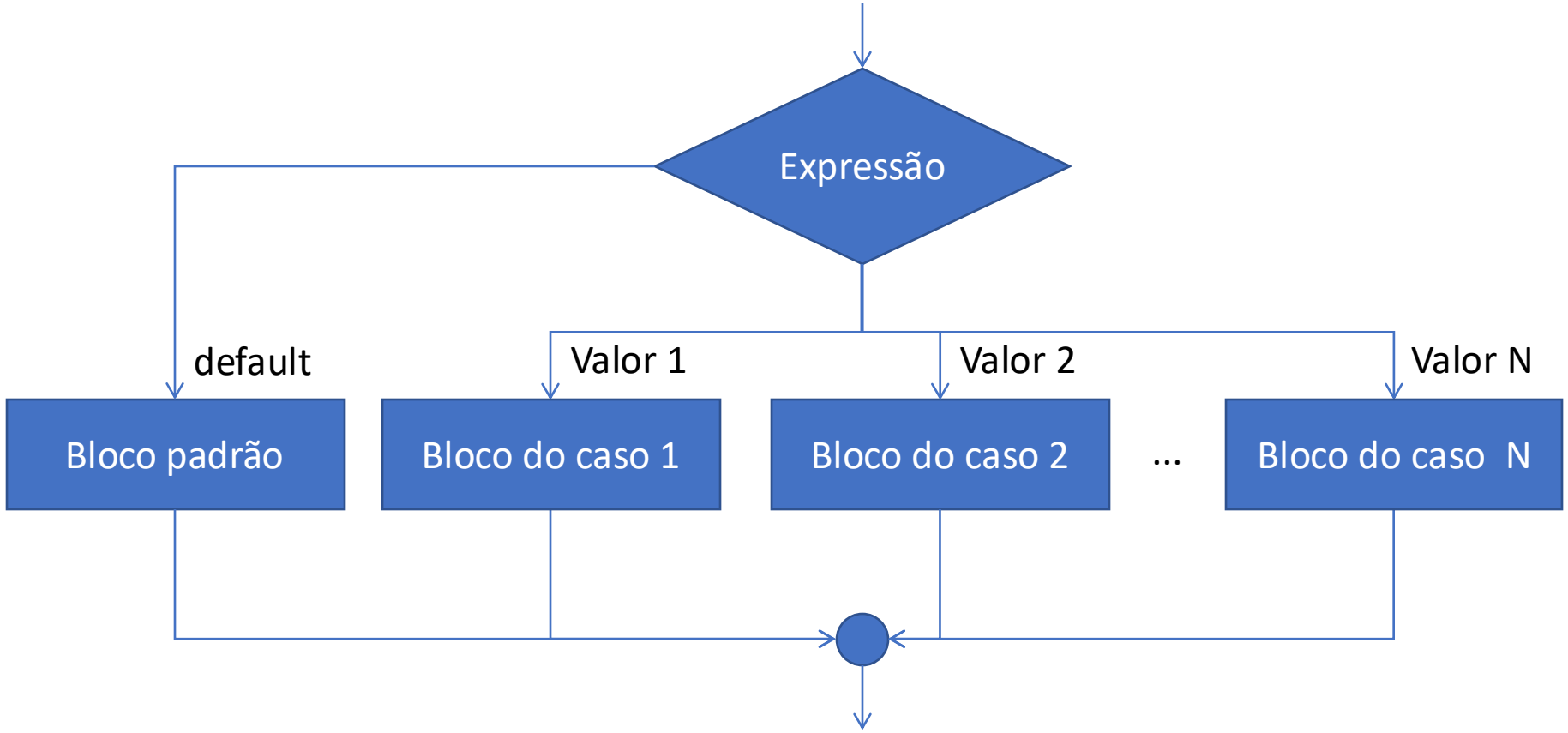
Programa para informar o número de dias de um mês

Exemplo de *if* aninhado

```
      System.out.println("Esse mês tem 30 dias!");
    else {
      System.out.print("Entre com o ano (4 dígitos): ");
      short ano = teclado.nextShort();
      if ((ano%400==0)||((ano%4==0)&&(ano%100!=0)))
        System.out.println("Esse mês tem 29 dias!");
      else
        System.out.println("Esse mês tem 28 dias!");
    }
  }
}
```

Programa para informar o número de dias de um mês

Decisão do tipo *switch...case*



Decisão do tipo *switch...case*

Java

```
...
switch (EXPRESSÃO) {
  case VALOR 1: INSTRUÇÃO 1;
                ...
                break;
  case VALOR 2: INSTRUÇÃO 1;
                ...
                break;
  ...
  case VALOR N: INSTRUÇÃO 1;
                ...
                break;
  default: INSTRUÇÃO 1;
           ...
}
...
```

Decisão do tipo *switch*...*case*

- Aceita expressões dos tipos byte, short, int, char e String
- É equivalente a *if* aninhado
  - Escolher o que tem melhor legibilidade
  - *Switch*...*case* é baseado em valores individuais
  - *If*...*then*...*else* pode ser baseado em intervalo de valores
- O uso de *break* é fundamental para a quebra do fluxo
  - A clausula *case* delimita somente o ponto de entrada
  - O programa executará todas as linhas seguintes até encontrar um *break* ou terminar o *switch*

Exemplo de *switch...case*

```
import java.util.Scanner;
public class DiasMes {
  public static void main(String[] args) {
    Scanner teclado = new Scanner(System.in);
    System.out.print("Entre com um mês (1 a 12): ");
    byte mes = teclado.nextByte();
    switch (mes) {
      case 1: case 3: case 5: case 7: case 8: case 10: case 12:
        System.out.println("Esse mês tem 31 dias!");
        break;
      case 4: case 6: case 9: case 11:
        System.out.println("Esse mês tem 30 dias!");
        break;
```

Exemplo de *switch...case*

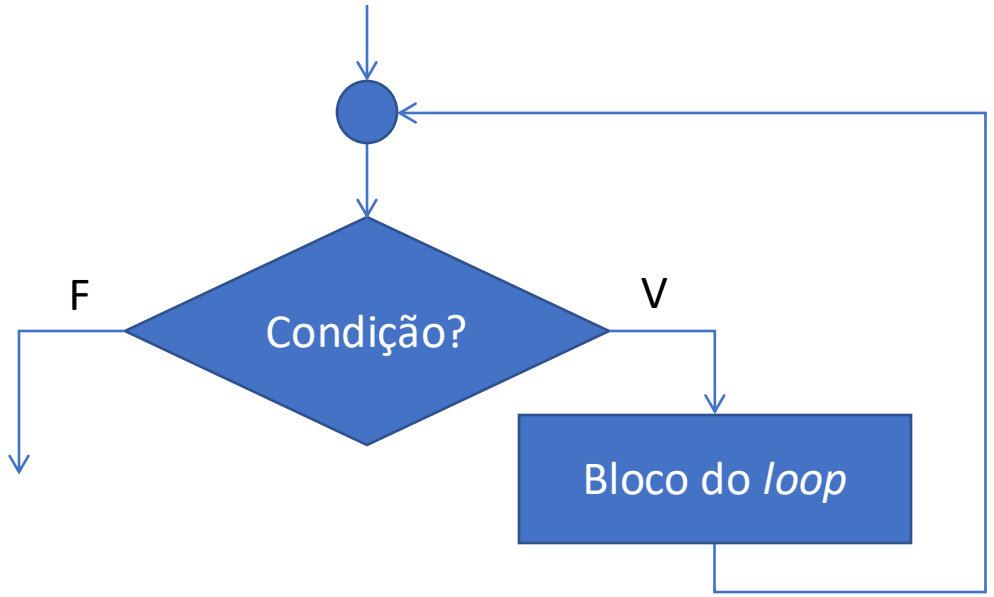
```
        case 2:
          System.out.print("Entre com o ano (4 dígitos): ");
          short ano = teclado.nextShort();
          if ((ano%400==0)||((ano%4==0)&&(ano%100!=0)))
            System.out.println("Esse mês tem 29 dias!");
          else
            System.out.println("Esse mês tem 28 dias!");
          break;
        default:
          System.out.println("Mês inválido!");
      }
    }
}
```

Estruturas de Repetição

- Permitem que um bloco de comandos seja executado diversas vezes
- **Repetição condicional**: executa um bloco de código enquanto uma condição lógica for verdadeira
  - *Do...while*
  - *While...do*
- **Repetição contável**: executa um bloco de código um número predeterminado de vezes
  - *For*

Repetição condicional do tipo *while...do*

Fluxograma



Pseudocódigo

```
...
Enquanto CONDIÇÃO faça
   INSTRUÇÃO 1
   INSTRUÇÃO 2
   ...
   INSTRUÇÃO N
...
```

Repetição condicional do tipo *while...do*

Java

```
...
while (CONDIÇÃO) {
   INSTRUÇÃO 1;
   INSTRUÇÃO 2;
   ...
   INSTRUÇÃO N;
}
...
```

Repetição condicional do tipo *while...do*

- Executa o bloco de instruções enquanto a condição for verdadeira
- A condição é uma expressão booleana que pode fazer uso de quaisquer operadores
- A condição deve sempre estar entre parênteses
- Pode omitir { e } caso execute somente uma instrução

Repetição condicional do tipo *while...do*

- Executa o bloco de instruções enquanto a condição for verdadeira
- **A condição é uma expressão booleana que pode fazer uso de quaisquer operadores**
- **A condição deve sempre estar entre parênteses**
- **Pode omitir { e } caso execute somente uma instrução**

Nenhuma novidade: igual ao if!!!

Exemplo de *while...do*

```
import java.util.Scanner;
public class Fatorial {
  public static void main(String[] args) {
    Scanner teclado = new Scanner(System.in);
    System.out.print("Entre com um número inteiro positivo: ");
    int numero = teclado.nextInt();
    long fatorial = 1;
    while (numero > 0)
      fatorial *= numero--;
    System.out.println("O fatorial desse número é " + fatorial);
  }
}
```

Programa para calcular fatorial de um número

Exemplo de *while...do*

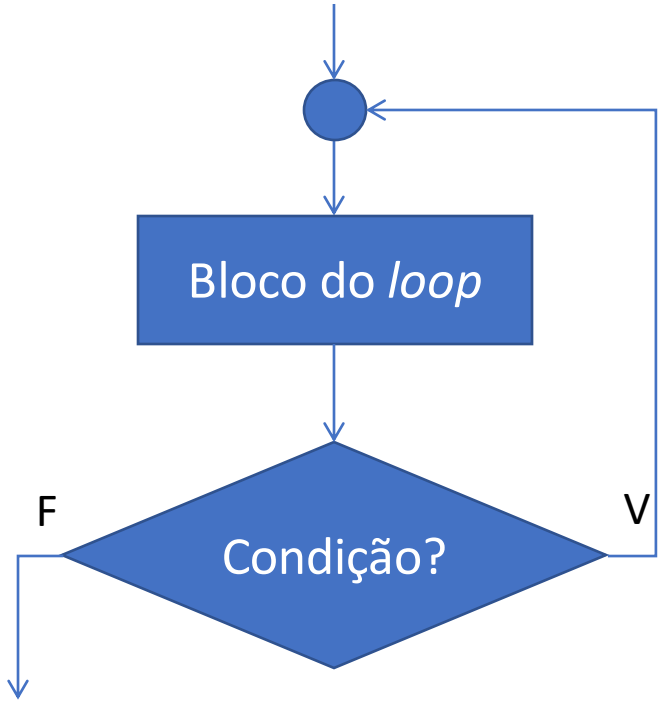
- Qual a saída do programa abaixo?

```
public class Loop {
  public static void main(String[] args) {
    int i = 0;
    while (true)
      System.out.println(i++);
  }
}
```

- Evitem forçar loops infinitos!

Repetição condicional do tipo *do...while*

Fluxograma



Pseudocódigo

```
...
Faça
   INSTRUÇÃO 1
   INSTRUÇÃO 2
   ...
   INSTRUÇÃO N
Enquanto CONDIÇÃO
...
```

Repetição condicional do tipo *do...while*

Java

```
...
do {
   INSTRUÇÃO 1;
   INSTRUÇÃO 2;
   ...
   INSTRUÇÃO N;
} while (CONDIÇÃO);
...
```

Repetição condicional do tipo *do...while*

- Executa o bloco de instruções enquanto a condição for verdadeira
- **Garante que ocorrerá ao menos uma execução**
  - A verificação da condição é feita depois do bloco de instruções
- Valem as mesmas condições do *while...do*

Exemplo de *do...while*

```
import java.util.Scanner;
public class Fatorial {
  public static void main(String[] args) {
    Scanner teclado = new Scanner(System.in);
    System.out.print("Entre com um número inteiro positivo: ");
    int numero = teclado.nextInt();
    long fatorial = 1;
    do {
      fatorial *= numero--;
    } while (numero > 0);
    System.out.println("O fatorial desse número é " + fatorial);
  }
}
```

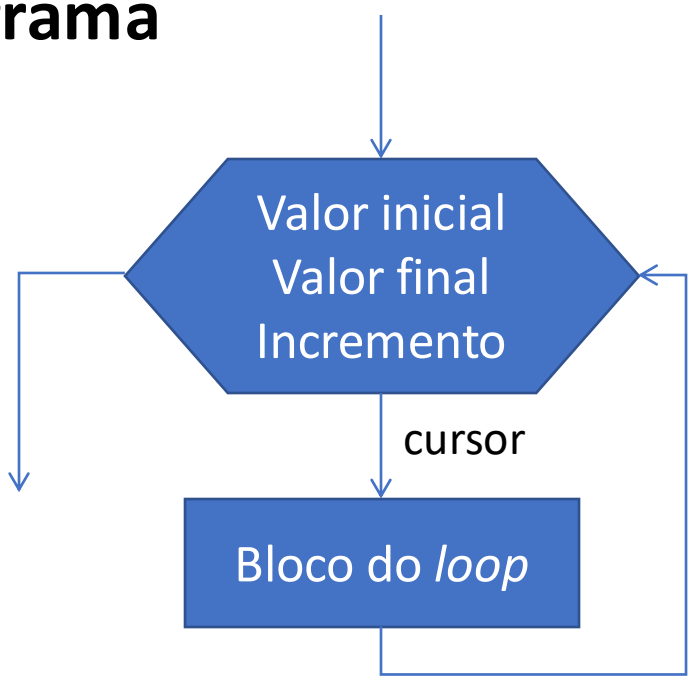
Programa para calcular fatorial de um número

Mas então... dá no mesmo?

- Naaaaaaaaaaaaaão!!!
- Reparem que pedimos para o usuário **"Entre com um número inteiro positivo: "**
  - Para esse cenário, ambas as estruturas funcionaram
- O que acontece se pedirmos para o usuário **"Entre com um número inteiro não negativo: “**
  - Qual das duas estruturas resolve o problema corretamente se o usuário entrar com zero?
  - Qual o resultado provido pela outra?
  - Lembrem: fatorial de zero é 1!

Repetição contável do tipo *for*

Fluxograma



Pseudocódigo

```
...
Para CURSOR variando de VALOR
INICIAL a VALOR FINAL com
passo INCREMENTO
   INSTRUÇÃO 1
   INSTRUÇÃO 2
   ...
   INSTRUÇÃO N
...
```

Repetição contável do tipo *for*

Java

```
...
for (INICIALIZAÇÃO; CONDIÇÃO; INCREMENTO) {
  INSTRUÇÃO 1;
  INSTRUÇÃO 2;
  ...
  INSTRUÇÃO N;
}
...
```

Repetição contável do tipo *for*

- Executa o bloco de instruções por um número predeterminado de vezes
- **Expressão de inicialização**
  - Utilizada para iniciar a variável de controle do *loop* (cursor)
  - Executada uma única vez, antes do primeiro *loop*
- **Expressão de condição**
  - Termina a execução do *loop* quando tiver o valor *false*
  - Verificada antes de cada *loop*

Repetição contável do tipo *for*

- Executa o bloco de instruções por um número predeterminado de vezes
- **Expressão de incremento**
  - Pode incrementar ou decrementar a variável de controle (cursor)
  - Executada no final de cada *loop*
- As expressões devem sempre estar entre parênteses e separadas por ponto-e-vírgula
- Pode omitir { e } caso execute somente uma instrução

Exemplo de *for*

```
import java.util.Scanner;
public class Fatorial {
  public static void main(String[] args) {
    Scanner teclado = new Scanner(System.in);
    System.out.print("Entre com um número inteiro positivo: ");
    int numero = teclado.nextInt();
    long fatorial = 1;
    for (int i = 2; i <= numero; i++) {
      fatorial *= i;
    }
    System.out.println("O fatorial desse número é " + fatorial);
  }
}
```

Programa para calcular fatorial de um número

String

- Classe em Java para representar variáveis textuais
- Possui uma variedade de métodos para manipulação de texto
- Métodos podem ser chamados a partir de uma variável ou do texto em si
  - System.out.println(texto.charAt(2));
  - System.out.println(“Texto”.charAt(2));
- Para manipulações mais eficientes com strings, veja a classe **StringBuffer**

Alguns métodos de String

- equals(Object)
  - Informa se duas Strings são iguais
  - Ex.: "Flamengo".equals("flamengo") → false
  - Ex.: "Flamengo".equals("Flamengo") → true
- length()
  - Retorna o tamanho da String
  - Ex.: "Flamengo".length() → 8

Alguns métodos de String

- concat(String)
  - Concatena duas strings, de forma equivalente ao operador +
  - Ex.: "Fla".concat("mengo") → “Flamengo”
- charAt(int)
  - Retorna o caractere na posição informada
  - A primeira posição é zero
  - Ex.: "Flamengo".charAt(2) → ‘a’

Alguns métodos de String

- compareTo(String)
  - Retorna 0 se as strings forem iguais, <0 se a string for lexicamente menor e >0 se for lexicamente maior que o parâmetro
  - "Fla".compareTo(“Flu") → -20
- compareToIgnoreCase(String)
  - Idem ao anterior, sem considerar diferenças entre maiúsculas e minúsculas
  - "Fla".compareToIgnoreCase("fla") → 0

Alguns métodos de String

- indexOf(String, int)
  - Busca pela primeira ocorrência de uma substring ou caractere a partir de uma posição informada
  - Retorna -1 se não encontrar a substring
  - Ex.: "Fla x Flu".indexOf("Fl", 0)) → 0
  - Ex.: "Fla x Flu".indexOf("Fl", 1)) → 6

Alguns métodos de String

- substring(int, int)
  - Retorna a substring que vai da posição inicial (inclusive) até a posição final (exclusive), ambas informadas
  - Ex.: "Flamengo".substring(3,6)) → “men”
- toLowerCase()
  - Retorna a string em minúsculas
  - Ex.: "Flamengo".toLowerCase() → “flamengo”

Alguns métodos de String

- toUpperCase()
  - Retorna a string em maiúsculas
  - Ex.: "Flamengo".toUpperCase() → “FLAMENGO”
- trim()
  - Remove espaços antes e depois da string
  - Ex.: " Flamengo ".trim() → “Flamengo”

Alguns métodos de String

- Veja os demais métodos em
  - http://docs.oracle.com/javase/8/docs/api/java/lang/String.html

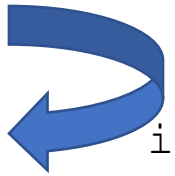
- Na verdade, todas as classes de apoio do Java podem ser consultadas em
  - http://docs.oracle.com/javase/8/docs/api/

Exemplo

- Programa para gerar a citação a partir de um nome
  - Ex.: Gleiph Ghiotto Lima de Menezes → MENEZES, G. G. L.

```
import java.util.Scanner;

public class Citacao {
  public static void main(String[] args) {
    Scanner teclado = new Scanner(System.in);
    String iniciais = "";
    String sobrenome = "";

    System.out.print("Entre com um nome completo: ");
    String nome = teclado.nextLine().trim();
```

Exemplo



```
    int inicio = 0;
    int fim = nome.indexOf(" ", inicio);
    while (fim != -1) {
      iniciais += nome.substring(inicio, inicio + 1) + ". ";
      inicio = fim + 1;
      fim = nome.indexOf(" ", inicio);
    }
    sobrenome = nome.substring(inicio).toUpperCase();

    System.out.print(sobrenome + ", ");
    System.out.println(iniciais.toUpperCase().trim());
  }
}
```

Exercício 1

- Faça um programa que leia três coordenadas num espaço 2D e indique se formam um triângulo, juntamente com o seu tipo (equilátero, isósceles e escaleno)
  - Equilátero: todos os lados iguais
  - Isósceles: dois lados iguais
  - Escaleno: todos os lados diferentes

Exercício 2

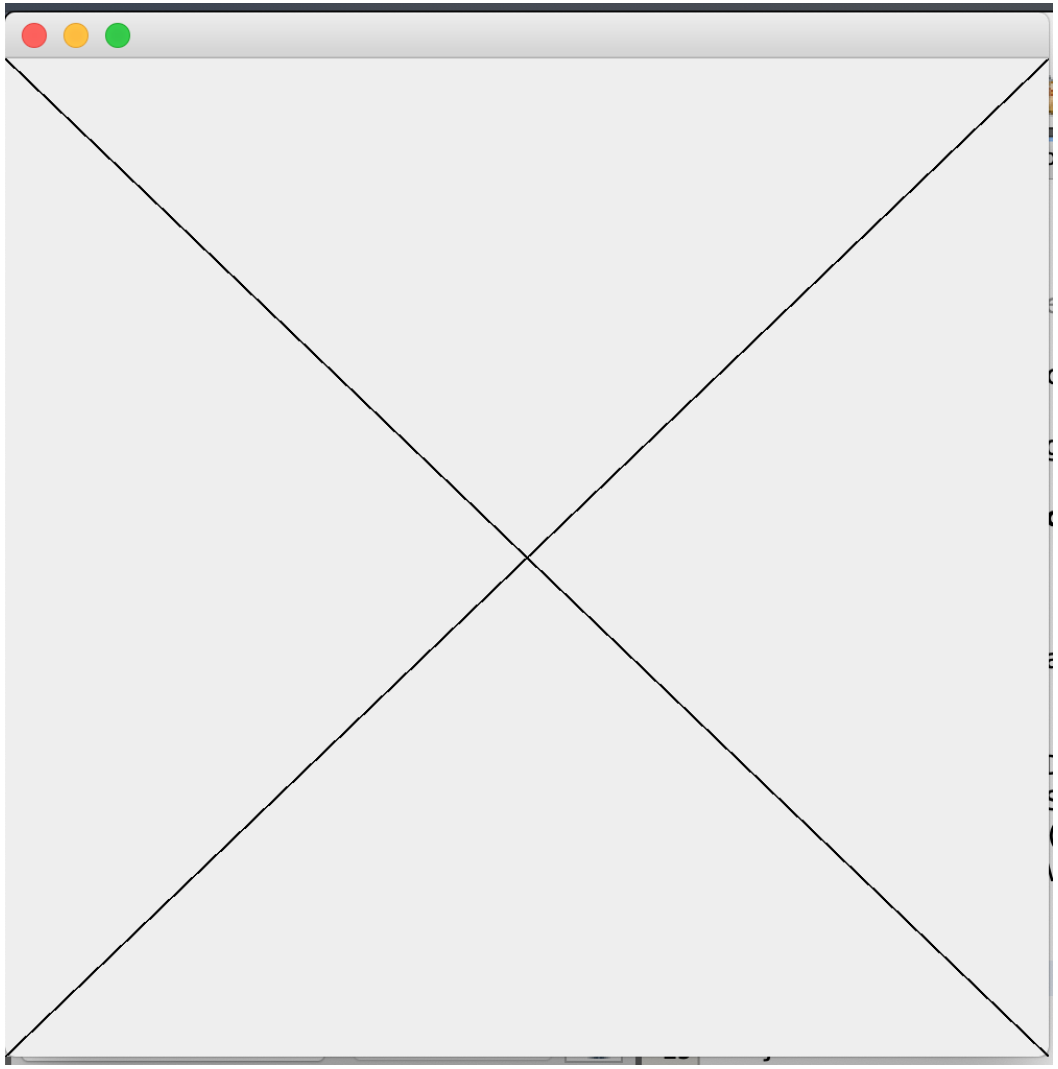
- Faça um programa para listar todos os divisores de um número ou dizer que o número é primo caso não existam divisores
  - Ao final, verifique se o usuário deseja analisar outro número

Extra (Exemplo 1)

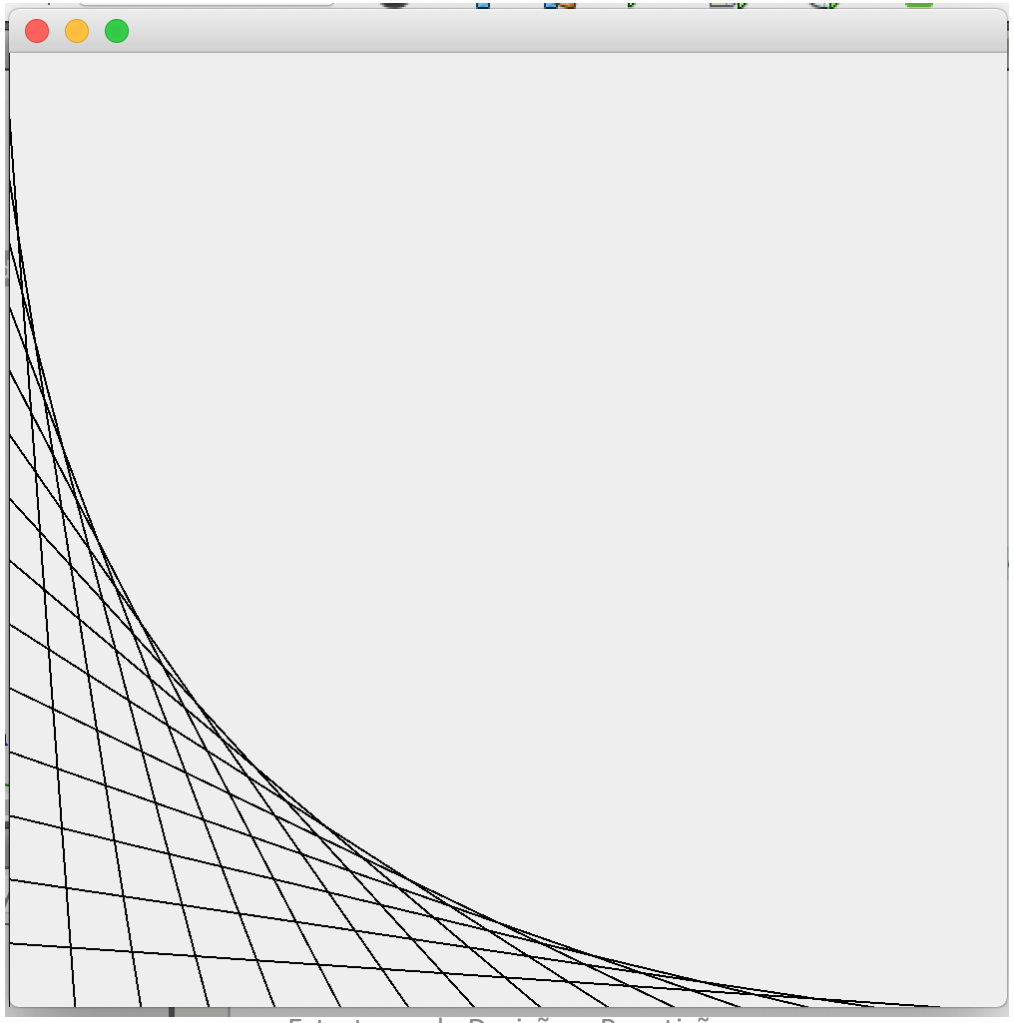
```java
package br.ufjf.dcc.oo.desenho;

import javax.swing.JFrame;

public class Exercicio {

    public static void main(String[] args) {

        JFrame frame = new JFrame();
        DrawPanel panel = new DrawPanel();

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(500, 500);
        frame.add(panel);
        frame.setVisible(true);


    }

}
```

```java
package br.ufjf.dcc.oo.desenho;

import java.awt.Graphics;
import javax.swing.JPanel;

public class DrawPanel extends JPanel {

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        int width = getWidth();
        int height = getHeight();
        g.drawLine(0, 0, width, height);
        g.drawLine(0, height, width, 0);
    }

}
```

Extra



Extra (Exemplo 2)



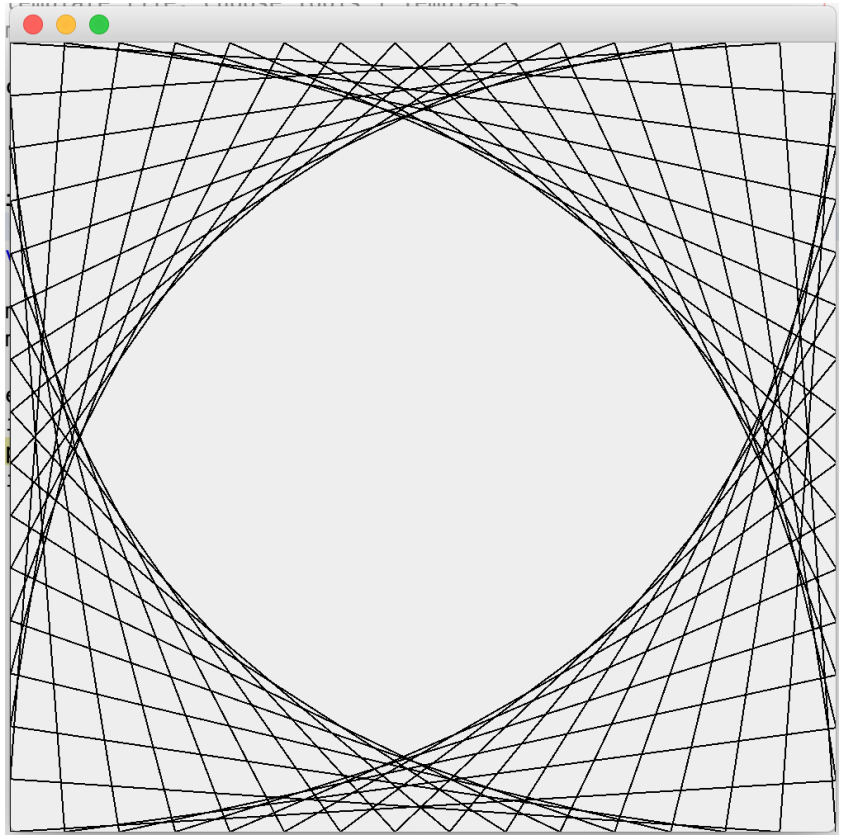
Extra (Exemplo 2)

```java
package br.ufjf.dcc.oo.desenho;

import javax.swing.JFrame;

public class Exercicio {

    public static void main(String[] args) {

        JFrame frame = new JFrame();
        DrawPanelImproved panel = new DrawPanelImproved();

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(500, 500);
        frame.add(panel);
        frame.setVisible(true);


    }

}
```

```java
package br.ufjf.dcc.oo.desenho;

import java.awt.Graphics;
import javax.swing.JPanel;

public class DrawPanelImproved extends JPanel {

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        int width = getWidth();
        int height = getHeight();

        double slices = 15;
        int x1, y1, x2, y2;
        double sliceWidth = width / slices;
        double sliceHeight = height / slices;

        for (int i = 0; i < slices; i++) {
            x1 = 0;
            y1 = (int) (i * sliceHeight);
            x2 = (int) (i * sliceWidth);
            y2 = (int) height;
            g.drawLine(x1, y1, x2, y2);
        }
    }
}
```

Desafio

- Implemente o programa que tenha a seguinte saída



Operadores e Estruturas de Decisão

Orientação a Objetos – DCC025

Gleiph Ghiotto Lima de Menezes

gleiph.ghiotto@ufjf.br