

# Estruturas de Repetição e String

Leonardo Gresta Paulino Murta

leomurta@ic.uff.br

# Aula de hoje...

- Estruturas de repetição
  - *while...do*
  - *do...while*
  - *for*
- String
  - Manipulação de textos

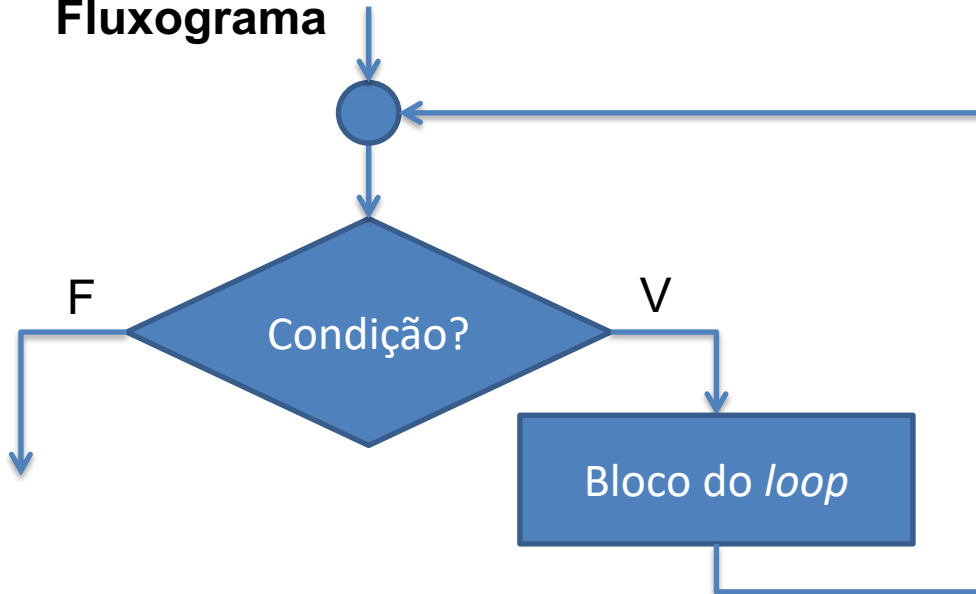
# Estruturas de Repetição

- 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*

- Programa para calcular fatorial de um número:

```
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);
    }
}
```



# Exemplo de *while...do*

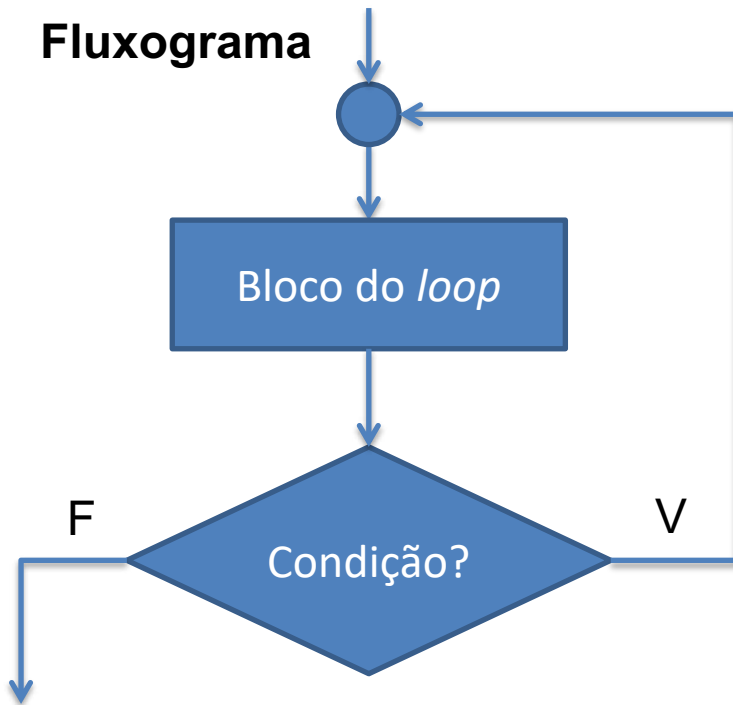
- Qual a saída do programa abaixo?

```
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*

- Programa para calcular fatorial de um número:

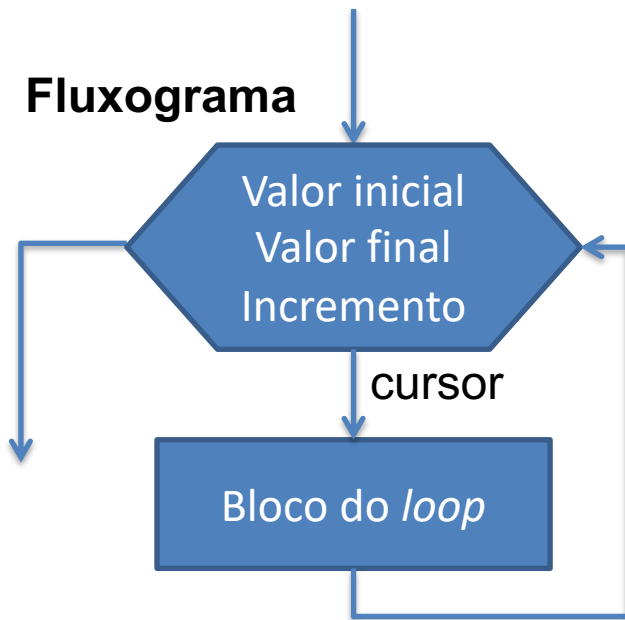
```
import java.util.Scanner;
public class Fatorial {
    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);
        System.out.print("Entre com um número inteiro positivo: ");
        int numero = teclado.nextInt();
        long fatorial = 1;
        do {
   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

# Mas então... dá no mesmo?

- Naaaaaaaaaaaaaaaaão!!!
- 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*
- **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*

- Programa para calcular fatorial de um número:

```
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);
    }
}
```

# Exemplo de *for*

- Qual a diferença de

```
for (int i = 2; i <= numero; i++) {
    fatorial *= i;
}
```

- Para

```
for (int i = numero; i > 1; i--) {
    fatorial *= i;
}
```

- ?

# 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
- 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`
- `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"
- `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.: Leonardo Gresta Paulino Murta → MURTA, L. G. P.

```
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

- 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

# Estruturas de Repetição e String

Leonardo Gresta Paulino Murta

leomurta@ic.uff.br