Controle de Fluxo

Operadores Relacionais

- Os **operadores relacionais** são utilizados em uma **condição booleana**;
- Uma **condição booleana** é **qualquer** expressão que **retorne true** ou **false**;
- Para isso, você pode usar os **operadores <**, **>**, **<=**, **>=**, **==** e outros.
- Os **operadores relacionais** são utilizados nas **diversas estruturas** da linguagem java.

O if-else

- O IF é uma **estrutura de seleção simples**;
- A **sintaxe** do if no **Java** é a seguinte:

```
if (condicaoBooleana) {
    codigo;
}
```

- Exemplo:

```
int idade = 15;

if (idade < 18) {
    System.out.println("Não pode entrar");
}
```

O if-else

- Além disso, você pode usar a **cláusula else** para indicar o comportamento que deve ser executado no **caso da expressão booleana ser falsa**:

```
int idade = 15;
if (idade < 18) {
    System.out.println("Não pode entrar");
} else {
    System.out.println("Pode entrar");
}
```

- Você pode **concatenar expressões booleanas** através dos **operadores lógicos "E"** e **"OU".** O **"E"** é representado pelo **&&** e o **"OU"** é representado pelo **||.**

O if-else

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- Exemplo

```
int idade = 15;
boolean amigoDoDono = true;
if (idade < 18 && amigoDoDono == false) {
    System.out.println("Não pode entrar");
}
else {
    System.out.println("Pode entrar");
}
```

- Para comparar se uma variável tem o mesmo valor, utilizamos o **operador ==**

```
int mes = 1;
if (mes == 1) {
    System.out.println("Você deveria estar de férias");
}
```

Switch

- Permite selecionar o **bloco de código** a ser executado baseado no **teste lógico** de uma **expressão**;

- O switch é a forma **evoluída** para o **if**, podendo tratar mais de dois blocos de execução.

```
public class ExemploSwitch {
  public static void main(String[] args) {
    int i = 2;
    // Switch que irá imprimir na tela o valor 2
    switch (i) {
      case 1 : System.out.println("Valor de i é 1");
      break;
      case 2 : System.out.println("Valor de i é 2");
      break;
      case 3 : System.out.println("Valor de i é 3");
      break;
      default: System.out.println("Default");
    }
```

While

- O while é um comando usado para fazer um **laço (loop):**

```
int idade = 15;
while (idade < 18) {
    System.out.println(idade);
    idade = idade + 1;
}
```

Do-while

- O **Do While** é similar **ao While**, porem ele exige que o **bloco de comandos** seja **executado** pelo menos uma vez:

```
public class ExemploDoWhile {
  public static void main(String[] args) {
    int i=0;
    do {
      System.out.println("Contador é " + i);
    } while(++i<10)

    do {
      System.out.println("Laço infinito.");
    } while(true);
  }
}
```

For

- Outro **comando de loop** extremamente utilizado é o **for**;
- A idéia é a mesma do **while**: fazer um trecho de código ser **repetido** enquanto uma **condição continuar verdadeira**;
- Mas além disso, o **for** isola também um **espaço para inicialização** de variáveis e o modificador dessas variáveis;

```
for (inicializacao; condicao; incremento) {
    codigo;
}
```

For

- Exemplo:

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

- **Pós incremento ++**

- i = i + 1 pode realmente ser **substituído** por i++ quando isolado, no entanto em atribuições essa operação tem implicações;

```
int i = 5;

int x = i++;
```

- Qual é o valor de x? O de i, após essa linha, é 6.

- O **operador ++**, quando vem **após** a **variável**, retorna o valor **antigo**, e **incrementa** (pós incremento), fazendo x valer 5.

Pré incremento

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- Se você tivesse usado o ++ **antes da variável** (**pré incremento**), o resultado seria 6:

```
int i = 5;

int x = ++i; // aqui x valera 6
```

Controlando Laços

- Apesar de termos **condições booleanas** nos nossos laços, em algum momento, podemos **decidir parar o** loop por algum motivo especial sem que o resto do laço seja executado.

```
for (int i = x; i < y; i++) {
    if (i % 19 == 0) {
        System.out.println("Achei um número divisível por 19 entre x e y");
        break;
    }
}
```

- Da mesma maneira, é possível **obrigar o loop a executar** o próximo laço. Para isso usamos a palavra chave **continue**.

```
for (int i = 0; i < 100; i++) {
    if (i > 50 && i < 60) {
        continue;
    }
    System.out.println(i);
}
```

Escopo das variáveis

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- No Java, podemos **declarar variáveis** a **qualquer momento**. Porém, dependendo de onde você as declarou, ela vai valer de um **determinado** ponto a outro.

```
// aqui a variável i não existe
int i = 5;
// a partir daqui ela existe
```

- O **escopo da variável** é o nome dado ao **trecho de código** em que aquela variável **existe** e onde é **possível acessá-la**.
- Quando abrimos um **novo bloco** com as **chaves**, as variáveis declaradas ali dentro **só valem até o fim daquele bloco**.

```
// aqui a variável i não existe
int i = 5;
// a partir daqui ela existe
while (condicao) {
    // o i ainda vale aqui
    int j = 7;
    // o j passa a existir
}
// aqui o j não existe mais, mas o i continua dentro do escopo
```

Escopo das variáveis

- Se você tentar acessar uma **variável fora** de seu escopo, **ocorrerá** um **erro** de **compilação**;

```
EscopoDeVariavel.java:8: cannot find symbol
symbol  : variable j
location: class EscopoDeVariavel
                System.out.println(j);
                                   ^
1 error
```

- O **mesmo** vale **para** um **if**:

```
if (algumBooleano) {
    int i = 5;
}
else {
    int i = 10;
}
System.out.println(i); // cuidado!
```

Escopo das variáveis

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- Forma correta do código anterior

```
int i;
if (algumBooleano) {
    i = 5;
}
else {
    i = 10;
}
System.out.println(i);
```

Entrada de Dados: Classe Scanner

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- Quem programa em **linguagens estruturadas**, como **C** e **Pascal**, e está aprendendo **Java**, depara-se com a seguinte situação: como **atribuir valores** para uma **variável** usando o **teclado**?

- Em **C**, por exemplo, tem-se a função **scanf()** e em **Pascal**, o procedimento **readln()**.

- No **Java**, está disponível a classe **Scanner** do pacote **java.util**. Essa classe implementa as operações de entrada de dados pelo teclado no console.

Entrada de Dados: Classe Scanner

- Para utilizar a classe Scanner em uma aplicação Java deve-se proceder da seguinte maneira:

```
importar o pacote java.util:
import java.util.Scanner;
Instanciar e criar um objeto Scanner:
Scanner ler = new Scanner(System.in);
Lendo valores através do teclado:
Lendo um valor inteiro:
int n;

System.out.printf("Informe um número para a tabuada: ");
n = ler.nextInt();
```

Entrada de Dados: Classe Scanner

- No exemplo anterior eu declarei a variavel **n** to tipo int(inteiro). Para cada tipo temos um .next , segue abaixo uma lista dos ".next's" :

- **int ->** .next**Int**();
  float -> .next**Float**();
  double -> next**Double**();
  char -> next**Char**();
  String -> next**Line**();

**String não é nextString() e sim nextLine().*

Entrada de Dados: JOptionPane

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- No exemplo anterior eu declarei a variavel **n** to tipo int(inteiro). Para cada tipo temos um .next , segue abaixo uma lista dos ".next's" :

- **int ->** .next**Int**();
  float -> .next**Float**();
  double -> next**Double**();
  char -> next**Char**();
  String -> next**Line**();

**String não é nextString() e sim nextLine().*

Exercício