

Objetivos: Compreender como o computador faz escolhas durante a execução de programas;
Aprender o funcionamento dos comandos IF e ELSE;
Trabalhar com comparação de Strings;

Recursos: Aula em laboratório com uso do computador; Quadro e pincel; Roteiro de Aula;

TOMADA DE DECISÕES

Muitas vezes precisamos optar entre situações diferentes ou testar condições diversas durante a execução de um programa. Em Java existem diversas maneiras de realizarmos essas comparações, uma delas é através do comando IF. Observe o trecho abaixo:

Arquivo: Decisao.java

```
1. int x = 15;  
2. if (x > 10) {  
3.     JOptionPane.showMessageDialog(null, "O número é maior que 10.");  
4. }
```

A linha 1 declara uma variável x com valor 15.

A linha 2 realiza o teste do valor de x, perguntando se ele é maior que 10. Caso essa condição seja verdadeira, o código que está entre as chaves será executado, neste caso é verdadeiro.

A linha 3 exibe uma janela de mensagem no caso da condição ser verdadeira.

Se o valor de x fosse 8, o programa não exibiria nenhuma mensagem.

OPTANDO POR DOIS CÓDIGOS (IF, ELSE)

Existem casos onde queremos executar um código quando a condição for verdadeira e outro código diferente quando a condição for falsa. Para estes casos usamos a seguinte estrutura:

Arquivo: Decisao2.java

```
1. int x = 8;  
2. if (x > 10) {  
3.     JOptionPane.showMessageDialog(null, "Acima da meta.");  
4. }else{  
5.     JOptionPane.showMessageDialog(null, "Abaixo da meta.");  
6. }
```

Neste segundo código, a linha 4 adiciona um complemento ao comando if que permite executar um código diferente caso o teste realizado falso. Observe que o valor de x é 8, o teste quer saber se 8 é maior que 10, como a resposta a essa pergunta é NÃO, o código da linha 3 não será executado, o programa passa para a linha 4 e executa o comando da linha 5.

OPERADORES RELACIONAIS

Os operadores que podem ser usados nos testes do método IF são:

== (igual)	> (maior que)	< (menor que)	
>= (maior ou igual)	<= (menor ou igual)	!= (diferente)	! (não, negativa)

MAIS OPÇÕES DE CÓDIGO

Há ainda uma outra forma de usar if/else para trabalhar com múltiplas opções de códigos. Observe o programa abaixo:

Arquivo: Decisao3.java

```
1. int x = 5;
2. if (x >= 10) {
3.     JOptionPane.showMessageDialog(null, "Maior que 10.");
4. }else if (x >= 5){
5.     JOptionPane.showMessageDialog(null, "Maior que 5.");
6. }else{
7.     JOptionPane.showMessageDialog(null, "Quatro ou menos.");
8. }
```

Este trecho utiliza a construção **if... else if... else** para permitir que diversas comparações sejam feitas, porém apenas um código do bloco será executado. Na hora que o programa detecta uma condição verdadeira, todo o resto é descartado. Caso não haja nenhuma condição verdadeira, o que estiver escrito em **else** será executado.

CONDIÇÕES MÚLTIPLAS

É possível agrupar duas ou mais condições dentro do mesmo if. Para isso usamos os operadores lógicos:

E (&&) - só é verdade quando **todos** os testes são verdade
OU (||) – é verdade quando **qualquer** dos testes é verdade

Veja o exemplo abaixo:

Arquivo: Decisao4.java

```
1. int x = 5, y = 10;
2. if (x > 0 && y > 0) {
3.     JOptionPane.showMessageDialog(null, "Todos Positivos");
4. }
```

O programa vai testar se $x > 0$, se for verdade ele testa se $y > 0$, se for verdade também ele exibirá a mensagem. Porém se um dos testes for falso ele não mostra nada.

Veja esse outro exemplo:

Arquivo: Decisao5.java

```
1. int x = 5, y = 10;
2. if (x > 0 || y < 0) {
3.     JOptionPane.showMessageDialog(null, "Todos Positivos");
4. }
```

O programa vai testar se $x > 0$, se for verdade ele já executa o código, pois o operador OU só precisa que 1 teste seja verdade para executar o código. Mas caso o primeiro teste seja falso, ele vai testar os demais até encontrar um verdadeiro. Caso não haja nenhum verdadeiro, o código será ignorado.

TESTANDO STRINGS

Em Java as strings se comportam de forma diferente das variáveis int, double e outros tipos padrão, assim não podemos usar os mesmos operadores de antes. Existe uma coleção de métodos que são usados para comparar as Strings, observe o código abaixo:

Arquivo: Decisao6.java

```
1. String nome = "Pedro";  
2. if (nome.equals("Pedro")) {  
3.     JOptionPane.showMessageDialog(null, "Nome correto");  
4. }
```

O método equals do objeto nome compara letra por letra da string, inclusive se estas são maiúsculas/minúsculas. Se a palavra for exatamente igual, o método retorna **verdadeiro**.

Uma alternativa para este método é o **equalsIgnoreCase()** que testa a palavra mas ignora maiúsculas e minúsculas.

As strings não precisam ser colocadas direto no código, podem ser lidas de campos ou obtidas de outras formas.

EXERCÍCIOS

1. Crie um programa em que o usuário digite a meta mensal de produção de uma indústria (em unidades) e num outro campo digite a quantidade de unidades produzidas até o momento. Ao clicar num botão, o programa deve verificar se a meta foi atingida/ultrapassada e informar ao usuário com uma caixa de mensagem. Se a produção ainda está abaixo da meta, o programa deve informar quantas unidades ainda faltam para atingir a meta.

2. Escreva um programa que possua 5 campos onde o usuário poderá digitar nomes de pessoas e um outro campo que será usado para fazer um tipo de pesquisa. Coloque dois botões, que ao serem clicados o programa deve testar se o nome digitado na pesquisa existe em um dos campos acima e em qual campo está (use caixas de diálogo) ou dizer que não existe. A diferença entre os botões é que um só deve aceitar escritas idênticas (ex.: "João" e "João") e o outro aceita escritas diferentes (ex.: "João" e "JOÃO").

