Capítulo - 07

7.1 - Estruturas condicionais e de repetição

As estruturas condicionais existem em todas as linguagens de programação e possibilitam que a execução de um programa seja desviada de acordo com certas condições. Os comandos condicionais ou ainda instruções condicionais usados em Java são, **`if-else`** e **`switch case`.** Essas duas estruturas de desvio existentes na linguagem possibilitam executar diferentes trechos de um programa com base em certas condições

7.2 Operadores relacionais

Para comparar valores a linguagem Java utiliza estes operadores para retornar o resultado de uma pergunta lógica ou condição. Sendo a resposta um valor verdadeiro ou falso. A tabela abaixo mostra quais são estes operadores.

Função	Símbolo utilizado em Java	Exemplo
Igual	= =	(x = = y) {“São iguais”}
Diferente	!=	(x != y) {“São diferentes”}
Maior que	>	(x > y) {“x é maior que y”}
Maior ou igual a	>=	(x >= y) {“x é maior ou igual a y”}
Menor que	<	(x < y) {“x é menor que y”}
Menor ou igual a	<=	(x<=y) {“x é menor ou igual a y”}

Sendo: **x = 5** **y = 7**	**Temos as seguintes respostas do sistema**
(x = = y)	Falso
(x != y) {“São diferentes”}	Verdadeiro
(x > y) {“x é maior que y”}	Falso
(x >= y) {“x é maior ou igual a y”}	Falso
(x < y) {“x é menor que y”}	Verdadeiro
(x<=y) {“x é menor ou igual a y”}	Verdadeiro

O `if` em conjunto com o `else`, forma uma estrutura que permite seleção entre dois caminhos distintos para a execução, dependendo do resultado (verdadeiro ou falso) de uma expressão lógica. Nesse tipo de estrutura, se a condição for verdadeira, serão executadas as instruções que estiverem posicionadas entre o `if/else`. Sendo a condição falsa, serão executadas as instruções que estiverem após a instrução `else`. A sintaxe para a utilização do conjunto `if-else` é demonstrada em seguida. Observe que a condição sempre deve aparecer entre parênteses, item obrigatório na linguagem Java.

```
if (<Condição>) {
     <instruções para condição se verdadeira>;
}
else {
    <instruções para condição se falsa>;
}
```

Dica! Assim como na maioria das instruções em Java, o conjunto `if-else` deve ser utilizado com minúsculas e, caso haja apenas uma instrução as ser executada, tanto no `if` como no `else`, o uso de chaves é desnecessário. **Lembre-e** de que apenas as chaves são utilizadas quando um bloco de instruções precisa ser executado, isto e, mais do que uma instrução.

Existe ainda outras forma de associar `if-else`:

if sem else;

if com else;

if com else aninhado.

7.3 - Operadores lógicos

São operadores que permitem avaliar o resultado lógico de diferentes operações aritméticas em uma expressão.

Função	Símbolo utilizado no Java	Exemplo	Se lê
E lógico ou AND	&&	x && y	x e y
Ou lógico ou OR	||	x || y	x ou y
Negação ou not	!	!x	diferente de x

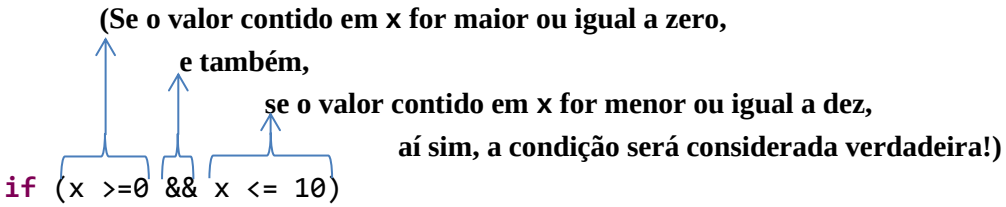
7.4 - Exemplo de uso de operadores lógicos:

Você pode concatenar expressões booleanas através dos operadores lógicos "E" e "OU". O "E" é representado pelo *&&* e o "OU" é representado pelo ||. Um exemplo seria verificar se ele tem menos de18 anos e se ele não é amigo do dono:

```
1  class AmigoDono {
2      public static void main(String[] args)
3      {
4          int idade = 15;
5          boolean amigoDoDono = true;
6
7          if (idade < 18 && amigoDoDono == false)
8          {
9              System.out.println("Não pode entrar");
10         }else {
11             System.out.println("Pode entrar");
12         }
13     }
14 }
```

Esse código poderia ficar ainda mais legível, utilizando-se o operador de negação '!'. Esse operador transforma o resultado de uma expressão booleana de **`false`** para **`true`** e vice-versa.

```
1  public class AmigoDono {
2
3      public static void main(String[] args)
4      {
5          int idade = 15;
6          boolean amigoDoDono = true;
7
8          if (idade < 18 && !amigoDoDono){
9              System.out.println("Não pode entrar");
10         }else{
11             System.out.println("Pode entrar");
12             }
13     }
14 }
```



7.4 - `If` sem `else`

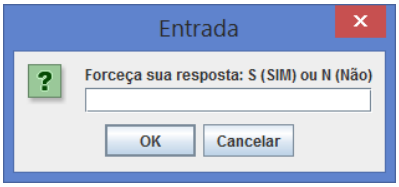
O exemplo abaixo mostra o uso prático e simples do `if` sem a presença do `else`. Trata-se de uma classe que o usuário fornece uma letra (S ou N) e em que é apresentada em tela com uma mensagem do tipo "Letra Fornecida: S".

Caso a letra fornecida não seja S ou N, a mensagem apresentada será diferente, como, por exemplo, "Letra fornecida: F **é inválida**". Neste caso estamos utilizando o operador lógico **&&** and para avaliar duas possibilidades.

```
1 package br.com.capitulo8.apostilaEtec;
2
3 import javax.swing.JOptionPane;
4
5 public class If {
6
7     public static void main(String[] args) {
8         String resposta = JOptionPane.showInputDialog("Forneça sua resposta: S (SIM) ou N (Não)");
9         String mensagem = "Letra fornecida: " + resposta;
10
11        if (!resposta.equals("S") && !resposta.equals("N")) {
12            mensagem = mensagem + " é inválida";
13        }
14
15        JOptionPane.showMessageDialog(null, mensagem);
16    }
17 }
```

Digite o exemplo acima e verifique o resultado. O usuário deve entrar com uma letra (S ou N) por meio da caixa de diálogo gerada pela classe `JOptionPane`.

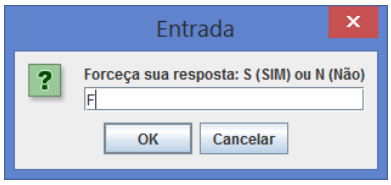
Na linha 5 conforme abordado no capítulo 6 no ítem 6.4, o método `showInputDialog` da caixa de diálogo `JOptionPane` permite que um valor seja fornecido pelo usuário. Esse valor é armazenado na variável `resposta`. Como você já deve ter percebido a caixa de diálogo possui os botões OK e Cancelar.

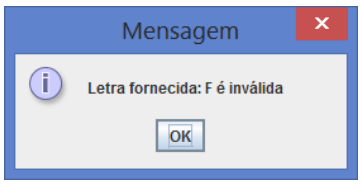


Caso o usuário pressione o botão Cancelar, o valor da variável `resposta` será nulo (`null`). Como esse exemplo não realiza essa verificação (tratamento), se o Cancelar for pressionado, será gerado um erro na execução da classe. Essa funcionalidade será tratada no próximo exemplo.

Na linha 7 a variável `mensagem` armazena a frase "`Letra fornecida: `" acumulado ou juntado/concatenado com o conteúdo da variável `resposta`. Supondo que o usuário tenha pressionado a letra S, o conteúdo da variável mensagem será: "`Letra fornecida: S`". O conteúdo dessa variável é apresentada no final da execução da classe. Esse texto será modificado em tempo de execução e seu conteúdo será concatenado dependendo da interação do usuário.

Na linha 9 temos a instrução `if` responsável por verificar se a letra digitada pelo usuário é diferente de S e de N. Observe que o comparador de igualdade de uma `String` não é o duplo (`= =`) e sim o método `equals`. Observe também o caractere exclamação (`!`) que significa negação, isto é, se o conteúdo da variável `resposta` não for S e nem N. Perceba ainda o operador lógico E/AND (`&&`) usado para agrupar os dois testes condicionais de uma só vez. Ou seja, a comparação do `if` será verdadeira se as duas condições forem verdadeiras. Se isso ocorrer, a linha 10 é executada e o conteúdo da variável `mensagem` é acumulado/concatenado com o texto "é inválida!". Note que este texto será acumulado somente se a letra digitada pelo usuário não for S ou não for N.





7.5 - If com else, tratamento de erros com try catch finally(visão preliminar).

O exemplo abaixo mostra o uso prático do `if-else` com duas comparações usadas para validar as entradas do usuário. Este exemplo realiza três validações. Em primeiro lugar, verifica se o usuário realmente entrou com um valor na caixa de diálogo, depois verifica se o valor digitado é numérico, logo a seguir verifica se esse valor está entre 0 e 10 (a faixa de valores possíveis, uma vez que a nota deve assumir apenas valores entre 0 e 10).

```
1  import javax.swing.JOptionPane;
2
3  public class IfComElse {
4
5      public static void main(String[] args) {
6          String aux = JOptionPane.showInputDialog(null, "Forneça um valor numérico entre 0 e 10");
7          if (aux !=  null) {
8              try
9              {
10                 float x = Float.parseFloat(aux);
11                 if (x >=0 && x <= 10){
12                     JOptionPane.showMessageDialog(null, "Nota = " + x + " valor válido");
13                 }
14                 else{
15                     JOptionPane.showMessageDialog(null, "Nota = " + x + " valor inválido");
16                 }
17             }
18             catch (NumberFormatException erro) {
19                 JOptionPane.showMessageDialog(null, "Digite apenas valores Numéricos - \n"
20                         + erro.toString());
21             }
22         }
23         System.exit(0);
24     }
25 }
```

Na linha 6 conforme abordado anteriormente, o método `ShowInputDialog` da caixa de diálogo `JOptionPane` permite que um valor seja inserido pelo usuário. Esse valor é armazenado na variável `aux`. Como você pode notar, a caixa de diálogo possui dois botões, OK e Cancelar. Caso o usuário pressione o botão Cancelar, o valor da variável `aux` será nulo (`null`).

Na linha 7 verificamos se o usuário pressionou o botão Cancelar da caixa de diálogo que foi exibida, ou seja, se o valor da variável `aux` for diferente de nulo (`!=`) ele executará o trecho entre chaves das linhas 7 até alinha 22.

Na linha 8 temos a estrutura **`try-catch`**. O **`try`**, na linha 8 e **`catch`** na linha 18. O objetivo de sua utilização é a **previsão de erros** de execução. Que será abordado no próximo tópico.

Na linha 9 temos a instrução `if` responsável por verificar se o valor da nota, digitada pelo usuário, está na faixa de valores entre 0 e 10. Se a comparação for verdadeira, será enviada uma mensagem positiva na (linha 12); caso seja falsa, executa a instrução `else` na (linha 14) e envia uma mensagem negativa na (linha 15)

Na linha 18 temos o **`catch`**, que finaliza a estrutura **`try-catch`** e tem como função, desviar a execução de um programa caso ocorram certos tipos de erro predefinidos durante o processamento das linhas. O emprego de **`try-catch`** é para evitar que o programador precise fazer testes de verificação e avaliação antes de realizar certas operações. Quando um erro ocorre, ele gera uma exceção que pode ser tratada pelo programa. A estrutura **`try-catch-finally`** pode ser usada tanto com checked exceptions como com unchecked exceptions.

7.6 - `If` com `else` aninhado

```
1 import javax.swing.JOptionPane;
2 public class IfComElseAninhado {
3
4     public static void main(String[] args) {
5         String diaDaSemana = JOptionPane.showInputDialog("Forneça um valor entre 1 e 7");
6         if (diaDaSemana != null)
7         {
8
9                 int dia = Integer.parseInt(diaDaSemana);
10                if (dia==1)
11                    diaDaSemana = "Domingo";
12                else if (dia==2)
13                    diaDaSemana = "Segunda";
14                else if (dia==3)
15                    diaDaSemana = "Terça";
16                else if (dia==4)
17                    diaDaSemana = "Quarta";
18                else if (dia==5)
19                    diaDaSemana = "Quinta";
20                else if (dia==6)
21                    diaDaSemana = "Sexta";
22                else if (dia==7)
23                    diaDaSemana = "Sábado";
24                else
25                    diaDaSemana = "Dia da Semana desconhecido!";
26
27                JOptionPane.showMessageDialog(null, diaDaSemana);
28        }else {
29            JOptionPane.showMessageDialog(null, "Cancelando sistema...");
30            System.exit(0);
31        }
32
33    }
34 }
```

O Exemplo acima mostrou o uso prático do uso do `if-else` aninhado, simulando os dias da semana. Este exemplo realiza três validações. Assim como o exemplo anterior, este verifica se o usuário realmente entrou com um valor na caixa de diálogo, depois confere se o valor digitado é numérico, logo a seguir verifica se esse valor está entre 1 e 7 (a faixa de valores possíveis, uma vez que existem sete dias na semana)

Dica!

Cuidado ao comparar valores com `if` contidos em variáveis não primitivas como `String`. Lembre-se o tipo String guarda as informações por referência, ou seja, String é uma classe.

Vou resumir antes para depois explicar:

Devemos utilizar `.equals` para fazer comparações **dentro** do Heap da JVM pois o tipo `String` utiliza esta área.

Devemos utilizar `==` para fazer comparações **fora** do Heap da JVM utilizada por todas as variáveis primitivas.

Mas você deve estar se perguntando! O que é o Heap?

O **Heap** é um local reservado e protegido da JVM (Java Virtual Machine) onde ficam alocados todos os objetos instanciados durante a execução de seu programa. Cada um destes objetos possui um endereço de memoria onde está armazenado, e o único acesso a ele, é por via de uma referencia (armazenada fora do heap). Esta referencia, também possui o endereço de memoria, tornando o seu acesso único e exclusivo a este objeto.

Lembre-se que tipos primitivos (`int`, `double`, `float`) também ficam fora do HEAP com as referencias para objetos.

7.7 - Comparação utilizando ('==') operador igual.

Veja o código abaixo:

```
1.  int a = 1;
2.  int b = 1;
3.
4.  if (a == b) {
5.     //instrucao
6.  }
```

Quando você usa o operador "==" para comparar, você está comparando fora do heap. Aí tudo bem, é uma comparação externa ao heap da jvm, ou seja, vai comparar os tipos primitivos, ou seja, se o valor em 'a' é o mesmo valor em 'b'.
O resultando é claro, será t**rue.**

7.8 - Comparação utilizando .equals

É o inverso da '==' compara os objetos dentro do heap, ou seja suas características.

Eis um exemplo básico e autoexplicativo:

```
1  package br.com.Capitulo9.ApostilaEtec;
2  import java.util.Scanner;
3  public class Heap {
4
5      public static void main(String[] args) {
6          Scanner dado = new Scanner(System.in);
7          System.out.print("Forceça valor: ");
8          String aux = dado.next();
9
10         System.out.println();
11         System.out.println("Valor: " + aux);
12
13         if (aux.equals("56"))
14             System.out.println("Eita");
15     }
16 }
```

Neste caso a mensagem “Eita” será executada!

Eis mais um exemplo:

```
1.  String nome1 = "Lucas";
2.  String nome2 = "Jose";
3.
4.  if (nome1.equals(nome2)) {
5.     //isntrucao
6.  }
```

Seria false, pois o VALOR das 2 instâncias são diferentes.

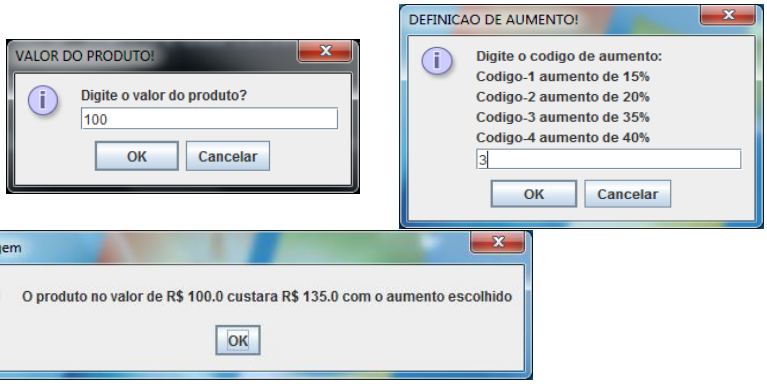
ATIVIDADES – 07

1) Crie uma classe que receba duas notas escolares de 0 a 10 e exiba na tela a média e se o status do aluno conforme a tabela abaixo

Faixa de Médias	Status Mensagem
De 0 à 3	Retido
Entre 3.1 à 6	Progressão Parcial
Entre 6.1 à 10	Aprovado!

2) Crie uma classe que receba dois valores (use `JOptionPane`) para receber o valor de um produto e um código de aumento, segundo a tabela abaixo:

CÓDIGO	%AUMENTO
1	15
2	20
3	35
4	40



3) Faça uma classe que receba três números inteiros em qualquer ordem e mostre o maior dentre eles, conforme indica a figura. Observação: para quebrar uma linha dentro da caixa de diálogo, utilize "\n".