UFABC

# BC0501 – Linguagens de Programação

*Aula Prática: 03* *Assunto:* **Comandos de Seleção**

## 1. Introdução

Um comando de seleção define uma condição em um programa, que permite que grupos de comandos sejam executados de maneira condicional, de acordo com o resultado da avaliação de um determinado teste (verdadeiro ou falso). Ou seja, programas utilizam comandos de seleção para escolher entre cursos alternativos de ações. As estruturas de seleção podem ser do tipo simples, composto ou encadeado.

A estrutura de seleção simples é utilizada para verificar se dada condição é atendida, se for, um conjunto de instruções deverá ser realizada; se não for, o fluxo de execução do algoritmo seguirá após o fim do bloco de decisão. Sua representação em Java:

```
if (condição){
      <conjunto de instruções>;
}
```

A estrutura de seleção composta prevê dois conjuntos de instruções para serem realizados de acordo com a condição: um conjunto que será realizado quando a condição resultar verdadeiro e um conjunto de instruções para resultado falso. Sua representação em Java:

```
if (condição){
      <conjunto de instruções A>;
} else{
      <conjunto de instruções B>;
}
```

Outra forma é a estrutura de seleção encadeada. Essa estrutura é uma sequência de testes de seleção, os quais serão executados ou não de acordo com o resultado das condições e com o encadeamento dos testes. Isto é, um teste de seleção pode ter dois conjuntos de instruções, conforme descrito em estruturas de seleção composta, um para resultado verdadeiro e outra para falso, porém esses conjuntos de instruções podem conter outros testes de seleção, que, por sua vez também podem conter outros. Sua representação em Java:

```
if (condição 1 ){
      if ( condição 2) {
            <conjunto de instruções A>;
      } else{
            <conjunto de instruções B>;
      }
} else{
      <conjunto de instruções C>;
}
```

Comandos de seleção são estruturas de controle básicas de qualquer linguagem de programação e, por isso, devem ser largamente estudadas e praticadas pelos alunos (ou seja, os alunos são aconselhados a resolverem vários tipos de problemas diferentes usando comandos de seleção).

## 2. Comandos de Seleção - Comparação

A comparação entre variáveis do tipo numérico pode ser feita utilizando a seguinte tabela de comandos muito parecidos com a linguagem matemática tradicional:

| Comparação | Comando JAVA |
|---|---|
| Igual ( = ) | = = |
| Diferente ( ≠ ) | != |
| Maior ( > ) | > |
| Menor ( < ) | < |
| Maior ou igual ( ≥ ) | >= |
| Menor ou igual (≤) | <= |

Além de comparar duas variáveis nós podemos comparar também expressões matemáticas. Exemplo:

```
if ((variável1 * 10.0 – 2.4) >= (variável2 / 2.1 + 3.0))
      < conjunto de instruções A >
else < conjunto de instruções B >
```

## 3. Atividade número 1 – Calcular a contribuição de imposto de renda

O objetivo desta atividade prática é calcular a contribuição de imposto de renda de contribuinte. O intuito é criar um programa que calcule a contribuição de IR baseado no salário do contribuinte. Uma das opções que será dada ao contribuinte é informar o seu salário mensal. Assim sendo será necessário fazer a conversão para o salário anual (multiplicando por 13). Finalmente, calcule o valor do imposto de renda a ser deduzido do salário anual. A alíquota de imposto de renda segue a tabela abaixo:

| Base de Cálculo em R$ | Alíquota % |
|---|---|
| Até 15.764,28 | 0,0 % |
| De 15.764,28 até 31.501,44 | 15% |
| Acima de 31.501,44 | 27,5% |

```java
package impostorenda;
import java.util.Scanner;

public class ImpostoRenda {
    public static void main(String[] args) {
        String NomeContribuinte;
        String CPFContribuinte;
        String salario;
        String pergunta1;
        double salarioinput;
        double imposto;

        //Entrada de Dados
        Scanner input=new Scanner(System.in);
        System.out.println("Digite o nome do contribuinte: ");
        NomeContribuinte = input.next();
        System.out.println("Digite o CPF do contribuinte: ");
        CPFContribuinte = input.next();
        System.out.println("O valor do salário é Anual (a) ou Mensal (m)? ");
        pergunta1 = input.next();
        System.out.println("Digite o valor do salário do contribuinte (em R$): ");
        salario = input.next();

        //Conversão de String para double
        salarioinput = Double.parseDouble(salario);

        if (pergunta1.equals("m")){
            salarioinput = salarioinput * 13;
        }

        if (salarioinput <=15764.28){
            imposto=0.0;
        } else if (salarioinput > 31501.44){
            imposto=(salarioinput-31501.44)*0.275 + 2360.57;
        } else {
            imposto = (salarioinput-15764.28)*0.15;
        }
        System.out.printf("Contribuinte:\n %s\n",NomeContribuinte);
        System.out.printf("CPF:\n %s \n",CPFContribuinte);
        System.out.printf("Total de imposto devido:\n R$ %.2f \n",imposto);;
    }
}
```

a) Salve, compile e execute o seu projeto. Alimente as variáveis e análise os resultados.

**Aumentando a complexidade da expressão booleana (operadores && e ||)**

Nesta seção será apresentado o uso de expressões booleanas mais complexas, usando os operadores && (E lógico) e || (OU lógico). A tabela verdade para esses operadores booleanos é:

E lógico

| p | q | $p \wedge q$ |
|---|---|---|
| V | V | V |
| V | F | F |
| F | V | F |
| F | F | F |

OU lógico

| p | q | $p \vee q$ |
|---|---|---|
| V | V | V |
| V | F | V |
| F | V | V |
| F | F | F |

Usando os operadores E e OU, é possível alterar parte do código para calcular o imposto de renda:

```
if (salarioinput <=15764.28){
       imposto=0.0;
} else if ((salarioinput > 31501.44) && (salarioinput <= 31501.44)){
       imposto=(salarioinput-31501.44)*0.275 + 2360.57;
} else if (salarioinput > 31501.44){
       imposto = (salarioinput-15764.28)*0.15;
}
```

b) Modifique o código do programa. Salve, compile e execute o seu projeto. Atribua valores as variáveis e analise os resultados.

c) Reescreva o programa de tal maneira que seja possível ler o número de dependentes. Antes do cálculo do imposto, calcule a nova base de cálculo do IR (subtraindo R$ 1.537,00 para cada dependente). Tanto a base de cálculo quanto os descontos devem armazenados em duas novas variáveis **BasedeCalculo** (double) e **Descontos** (double). Estes dois valores devem ser impressos juntamente com o imposto total cobrado.

## 4. Atividade número 2 – Determine a ordem crescente

Faça um programa em Java que leia três números inteiros e imprima-os em ordem crescente. Nesse programa será utilizada a estrutura de seleção encadeada para determinar a ordem entre os números.

```java
package ordemcrescente;
import java.util.Scanner;

public class OrdemCrescente {
   public static void main(String[] args) {
     int valor1;
     int valor2;
     int valor3;

     //Entrada de Dados
     Scanner input=new Scanner(System.in);
     System.out.println("Digite o primeiro valor: ");
     valor1 = input.nextInt();
     System.out.println("Digite o segundo valor: ");
     valor2 = input.nextInt();
     System.out.println("Digite o terceiro valor: ");
     valor3 = input.nextInt();

     if (valor1 > valor2){
        if (valor2 > valor3){
           System.out.printf("Segue a ordem crescente:\n %d %d %d",valor3,valor2,valor1);
        } else if (valor1 > valor3){
            System.out.printf("Segue a ordem crescente:\n %d %d %d",valor2,valor3,valor1);
        } else {
           System.out.printf("Segue a ordem crescente:\n %d %d %d", valor2, valor1, valor3);
        }
     } else if (valor1 > valor3){
        System.out.printf("Segue a ordem crescente:\n %d %d %d", valor3, valor1, valor2);
     } else if (valor2 > valor3){
        System.out.printf("Segue a ordem crescente:\n %d %d %d", valor1, valor3, valor2);
     } else {
        System.out.printf("Segue a ordem crescente:\n %d %d %d", valor1, valor2, valor3);
     }
   }
}
```

Salve, compile e execute o seu projeto. Analise os resultados. Modifique o código para apresentar os números na ordem decrescente.

## 5. Atividade número 3 – Determine o nome do dia da semana a partir de um número.

**Faça um programa em Java que um número inteiro e retorne o noem do dia da semana correspondente. Nesse programa será utilizada a estrutura de seleção múltipla *switch*.**

```java
package pkgswitch;
import java.util.Scanner;
```

```
public class Switch {
   public static void main(String[] args) {
        int dia = 1;

        System.out.print("Digite um número: ");
        Scanner input=new Scanner(System.in);
        dia = input.nextInt();

        // múltipla seleção. Só funciona com char, int, short ou byte
        switch (dia) {
            case 1:
                System.out.println("Domingo");
                break;
            case 2:
                System.out.println("Segunda-feira");
                break;
            case 3:
                System.out.println("Terça-feira");
                break;
            case 4:
                System.out.println("Quarta-feira");
                break;
            case 5:
                System.out.println("Quinta-feira");
                break;
            case 6:
                System.out.println("Sexta-feira");
                break;
            case 7:
                System.out.println("Sábado");
                break;
            default:
                System.out.println("Este não é um dia válido!");
         }
    }
}
```

## 6. Exercícios

6.1 Faça um programa em Java que pede como entrada o peso (em kilogramas) e a altura (em metros) da pessoa e calcula o IMC – Índice de Massa Corporal (IMC=massa/(altura . altura)). Adicionalmente, o programa deve emitir as mensagens correspondentes conforme a tabela a seguir:

Cálculo IMC:

| | |
|---|---|
| Abaixo de 18,5: | Você está abaixo do peso ideal |
| Entre 18,5 e 24,9: | Parabéns — você está em seu peso normal! |
| Entre 25,0 e 29,9: | Você está acima de seu peso (sobrepeso) |
| Entre 30,0 e 34,9: | Obesidade grau I |
| Entre 35,0 e 39,9: | Obesidade grau II |
| 40,0 e acima: | Obesidade grau III |

6.2 Faça um programa em Java que peça como entrada os coeficientes ***a***, ***b*** e ***c*** de uma equação de 2º grau e forneça como saída as suas raízes. Não se esqueça de prever os seguintes casos:

(1) $\boldsymbol{a = 0}$: Equação de 1º grau, então calcule a **única** raiz diretamente.

(2) $\boldsymbol{\Delta = 0}$: Calcule a **única** raiz pela fórmula de Báskara.

(3) $\boldsymbol{\Delta < 0}$: Calcule as duas raízes **complexas** pela fórmula de Báskara.

(4) $\boldsymbol{\Delta > 0}$: Calcule as duas raízes **reais** pela fórmula de Báskara.

6.3 Faça um programa em Java que receba a nota de um aluno de Processamento da Informação e retorne o conceito do aluno na disciplina. Classificação: A > 9,0; B > 7,5; C > 6,5; D > 5,0.

6.4 Faça um programa em Java que peça como entrada o conceito final de um aluno em uma disciplina cursada na UFABC e imprima a mensagem correta, conforme tabela abaixo:

CONCEITO FINAL SITUAÇÃO

A: Desempenho excepcional

B: Bom desempenho

C: Desempenho adequado

D: Aproveitamento mínimo

F: Reprovado

O: Reprovado por falta

I: Incompleto

6.5 Uma livraria está fazendo uma promoção para pagamento à vista em que o comprador pode escolher entre dois critérios de desconto:

Critério A: R$ 0,25 por livro + R$ 7,50 fixo.

Critério B: R$ 0,50 por livro + R$ 2,50 fixo.

Faça um programa em JAVA para o usuário digitar a quantidade de livros que deseja comprar e o programa diga qual é a melhor opção de desconto.