

# Aula 4



## Objetivos

- Tomadas de decisão
- Algoritmos;
- Estruturas de controle;
- A estrutura de seleção **if**;
- A estrutura de seleção **if/else**;
- A estrutura de repetição **while**.

# Aula 4



## Tomadas de decisão

Operadores de igualdade e operadores relacionais

Vamos introduzir uma versão simples da estrutura **if** de Java que permite a um programa tomar uma decisão com base na verdade ou falsidade de alguma condição. Se a condição for atendida (isto é, a condição for verdadeira), a instrução no corpo da estrutura **if** será executada. Se a condição não for atendida (isto é, a condição for falsa), a instrução no corpo não será executada.

As condições em estruturas **if** podem ser formadas utilizando-se operadores de igualdade e operadores relacionais:

# Aula 4



## Tomadas de decisão

### Operadores de igualdade

=

≠

em Java

==

!=

exemplo

x == y

x != y

### Operadores relacionais

>

<

≥

≤

>

<

>=

<=

x > y

x < y

x >= y

x <= y

# Aula 4



## Tomadas de decisão

Inverter os operadores `!=`, `>=` e `<=` como em `=!`, `=>` e `=<` é um erro de sintaxe.

Confundir o operador de igualdade `==` com o operador de atribuição, `=`, pode ser um erro de lógica ou de sintaxe. O operador de igualdade deve ser lido “é igual a” e o operador de atribuição deve ser lido “torna-se” ou “adquire o valor de”. Algumas pessoas preferem ler o operador de igualdade como “iguais duplos” ou “iguais iguais”

```
//Compara inteiros usando estrutura if e operadores relacionais
```

```
//Pacote de extensão Java
```

```
import javax.swing.JOptionPane;
```

```
public class Comparacao {
```

```
// main method begins execution of Java application
```

```
public static void main( String args[] )
```

```
{
```

```
String primeiroNumero; // primeira string de entrada do usuário
```

```
String segundoNumero; // segunda string de entrada do usuário
```

```
String resultado; // string que contem o resultado
```

```
int numero1; // primeiro número para comparação
```

```
int numero2; // segundo número para comparação
```

```
// lê o primeiro número digitado pelo usuário
```

```
primeiroNumero =
```

```
JOptionPane.showInputDialog( "Entre com o primeiro inteiro:" );
```

```
// lê o segundo número digitado pelo usuário
```

```
segundoNumero =
```

```
JOptionPane.showInputDialog( "Entre com o segundo inteiro:" );
```

```
// converte um número no formato string para int
```

```
numero1 = Integer.parseInt( primeiroNumero );
```

```
numero2 = Integer.parseInt( segundoNumero );
```

```
// inicializa a string de saída vázia
```

```
resultado = "";
```



```
resultado = "";
```

```
if ( numero1 == numero2 )  
    resultado = numero1 + " == " + numero2;
```

```
if ( numero1 != numero2 )  
    resultado = numero1 + " != " + numero2;
```

```
if ( numero1 < numero2 )  
    resultado = resultado + "\n" + numero1 + " < " + numero2;
```

```
if ( numero1 > numero2 )  
    resultado = resultado + "\n" + numero1 + " > " + numero2;
```

```
if ( numero1 <= numero2 )  
    resultado = resultado + "\n" + numero1 + " <= " + numero2;
```

```
if ( numero1 >= numero2 )  
    resultado = resultado + "\n" + numero1 + " >= " + numero2;
```

```
// Mostra o resultado
```

```
JOptionPane.showMessageDialog(  
    null, resultado, "Resultado de comparação",  
    JOptionPane.INFORMATION_MESSAGE );
```

```
System.exit( 0 ); // aplicação termina
```

```
} // fim do método main
```

```
} // fim da classe comparação
```



# Aula 4



## Tomadas de decisão

Procure não inicializar uma variável definida em método antes que a variável seja utilizada no corpo do método é um erro.

Recue a instrução no corpo de uma estrutura if para fazer o corpo da estrutura se destacar e para melhorar a legibilidade de programa.

Coloque apenas uma instrução por linha em um programa. Isso aprimora a legibilidade do programa.

# Aula 4



## Algoritmos

Qualquer problema de computação pode ser resolvido executando-se uma série de ações em uma ordem especificada. O procedimento para resolver um problema em termos:

- das ações a serem executadas;
- da ordem em que essas ações devem ser executadas.

Recebe o nome de algoritmo.

Considere o “algoritmo tomar café”, executado por um jovem que deseja tomar café numa xícara numa mesa com garrafa térmica e um açucareiro:

- pega uma xícara;
- coloca na mesa;
- pega a garrafa térmica que esta na mesa;

# Aula 4



## Algoritmos

- coloca o café na xícara;
- coloca de volta a garrafa térmica na mesa;
- pega a xícara de café;
- toma o café;
- coloca de volta a xícara na mesa.

Neste caso, o rapaz

# Aula 4



## Estrutura de controle

Normalmente, as instruções em um programa são executadas uma após a outra na ordem em que são escritas. Isso é chamado de execução sequencial. As instruções de controle em Java permitem aos programador especificar qual será a próxima instrução a ser executada que pode ser não próxima na sequência, que recebe o nome de transferência de controle.

A estrutura de seleção **if**.

Utiliza-se a estrutura de seleção para escolher entre cursos de ação alternativos em um programa. Por exemplo, suponha que a nota de aprovação em um exame seja 60 (em 100). Então podemos escrever em linguagem natural:

# Aula 4



## Estrutura de controle

A estrutura de seleção **if**.

*Se (If) a nota do aluno for maior do que ou igual a 60  
Imprimir “Aprovado”*

Em Java:

```
if (estudante >= 60)
    System.out.println("Aprovado");
```

# Aula 4



## Estrutura de controle

A estrutura de seleção **if/else**.

A estrutura de seleção **if** executa uma ação indicada quando a condição especificada é avaliada como **true**(verdadeira); caso contrário, a ação é pulada. A estrutura de seleção **if/else** permite especificar que, quando a condição for verdadeira, deve ser executada uma ação diferente da executada quando a condição for falsa.

*Se (If) a nota do aluno for maior do que ou igual a 60*

*Imprimir = “Aprovado”*

*senão (else)*

*Imprimir = “Reprovação”*

# Aula 4



## Estrutura de controle

A estrutura de seleção **if/else**.

Em Java:

```
if ( estudante >= 60 )  
    System.out.println(“Aprovado”);  
else  
    System.out.println(“Reprovado”);
```

# Aula 4



## Estrutura de controle

A estrutura de seleção **if/else**.

As estruturas aninhadas if/else testam múltiplos casos colocando estrutura if/else dentro de outras estruturas if/else.

# Aula 4



## Estrutura de controle

A estrutura de seleção **if/else**.

**Se a nota do aluno for maior do que ou igual a 90**

**Imprimir “A”**

**senão**

**Se a nota do aluno for maior do que ou igual a 80**

**Imprimir “B”**

**senão**

**Se a nota do aluno for maior do que ou igual a 70**

**Imprimir “C”**

**senão**

**Se a nota do aluno for maior do que igual ou igual a 60**

**Imprimir “D”**

**senão**

**Imprimir “F”**

# Aula 4



## Estrutura de controle

A estrutura de seleção **if/else**.

Em Java

```
if (notaEstudante >= 90)  
    System.out.println("A");  
else  
    if (notaEstudante >= 80)  
        System.out.println("B");  
    else  
        if (notaEstudante >= 70)  
            System.out.println("C");  
        else  
            if (notaEstudante >= 60)  
                System.out.println("D");  
            else  
                System.out.println("E");
```

# Aula 4



## Estrutura de controle

A estrutura de seleção **if/else**.

Outra forma em Java

```
if (notaEstudante >= 90)  
    System.out.println("A");  
else if (notaEstudante >= 80)  
    System.out.println("B");  
else if (notaEstudante >= 70)  
    System.out.println("C");  
else if (notaEstudante >= 60)  
    System.out.println("D");  
else  
    System.out.println("F");
```

# Aula 4



## Estrutura de controle

A estrutura de repetição while.

A estrutura de repetição especificar que uma ação deve ser repetida enquanto alguma condição permanecer verdadeira.

*Enquanto (While) houver mais itens em minha lista de compras*

*Comprar o próximo item e riscá-lo de minha lista*

Essa ação será executada repetidamente enquanto a condição permanecer verdadeira. A(s) instrução(ões) contidas(s) na estrutura de repetição while constitui(em) o corpo da estrutura while. O corpo da estrutura while pode ser uma instrução única ou um bloco. Em algum momento, a condição se tornará falso (quando o último item na lista de compras for comprado e riscado da lista. Neste ponto, a repetição termina e a primeira instrução depois da estrutura é executada.

```
// Programa da média da classe utilizando repetição controlada por contador
```

```
// Pacote de extensão Java
```

```
import javax.swing.JOptionPane;
```

```
public class Media {
```

```
// Método main inicia a execução da aplicação Java
```

```
public static void main( String args[] )
```

```
{
```

```
    int total,      // soma as notas inseridas pelo usuário
```

```
        contadorNota, // número de notas inseridas
```

```
        valorNota,   // valor da nota
```

```
        media;      // média total das notas
```

```
    String nota;    // tipo de nota do usuário
```

```
// Initialization Phase
```

```
total = 0;      // limpa total
```

```
contadorNota = 1; // preparação para loop
```

```
// Fase de processamento
```

```
while ( contadorNota <= 5 ) { // repetição de 10 vezes
```

```
    // entrada de notas pelo usuário
```

```
    nota = JOptionPane.showInputDialog(  
        "Entre com uma nota: " );
```

```
    // converte nota de literal para inteiro
```

```
    valorNota = Integer.parseInt( nota );
```

```
    // adiciona ao valor da nota no total
```

```
    total = total + valorNota;
```

```
    // adiciona 1 ao contador de nota
```

```
    contadorNota = contadorNota + 1;
```





```
} // fim da estrutura de repetição

// Fase de término
media = total / 5; // realiza a divisão do inteiro

// mostra média da relação de notas
JOptionPane.showMessageDialog( null,
    "A média da classe é " + media, "Média da classe",
    JOptionPane.INFORMATION_MESSAGE );

System.exit( 0 ); // termina o programa

} // fim do método main

} // fim da classe Media
```