Desenvolvimento de Sistemas

Cronograma da Aula

- Estruturas de Decisão
- Estruturas de Repetição

Programação com Decisões

Para que seja possível fazer um programa tomar decisões, é necessário primeiramente imputar uma condição. Uma condição é o estabelecimento de uma relação lógica entre variável versus variável ou entre variável e constante por meio de um operador relacional.

Operadores Relacionais

Símbolo	Significado
==	Igual a
!=	Diferente de (não igual)
>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a

Desvio Condicional Simples

Um desvio condicional será simples, quando houver uma condição que desvia a execução do programa, caso o resultado lógico avaliado seja verdadeiro. Se o resultado lógico avaliado for falso, nada acontece, e o programa simplesmente segue o seu fluxo de execução.

Desvio Condicional Simples

Sintaxe na Linguagem Java:

```
If (condiçãoBooleana){
        <instrução executada quando a condição for verdadeira>;
}
<instrução executada após cond. verd. e/ou falsa >;
```

Desvio Condicional Simples

Exemplo

```
2   package javaexemplo5;
3
4    //@author Rodrigo
5
6   import java.util.Scanner;
7   public class Main
8   {
9       public static void main(String[] args)
10      {
11          int a,b,r;
12          Scanner s=new Scanner(System.in);
13          System.out.println();
14          System.out.println("ENTRE COM O VALOR DE A: ");
15          a=s.nextInt();
16          System.out.println("ENTRE COM O VALOR DE B: ");
17          b=s.nextInt();
18          r=a+b;
19          System.out.println();
20          if (r>10)
21              System.out.println("RESULTADO = "+r);
22          System.out.println();
23      }
24  }
25
```

Desvio Condicional Composto

Numa instrução **if...else** se a condição for verdadeira, será executada a instrução que estiver posicionada entre a instrução **if** e a **else**. Sendo a condição falsa, será executada a instrução que estiver posicionada logo após a instrução **else**.

Desvio Condicional Composto

Sintaxe na Linguagem Java:

```
If (condiçãoBooleana){
        <instrução executada quando a condição for verdadeira>;
}
else{
        <instrução executada quando a condição for falsa>;
}
<instrução executada após cond. verd. ou falsa >;
```

Desvio Condicional Composto

Exemplo

```
1   package javaexemplo6;
2
3   import java.util.Scanner;
4
5   public class Main
6   {
7       public static void main(String[] args)
8       {
9           double a,b,r;
10          Scanner s=new Scanner(System.in);
11          System.out.println();
12          System.out.print("ENTRE COM O NOTA 1: ");
13          a=s.nextDouble();
14          System.out.print("ENTRE COM O NOTA 2: ");
15          b=s.nextDouble();
16          r=(a+b)/2;
17          System.out.println();
18          if (r>=7)
19              System.out.println("O ALUNO ESTÁ APROVADO");
20          else
21              System.out.println("O ALUNO ESTÁ DE EXAME/REPROVADO");
22          System.out.println();
23      }
24  }
25
```

Desvio Condicional Encadeado

Sintaxe na Linguagem Java:

```
If (condição1){
        <instrução executada quando a condição1 for verdadeira>;
}else if (condição2){
        <instrução executada quando a condição2 for verdadeira>;
}else if (condição3){
        <instrução executada quando a condição3 for verdadeira>;
}else{
        <instrução executada quando todas as condições forem falsas>;
}
```

Desvio Condicional Encadeado

Exemplo

```
1
2   package javaexemplo7;
3
4   import java.util.Scanner;
5
6   public class Main
7   {
8       public static void main(String[] args)
9       {
10          int mes;
11          Scanner s=new Scanner(System.in);
12          System.out.println();
13          System.out.println("ENTRE COM UM VALOR REFERENTE A UM MÊS ");
14          mes=s.nextInt();
15          System.out.println();
16          if(mes==1)
17              System.out.println("JANEIRO");
18          else if(mes==2)
19              System.out.println("FEVEREIRO");
20          else if(mes==3)
21              System.out.println("MARÇO");
22          else if(mes==4)
23              System.out.println("ABRIL");
24          else if(mes==5)
25              System.out.println("MAIO");
```

```
26              else if(mes==6)
27                  System.out.println("JUNHO");
28              else if(mes==7)
29                  System.out.println("JULHO");
30              else if(mes==8)
31                  System.out.println("AGOSTO");
32              else if(mes==9)
33                  System.out.println("SETEMBRO");
34              else if(mes==10)
35                  System.out.println("OUTUBRO");
36              else if(mes==11)
37                  System.out.println("NOVEMBRO");
38              else if(mes==12)
39                  System.out.println("DEZEMBRO");
40              else
41                  System.out.println("MÊS INVÁLIDO");
42              System.out.println();
43          }
44      }
45
```

Estrutura de Controle com Múltipla escolha

Estrutura switch, Sintaxe:

```
switch <variável>
{
        case <opção1:><operação 1>; break;
        case <opção2:><operação 2>; break;
        case <opçãoN:><operação N>; break;
        default         :<operação default>break;
}
```

Estrutura de Controle com Múltipla escolha

Exemplo

```
3  import java.util.Scanner;
4  public class Main
5  {
6      public static void main(String[] args)
7      {
8          int mes;
9          Scanner s=new Scanner(System.in);
10         System.out.println();
11         System.out.println("ENTRE COM UM VALOR REFERENTE A UM MÊS ");
12         mes=s.nextInt();
13         System.out.println();
14         switch (mes)
15         {
16             case 1: System.out.println("JANEIRO");break;
17             case 2: System.out.println("FEVEREIRO");break;
18             case 3: System.out.println("MARÇO");break;
19             case 4: System.out.println("ABRIL");break;
20             case 5: System.out.println("MAIO");break;
21             case 6: System.out.println("JUNHO");break;
22             case 7: System.out.println("JULHO");break;
23             case 8: System.out.println("AGOSTO");break;
24             case 9: System.out.println("SETEMBRO");break;
25             case 10: System.out.println("OUTUBRO");break;
26             case 11: System.out.println("NOVEMBRO");break;
27             case 12: System.out.println("DEZEMBRO");break;
28             default: System.out.println("MÊS INVÁLIDO");break;
29         }
30         System.out.println();
31     }
32 }
33
```

Operadores lógicos

Existem ocasiões em que é necessário trabalhar com o relacionamento de mais de uma condição para se tomar decisão.

Operador Lógico	Função
&&	Operador de conjunção
II	Operador de disjunção
^	Operador de disjunção exclusiva
!	Operador de negação

Operador lógico && (e)

O operador lógico de conjunção && é utilizado quando dois relacionamentos lógicos de uma determinada condição necessitam ser verdadeiros para obter-se um resultado lógico verdadeiro.

Operador Lógico &&		
Condição 1	Condição 2	Resultado
Falsa	Falsa	Falso
Verdadeira	Falsa	Falso
Falsa	Verdadeira	Falso
Verdadeira	Verdadeira	Verdadeiro

Operador lógico && (e)

Exemplo

```
1
2   package javaexemplo9;
3
4   import java.util.Scanner;
5   import javax.swing.JOptionPane;
6
7   public class Main
8   {
9       public static void main(String[] args)
10      {
11          String usuario, senha;
12          Scanner s = new Scanner(System.in);
13          //entrada de dados
14          System.out.print("DIGITE O USUÁRIO: ");
15          usuario=s.nextLine();
16          System.out.print("DIGITE  A SENHA: ");
17          senha=s.nextLine();
18          if(("Rodrigo".equals(usuario))&&("123456".equals(senha)))
19          {
20              JOptionPane.showMessageDialog(null,
21                      "BEM VINDO "+usuario);
22          }
23         else
24          {
25              JOptionPane.showMessageDialog(null,
26                      "USUÁRIO OU SENHA INVÁLIDOS...");
27          }
28      }
29  }
30
```

Operador lógico || (ou)

O operador lógico de disjunção || é utilizado quando pelo menos um dos relacionamentos lógicos de uma condição necessita ser verdadeiro para obter um resultado lógico verdadeiro.

Operador Lógico ||		
Condição 1	Condição 2	Resultado
Falsa	Falsa	Falso
Verdadeira	Falsa	Verdadeiro
Falsa	Verdadeira	Verdadeiro
Verdadeira	Verdadeira	Verdadeiro

Operador lógico || (ou)

Exemplo

```
1
2   package javaexemplo10;
3
4   import java.util.Scanner;
5
6   public class Main
7   {
8       public static void main(String[] args)
9       {
10          int CODIGO;
11          Scanner s = new Scanner(System.in);
12          System.out.println();
13          System.out.print("Entre o codigo de acesso: ");
14          CODIGO = s.nextInt();
15          if (CODIGO == 1 || CODIGO == 2 || CODIGO == 3)
16          {
17              if (CODIGO == 1)
18                  System.out.println("um");
19              if (CODIGO == 2)
20                  System.out.println("dois");
21              if (CODIGO == 3)
22                  System.out.println("tres");
23          }
24          else
25              System.out.println("codigo invalido");
26
27          System.out.println();
28      }
29
30  }
31
```

Operador lógico de Negação !

O operador lógico de negação ! é utilizado quando é necessário estabelecer que uma determinada condição deve ser não verdadeira ou deve ser não falsa.

Operador Lógico !	
Condição	Resultado
Verdadeiro	Falso
Falso	Verdadeira

Operador lógico de Negação !

Exemplo

```
1
2   package javaexemplo12;
3
4   import java.util.Scanner;
5
6   public class Main
7   {
8       public static void main(String[] args)
9       {
10          int A, B, C, X;
11          Scanner s = new Scanner(System.in);
12          System.out.println();
13          System.out.print("Entre o valor <A>: ");
14          A = s.nextInt();
15          System.out.print("Entre o valor <B>: ");
16          B = s.nextInt();
17          System.out.print("Entre o valor <C>: ");
18          C = s.nextInt();
19          if (!(C > 5))
20              X = (A + B) * C;
21          else
22              X = (A - B) * C;
23          System.out.println("O resultado de X equivale a: "
24                  + X);
25          System.out.println();
26      }
27  }
28
```

Exercícios

1- Faça um programa para ler dois valores numéricos inteiros e apresentar o resultado da diferença do maior em relação ao menor valor.

2- Faça um programa para ler quatro valores reais referente a 4 notas, calcular a média do aluno e mostrar se o aluno foi aprovado ou reprovado, considere a média escolar maior ou igual a 5.

Exercícios

3- Faça um programa para ler 3 valores inteiros e apresentá-los em ordem crescente.

4- Faça um programa para ler 4 valores inteiros e apresentar somente aqueles que forem divisíveis por 2 e 3.

Programação com Laços de Repetição

Também conhecidos como loopings ou malhas de repetição, que possibilitam a ação de repetir um determinado trecho de programa várias vezes. É possível determinar repetições com números variados de vezes, desde laços finitos até laços indeterminados.

Laço com verificação condicional inicial

É do tipo enquanto (while), que efetua um teste lógico no início de sua execução, verificando se é permitido executar o trecho de instruções subordinadas a esse laço.

while - Sintaxe

While (condição)

{

instrução (ões) executada enquanto a condição for verdadeira

}

while

Exemplo

```
1   package javaexemplo14;
2
3   import java.util.Scanner;
4   public class Main
5   {
6       public static void main(String[] args)
7       {
8           int i,qtd=0, resp=1;
9           Scanner s=new Scanner(System.in);
10          while(resp==1)
11          {
12              qtd++;
13              System.out.println();
14              i = 1;
15              while (i <= 5)
16              {
17                  System.out.println("Valor = " + i);
18                  i++; //i=i+1
19              }
20              System.out.println();
21              System.out.println("\nQuantidade de Execuções: "+qtd);
22              System.out.println("Deseja Continuar?");
23              System.out.print("Tecle[1] para SIM / [2] para NÃO: ");
24              System.out.println();
25              resp=s.nextInt();
26          }
27      }
28  }
29
```

Laço com verificação condicional final

O laço de repetição com a verificação condicional no final realiza um teste lógico no final de um laço. Sendo parecida com a estrutura **while (enquanto for)**, a estrutura em questão é denominada **do...while (faça enquanto)**

do...while - Sintaxe

```
do
{
        instrução (ões) executada
enquanto a condição for verdadeira
}
while (condição);
```

do...while

Exemplo

```
2   package javaexemplo15;
3
4   import java.util.Scanner;
5
6   public class Main
7   {
8       public static void main(String[] args)
9       {
10          int I,qtd=0, RESP = 1;
11          Scanner s = new Scanner(System.in);
12          do
13          {
14              System.out.println();
15              I = 1;
16              qtd++;
17              do
18              {
19                  System.out.println("Valor = " + I);
20                  I++;
21              }
22              while (I <= 5);
23          System.out.println();
24          System.out.println("\nQuantidade de Execuções: "+qtd);
25          System.out.println("Deseja continuar?");
26          System.out.print("Tecle: [1] para SIM / [2] para NAO: ");
27          RESP = s.nextInt();
28          }
29          while (RESP == 1);//termina aqui
30      }
31  }
32
```

Laço com variável de controle (incondicional)

Os laços que possuem um número finito de execuções podem ser processados por meio de uma laço de repetição executado pelo método **for()**, que tem seu funcionamento controlado por variável de controle do tipo contador, podendo ser crescente ou decrescente.

for - Sintaxe

```
for (início; fim; incremento ou decremento)
{
    instruções 1;
    instruções 2;
    instruções N;
}
```

for

Exemplo

```
1
2   package javaexemplo16;
3
4   import java.util.Scanner;
5
6   public class Main
7   {
8       public static void main(String[] args)
9       {
10      int I,resp=1,qtd=0;
11      System.out.println();
12      Scanner s = new Scanner(System.in);
13      while (resp==1)
14      {
15         qtd++;
16         for (I = 1; I <= 5; I++)
17              System.out.println("Valor = " + I);
18
19         System.out.println();
20          System.out.println("\nQuantidade de Execuções: "+qtd);
21          System.out.println("Deseja continuar?");
22          System.out.print("Tecle: [1] para SIM / [2] para NAO: ");
23          resp = s.nextInt();
24      }//termina aqui
25      }
26
27  }
28
```

for

Exemplo 2

```
6   public class Main
7   {
8       public static void main(String[] args)
9       {
10          int N, I, R,resp=1;
11          while (resp==1)
12          {
13              Scanner s = new Scanner(System.in);
14              System.out.println();
15              System.out.println("Programa: TABUADA");
16              System.out.println();
17              System.out.print("Entre um valor inteiro para o calculo: ");
18              N = s.nextInt();
19              System.out.println();
20              for (I = 1; I <= 10; I++)
21              {
22                  // Cálculo da tabuada propriamente dito
23                  R = N * I;
24                  // Apresentação da tabuada no formato: N X I = R
25                  System.out.println(N+" X "+I+" = "+R);
26              }
27              System.out.println();
28              System.out.println("Deseja continuar?");
29              System.out.print("Tecle: [1] para SIM / [2] para NAO: ");
30              resp = s.nextInt(); //até aqui
31          }
32      }
33  }
34
```

Controlando Loops

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;
    }
}
```

O código acima vai percorrer os números de x a y e parar quando encontrar um número divisível por 19, uma vez que foi utilizada a palavra chave **break**.

Controlando Loops

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

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.

Escopo das variáveis

```
// 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
```

No bloco acima, a variável j para de existir quando termina o bloco onde ela foi declarada. Se você tentar acessar uma variável fora de seu escopo, ocorrerá um erro de compilação.

Escopo das variáveis

O mesmo vale para um if:

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

Aqui a variável i não existe fora do if e do else! Se você declarar a variável antes do if, vai haver outro erro de compilação: dentro do if e do else a variável está sendo redeclarada! Então o código para compilar e fazer sentido fica:

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

Exercícios

1- Apresentar os quadrados dos números inteiros de 15 a 200.

2- Apresentar o total da soma obtida dos cem primeiros números inteiros, representados pela sequência 1+2+3+4+5+6+7+...+97+98+99+100.

3- Apresentar todos os números divisíveis por 4 que sejam menores que 30. Iniciar a contagem com o valor 1.

Referências desta aula

- MANZANO, José Augusto N.G. Java 7 Programação de Computadores
- DEITEL, Harvey M.; DEITEL, Paul J. Java: como programar 6.ed, Prentice-Hall, 2005.
- FJ11: Java e Orientação a Objetos. Caelum: ensino e inovação.

FIM

OBRIGADO

RODRIGO