Programação Avançada II

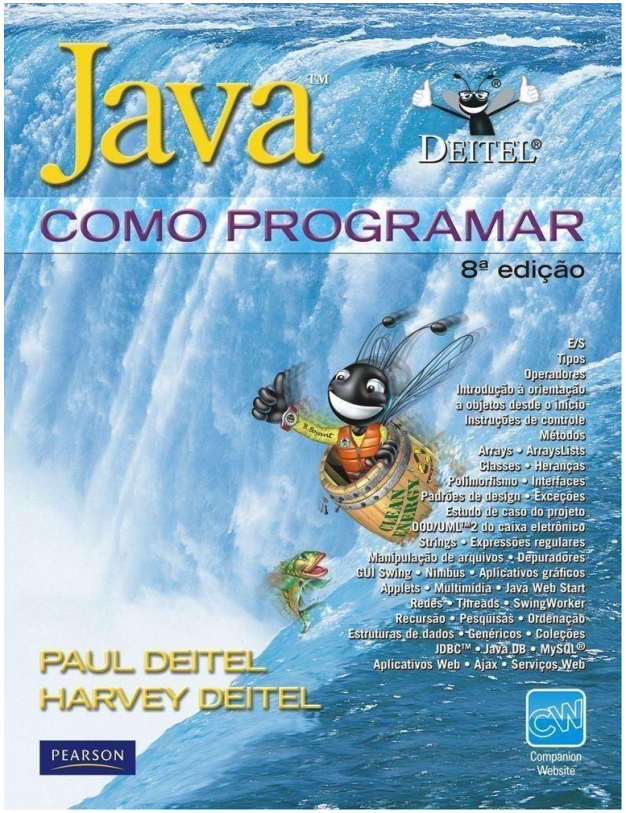
PROF. ISAAC

Bibliografia

- Deitel, Harvey M.; **Java : como programar**. 6. ed. Pearson,. 2005.

Bibliografia

- **Java Como Programar** - 8ª Ed. Deitel & Deitel



Bibliografia

Programação com Java: Uma Introdução Abrangente. Herbert Schildt & Dale Skrien

Bibliografia Complementar

- Design Patterns com Java - Projeto Orientado a Objetos Guiado por Padrões. Eduardo Guerra.

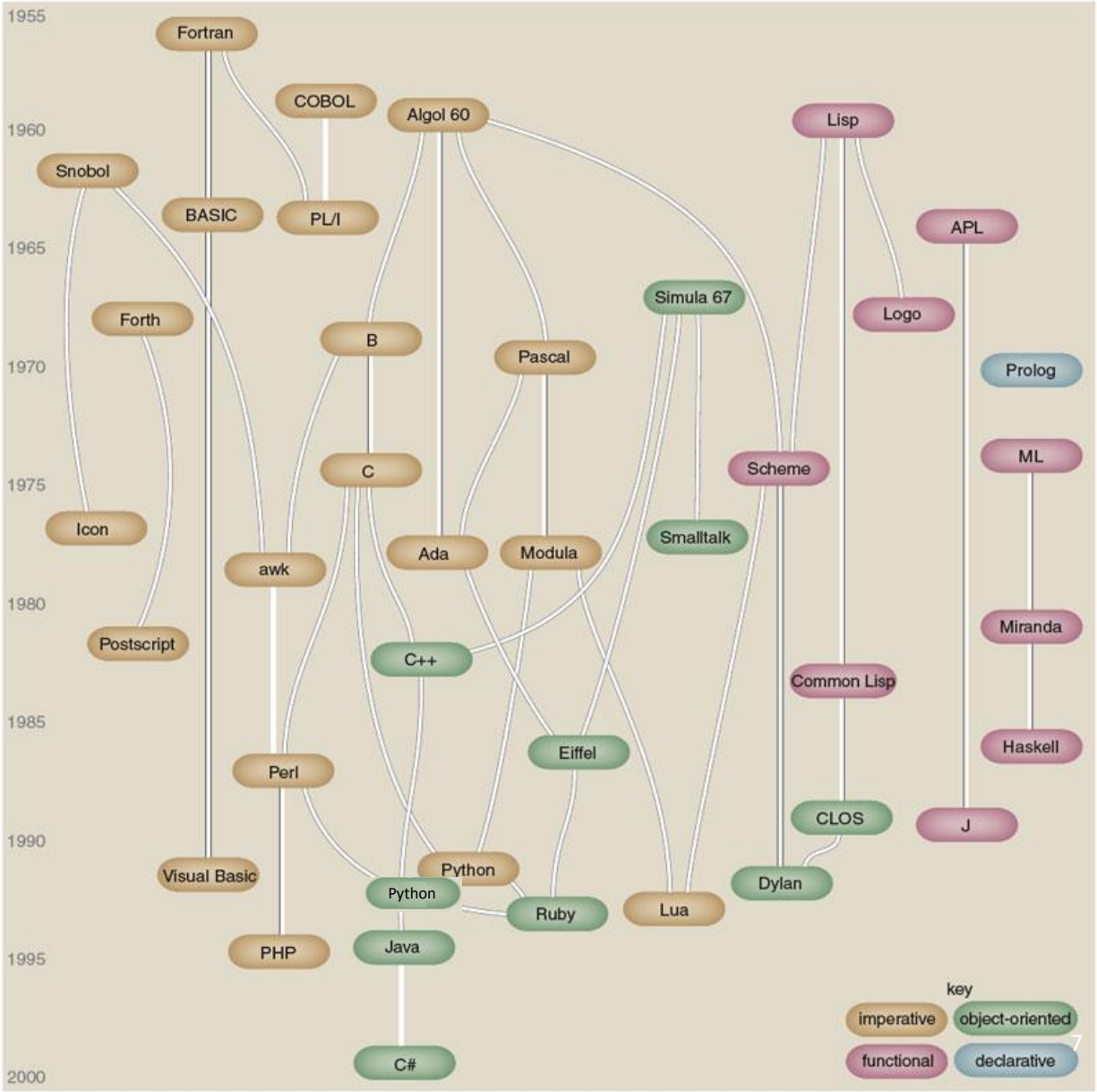
- UML 2 - Uma Abordagem Prática - 2ª Edição

Gilleanes T. A. Guedes

2012

Introdução ao Java

Genealogia das principais linguagens de programação



Programação Orientada a Objetos

Programação Orientada a Objetos (POO) é um paradigma de programação no qual pode-se abstrair um programa como uma coleção de objetos que interagem entre si.

O conceito formal de Orientação a Objetos foi introduzido em meados de 1960, com a linguagem Simula 67 (Centro Norueguês de Computação em Oslo).

O desenvolvimento completo da POO veio com o Smalltalk 80 (1980).

Algumas Linguagens Orientadas a Objetos

The Top Programming Languages - IEEE 2020

Rank	Language	Type	Score
1	Python	web, desktop, embedded	100.0
2	Java	web, mobile, desktop	95.3
3	C	mobile, desktop, embedded	94.6
4	C++	mobile, desktop, embedded	87.0
5	JavaScript	web	79.5
6	R	desktop	78.6
7	Arduino	embedded	73.2
8	Go	web, desktop	73.1
9	Swift	mobile, desktop	70.5
10	Matlab	desktop	68.4

Fonte: https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2020

Declaração de variáveis

- As variáveis são declaradas após a especificação de seus tipos
- Os tipos de dados mais utilizados são:
  - **int:** para números inteiros
  - **float** e **double**: para números reais
  - **char**: caracter
  - **String**: para textos
  - **boolean**: para verdadeiro ou falso

Saída de Dados no console

Usando o método **“printf”,** para obter uma saída formatada:

System.out.printf(“ %s a FEI ”, “Bem Vindo”);

E para concatenar valores, utilizamos o sinal “**+**”:

String x = “Fulano”;

System.out.printf("Olá %s ! “, x);

Saída de Dados no console

Usando o método “printf”, para obter uma saída formatada:

System.out.printf(“Nome: %s\nIdade: %d\nAltura: %f\n”, “Maria”, 19, 1.70);

Saída de Dados no console

Usando o método “printf”, para obter uma saída formatada:

```
6   package javaapplication3;
7
8   public class JavaApplication3 {
9       public static void main(String[] args) {
10          System.out.printf("Nome: %s\nIdade: %d\nAltura: %f\n", "Maria", 19, 1.70);
11      }
12
13  }
```

```
Saída - JavaApplication3 (run)
run:
Nome: Maria
Idade: 19
Altura: 1,700000
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Saída de Dados no console

Usando o método “printf”, para obter uma saída formatada:

```
String nome = “Maria”;
int idade = 19;
double altura = 1.69;
System.out.printf( “Nome: %s\nIdade: %d\nAltura: %f\n”, nome, idade, altura );
```

Saída de Dados no console

Usando o método “printf”, para obter uma saída formatada:

```
8   public class JavaApplication3 {
9       public static void main(String[] args) {
10          String nome = "Maria";
11          int idade = 19;
12          double altura = 1.69;
13          System.out.printf("Nome: %s\nIdade: %d\nAltura: %f\n", nome, idade, altura);
14
15      }
16  }
17
```

```
Saída - JavaApplication3 (run) ×
run:
Nome: Maria
Idade: 19
Altura: 1,690000
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Entrada de dados

- A entrada de dados pode ser feita pela classe **Scanner**
- Todas as entradas são feitas como texto, que devem ser convertidos conforme a necessidade:
  - ***nextInt(); nextLine(); nextDouble()***
- a classe Scanner precisa ser importada com o comando: ***java.util.Scanner***

Importação de pacotes

- Para se importar pacotes, utiliza-se a palavra ***import***, seguida do nome do pacote, no início do programa.
- Exemplos:
  - ***import java.util.Scanner;***
  - ***import java.lang.Math;***

Outro Exemplo simples com inteiros

```java
// Fig. 2.7: Addition.java
// Programa de adicao que exibe a soma de dois numeros.
import java.util.Scanner; // programa utiliza a classe Scanner

public class Addition
{
   // metodo principal inicia a execucao do aplicativo Java
   public static void main()
   {
      // cria Scanner para obter entrada a partir da janela de comando
      Scanner input = new Scanner( System.in );

      int number1; // primeiro numero a somar
      int number2; // segundo numero a adicionar
      int sum; // soma de number1 e number2

      System.out.print( "Enter first integer: " ); // prompt
      number1 = input.nextInt(); // le primeiro o numero fornecido pelo usuario

      System.out.print( "Enter second integer: " ); // prompt
      number2 = input.nextInt(); // le o segundo numero fornecido pelo usuario

      sum = number1 + number2; // soma os numeros

      System.out.printf( "Sum is %d\n", sum );

   } // fim do metodo principal

} // fim da classe Addition
```

```
perico@nuc:~/Dropbox/CC3642 - Orientação a Objetos$ javac Addition.java
perico@nuc:~/Dropbox/CC3642 - Orientação a Objetos$ java Addition
Enter first integer: 5
Enter second integer: 6
Sum is 11
perico@nuc:~/Dropbox/CC3642 - Orientação a Objetos$
```

Operadores Aritméticos

+	Soma
-	Subtração
*****	Multiplicação
/	Divisão
%	Módulo: retorna o resto da divisão

Operadores de atribuição compostos aritméticos

Operador	uso	Função	Exemplo
+=	variável += valor	variável = variável + valor	A+=2
-=	variável -= valor	variável = variável - valor	A-=2
*=	variável *= valor	variável = variável * valor	A*=2
/=	variável /= valor	variável = variável / valor	A/=2
%=	variável %= valor	variável = variável % valor	A%=2

Obs: o valor pode ser outra variável, exemplo: variável01+=variável02

Operadores Relacionais

operador	operação	Símbolo matemático
==	Igualdade	=
>	Maior que	>
<	Menor que	<
!=	diferente	≠
>=	Maior ou igual	≥
<=	Menor ou igual	≤

Classe math

- Todos os métodos da classe Math são static:

  Exemplo: Math.sqrt(25)

- A classe Math faz parte do pacote java.lang (implicitamente importado pelo compilador) e, portanto, não é necessário importar a classe Math para utilizar seus métodos.

  Math.PI e Math.E são campos final static da classe Math.

Classe math

abs(x)	Valor absoluto de x
sqrt(x)	Raiz quadrada de x
log(x)	Retorna o logaritmo natural de x (base e)
log10(x)	Retorna o logaritmo base-10 de x
sin(x)	Retorna o seno de x radianos
cos(x)	Retorna o cosseno de x radianos
exp(x)	Retorna e^x
ceil(x)	Arredonda x para o menor inteiro não menor que x
floor(x)	Arredonda x para o maior inteiro não maior que x
max(x,y)	Maior valor de x e y
min(x,y)	Menor valor de x e y
pow(x,y)	X elevado à potência de de y

Estrutura de Seleção

if … else if … else

```
if (condição){
        …
}
else if (condição 2){
        …
}
else{
        …
}
```

```
if ( nota >= 9 )
  System.out.println( "A" );
else if ( nota >= 8 )
  System.out.println( "B" );
else if ( nota >= 7 )
  System.out.println( "C" );
else if ( nota >= 6 )
  System.out.println( "D" );
else
  System.out.println( "F" );
```

Exemplo: Verificar se um número é par

- O programa deve pedir para o usuário digitar um número.
- O programa deve informar se este número é par ou impar.

Exemplo: Verificar se um número é par

```
8  import java.util.Scanner;
9
10 public class JavaApplication3 {
11     public static void main(String[] args) {
12         Scanner input = new Scanner(System.in);
13
14         System.out.println("Digite um número: ");
15         int number = input.nextInt();
16         if(number%2 == 0)
17         {
18             System.out.println("O número é PAR");
19         }
20         else
21         {
22             System.out.println("O número é IMPAR");
23         }
24     }
25 }
```

```
Saída - JavaApplication3 (run)
run:
Digite um número:
4
O número é PAR
CONSTRUÍDO COM SUCESSO (tempo total: 6 segundos)
```

Operadores Lógicos

Operadores Lógicos

- Podemos combinar condições para determinar como continuar o fluxo de um programa!
- Os operadores lógicos permitem a construção de condições mais complexas.
- Os operadores lógicos em Java são:
  - **&&** (E condicional)
  - **||** (OU condicional)
  - **&** (E lógico booleano)
  - **|** (OU inclusivo lógico booleano)
  - **^** (OU exclusivo lógico booleano)
  - **!** (NÃO lógico)

Operadores Lógicos

Operador E condicional (&&)

- Avalia o próximo operando apenas se o operando anterior for verdadeiro.

$$S = A \,\&\&\, B$$

S	A	B
False	False	False
False	False	True
False	True	False
True	True	True

Operador E lógico booleano (&)

- Funciona de maneira idêntica a && exceto que sempre avalia ambos os operandos

Operadores Lógicos

Operador OU condicional (||)

- Avalia o próximo operando apenas se o operando anterior for falso.

$$S = A \,||\, B$$

S	A	B
False	False	False
True	False	True
True	True	False
True	True	True

Operador OU inclusivo lógico booleano (|)

- Funciona de maneira idêntica a || exceto que sempre avalia ambos os operandos

Operadores Lógicos

OU exclusivo lógico booleano

$$S = A \wedge B$$

S	A	B
False	False	False
True	False	True
True	True	False
False	True	True

Operador de negação lógica ! (NÃO lógico)

$$S = !A$$

S	A
True	False
False	True

Exemplo: Verificar aprovação

- O programa deve pedir para o usuário digitar a média do Aluno.
- O programa deve informar se o aluno foi aprovado ou está de P3.
- O programa não deve aceitar notas maiores que 10 e menores que 0.

Exemplo: Verificar aprovação

```
import java.util.Scanner;

public class Principal {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.printf("Digite a média do aluno: ");
        double nota = input.nextDouble();
        if(nota <= 10 && nota >=5){
            System.out.println("Aluno Aprovado");
        } else if (nota < 5 && nota >=0){
            System.out.println("Aluno deve fazer P3");
        } else{
            System.out.println("Nota inválida");
        }
    }
}
```

Precedência dos Operadores

Ordem	Operador	Função
1º	( )	Parênteses que agrupam expressões
2º	*, /	Operadores aritméticos de multiplicação e divisão
3º	+, -	Operadores aritméticos de adição e subtração
4º	=	Operador de atribuição
5º	>, <, ==, >=, <=, !=	Operadores relacionais
6º	!	Operador lógico de negação
7º	&&	Operador lógico E
8º	| |	Operador lógico OU

Operador Condicional Ternário

Operador Condicional (?:)

Operador ternário do Java (recebe 3 operandos).

? : e seus três operandos formam uma expressão condicional.

- Se a expressão do primeiro operando for true será executado o segundo operando.
- Se a expressão do primeiro operando for false será executado o terceiro operando.

Sintaxe:

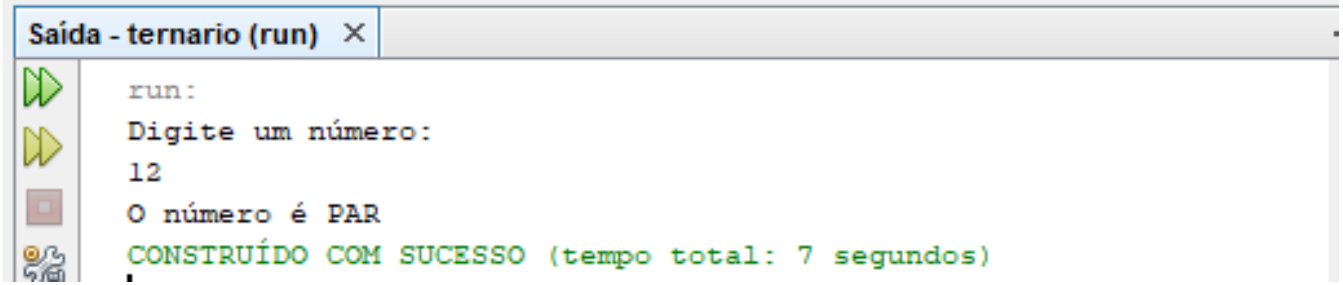
Condição ? se verdadeiro : se falso

Operador Condicional (?:)

EXEMPLO: Verificar se um número é par.

```java
import java.util.Scanner;

public class Principal {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Digite um número: ");
        int numero = input.nextInt();
        String texto = numero%2==0 ? "PAR" : "IMPAR";
        System.out.println("O número é " + texto);
    }
}
```



Estrutura de Seleção múltipla switch

Estrutura de Seleção múltipla switch

```
switch (variável ou valor){
        case valor1:
        ...
        break;

        case valor2:
        ...
        break;
        ...
        ...
        default:
        ...
        break;
}
```

Estrutura de Seleção múltipla switch

```
Scanner input = new Scanner(System.in);
System.out.printf("Digite a média do aluno: ");
int nota = (int)input.nextDouble();

switch(nota){
    case 9:
        System.out.println( "A" );
    break;
    case 8:
        System.out.println( "B" );
    break;
    case 7:
        System.out.println( "C" );
    break;
    case 6:
        System.out.println( "D" );
    break;
    default:
        System.out.println( "F" );
    break;
}
```

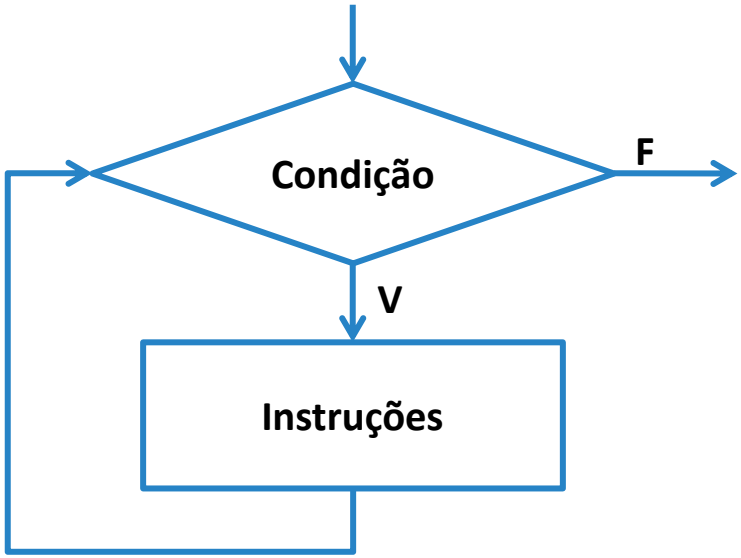
Estrutura de Repetição

Instruções de repetição

- **while** - *enquanto*
- **for** - *para*
- **do … while** - *faça … enquanto*

Comando WHILE

Condição ANTES de executar instruções




```
...
while (<condição>) {
    <instruções>
}
...
```

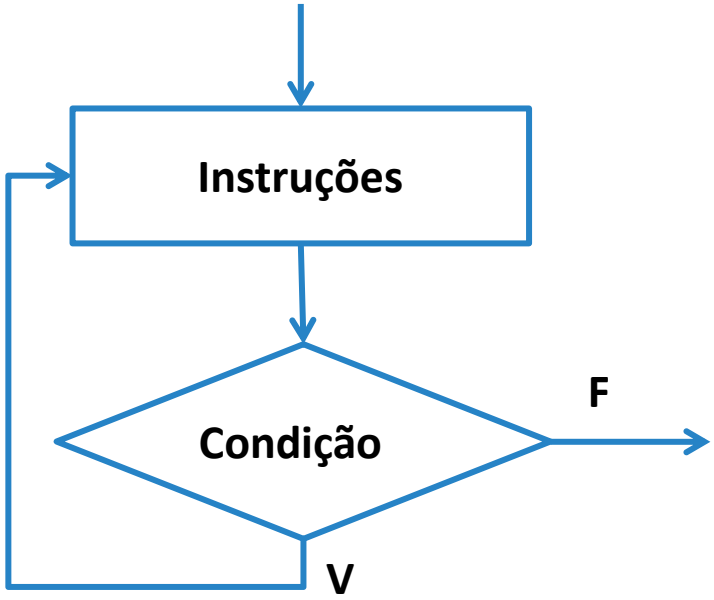
while: Exemplo

```
int contador = 0;

while ( contador <= 10 ) // faz o loop 10 vezes
{
      System.out.printf( “%d  ”, contador );
       contador = contador + 1; // incrementa o contador por 1
} // fim do while
```

Comando DO...WHILE

Condição DEPOIS de executar instruções



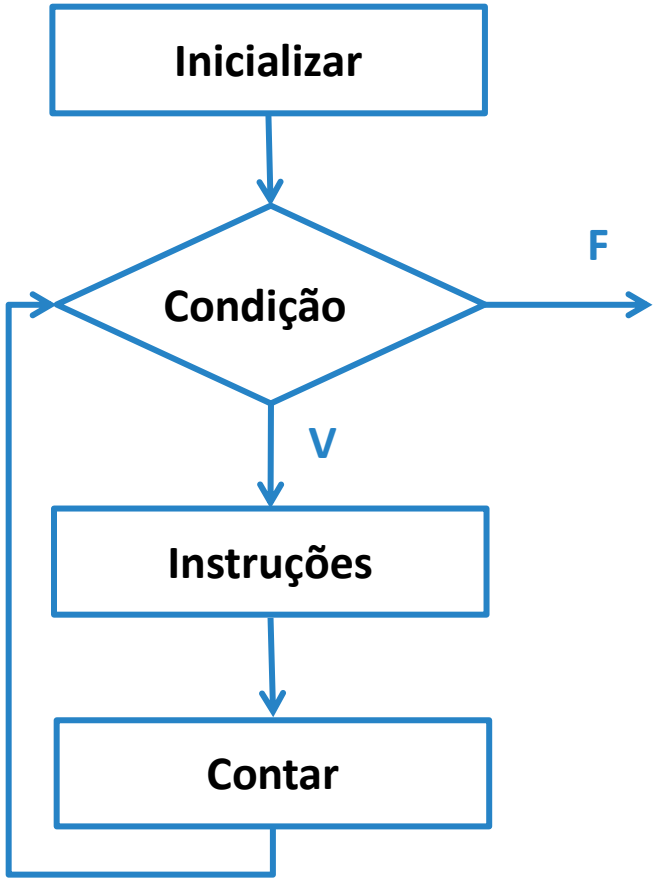

```
...
do {
        <instruções>
}  while (<condição>);
...
```

Do...while: Exemplo

```
10    public class Principal {
11        public static void main(String[] args) {
12            // TODO code application logic here
13            Scanner teclado = new Scanner(System.in);
14            int numero;
15            do{
16                System.out.println("Digite um número inteiro entre 0 e 5");
17                numero = teclado.nextInt();
18            }while(numero<0 || numero>5);
19            System.out.println("Você digitou o número: "+numero);
20        }
21    }
```

```
run:
Digite um número inteiro entre 0 e 5
9
Digite um número inteiro entre 0 e 5
3
Você digitou o número: 3
BUILD SUCCESSFUL (total time: 6 seconds)
```

Comando FOR




```
...
for (<inicializar>; <condição>; <contar>) {
    <instruções>
}
...
```

Instruções break e continue

Instrução break

- Causa saída imediata da estrutura de controle.

Instrução continue

- Pula as instruções restantes no corpo do loop.
- Prossegue para a próxima iteração.

Exemplo break

```
int contador = 0;

while ( contador <= 10 ) // faz o loop 10 vezes
{
      System.out.printf( “%d  ”, contador );
       contador = contador + 1; // incrementa o contador por 1
       if (contador > 5)
            break;
} // fim do while
```

Exemplo continue

```
int contador = 0;

while ( contador <= 10 ) // faz o loop 10 vezes
{
      contador = contador + 1; // incrementa o contador por 1
       if (contador%2 == 0)
            continue;
       System.out.printf( “%d  ”, contador );
} // fim do while
```

Exemplo continue

```
3   public class Principal {
4       public static void main(String[] args) {
5           int contador = 0;
6
7           while ( contador <= 10 ) // faz o loop 10 vezes
8           {
9           contador = contador + 1; // incrementa o contador por 1
10              if (contador%2 == 0)
11                  continue;
12              System.out.printf( "%d  ", contador );
13          } // fim do while
14      }
15  }
```

```
Saída - ternario (run)
run:
1  3  5  7  9  11  CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Exercícios

Exercícios

1. Faça um programa que receba um número e retorne se o valor é negativo ou positivo (usar o operador ternário).
2. Leia 3 valores inteiros e os ordene em ordem crescente. No final, mostre os 3 valores ordenados.
3. Faça um programa que exibe os números ímpares entre 1 e 1000.
4. Faça um programa que lê 20 números. Então, mostre a quantidade de valores positivos que foram digitados.

Exercício 5 - URI (1048)

A empresa ABC resolveu conceder um aumento de salários a seus funcionários de acordo com a tabela abaixo:

Salário	Percentual de Reajuste
0 - 400.00	15%
400.01 - 800.00	12%
800.01 - 1200.00	10%
1200.01 - 2000.00	7%
Acima de 2000.00	4%

Leia o salário do funcionário e calcule e mostre o novo salário, bem como o valor de reajuste ganho e o índice reajustado, em percentual.

Entrada

A entrada contém apenas um valor de ponto flutuante, com duas casas decimais.

Saída

Imprima 3 linhas na saída: o novo salário, o valor ganho de reajuste e o percentual de reajuste ganho, conforme exemplo abaixo.

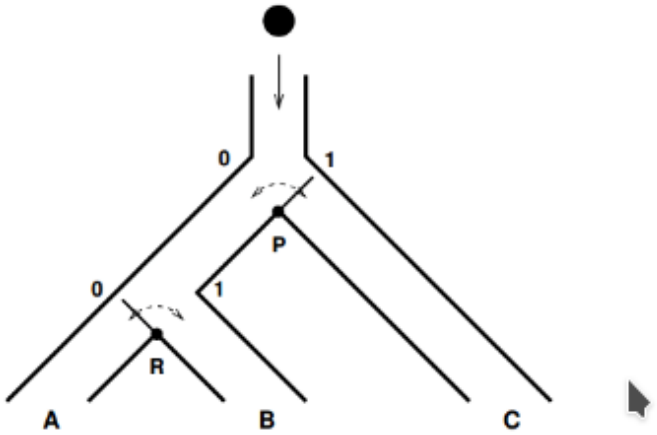
Exercício 6

(*5.17 Deitel*) Uma empresa que faz negócio por reembolso postal vende cinco produtos cujos preços de varejo são como segue: Produto 1, $ 2.98; Produto 2, $ 4.50; Produto 3, $ 3.98; Produto 4, $ 4.49, Produto 5, $ 6.87. Escreva um aplicativo que leia uma série de pares de números como segue: a) número do produto; b) quantia vendida.

Seu programa deve utilizar uma instrução *switch* para determinar o preço de venda de cada produto. Você deve calcular e exibir o valor de varejo total de todos os produtos vendidos. Utilize um *loop* controlado por sentinela para determinar quando o programa deve parar o *loop* e exibir os resultados finais.

Exercício 7 - URI (2454)

Flíper é um tipo de jogo onde uma bolinha de metal cai por um labirinto de caminhos até chegar na parte de baixo do labirinto. A quantidade de pontos que o jogador ganha depende do caminho que a bolinha seguir. O jogador pode controlar o percurso da bolinha mudando a posição de algumas portinhas do labirinto. Cada portinha pode estar na posição 0, que significa virada para a esquerda, ou na posição 1 que quer dizer virada para a direita. Considere o flíper da figura abaixo, que tem duas portinhas. A portinha P está na posição 1 e a portinha R, na posição 0. Desse jeito, a bolinha vai cair pelo caminho B.



Você deve escrever um programa que, dadas as posições das portinhas P e R, neste flíper da figura, diga por qual dos três caminhos, A, B ou C, a bolinha vai cair!

Exercício 8 (Deitel 5.16)

Escreva um aplicativo que lê 5 números entre 1 e 30. Para cada número lido, seu programa deve imprimir o mesmo número de asteriscos adjacentes. Por exemplo, se seu programa lê os números: 1, 3, 10, 2 e 5, ele deve imprimir:

```
*
***
**********
**
*****
```

Exercício 9 - *Desafio* (Deitel 5.21)

(Triplos de Pitágoras) Um triângulo retângulo pode ter lados que são todos inteiros. Um conjunto de três valores inteiros para os lados de um triângulo retângulo é chamado de triplo de Pitágoras. Esses três lados devem satisfazer o relacionamento de que a soma dos quadrados de dois dos lados seja igual ao quadrado da hipotenusa. Localize todos os triplos de Pitágoras para lado1, lado2 e hipotenusa, todos não maiores que 500. Utilize um loop for triplamente aninhado que tente todas as possibilidades. Esse é um exemplo de computação baseada na **força bruta**.