

|    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|
| 53 | 45 | 52 | 45 | 49 | 20 | 46 | 49 | 45 | 4c | 20 |
| 41 | 4f | 53 | 20 | 50 | 52 | 45 | 43 | 45 | 49 | 54 |
| 4f | 53 | 20 | 44 | 41 | 20 | 48 | 4f | 4e | 52 | 41 |
| 20 | 45 | 20 | 44 | 41 | 20 | 43 | 49 | c3 | 8a | 4e |
| 43 | 49 | 41 | 2c | 20 | 50 | 52 | 4f | 4d | 4f | 56 |
| 45 | 4e | 44 | 4f | 20 | 4f | 20 | 55 | 53 | 4f | 20 |
| 45 | 20 | 4f | 20 | 44 | 45 | 53 | 45 | 4e | 56 | 4f |
| 4c | 56 | 49 | 4d | 45 | 4e | 54 | 4f | 20 | 44 | 41 |
| 20 | 49 | 4e | 46 | 4f | 52 | 4d | c3 | 81 | 54 | 49 |
| 43 | 41 | 20 | 45 | 4d | 20 | 42 | 45 | 4e | 45 | 46 |
| c3 | 8d | 43 | 49 | 4f | 20 | 44 | 4f | 20 | 43 | 49 |
| 44 | 41 | 44 | c3 | 83 | 4f | 20 | 45 | 20 | 44 | 41 |
| 20 | 53 | 4f | 43 | 49 | 45 | 44 | 41 | 44 | 45 | 2e |

## RESIDÊNCIA DE SOFTWARE

**CAPACITAR**  
**TREINAR**  
**EMPREGAR**

**TRANSFORMAR**

**Firjan** SENAI  




Variáveis, Tipo

Variáveis

Tipos primitivos

Estruturas de Seleção

Estruturas de Repetição

Conversões

Operadores

# PALAVRAS CHAVE

Palavras-chave, também conhecidas como palavras reservadas da linguagem, são palavras que não podem ser usadas como identificadores, ou seja, não podem ser usadas para representar variáveis, classes ou nomes de métodos.

|          |          |            |           |              |
|----------|----------|------------|-----------|--------------|
| abstract | continue | for        | new       | switch       |
| assert   | default  | goto       | package   | synchronized |
| boolean  | do       | if         | private   | this         |
| break    | double   | implements | protected | throw        |
| byte     | else     | import     | public    | throws       |
| case     | enum     | instanceof | return    | transient    |
| catch    | extends  | int        | short     | try          |
| char     | final    | interface  | static    | void         |
| class    | finally  | long       | strictfp  | volatile     |
| const    | float    | native     | super     | while        |

# VARIÁVEIS

São armazenadas na memória RAM da máquina. As variáveis podem guardar dados de tipos numéricos, textos, booleanos e referências de objetos. O nome de uma variável não pode começar com um número e não pode ser uma palavra reservada.

## Declaração

- Tipo da variável mais o nome da variável.

```
int numero;  
double media;
```

- A declaração de uma variável pode ser realizada em qualquer linha de um bloco. Não é necessário declarar todas as variáveis no começo do bloco.

```
int numero = 30;  
System.out.println ( numero );  
double numero2= 87.3;  
System.out.println ( numero2);
```

## Variáveis de instância ou atributos

As variáveis de instâncias são definidas dentro de um classe, e só são inicializadas quando a classe é instanciada..

# TIPOS PRIMITIVOS

Um variável do tipo primitivo armazena um valor do seu tipo que foi declarado.

Abaixo uma lista do tipos primitivos. As variáveis devem ser declaradas respeitando-se a sintaxe básica “tipo nomeVariavel” esta convenção é chamada de CamelCase

| Tipo    | Tamanho |
|---------|---------|
| byte    | 1 byte  |
| short   | 2 bytes |
| int     | 4 bytes |
| long    | 8 bytes |
| float   | 4 bytes |
| double  | 8 bytes |
| boolean | 1 bit   |
| char    | 2 bytes |

O tipo primitivo char armazena apenas um caractere. Quando é necessário armazenar um texto, devemos utilizar o tipo String.

# EXERCÍCIOS

- 1) Criar um novo projeto no Eclipse com o nome **aula2**
  - Criar o pacote com o nome **exercicios**
  - Criar a classe **ExercicioVariaveis** no pacote **exercicios**
  - Declarar as variáveis: idade, peso e altura

Deverá ser impresso no console o seguinte resultado:

```
O funcionario João tem:  
idade:20  
altura:1.75  
peso:52.5
```

- 2) Criar uma nova classe com o nome **CalculadoraMedia**. Criar 4 variáveis com o nome nota1, nota2, nota3 e nota 4 com valores iniciais qualquer e exibir a média no console

```
A média é:9.0
```

# RESOLUÇÃO

```
package exercicios;

public class ExercicioVariaveis {
    public static void main(String[] args){
        int idade = 20;
        double altura = 1.75;
        double peso = 52.5;
        System.out.println("O funcionario João tem:" + "\nidade:" + idade + "\naltura:" + altura + "\npeso:" + peso);
    }
}
```

```
package exercicios;

public class CalculaMedia {
    public static void main(String[] args) {
        double nota1=10, nota2=9, nota3=8, nota4=9;
        System.out.println("A média é:" + (nota1 + nota2 + nota3 + nota4)/4);
    }
}
```



# CASTING DE TIPOS PRIMITIVOS

É possível atribuímos o valor de um tipo de variável a uma de outro tipo. Conversões de tipos primitivos boolean não podem ser feitas.

```
*Conversao.java X
package aula;

public class Conversao {
    public static void main(String[] args) {
        int a = 100;
        float b = a;

        double c = 4.19;
        // int d = c; não executa
        int d = (int) c;

        float e = (float) c;
        float f = 6.18f;

        System.out.println(b + "\n" + c + "\n" + d + "\n" + e + "\n" + f);
    }
}
```

Conversão implícita. A variável **b** de um tipo maior receberá o valor da variável **a**.

casting de um double para um inteiro.

Uma variável **float** não pode receber um **double** sem conversão pois todos os literais com ponto flutuante são **double**. A letra **f** indica que a variável é um do tipo **float**.



# CASTING DE TIPOS PRIMITIVOS

## Casting possíveis

Abaixo os tipos possíveis de casting em Java. A indicação impl. Quer dizer que o cast é implícito e automático, ou seja, você não precisa indicar o cast explicitamente. Além disso, o tipo boolean não pode ser convertido para outro tipo.

| PARA:         | byte   | short        | char   | int          | long         | float        | double       |
|---------------|--------|--------------|--------|--------------|--------------|--------------|--------------|
| DE:           |        |              |        |              |              |              |              |
| <b>byte</b>   | ----   | <i>Impl.</i> | (char) | <i>Impl.</i> | <i>Impl.</i> | <i>Impl.</i> | <i>Impl.</i> |
| <b>short</b>  | (byte) | ----         | (char) | <i>Impl.</i> | <i>Impl.</i> | <i>Impl.</i> | <i>Impl.</i> |
| <b>char</b>   | (byte) | (short)      | ----   | <i>Impl.</i> | <i>Impl.</i> | <i>Impl.</i> | <i>Impl.</i> |
| <b>int</b>    | (byte) | (short)      | (char) | ----         | <i>Impl.</i> | <i>Impl.</i> | <i>Impl.</i> |
| <b>long</b>   | (byte) | (short)      | (char) | (int)        | ----         | <i>Impl.</i> | <i>Impl.</i> |
| <b>float</b>  | (byte) | (short)      | (char) | (int)        | (long)       | ----         | <i>Impl.</i> |
| <b>double</b> | (byte) | (short)      | (char) | (int)        | (long)       | (float)      | ----         |

# EXERCÍCIOS CASTING DE TIPOS PRIMITIVOS

Declare duas variáveis do tipo **int** e realize sua soma.

Em seguida, realize o casting destes dois inteiros para **double** para realizar sua divisão.

```
package exercicios;

public class Casting {
    public static void main(String[] args) {
        int a = 10;
        int b = 2;
        int soma = a + b;

        System.out.println("Soma:" + soma);

        double divisao = ((double) a + (double) b) / 2;

        System.out.println("Divisão:" + divisao);
    }
}
```

# OPERADORES ARITMÉTICOS

Os operadores aritméticos seguem as mesmas regras seguidas em álgebra. Quando existem vários operadores de mesma precedência, ela é avaliada da esquerda pra direita.

| Operador      | Símbolo | Precedência |
|---------------|---------|-------------|
| Multiplicação | *       | 1º          |
| Divisão       | /       | 1º          |
| Resto         | %       | 1º          |
| Soma          | +       | 2º          |
| Subtração     | -       | 2º          |

Importante: a precedência também é válida para parênteses mais internos quando presente, assim como na álgebra.

# OPERADORES RELACIONAL

Os operadores relacionais avaliam dois operandos retornando um valor booleano

| Operador       | Símbolo |
|----------------|---------|
| Igual          | ==      |
| Diferente      | !=      |
| Menor que      | <       |
| Menor ou igual | <=      |
| Maior que      | >       |
| Maior ou igual | >=      |

# OPERADORES ATRIBUIÇÃO

Os operadores de atribuição como o próprio nome diz, fazem a atribuição de um valor a uma variável .

| Operador               | Símbolo | Equivalente  |
|------------------------|---------|--------------|
| Atribuição             | =       |              |
| Soma e atribui         | +=      | $a = a + b$  |
| Subtrai e atribui      | -=      | $a = a - b$  |
| Multiplica e atribui   | *=      | $a = a * b$  |
| Divide e atribui       | /=      | $a = a / b$  |
| Pega o resto e atribui | %=      | $a = a \% b$ |

# OPERADORES LÓGICOS

Os operadores lógicos representam o recurso que nos permite criar expressões lógicas maiores a partir da junção de duas ou mais expressões.

| Operador | Símbolo |
|----------|---------|
| Negação  | !       |
| E        | &&      |
| Ou       |         |

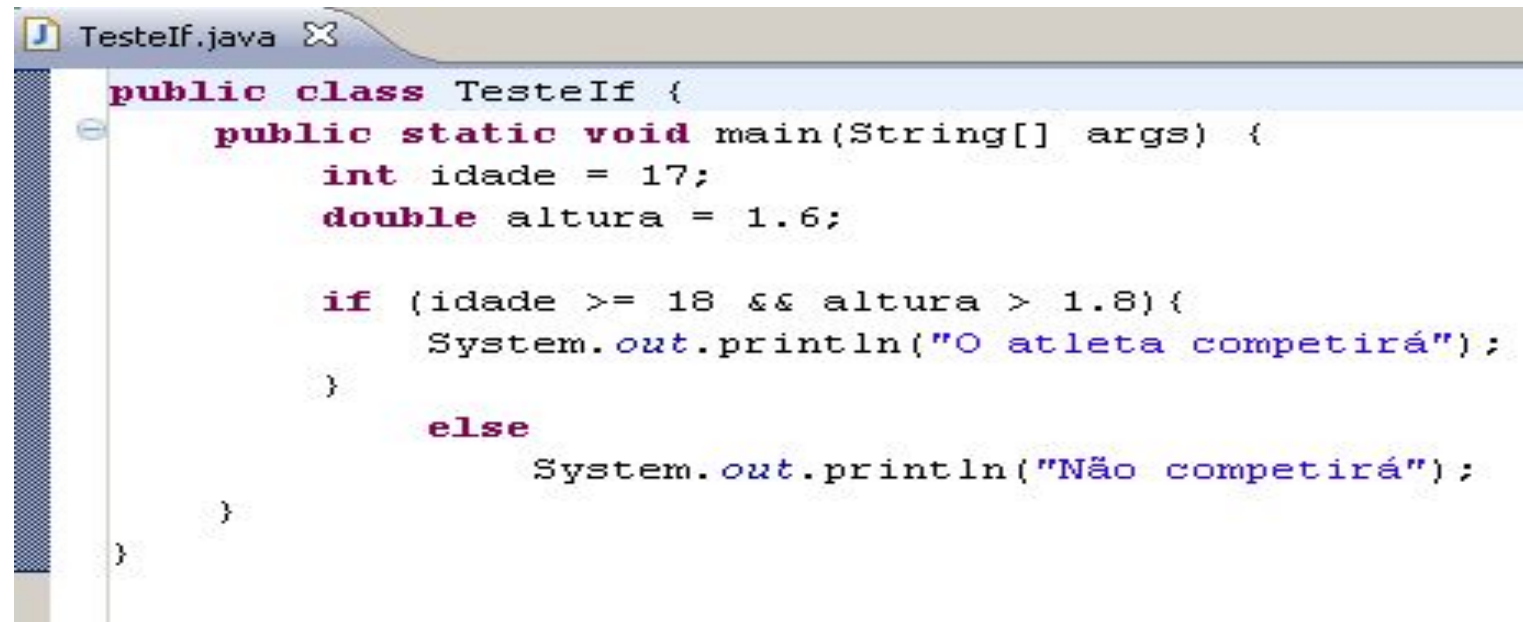
# IF/ELSE

Definem qual bloco de comandos deverá ser executado uma determinada condição.

Caso a condição do comando if for avaliada como verdadeira será executado o bloco de comandos dentro do if caso contrário o else. A condição é uma expressão que retorna true ou false.

sintaxe:

```
if (condicao) {  
    codigo;  
}  
else  
{  
    codigo;  
}
```



```
TesteIf.java X  
  
public class TesteIf {  
    public static void main(String[] args) {  
        int idade = 17;  
        double altura = 1.6;  
  
        if (idade >= 18 && altura > 1.8){  
            System.out.println("O atleta competirá");  
        }  
        else  
            System.out.println("Não competirá");  
    }  
}
```



**Curto-circuito:** ao avaliar expressões booleanas (lógicas AND e OR), a avaliação pode parar assim que encontrar a primeira condição que satisfaça ou negue a expressão

```
public class CurtoCircuito {  
    public static void main(String[] args) {  
        boolean a = 1==1; //true  
        boolean b = 1==2; //false  
        boolean c = 10==10; //true  
        boolean d = 1==20; //false  
  
        if(a && b) // Avalia as duas expressões  
        if(b && c) // Avalia apenas a primeira expressões  
        if(a || c) // Avalia apenas a primeira expressões  
        if(b || c) // Avalia as duas expressões  
    }  
}
```

# EXERCÍCIOS

1) Criar uma classe com o nome SituacaoAluno.

Faça um programa com duas variáveis nota1 e nota2 com valor inicial definido.

Calcule a média e caso o valor maior ou igual a 7 deverá ser exibida a mensagem “Aprovado”.

Caso a média for menor que 7 “Reprovado” e se a media for igual 10 “Aprovado Parabéns”.

```
package exercicios;

public class SituacaoAluno {
    public static void main(String[] args) {
        double nota1 = 8, nota2 = 6;
        double media = (nota1 + nota2) / 2;
        if (media == 10) {
            System.out.println("Aprovado Parabéns");
        } else if (media >= 7 && media < 10) {
            System.out.println("Aprovado");
        } else {
            System.out.println("Reprovado");
        }
    }
}
```

# EXERCÍCIOS

2) Crie uma classe com o nome CalculadoraSalario defina uma variável com o nome salário, inicialize a variável com algum valor e exiba no console o valor do salário com desconto do INSS.

## Tabela INSS

até 1.751,81 descontará 8%

entre 1.751,82 até 2.919,72 descontará 9%

entre 2.919,73 até 5.839,45 descontará 10%

Acima 5.839,456 descontará 11%

# RESOLUÇÃO

```
package exercicios;

public class CalculadoraSalario {
    public static void main(String[] args) {
        double salario = 2900;

        if(salario <= 1751.81) {
            salario = salario - salario * 8/100;
        }else if(salario >= 1751.82 && salario <=2912.72) {
            salario = salario - salario * 9/100;
        }else if(salario >= 2912.73 && salario <= 5839.45) {
            salario = salario - salario * 10/100;
        }else {
            salario = salario - salario * 11/100;
        }
        System.out.println("Salario com desconto: " + salario);
    }
}
```

# SWITCH

O **switch** testa o valor de uma variável, e dependendo do valor contido nessa variável, permite executar uma entre múltiplas escolhas de ações, com isto podemos substituir os múltiplos ifs utilizados em uma estrutura.

## Exemplo:

```
package exercicios;

public class ExemploSwitch {
    public static void main(String[] args) {
        int avaliacao= 0;

        switch (avaliacao) {
            case 5:
                System.out.println("Excelente");
                break;
            case 4:
                System.out.println("Bom");
                break;
            case 3:
                System.out.println("Regular");
                break;
            case 2:
                System.out.println("Ruim");
                break;
            case 1:
                System.out.println("Péssimo");
                break;

            default:
                System.out.println("Sem resposta");
                break;
        }
    }
}
```

# WHILE / DO WHILE

O **while** é um comando usado para fazer um loop, repetir um trecho de código várias vezes.

Exemplo

```
package exercicios;

public class TesteWhile {
    public static void main(String[] args) {
        int cont = 1;
        while(cont <= 10) {
            System.out.println(cont);
            cont += 1;
        }
    }
}
```

## Do - While

Nesta estrutura a verificação se o laço deve ser ou não repetido é no final do bloco.

```
package exercicios;

public class TesteDoWhile {
    public static void main(String[] args) {
        int cont = 1;
        do {
            System.out.println(cont);
            cont ++;
        } while (cont <= 10);
    }
}
```

# FOR

O for é outro comando de repetição que recebe 3 argumentos.

```
package exercicios;

public class TesteFor {
    public static void main(String[] args) {
        for(int i =1; i<=10; i++) {
            System.out.println(i);
        }
    }
}
```

## For - break

A mensagem será exibida até quando i for igual a 4.

```
package exercicios;

public class TesteForBreakContinue {
    public static void main(String[] args) {
        for (int i = 1; i <= 10; i++) {
            if (i == 5) {
                break;
            }
            System.out.println(i);
        }
    }
}
```

## For - continue

A mensagem não será exibida quando i for igual a 5 e 6.

```
package exercicios;

public class Continue {
    public static void main(String[] args) {
        for (int i = 0; i < 10; i++) {
            if(i < 2) {
                continue;
            }
            System.out.println(i);
        }
    }
}
```



# EXERCÍCIOS

- 1) Faça um programa que percorra números entre 0 e 30 e exibe a quantidade de números pares e ímpares.
- 2) Faça um programa que percorra todos os números de 1 até 22. Para os números múltiplos de 2, imprima a palavra “Java”, e mostre o total de múltiplos de 2 encontrado.
- 3) Faça uma tabela de multiplicação para o número 2 multiplicando do 1 até 10.
- 4) Liste os fatoriais de 1 a 10

# RESOLUÇÃO

1)

```
package exercicios;

public class Par {
    public static void main(String[] args) {
        int contPar = 0;
        int contImpar = 0;

        for (int i = 0; i < 30; i++) {
            if (i % 2 == 0) {
                contPar += 1;
            } else {
                contImpar += 1;
            }
        }
        System.out.println("Total de números pares:" + contPar);
        System.out.println("Total de números impares:" + contImpar);
    }
}
```

2)

```
package exercicios;

public class Multiplo2 {
    public static void main(String[] args) {
        int total = 0;
        for (int contador = 1; contador <= 22; contador++) {
            int resto = contador % 2;
            if (resto == 0) {
                System.out.println("JAVA");
                total += 1;
            }
        }
        System.out.println("Total:" + total);
    }
}
```

3)

```
package exercicios;

public class Tabuada {
    public static void main(String[] args) {
        int numero = 2;
        int resultado = 0;
        for (int i=1; i<=10; i++) {
            resultado = numero * i;
            System.out.println(numero + "x"
                + i + "=" + resultado);
        }
    }
}
```

4)

```
package exercicios;

public class Fatorial {
    public static void main(String[] args){
        for (int n = 1, fatorial = 1; n <= 10; n++){
            fatorial = fatorial * n;
            System.out.println(" O Fatorial de " + n + ": " + fatorial);
        }
    }
}
```

# ORIENTAÇÃO A OBJETOS

É uma tecnologia de desenvolvimento composta por metodologias e linguagens usadas na análise, no projeto e implementação de sistemas

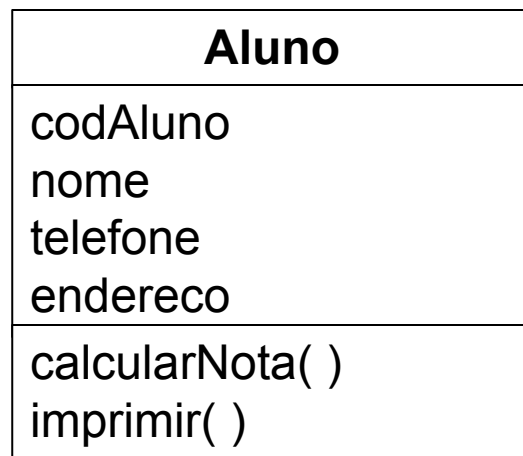
Os principais conceitos de orientação a objetos são:

- Classes
- Objetos
- Atributos
- Métodos
- Abstração
- Encapsulamento
- Polimorfismo.

## Classes

Uma classe é definida pelos seus atributos e métodos. A partir de uma classe, podemos construir objetos na memória do computador.

Um exemplo da classe Aluno no diagrama de UML abaixo, composta pelo nome da classe, atributos e métodos.



# CLASSE EM JAVA

```
class Aluno {  
    int codAluno;  
    String nome;  
    String telefone;  
    String endereco;  
}
```

Declaramos os atributos da classe.

## Criando Objetos em Java

Após a definição da classe Aluno podemos **contruir ou instanciar**, objetos que ficaram em memória. O comando usado para criação de objetos é o **new**. A classe testaAluno serve apenas para uso do método de chamada main.

```
class TestaAluno {  
    public static void main(String[] args){  
        new Aluno();  
    }  
}
```

O objeto foi criado agora como vamos acessá-lo? O comando **new** aloca o objeto em algum lugar da memória. Para acessá-lo precisamos de sua referência. Para guardar a referência utilizamos variáveis do tipo do objeto.

```
class testaAluno {  
    public static void main(String[] args){  
        Aluno a = new Aluno();  
    }  
}
```

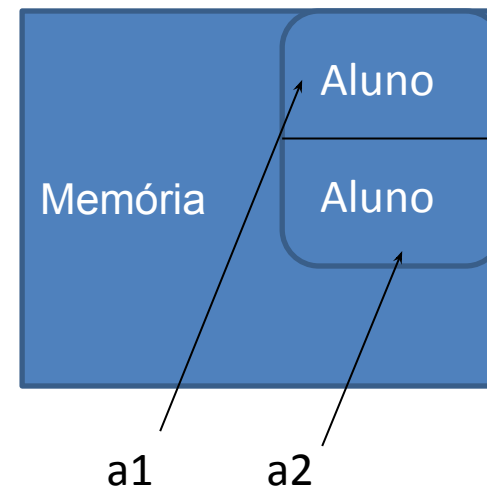
A variável **a** só pode referenciar objetos do tipo Aluno

# REFERÊNCIAS

Quando um objeto é criado é atribuído a variável, através do comando **new**, que a máquina virtual aloca o espaço necessário para armazenar os valores dos membros dos objetos

As variáveis a1 e a2 fazem referência a objetos diferentes.

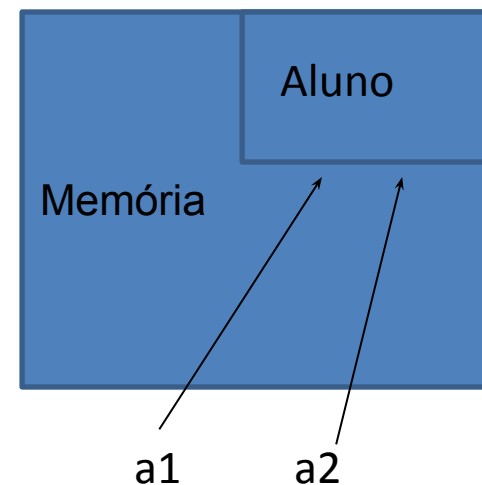
```
class testaAluno {  
    public static void main(String[] args){  
        Aluno a1 = new Aluno();  
        Aluno a2 = new Aluno();  
        System.out.println(a1);  
        System.out.println(a2);  
        if (a1 == a2)  
            System.out.println("Ref. iguais");  
        else  
            System.out.println("Ref. diferentes");  
    }  
}
```



# REFERÊNCIAS

A variável a2 faz referência ao mesmo objeto que a1.

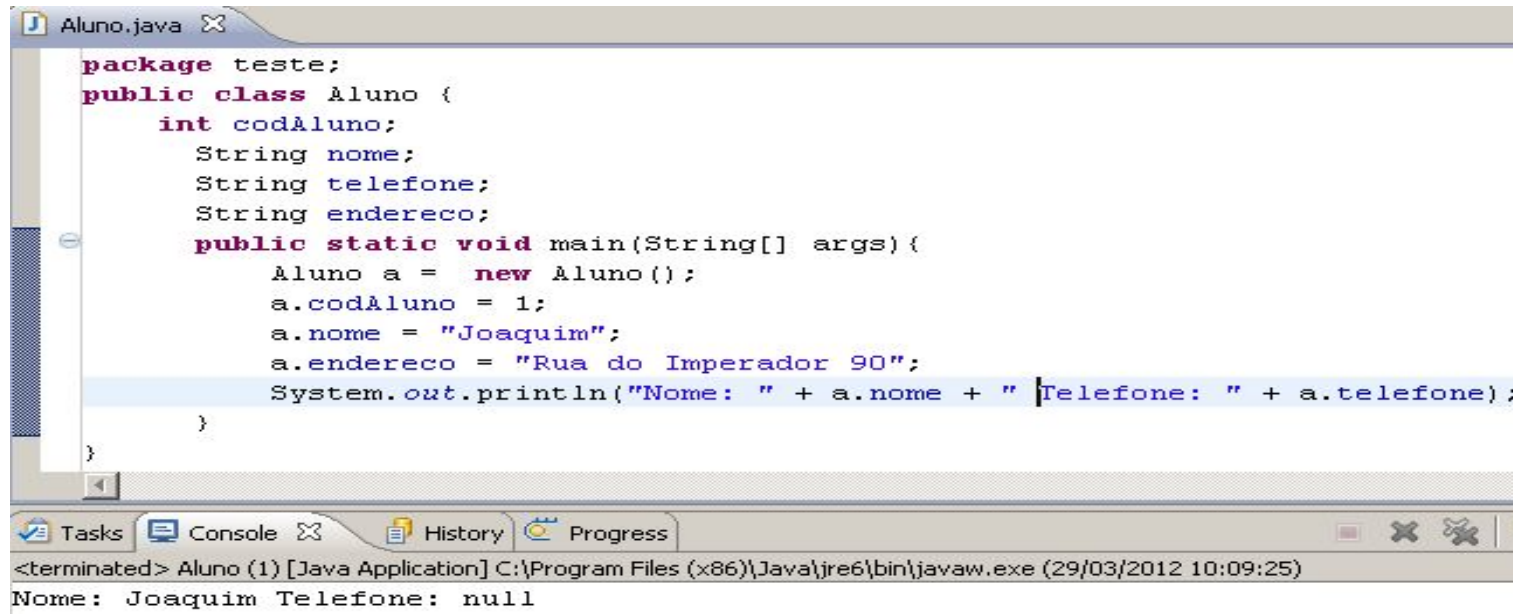
```
class testaAluno {  
    public static void main(String[] args){  
        Aluno a1 = new Aluno();  
        Aluno a2 = a1;  
        System.out.println(a1);  
        System.out.println(a2);  
        if (a1 == a2)  
            System.out.println("Ref. iguais");  
        else  
            System.out.println("Ref. diferentes");  
    }  
}
```





# ACESSANDO ATRIBUTOS DA CLASSE

No Java acessa-se um atributo ou um método por meio do operador “.” Para alterarmos os valores guardados nos atributos de um objeto os atributos são acessados pelo nome.



```
package teste;
public class Aluno {
    int codAluno;
    String nome;
    String telefone;
    String endereco;
    public static void main(String[] args){
        Aluno a = new Aluno();
        a.codAluno = 1;
        a.nome = "Joaquim";
        a.endereco = "Rua do Imperador 90";
        System.out.println("Nome: " + a.nome + " Telefone: " + a.telefone);
    }
}
```

Tasks Console History Progress

<terminated> Aluno (1) [Java Application] C:\Program Files (x86)\Java\jre6\bin\javaw.exe (29/03/2012 10:09:25)

Nome: Joaquim Telefone: null

A variável **a** faz referência ao objeto **Aluno**.

Quando criamos um objeto os atributos de tipos **numéricos** são inicializados com 0, os atributos do tipo **boolean** são inicializados com **false** e os demais atributos com **null** (vazio).

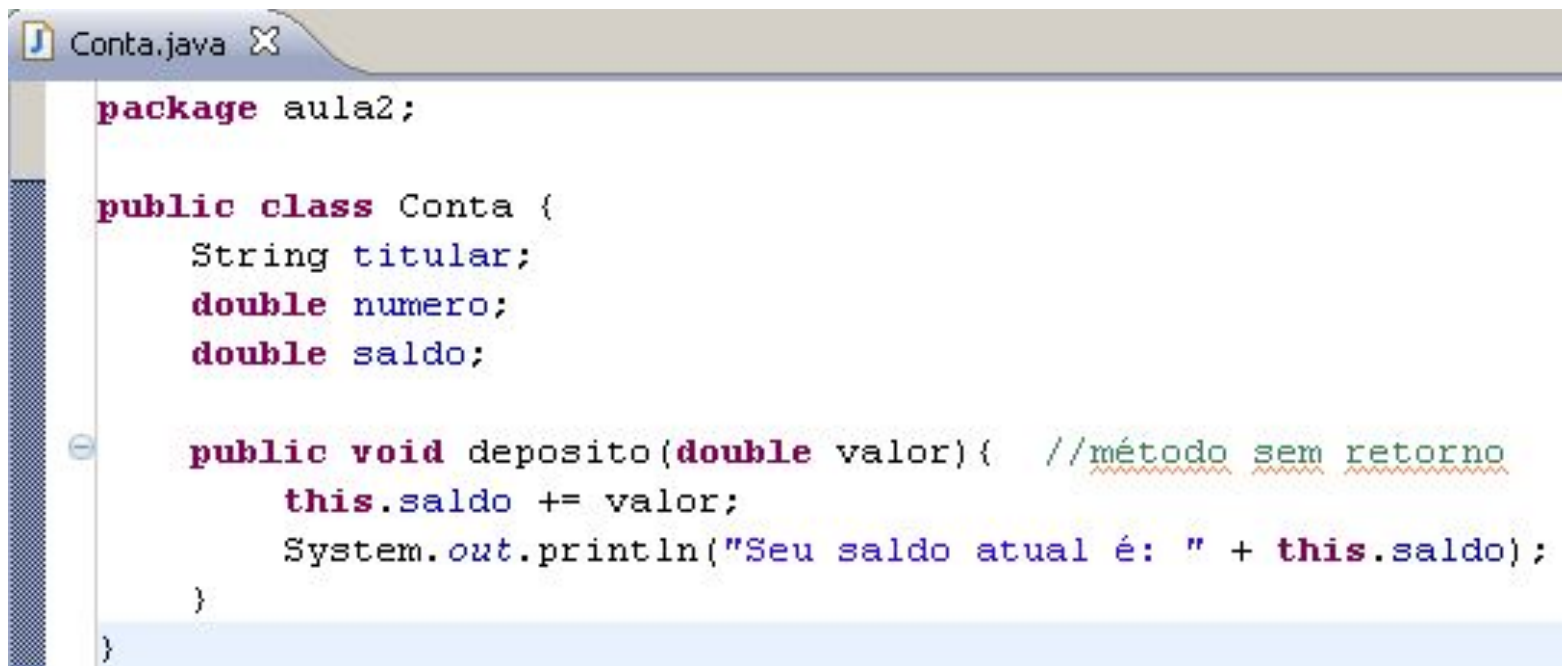
Não especificamos um valor para telefone por isto foi exibido **null** na execução.

# MÉTODOS

Os comportamentos da classe são implementados nos métodos de uma classe. Um método realiza diversas operações nos objetos.

## Métodos sem retorno

Para que um método não tenha retorno deve ser digitada a palavra **void** na definição do método.



```
Conta.java X
package aula2;

public class Conta {
    String titular;
    double numero;
    double saldo;

    public void deposito(double valor){ //método sem retorno
        this.saldo += valor;
        System.out.println("Seu saldo atual é: " + this.saldo);
    }
}
```

O **this** é usado para mostrar que estamos fazendo referência a um atributo e não a uma variável.

## Métodos com retorno

Para que um método tenha retorno deve ser inserido o tipo de retorno na definição do método.

```
public boolean saque(double valor) {  
    if (this.saldo < valor)  
        return false;  
    else {  
        this.saldo -= valor;  
        return true;  
    }  
}
```

# EXERCÍCIOS

- 1- Crie uma classe com o nome **Conta** no pacote **exercícios** com seus atributos e métodos.
- 2- Crie a classe **SaqueDeposito** que irá conter o método de chamada do **main** para execução.
- 3- Construa mais uma conta e atribua valores para teste.

# RESOLUÇÃO

```
Conta.java X
package aula2;

public class Conta {
    String titular;
    double numero;
    double saldo;

    public void deposito(double valor) { // método sem retorno
        this.saldo += valor;
        System.out.println("Seu saldo atual é: " + this.saldo);
    }

    public boolean saque(double valor) { // método com retorno
        if (this.saldo < valor)
            return false;
        else {
            this.saldo -= valor;
            return true;
        }
    }
}
```

```
Conta.java SaqueDeposito.java X
package aula2;

public class SaqueDeposito {
    public static void main(String[] args) {
        Conta c = new Conta();
        c.titular = "Amaral";
        c.numero = 12345;
        c.saldo = 1200.;
        c.deposito(500);
        if (c.saque(1000)) {
            System.out.println("Saque efetuado");
            System.out.println("Seu saldo Atual é: " + c.saldo);
        } else {
            System.out.println("Saldo insuficiente");
        }
    }
}
```

# ENTRADA DE DADOS

Quando falamos de processamento de dados por um computador, a entrada de dados são os dados obtidos de forma bruta, colhidos do mundo real através de algum dispositivo de entrada.

Exemplos:

- Teclado
- Arquivo
- Leitores
- Mouse
- Sensores

# ENTRADA DE DADOS

Para realizarmos entrada através do teclado podemos utilizar a classe **Scanner**. Essa classe possui vários métodos que possibilitam diferentes entradas de diferentes tipos:

```
package exercicios;

import java.util.Scanner;

public class Exercicio {
    public static void main(String[] args) {
        Scanner ler = new Scanner(System.in);
        int a, b;

        System.out.println("Informe o primeiro valor: ");
        a = ler.nextInt();

        System.out.println("Informe o segundo valor.: ");
        b = ler.nextInt();

        System.out.println("\nResultados da soma:\n");
        System.out.println(a + b);

        ler.close();
    }
}
```



# ENTRADA DE DADOS

Para entrada com a classe **Scanner** podemos utilizar de vários métodos para ler os diferentes tipo.

```
package exercicios;

import java.util.Scanner;

public class Exercicio {
    public static void main(String[] args){
        Scanner ler = new Scanner(System.in);

        System.out.println("Informe um número inteiro:");
        System.out.println(ler.nextInt());

        System.out.println("Informe um valor em reais:");
        System.out.println(ler.nextDouble());

        ler.close();
    }
}
```

# SAÍDA DE DADOS COM FORMATAÇÃO

Para saída de dados formatada podemos utilizar o método **printf()**

```
System.out.printf(expressão_de_controle, argumento1, argumento2, ...);
```

```
package exercicios;

public class Exercicio {
    public static void main(String[] args){
        double a = 135.4528;
        double b = 23050.568;
        double c = 5.0;

        System.out.printf("Variavel 'a' = %8.2f\n", a);
        System.out.printf("Variável 'b' = %8.2f\n", b);
        System.out.printf("Variável 'c' = %8.2f\n", c);

        int d = 1, e = 10, f = 100;
        System.out.printf("\n\n'd' = %3d\n'e' = %3d\n'f' = %3d\n", d, e, f);
    }
}
```

```
Variavel 'a' = 135,45
Variável 'b' = 23050,57
Variável 'c' = 5,00
```

```
'd' = 1
'e' = 10
'f' = 1000
```