



CURSO DE PROGRAMAÇÃO EM JAVA

Aula 3 
Tipos de dados

1.

Identificadores

Definição e regras para usá-los

Identificadores

Identificador é o nome que utilizamos para representar as variáveis, classes, objetos, etc. Por exemplo, na Matemática utilizamos um nome para as incógnitas (x , y , z , etc.) que é o identificador daquela incógnita.



Regras para a criação de identificadores:



Primeiro caracter - letra, underline (_) ou símbolo monetário (\$)



Caracteres seguintes - quaisquer desses ou numerais



Não precisam ser letras Latinas ou numerais - qualquer alfabeto - Unicode



Acentuação pode ser usada (apesar de não ser recomendado)



Identificadores *case sensitive*



Não pode ser palavra-reservada

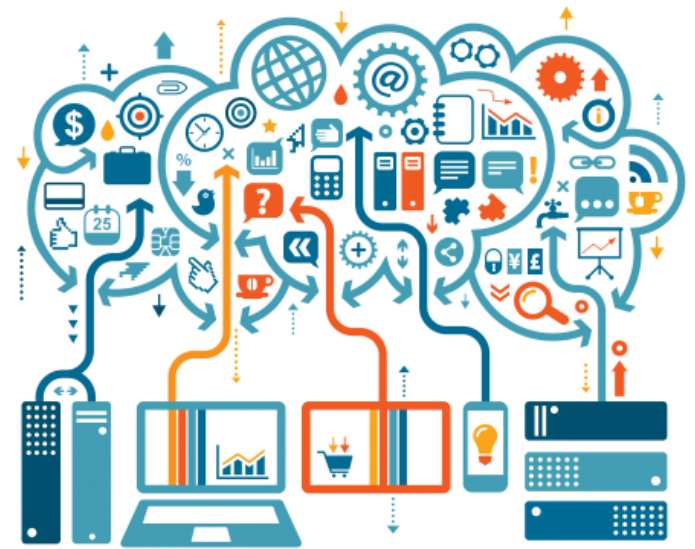
Alguns exemplos

x
abóbora
άγγελος
carro1
carro_1
\$1
_carro
\$James

2.

PALAVRAS-CHAVES

Quais as palavras reservadas em Java?



Palavras-chaves

Em programação, palavras-chave, ou palavras reservadas, são as palavras que não podem ser usadas como identificadores, ou seja, não podem ser usadas como nome de variáveis, nome de classes, etc. Estas palavras são assim definidas ou porque já têm uso na sintaxe da linguagem ou porque serão usadas em alguns momento, seja para manter compatibilidade com versões anteriores ou mesmo com outras linguagens.

No caso do Java temos as seguintes palavras-chave:

<i>abstract</i>	<i>continue</i>	<i>for</i>	<i>new</i>	<i>switch</i>
<i>assert(3)</i>	<i>default</i>	<i>goto(1)</i>	<i>package</i>	<i>synchronized</i>
<i>boolean</i>	<i>do</i>	<i>if</i>	<i>private</i>	<i>this</i>
<i>break</i>	<i>double</i>	<i>implements</i>	<i>protected</i>	<i>throw</i>
<i>byte</i>	<i>else</i>	<i>import</i>	<i>public</i>	<i>throws</i>
<i>case</i>	<i>enum(4)</i>	<i>instanceof</i>	<i>return</i>	<i>transient</i>
<i>catch</i>	<i>extends</i>	<i>int</i>	<i>short</i>	<i>try</i>
<i>char</i>	<i>final</i>	<i>interface</i>	<i>static</i>	<i>void</i>
<i>class</i>	<i>finally</i>	<i>long</i>	<i>strictfp(2)</i>	<i>volatile</i>
<i>const(1)</i>	<i>float</i>	<i>native</i>	<i>super</i>	<i>while</i>
<i>null</i>				

(1) sem uso na linguagem

(2) somente a partir da versão 1.2

(3) somente a partir da versão 1.4

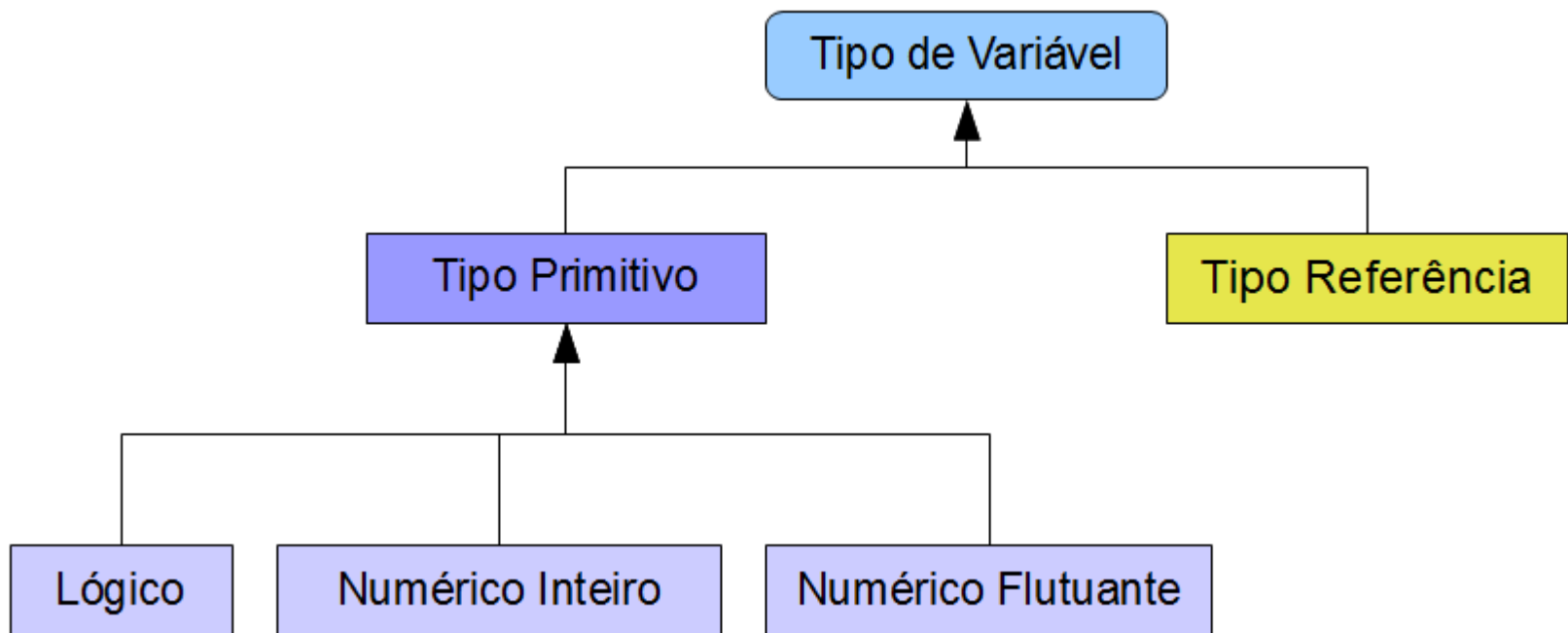
(4) somente a partir da versão 5.0

3.

TIPOS DE DADOS - PRIMITIVOS

Lógicos, numéricos inteiros e numéricos flutuantes

Famílias de Tipos de Variável



Tipos primitivos

No Java, existem algumas palavras reservadas para a representação dos tipos de dados básicos que precisam ser manipulados para a construção de programas. Estes tipos de dados são conhecidos como tipos primitivos.

Existem milhares de classes disponíveis na API do Java e todas são tipos de dados, porém uma classe pode armazenar diversos dados ao mesmo tempo em seus atributos, e realizar tarefas através de seus métodos. Um tipo primitivo por outro lado, só armazena um único dado e não contém quaisquer métodos para realizar tarefas.



Tipos primitivos

Boolean: Não é um valor numérico, só admite os valores true ou false.

Char: Usa o código UNICODE e ocupa cada caractere 2 bytes.

Inteiros: Diferem nas precisões e podem ser positivos ou negativos.

Byte: 1 byte.

Short: 2 bytes.

Int: 4 bytes.

Long: 8 bytes.

Reais em ponto flutuante: igual que os inteiros também diferem nas precisões e podem ser positivos ou negativos.

Float: 4 bytes.

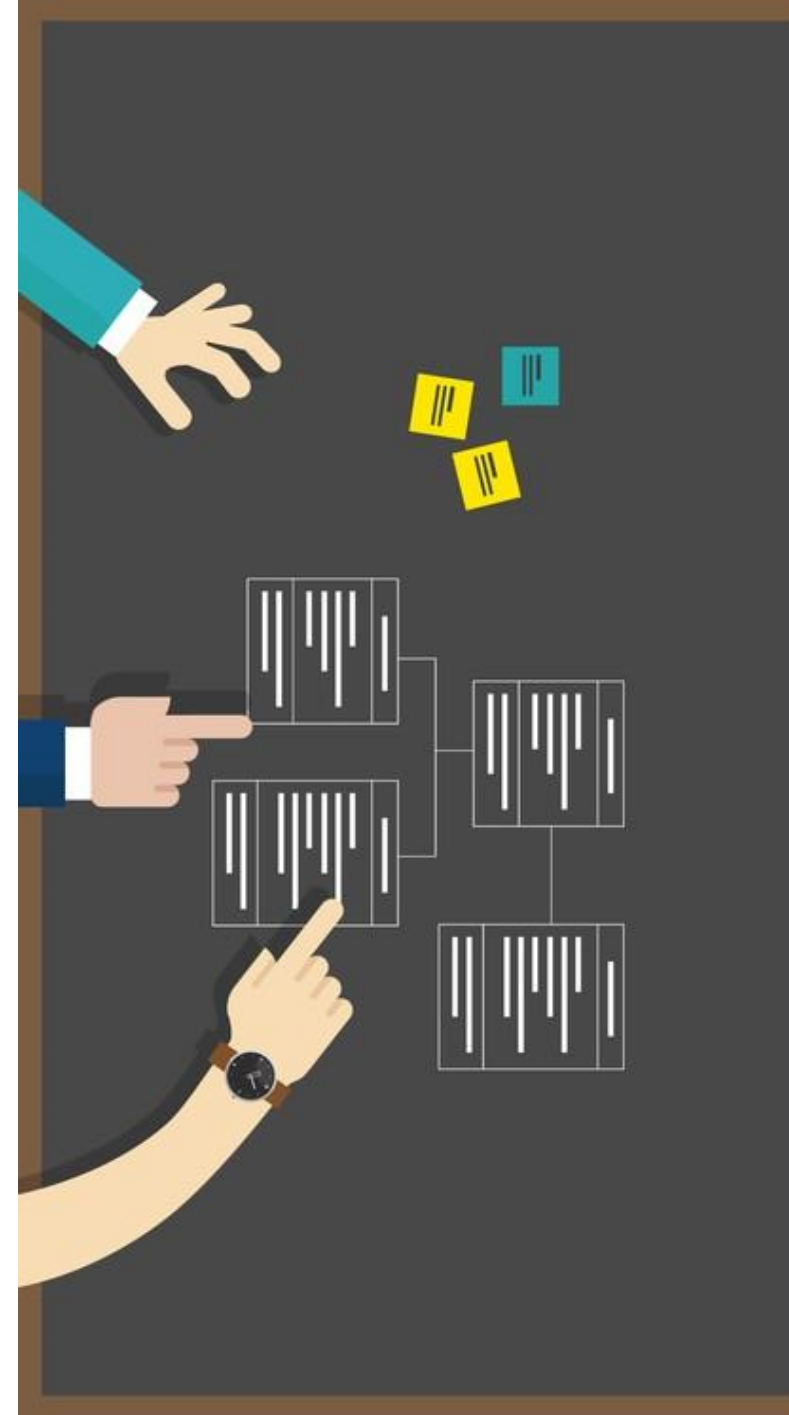
Double: 8 bytes.

Tabela Unicode

Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code
	160	¡	161	¢	162	£	163	¤	164	¥	165	¦	166	§	167
¨	168	©	169	ª	170	«	171	¬	172		173	®	174	¯	175
°	176	±	177	²	178	³	179	´	180	µ	181	¶	182	·	183
,	184	¹	185	º	186	»	187	¼	188	½	189	¾	190	¿	191
À	192	-	193	Â	194	Ã	195	Ä	196	Å	197	Æ	198	Ç	199
È	200	É	201	Ê	202	Ë	203	Ì	204	Í	205	Î	206	Ï	207
Ð	208	Ñ	209	Ò	210	Ó	211	Ô	212	Õ	213	Ö	214	×	215
Ø	216	Ù	217	Ú	218	Û	219	Ü	220	Ý	221	Þ	222	ß	223
à	224	á	225	â	226	ã	227	ä	228	å	229	æ	230	ç	231
è	232	é	233	ê	234	ë	235	ì	236	í	237	î	238	ï	239
ð	240	ñ	241	ò	242	ó	243	ô	244	õ	245	ö	246	÷	247
ø	248	ù	249	ú	250	û	251	ü	252	ý	253	þ	254	ÿ	255

Tipos primitivos

Também existem classes para representar cada um dos tipos primitivos. Sempre que for preciso realizar uma operação mais complexa com algum dado, você poderá armazená-la em um objeto da classe correspondente ao invés de utilizar um tipo primitivo. Assim, poderá fazer uso dos métodos disponíveis nessa classe para realizar diversas operações com o dado armazenado.



Tamanhos de tipos primitivos

```
public class Tipos_de_Dados {  
  
    public static void main(String[] args) {  
  
        System.out.println("Tipos de dados em Java: \n" +  
            "\nMenor Byte: " + Byte.MIN_VALUE +  
            "\nMaior Byte: " + Byte.MAX_VALUE +  
            "\nMenor Short Int: " + Short.MIN_VALUE +  
            "\nMaior Short Int: " + Short.MAX_VALUE +  
            "\nMenor Int: " + Integer.MIN_VALUE +  
            "\nMaior Int: " + Integer.MAX_VALUE +  
            "\nMenor Long: " + Long.MIN_VALUE +  
            "\nMaior Long: " + Long.MAX_VALUE +  
            "\nMenor Float: " + Float.MIN_VALUE +  
            "\nMaior Float: " + Float.MAX_VALUE +  
            "\nMenor Double: " + Double.MIN_VALUE +  
            "\nMaior Double: " + Double.MAX_VALUE);  
  
    }  
}
```

Termos básicos



DADOS NUMÉRICOS

A representação dos números inteiros é feita pelo **byte, short, int, long**. E dos números reais pelo **float e double**.



DADOS TEXTUAIS

A representação de um caractere solitário é feita pelo tipo **char** e a representação de textos é feita pela classe **String**.

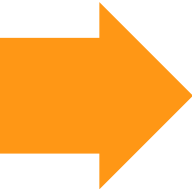


DADOS LÓGICOS

O tipo lógico é representado, em Java, pelo tipo **booleano**. Este tipo pode armazenar um de dois valores possíveis: **true** ou **false**. Ele é empregado para realizar testes lógicos em conjunto com operadores relacionais e dentro de estruturas de decisão e repetição.

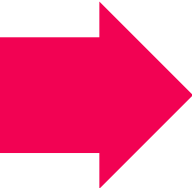
Declaração dos tipos primitivos

```
public class Tipos_Primitivos {  
    public static void main(String[] args) {  
        byte tipoByte = 127;  
        short tipoShort = 32767;  
        char tipoChar = 'C';  
        float tipoFloat = 2.6f;  
        double tipoDouble = 3.59;  
        int tipoInt = 2147483647;  
        long tipoLong = 9223372036854775807L;  
        boolean tipoBooleano = true;  
        System.out.println("Valor do tipoByte = " + tipoByte);  
        System.out.println("Valor do tipoShort = " + tipoShort);  
        System.out.println("Valor do tipoChar = " + tipoChar);  
        System.out.println("Valor do tipoFloat = " + tipoFloat);  
        System.out.println("Valor do tipoDouble = " + tipoDouble);  
        System.out.println("Valor do tipoInt = " + tipoInt);  
        System.out.println("Valor do tipoLong = " + tipoLong);  
        System.out.println("Valor do tipoBooleano = " + tipoBooleano);  
    }  
}
```



Uma **variável** representa a unidade básica de armazenamento temporário de dados e compõe-se de um tipo, um identificador e um escopo. Seu objetivo é armazenar um dado de determinado tipo primitivo para que possa ser recuperado e aplicado em operações posteriores.

```
< tipo > < identificador > = < valor >;
```



As **constantes** são unidades básicas de armazenamento de dados que não devem sofrer alterações ao longo da execução do aplicativo. O uso de constantes é menos frequente que o uso de variáveis. No entanto, há situações em que elas são requeridas.

```
final < tipo > < identificador > = < valor >;
```

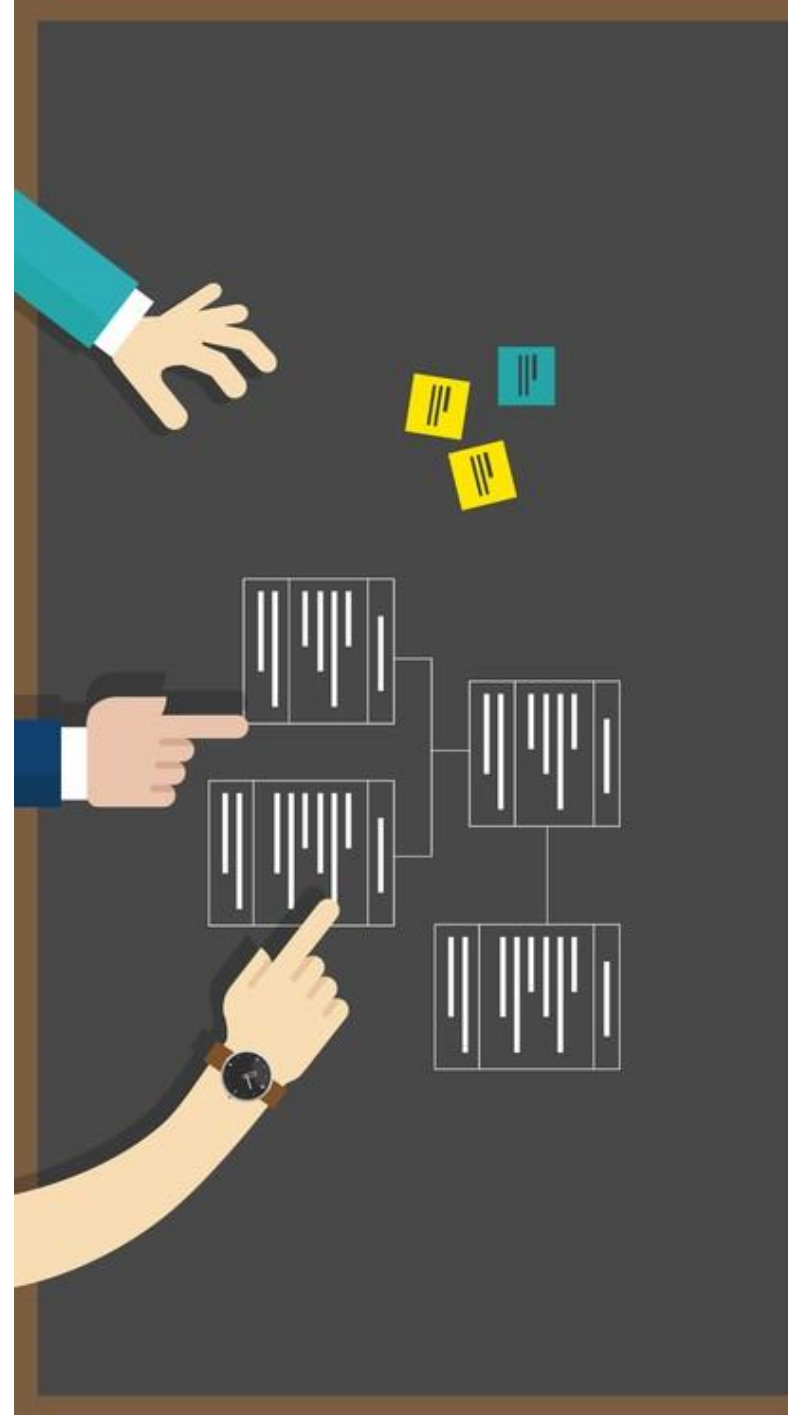
4.

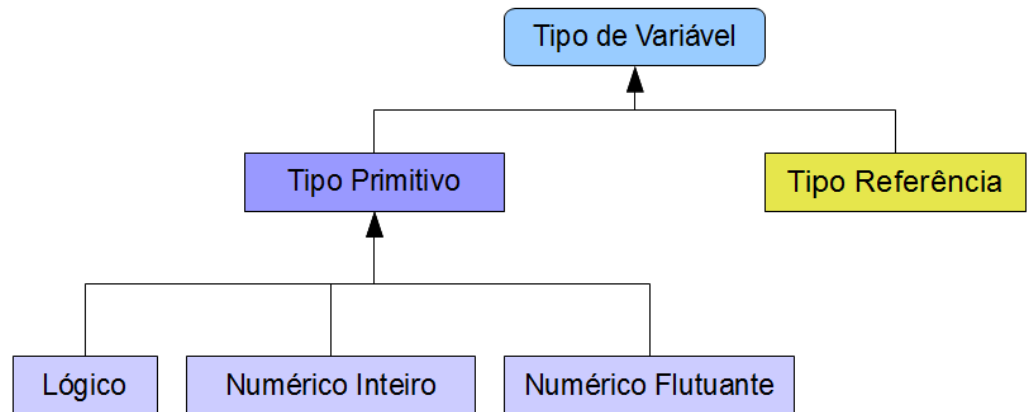
TIPOS DE DADOS - POR REFERÊNCIA

O Java possui dois tipos de dados que são divididos em por valor (tipos primitivos) e por referência (tipos por referência).

Tipos por referência

Os tipos primitivos são **boolean**, **byte**, **char**, **short**, **int**, **long**, **float** e **double**. Os tipos por referência, são **classes** que especificam os tipos de objeto **Strings**, **Arrays** **Primitivos** e **Objetos**.





Tipos por referência

Os programas utilizam as variáveis de tipos por referência para armazenar as localizações de objetos na memória do computador. Esses objetos que são referenciados podem conter várias variáveis de instância e métodos dentro do objeto apontado.

Para trazer em um objeto os seus métodos de instância, é preciso ter referência a algum objeto. As variáveis de referência são inicializadas com o valor "null" (nulo).

Exemplos

Por exemplo, **ClasseConta acao = new ClasseConta()**, cria um objeto de classe **ClasseConta** e a variável **acao** contém uma referência a esse objeto ClasseConta, onde poderá invocar todos os seus métodos e atributos da classe. A palavra chave **new** solicita a memória do sistema para armazenar um objeto e inicializa o objeto.

Exemplos

```
public class AcessoMetodo {  
  
    public void imprime(){  
        System.out.println("Bem Vindo ao Java!");  
    }  
  
    public static void main(String[] args) {  
        AcessoMetodo acesso = new AcessoMetodo ();  
        acesso.imprime();  
    }  
}
```

A saída desse código acima irá ser reproduzida através da ação `acesso.imprime()`, porque está sendo acessado o método do objeto que foi inicializado com a variável definida como "acesso".

5.

Inicializando objetos com
construtores

Construtores



São os responsáveis por criar o objeto em memória, ou seja, instanciar a classe que foi definida.



Possuem o mesmo nome que a classe.



Pode existir mais de um construtor em uma classe, porém com diferentes parâmetros.



O Java já cria um construtor default para nós.



Na declaração do Objeto o new é o responsável de chamar o construtor.



É o valor default dos seus objetos, do mesmo modo que 0 é o valor default para int.



Não possui um tipo de retorno.

Exemplos

```
public class Carro{

    /* CONSTRUTOR DA CLASSE Carro */
    /*modificadores de acesso (public nesse caso) + nome da classe (Carro
nesse caso) + parâmetros (nenhum definido neste caso).*/

    public Carro(){
        //Faça o que desejar na construção do objeto
    }
}

public class Aplicacao {
    public static void main(String[] args) {
        //Chamamos o construtor sem nenhum parâmetro
        Carro fiat = new Carro();
    }
}
```

Exemplos

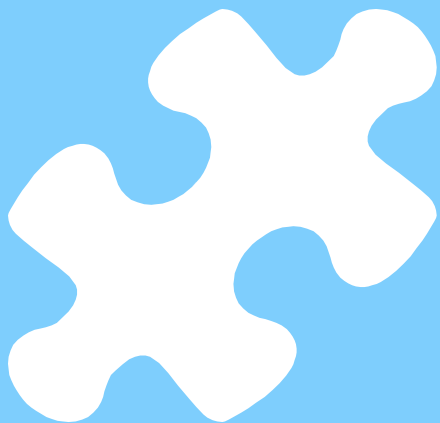
```
public class Carro{  
  
    private String cor;  
    private double preco;  
    private String modelo;  
  
    /* CONSTRUTOR PADRÃO */  
    public Carro(){  
  
    }  
  
    /* CONSTRUTOR COM 2 PARÂMETROS */  
    public Carro(String modelo, double preco){  
        //Se for escolhido o construtor sem a COR do veículo  
        // definimos a cor padrão como sendo PRETA  
        this.cor = "PRETA";  
        this.modelo = modelo;  
        this.preco = preco;  
    }  
}
```

Exemplos

```
/* CONSTRUTOR COM 3 PARÂMETROS */
public Carro(String cor, String modelo, double preco){
    this.cor = cor;
    this.modelo = modelo;
    this.preco = preco;
}
}
public class Aplicacao {
    public static void main(String[] args) {
        //Construtor sem parâmetros
        Carro prototipoDeCarro = new Carro();

        //Construtor com 2 parâmetros
        Carro civicPreto = new Carro("New Civic", "40000");

        //Construtor com 3 parâmetros
        Carro golfAmarelo = new Carro("Amarelo", "Golf", "38000");
    }
}
```



DESAFIO

E aí, vamos praticar?

Média

Leia 3 valores, no caso, variáveis A, B e C, que são as três notas de um aluno. A seguir, calcule a média do aluno, sabendo que a nota A tem peso 2, a nota B tem peso 3 e a nota C tem peso 5. Considere que cada nota pode ir de 0 até 10.0, sempre com uma casa decimal.

Exemplos de Entrada	Exemplos de Saída
5.0 6.0 7.0	MEDIA = 6.3
5.0 10.0 10.0	MEDIA = 9.0

Utilização de classes e métodos:

Classe Media com atributos: notaA, notaB, notaC.

Métodos: obterMedia, imprimir.

Área do Círculo

A fórmula para calcular a área de uma circunferência é: **area** = π . **raio**². Considerando para este problema que π = 3.14159:

- Efetue o cálculo da área, elevando o valor de **Raio** ao quadrado e multiplicando por π .

Exemplos de Entrada	Exemplos de Saída
2.00	A=12.5664
100.64	A=31819.3103
150.00	A=70685.7750

Utilização de classes e métodos:

Classe Circulo com atributos: raio, pi (constante).

Métodos: obterArea, imprimir.

Gasto de Combustível

Joaozinho quer calcular e mostrar a quantidade de litros de combustível gastos em uma viagem, ao utilizar um automóvel que faz 12 KM/L. Para isso, ele gostaria que você o auxiliasse através de um simples programa. Para efetuar o cálculo, deve-se fornecer o tempo gasto na viagem (em horas) e a velocidade média durante a mesma (em km/h). Assim, pode-se obter distância percorrida e, em seguida, calcular quantos litros seriam necessários. Mostre o valor com 3 casas decimais após o ponto.

Utilização de classes e métodos:

Classe CalcCombustivel com atributos: tempo, velocidade, qntLitros.

Métodos: obterMedia, imprimir.

Gasto de Combustível

Exemplo de Entrada	Exemplo de Saída
10 85	70.833
2 92	15.333
22 67	122.833

Utilização de classes e métodos:

Classe CalcCombustivel com atributos: tempo, velocidade, qntLitros.

Métodos: obterMedia, imprimir.

Obrigado!

Alguma pergunta?

Você pode me contatar em:
ywassef@hotmail.com