

Programação Orientada a Objetos

Encapsulamento e Método Construtor

profº Mauricio Conceição Mario

Encapsulamento

Define o que está acessível na classe, e é a forma como os elementos da classe podem ser vistos e utilizados por outras classes.

*Declaração de variável de instância como **static**: o conteúdo da variável será constante, estático para todos os objetos:*

public: O modificador public deixará visível a classe ou membro para todas as outras classes, subclasses e pacotes do projeto Java.

protected: O modificador protected deixará **visível** o atributo para todas as outras classes e subclasses que pertencem ao mesmo **pacote**. A principal diferença é que apenas as classes do mesmo pacote têm acesso ao membro. O pacote da subclasse não tem acesso ao membro.

private: O modificador private deixará visível o atributo apenas para a classe em que este atributo se encontra.

package-private: é o modificador padrão quando outro não é definido. Isto torna acessível na própria classe, nas classes e subclasses do mesmo pacote. Ele geralmente é utilizado para construtores e métodos que só devem ser invocados pelas classes e subclasses do pacote, constantes estáticas que são úteis apenas dentro do pacote em que estive inserido.

<http://www.devmedia.com.br/encapsulamento-polimorfismo-heranca-em-Java/12991#ixzz3kOJYBbZ1>

```
package encapsulamento_1;

public class Acesso_Numeros {

    int a;
    public int b;
    private int c;
    protected int d;







    public void setNumero(String id, int numero){
        if (id == "a")
            this.a = numero;

        if (id == "c")
            this.c = numero;

        if (id == "d")
            this.d = numero;
    }

    public void mostra_numero(){
        System.out.println("numero a = " + a);
        System.out.println("numero b = " + b);
        System.out.println("numero c = " + c);
        System.out.println("numero d = " + d);
    }

}
```

- ▼  encapsulamento_1
 - >  Acesso_Numeros.java
 - >  Verifica_Encapsulamento_1.java
- ▼  encapsulamento_2
 - >  Verifica_Encapsulamento_2.java
- >  JRE System Library [jdk1.8.0_45]

```
package encapsulamento_1;

public class Verifica_Encapsulamento_1 {

    public static void main(String args []){

        Acesso_Numeros chave = new Acesso_Numeros();

        chave.a = 10;
        chave.b = 20;
        //chave.c = 30; NÃO É POSSÍVEL ACESSAR VARIÁVEL PRIVATE DIRETAMENTE
        chave.setNumero("c", 30);
        chave.d = 40;
        chave.mostra_numero();

    }

}
```

Problems @ Javadoc Declaration Console

<terminated> Verifica_Encapsulamento_1 [Java Application] C:\

```
numero a = 10
numero b = 20
numero c = 30
numero d = 40
```

```

package encapsulamento_2;

import encapsulamento_1.Acesso_Numeros;

public class Verifica_Encapsulamento_2 {
    public static void main(String args []){

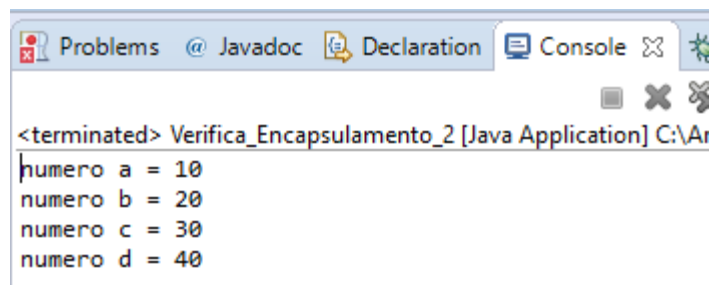
        Acesso_Numeros chave = new Acesso_Numeros();

        /*chave.a = 10; ENCAPSULAMENTO PUBLIC NÃO EXPLÍCITO
        NÃO PERMITE ACESSO DE FORA DO PACOTE*/
        chave.setNumero("a", 10);
        chave.b = 20;
        //chave.c = 30; NÃO É POSSÍVEL ACESSAR VARIÁVEL PRIVATE DIRETAMENTE
        chave.setNumero("c", 30);
        /*chave.d = 40; NÃO É POSSÍVEL ACESSAR VARIÁVEL PROTECTED DIRETAMENTE
        DE FORA DO PACOTE*/
        chave.setNumero("d", 40);

        chave.mostra_numero();

    }
}

```



The screenshot shows the 'Console' tab of a Java IDE. The title bar indicates the application is 'Verifica_Encapsulamento_2 [Java Application]' running at 'C:\Ar'. The output text in the console is as follows:

```

<terminated> Verifica_Encapsulamento_2 [Java Application] C:\Ar
numero a = 10
numero b = 20
numero c = 30
numero d = 40

```

Métodos Construtores

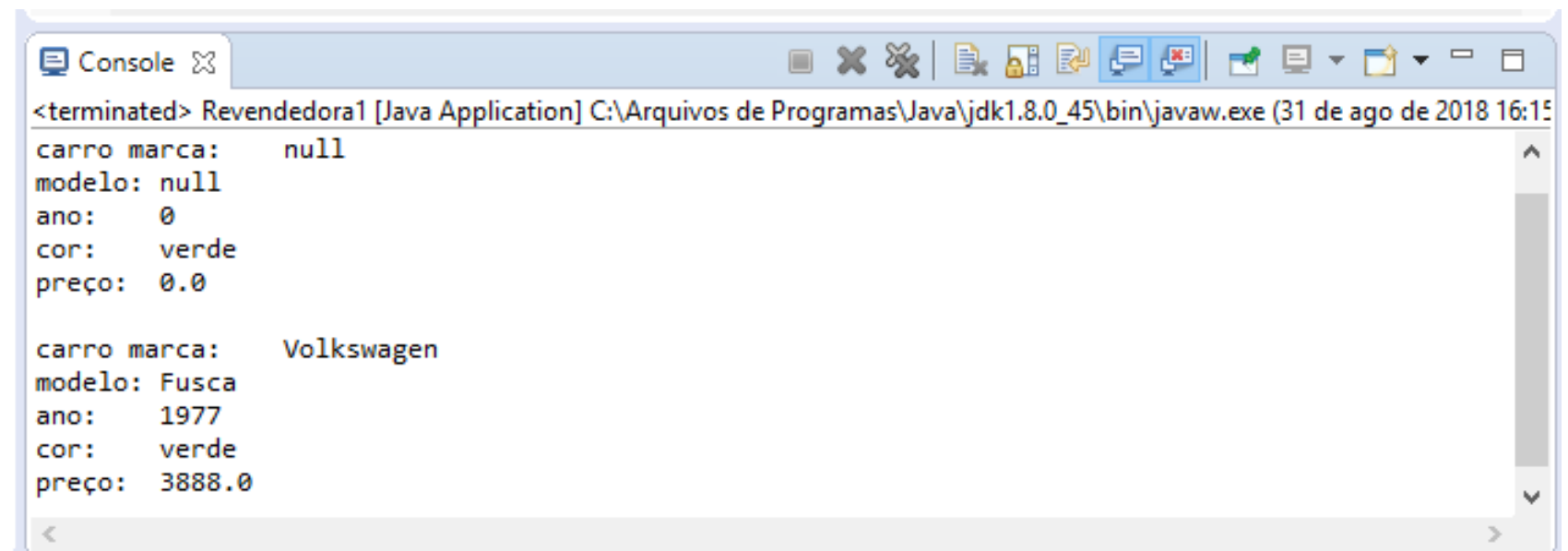
*Operador new atribui valores default a um objeto (variáveis numéricas = 0, valores lógicos = false, objetos = **null**). O método construtor constrói o objeto com valores; é invocado pelo operador new quando o objeto é criado e aloca espaço na memória para manipulação do objeto. Deve possuir o mesmo nome da classe.*

Classe Automovel

```
-
3 public class Automovel {
4
5     private int ano;
6     private String marca;
7     private String modelo;
8     private static String cor;
9     private double preco;
10
11     //método construtor
12     Automovel () {}
13     //método construtor
14     Automovel (int ano, String marca, String modelo, double preco) {
15         this.ano = ano;
16         this.marca = marca;
17         this.modelo = modelo;
18         this.preco = preco;
19     }
20
21     static {
22         cor = "verde";
23     }
24
25     public void mostracarro ()
26     {
27         System.out.println("\n" + "carro marca:" + "\t" + marca + "\n"
28             + "modelo:" + "\t" + modelo + "\n" + "ano:" + "\t" + ano + "\n"
29             + "cor:" + "\t" + cor + "\n" + "preço:" + "\t" + preco);
30     }
31 }
32
```

Classe Revendedora1

```
3 public class Revendedora1 {  
4  
5     public static void main (String args[]) {  
6  
7         Automovel A = new Automovel();  
8         A.mostracarro();  
9  
10        Automovel B = new Automovel(1977, "Volkswagen", "Fusca", 3888.00);  
11        B.mostracarro();  
12  
13    }  
14  
15 }  
16
```



The screenshot shows a Java IDE console window titled "Console". The output of the program is as follows:

```
<terminated> Revendedora1 [Java Application] C:\Arquivos de Programas\Java\jdk1.8.0_45\bin\javaw.exe (31 de ago de 2018 16:15  
carro marca:    null  
modelo: null  
ano:    0  
cor:    verde  
preço: 0.0  
  
carro marca:    Volkswagen  
modelo: Fusca  
ano:    1977  
cor:    verde  
preço: 3888.0
```


Métodos Destrutores (finalizers)

Liberam os recursos usados pelos objetos durante a execução do programa.

A linguagem Java possui um processo automático para limpeza de objetos não utilizados depois de um certo tempo, nomeado como “Coleta Automática de Lixo”(automatic garbage collection).

Referência this

Usa-se a referência this implicitamente para fazer referências às variáveis de instância e aos métodos de um objeto.

classe Automovel_II

```
3 import java.text.*;
4
5 public class Automovel_II {
6
7     private static int ano;
8     String marca;
9     String modelo;
10    String cor;
11    private static double preco;
12
13    NumberFormat nf = NumberFormat.getNumberInstance();
14
15    Automovel_II(int ano, String marca, double preco)
16        /* parâmetros dos atributos com
17        os mesmos nomes:
18        usa-se a referência this*/
19    {
20        this.ano = ano;
21        this.marca = marca;
22        modelo = "Scort";
23        cor = "amarelo";
24        this.preco = preco;
25    }
26
27    public static double atualizaPreco(double valor) {
28        return preco = ( valor * ano / 1988);
29    }
30
31    public void mostracarro () {
32        System.out.println("\n" + "carro marca" + "\t" + marca + "\n" + "modelo"
33        + "\t" + modelo + "\n" + "ano" + "\t" + ano + "\n" + "cor" + "\t" + cor +
34        "\n" + "preco" + "\t" + nf.format(atualizaPreco(7654.77)));
35    }
36 }
```

classe Revendedora2

```
3 public class Revendedora2 {  
4  
5     public static void main (String args[])  
6  
7     {  
8         Automovel_II C = new Automovel_II(2000, "Ford", 5000.00);  
9  
10        C.mostracarro();  
11  
12        //método finalizador  
13        C = null;  
14        System.gc();  
15        C.mostracarro();  
16    }  
17 }
```

Console

<terminated> Revendedora2 [Java Application] C:\Arquivos de Programas\Java\jdk1.8.0_45\bin\javaw.exe (31 de ago

```
carro marca      Ford  
modelo  Scort  
ano      2000  
cor      amarelo  
preco    7.700,976
```

Exception in thread "main" [java.lang.NullPointerException](#)
at encapsulamento.Revendedora2.main([Revendedora2.java:15](#))

Exercícios:

1. Compilar a classe Automovel e executar a classes Revendedora1; verificar os resultados.
2. Compilar a classe Automovel_II e executar a classe Revendedora2; verificar os resultados.