Programação Orientada a Objetos
ITE - 003

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 dever 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

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.*

Superclasse Automovel

```
public class Automovel {
    //variáveis de instância = atributos
    int ano;
    protected String marca;
    String modelo;
    protected static String cor;
    double preco;

    //método construtor
    Automovel( )
    {
      ano = 0;
      marca = "";
      modelo = "";
      cor = "";
      preco = 0.0;
    }

     public void atualizacor( )
  {
   cor = "azul";
  }


 public void mostracarro ()
     {
       System.out.println("carro marca" + "\t" + marca + "\n" + "modelo" + "\t" +
       modelo + "\n" + "ano" + "\t" + ano + "\n" + "cor" + "\t" + cor + "\n" + "preco" + "\t" + preco);
      }

}
```

subclasse Revendedora1

```
public class Revendedora1 extends Automovel {
    public static void main (String args[])

    {
        //objetos A e B
    Automovel B = new Automovel();
    Automovel A = new Automovel();

    B.mostracarro();
    A.mostracarro();


    B.marca = "Ford";
    B.modelo = "Belina";
    B.ano = 1980;
    B.cor = "verde";
    B.preco = 4800.00;

    B.mostracarro();

    B.atualizacor( );

    B.mostracarro();

    A.marca = "Volksvagen";
    A.modelo = "fusca";
    A.ano = 1977;
    A.preco = 3700.00;

    A.mostracarro();

    }
}
```

```
carro marca
modelo
ano0
cor
preco0.0
carro marca
modelo
ano0
cor
preco0.0
carro marcaFord
modeloBelina
ano1980
corverde
preco4800.0
carro marcaFord
modeloBelina
ano1980
corazul
preco4800.0
carro marcaVolksvagen
modelofusca
ano1977
corazul
preco3700.0
```

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 Revendedora2

```
public class Revendedora2 {

    public static void main (String args[])

    {
        //objetos C
    Revendedora1 C = new Revendedora1( );
    C.marca = "Volkswagen";
    C.modelo = "Kombi";
    C.ano = 1980;
    C.cor = "branco";
    C.preco = 5000.00;

    C.mostracarro();
    C.atualizacor();
    C.mostracarro();

    /*metodo finalizador*/
    C = null;
    System.gc();

    C.mostracarro();

    }
}
```

```
carro marcaVolkswagen
modeloKombi
ano1980
corbranco
preco5000.0
carro marcaVolkswagen
modeloKombi
ano1980
corazul
preco5000.0
Exception in thread "main"
java.lang.NullPointerException
at Revendedora2.main(Revendedora2.java:23)
```

Exercícios:

5. Compilar a classe Automovel e executar as classes Revendedora1 e Revendedora2; verificar os resultados.

6. Encapsular os atributos de Automovel como *private*; modificar as classes Revendedora1 e Revendedora2 de modo que os valores dos atributos possam ser inseridos e acessados através de métodos *get*( ) e *set*( ).

7. Modificar o método construtor na superclasse Automovel de modo que o mesmo possa receber valores que não sejam default aos atributos; modificar a classe Revendedora2 de modo que a mesma possa inserir valores aos atributos da superclasse Automovel através do método construtor e depois possa acessá-los.