

Encapsulamento



Prof. Jeferson Souza, MSc.

(thejefecom)

jeferson.souza@udesc.br



JOINVILLE
CENTRO DE CIÊNCIAS
TECNOLÓGICAS

Afinal, o que é Encapsulamento?

Definição de Encapsulamento

Encapsulamento não é:

- ▶ uma forma de embalar medicamentos;
- ▶ uma forma de empacotar programas;
- ▶ uma forma de virar borboleta;
- ▶ uma forma de proteger-se do frio :-D.

Afinal, o que é Encapsulamento?

Definição de Encapsulamento

Portanto, **Encapsulamento** pode ser definido como uma característica intrínseca presente nas linguagens de programação, onde seus elementos (e.g. variáveis, atributos, métodos, funções, classes, etc...) fazem parte de um contexto bem definido, o qual estabelece os limites da existência e do acesso aos referidos elementos.

Afinal, o que é Encapsulamento?

Definição de Encapsulamento

Encapsulamento também pode ser visto como uma boa prática de programação, a definir e restringir a existência e o acesso a elementos específicos, por meio de sua declaração em um contexto escolhido.

Pense em Encapsulamento Como...



Imagem: Pixabay (<https://pixabay.com>)

Encapsulamento como um cofrinho de porquinho. Pôr a moedinha no porquinho significa encapsular a mesma.

Qual é a definição de Escopo?

Definição de Escopo

Dentro do **Encapsulamento** o **Escopo** representa o contexto e as restrições associadas ao mesmo. Diz-se que um elemento está dentro do **Escopo** quando encontra-se dentro do contexto de sua definição; e diz-se que um elemento está fora do **Escopo** caso contrário.

Qual é a definição de Escopo?

Estou fora do Escopo, então...

Diz-se ainda que um elemento a tentar ser acessado fora do **Escopo** representa a violação do **Encapsulamento**.

Escopo de Variáveis na Orientação à Objetos

O que é uma variável local?

Uma **variável local** é uma variável definida dentro de um método.

Escopo de Variáveis na Orientação à Objetos

O que é uma variável local?

Uma **variável local** é uma variável definida dentro de um método.

O que é uma variável de instância?

Uma **variável de instância** é uma variável que representa um atributo da classe, o qual é instanciado juntamente com o objeto para poder ser utilizado.

O que é uma variável de classe?

Uma **variável de classe** é uma variável (i.e. atributo) que é partilhada por múltiplas instâncias da classe, a implicar que sua instanciação não ocorre com cada um dos objetos.

Exemplo de Variável Local em Java

```
public void executarAcao(){
```

```
    Boolean iniciaAcao = true; //A variável iniciaAcao representa uma  
    variável local.
```

```
}
```

Exemplo de Variável de Instância em Java

```
public class Pessoa {
```

```
    String nome; //A variável nome representa uma variável de instância,  
    i.e., um atributo da classe.
```

```
}
```

Exemplo de Variável de Classe em Java

```
public class Quadrado {
```

```
    static Double AREA_MAXIMA = 100.00; //A variável  
    AREA_MAXIMA representa uma variável de classe, i.e., um atributo  
    estático da classe.
```

```
}
```

Exemplo de Encapsulamento de Variáveis em múltiplas camadas em Java

```
public void executarAcao(){
```

```
    Boolean iniciaAcao = true; //A variável iniciaAcao representa uma  
    variável local.
```

```
    {
```

```
        boolean variavelEncapsuladaEmCamadas = true;
```

```
    }
```

```
    //Neste ponto a variável variavelEncapsuladaEmCamadas está fora de  
    escopo, mesmo a estar dentro do método.
```

```
}
```


Modificadores de Acesso em Java

O que são os Modificadores de Acesso?

Os modificadores de acesso estabelecem restrições ao *Encapsulamento* dos elementos presentes na linguagem. Cada modificador de acesso permite, ou não, o acesso de um elemento fora de seu escopo de definição.

Tipos de Modificadores de Acesso em Java

Podem ser de quatro tipos

- ▶ **public**: elemento pode ser acessado por qualquer classe;
- ▶ **private**: elemento pode ser acessado somente dentro da classe;
- ▶ **protected**: elemento pode ser acessado de classes do mesmo pacote e/ou subclasses;
- ▶ **Acesso padrão (privado no pacote)**: elemento pode ser acessado por qualquer classe dentro do mesmo pacote, ou por subclasses. Não existe modificador de acesso, basta omitir.

Classificadores Opcionais

Podem ser de seis tipos

- ▶ **static**: elemento faz parte da classe, sem precisar de uma instância da respectiva classe para ser acessado;
- ▶ **abstract**: classe precisa ser estendida e método precisa ter uma implementação realizada por uma subclasse (Veremos maiores detalhes em Herança);
- ▶ **final**: variável pode ser atribuída somente quando inicializada, e método não pode ser sobrescrito por uma subclasse;

Classificadores Opcionais (Continuação)

Podem ser de seis tipos (Continuação)

- ▶ **synchronized**: garante que um dado bloco somente poderá ser acessado de forma serializada, ie., uma linha de execução por vez;
- ▶ **native**: utilizado para interagir com código escrito em outra linguagem de programação, e.g., C++;
- ▶ **strictfp**: utilizado para tornar os cálculos de ponto flutuante portáteis.

PS: Não veremos com muita frequência a utilização destes classificadores opcionais por serem aplicados em domínios mais avançados, tais como Programação Paralela e Distribuída.

Exemplo de acesso a Elemento com Modificador de Acesso **[public]**

```
public class Principal {  
    public static void main(String ...args){  
        Animal animal = new Animal();  
        animal.setDescricao("Golfinho-Chileno (Cephalorhynchus eutropia)");  
    }  
    public class Animal {  
        private String descricao;  
        public void setDescricao(String descricao){  
            this.descricao = descricao;  
        }  
    }  
}
```

Os método público **setDescricao()** da classe **Animal** está a ser acessado dentro da classe **Principal**.

Exemplo de acesso a elemento com Modificador de Acesso **private**

```
public class Pessoa {  
    private String nome;  
    public String getNome(){  
        return this.nome;  
    }  
    public void setNome(String nome){  
        this.nome = nome;  
    }  
}
```

O atributo **nome** está a ser acessado dentro da classe nos métodos **getNome()** e **setNome()**.

Exemplo de acesso a Elemento com Modificador de Acesso **protected**

```
public class Principal {  
    public static void main(String ...args){  
        Pessoa pessoa = new Pessoa();  
        pessoa.nome = "Odete";  
        System.out.println(pessoa.nome);  
    }  
    class Pessoa {  
        protected String nome;  
    }  
}
```

O atributo **nome** da classe **Pessoa** está a ser acessado dentro da classe **Principal** (acesso **protected**).

Exemplo de acesso a Elemento sem Modificador de Acesso [Acesso padrão (privado no pacote)]

```
public class Principal {  
    public static void main(String ...args){  
        Carro carro = new Carro();  
        carro.denominacao = "Fusca Bola";  
        System.out.println(carro.denominacao);  
    }  
}  
class Carro {  
    String denominacao;  
}
```

O atributo **denominacao** da classe **Carro** está a ser acessado dentro da classe **Principal** [Acesso padrão (privado no pacote)].

Classes Aninhadas (Nested Classes) em Java

O que são Classes Aninhadas (Nested Classes)?

As **Classes Aninhadas (Nested Classes)** são classes que podem ser declaradas dentro de outras classes. São de quatro tipos:

- ▶ **Classe Membro (Member Inner Class):** são declaradas como variáveis de instância da classe;
- ▶ **Classe Local (Local Inner Class):** são declaradas dentro de métodos;
- ▶ **Classe Anônima (Anonymous Class):** tipo especial de classe local que não possui nome;
- ▶ **Classe Estática (Static Nested Class):** são declaradas como variáveis de classe.

Exemplo de Classe Membro (Member Inner Class)

```
public class Veiculo {  
    private String nome;  
    private class Interior {  
        private Boolean comercial;  
        public void setComercial(Boolean comercial){  
            this.comercial = comercial;  
        }  
    }  
    private Interior interior;  
    public void criaInterior(Boolean comercial){  
        this.interior = new Interior();  
        this.interior.setComercial(comercial);  
    }  
}
```

Neste exemplo a classe *Interior* é uma **Classe Membro** da classe **Veiculo**.

Exemplo de Classe Local (Local Inner Class) [Boyarsky&Selikoff, 2015]

```
public class Outer {  
    private int tamanho = 5;  
    public void calcular() {  
        final int largura;  
        class Inner {  
            public void multiplicar(){  
                System.out.println(tamanho * largura);  
            }  
        }  
        Inner inner = new Inner();  
        inner.multiplicar();  
    }  
    public static void main(String[] args){  
        Outer outer = new Outer();  
        outer.calcular();  
    }  
}
```

Neste exemplo a classe *Inner* é uma **Classe Local** do método **calcular()**.

Exemplo de Classe Anônima (Anonymous Class) [Boyarsky&Selikoff, 2015]

```
public class AnonInner {  
    abstract class VendaSomenteHoje {  
        abstract int descontoDollar();  
    }  
    public int admissao(int precoBase){  
        VendaSomenteHoje venda = new VendaSomenteHoje(){  
            int descontoDollar() { return 3;}  
        };  
        return precoBase - venda.descontoDollar();  
    }  
}
```

Neste exemplo a declaração de uma subclasse da classe *VendaSomenteHoje* é uma **Classe Anônima** do método **admissao()**.

Exemplo de Classe Estática (Static Nested Class)

```
public class Principal {  
    private static class Carro {  
        private String denominacao;  
    }  
    public static void main(String ...args){  
        Carro carro = new Carro();  
        carro.denominacao = "Fusca Bola";  
        System.out.println(carro.denominacao);  
    }  
}
```

Neste exemplo a classe *Carro* é uma **Classe Estática** da classe **Principal**.

Bibliografia

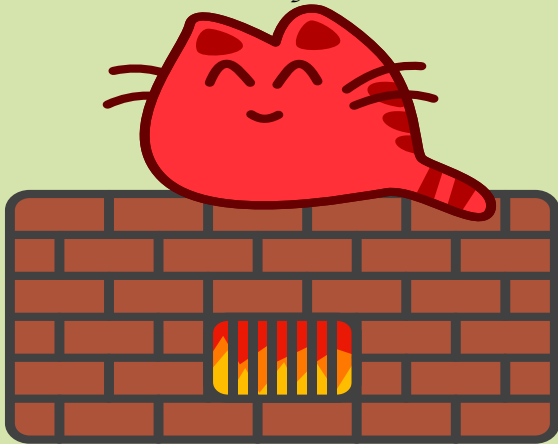


BOYARSKY, J. and SELIKOFF, S. *“Oracle Certified Associate Java SE 8 Programmer I: Study Guide”*. Sybex. Indianápolis, Indiana. 2015.



BOYARSKY, J. and SELIKOFF, S. *“Oracle Certified Associate Java SE 8 Programmer II: Study Guide”*. Sybex. Indianápolis, Indiana. 2016.

That's it folks!



Thank you for your attention!