Sobre Encapsulamento

Mário Leite

...

Encapsulamento é um conceito muito importante para a Programação Orientada a Objetos (OOP) porque contribui, objetivamente, para diminuir os malefícios causados pela interferência externa sobre os *objetos*. Partindo deste princípio, toda e qualquer transação com os *objetos* só pode ser feita através de procedimentos colocados "dentro" deles, acessados por meio do envio de mensagens. Desta maneira, dizemos que um elemento (dado ou rotina) está *encapsulado* quando envolvido por código de forma que só é visível na rotina onde foi criado. O mesmo acontece com uma rotina, que, sendo encapsulada, suas operações internas são invisíveis às outras rotinas. Isto deve ser sempre a preocupação dos programadores, mesmo em Programação Estruturada Procedural; e uma boa técnica é sempre definir variáveis locais. E caso essas variáveis devam ser vistas por outras rotinas, é melhor serem passadas como parâmetros, mas deve ser evitado declará-las como globais; neste contexto, isto seria uma maneira correta de abstrair. No caso da OOP, preservar a integridade dos campos das propriedades de uma classe através de encapsulamentos é fundamental. Como exemplo de emprego do Encapsulamento na vida real, podemos considerar um aparelho de DVD (para os mais antigos pode ser Vídeo Cassete), onde existem os tradicionais botões de **[Power]**, **[Stop]**, **[Rewind]**, **[Start]**, etc, entre outras funcionalidades. Esses botões executam operações existentes no aparelho, ativadas pelos componentes **internos**. Não interessa ao operador saber como é o funcionamento do equipamento, pois essa informação só é relevante para o projetista do aparelho. As informações disponíveis ao usuário do equipamento são as existentes no meio externo (*console* e *controle remoto*) que ativam/desativam as operações. Deste modo, o aparelho pode evoluir com os avanços tecnológicos e os usuários que o utilizam continuam sabendo utilizá-lo, sem a necessidade de um novo treinamento básico dos recursos preexistentes. A ideia por trás deste conceito é de que a utilização de um objeto não deve depender de sua implementação interna, mas sim de sua "aparência" (interface) externa. Em outras palavras, "o que o usuário vê" deve ser apenas aquilo que ele "precisa ver", e não como o objeto funciona. O esquema da **figura 1** mostra um exemplo clássico de um aparelho de DVD. Nesta figura as partes visíveis (*painel de controle* + *controle remoto*) formam a interface do "objeto" DVD (na realidade uma classe DVD, pois todos os DVD's possuem uma interface pública). Seu mecanismo de funcionamento é invisível ao usuário; diz-se que esse mecanismo está ENCAPSULADO, protegido. Assim, para dar um ***Rewind*** ou um ***Play*** no disco basta acionar um botão do controle remoto, ou no próprio console; mas como o DVD faz isto não diz respeito ao usuário, pois, essa complexidade é resolvida pelo objeto e não por quem o acessa. Deste modo, deve estar bem separado o que "o objeto expõe" na interface e "o que ele faz internamente" para responder aos requisitos na interface. A **figura 2** ilustra, de modo esquemático, o nosso objeto DVD tratado como uma classe no contexto da orientação a objetos. A **figura 3** mostra o resultado da solicitação de desligar o DVD ao pressionar o botão **[Stop]** do console. A ação do usuário é externa, mas, em vez dessa ação afetar diretamente o aparelho, o que acontece, de verdade é que esta solicitação é passada para a funcionalidade interna (*método privado*) para que o aparelho seja, finalmente, desligado. Isto quer dizer que, embora possa parecer que o aparelho é desligado diretamente pelo usuário, na verdade, isto é feito por um mecanismo interno (método privado) do aparelho. Assim, caso haja alguma melhoria nesse mecanismo interno de desligamento isto ficará transparente ao usuário, que nunca irá saber como é feita essa operação: ele vai continuar pressionando a tecla **[Stop]** sem saber o que se passa no interior do aparelho; esta é a missão do Encapsulamento. Já pensou se o usuário pudesse, ele mesmo, meter o dedo no interior do aparelho para desliga-lo?! Portanto, a ação de desligar se passa assim:

Pressiona botão [Stop] ==> Aciona método externo ==> Aciona método interno ==> DVD Desliga

---

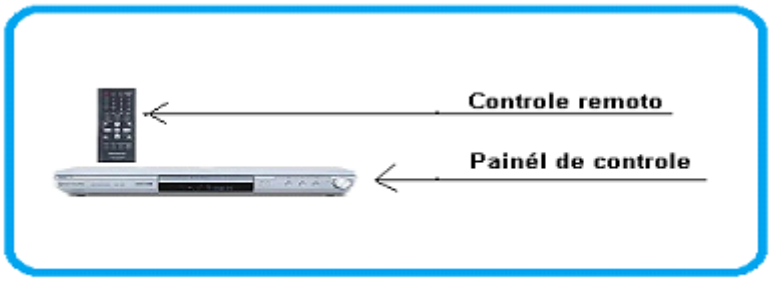

Figura 1 - O "objeto" DVD visto pelo usuário

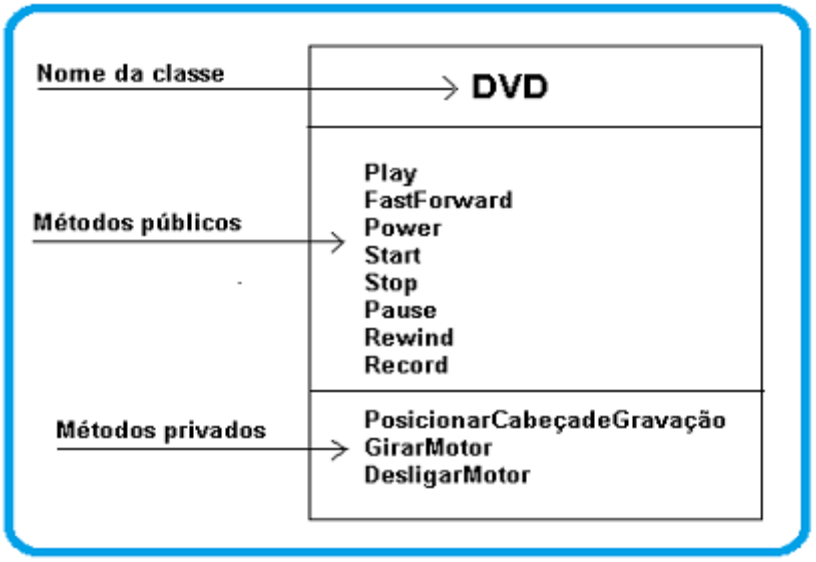

Figura 2 - O objeto DVD no contexto da orientação a objetos

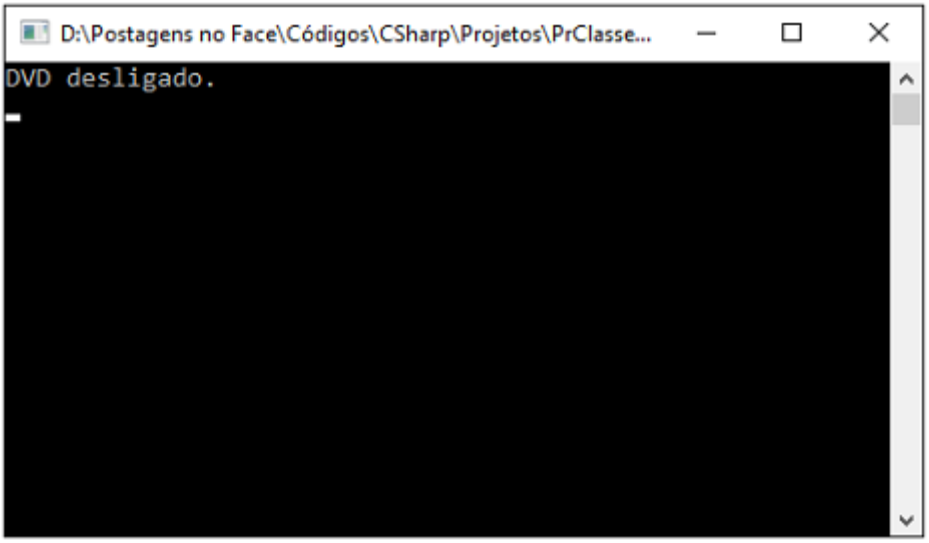

Figura 3 - O desligamento do DVD

```
//Programa "Encapsulamento"
//Cria uma classe para explicar como funciona o Encapsulamento
//Codificado em C#
//Autor: Mário Leite
//-----------------------------------------------------------------------
class AparelhoDVD  //cria a classe
{
    //Propriedades (campos) da classe
    private string _marca;
    private string _modelo;
    private int _quadros;

    //Métodos (funcionalidades) da classe
    private static void DesligarMotor()  //método privado
    {
        //O DVD é desligado de acordo com o projeto do aparelho
        Console.WriteLine("DVD desligado.");
    }
    public void Stop()  //método público
    {
        AparelhoDVD.DesligarMotor();  //invoca (internamente) o método privado
    }
}

static void Main(string[] args)
{
    AparelhoDVD objDVD = new AparelhoDVD();  //cria instância da classe
    objDVD.Stop();  //invoca (externamente) o método público
    Console.ReadKey();
}
```