UNIP
UNIVERSIDADE PAULISTA

UNICLUSTER

Encapsulamento

Carlos Arruda Baltazar
UNIP – Cidade Universitária

Encapsulamento

Encapsulamento é uma técnica associada ao paradigma de programação orientada a objetos que possibilita que os detalhes internos de uma classe permaneçam ocultos para outros objetos. O conhecimento a respeito da implementação interna da classe é desnecessário do ponto de vista do objeto, uma vez que isso é responsabilidade dos métodos implementados na classe.

Encapsulamento

No ato de desenvolver uma classe, seu código e seus atributos, também denominados de membros da classe, são determinados. Uma vez que os atributos e métodos de uma classe podem possuir modificadores públicos ou privados, temos a seguinte situação:

Encapsulamento

- Tudo que objeto externo precisa conhecer a respeito de uma classe encontra-se em atributos e métodos que possuem um modificador público;

- Os métodos e atributos que possuem um modificador privado só podem ser conhecidos e invocados por membros desta mesma classe. Isso garante que não ocorrerão ações inadequadas.

Encapsulamento

Compreendido tudo isso, pode-se concluir que a única forma de conhecer ou alterar os atributos de um objeto é por meio de seus métodos. Dentre as vantagens fornecidas pela técnica do encapsulamento estão:

Encapsulamento

- O objeto é disponibilizado com todas as suas funcionalidades sem que seja necessário conhecer seu funcionamento e armazenamento internos;

- É possível editar um objeto internamente, acrescentando métodos, sem que cause impactos em outros componentes do sistema que utilizem o objeto modificado;

Encapsulamento

- O processo de desenvolvimento é otimizado, já que para utilizar um objeto não se faz necessário conhecer sua constituição e funcionamento interno;

- Os comportamentos do objeto podem ser modificados sem que haja impacto no resto do programa, pois o resto do programa não tem dependência com a forma que ele é implementado.

Encapsulamento

A ideia é tornar o software mais flexível, fácil de modificar e de criar novas implementações. Serve para controlar o acesso aos atributos e métodos de uma classe. É uma forma eficiente de proteger os dados manipulados dentro da classe, além de determinar onde esta classe poderá ser manipulada.

Encapsulamento

Usamos o nível de acesso mais restritivo, private, que faça sentido para um membro particular. Sempre usamos private, a menos que tenhamos um bom motivo para deixá-lo com outro nível de acesso. Não devemos permitir o acesso público aos membros, exceto em caso de ser constantes. Isso porque membros públicos tendem a nos ligar a uma implementação em particular e limita a nossa flexibilidade em mudar o código.

Encapsulamento

- O encapsulamento que é dividido em dois níveis:
  - Nível de classe: Quando determinamos o acesso de uma classe inteira que pode ser public ou Package-Private (padrão);
  - Nível de membro: Quando determinamos o acesso de atributos ou métodos de uma classe que podem ser public, private, protected ou Package-Private (padrão).

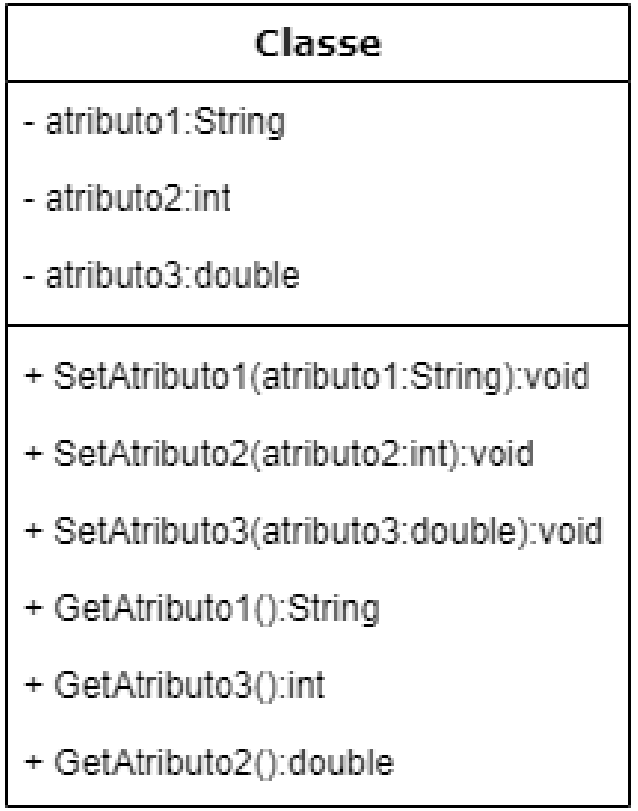
Encapsulamento

Então para ter um método encapsulado utilizamos um modificador de acesso que geralmente é public, além do tipo de retorno dele. Para se ter acesso a algum atributo ou método que esteja encapsulado utiliza-se o conceito de get e set. Por definição, com SET é feita uma atribuição a algum atributo, ou seja, define, diz o valor que algum atributo deve ter. E com GET é possível recuperar esse valor.

Encapsulamento de Atributos

- ***Setters***: métodos responsáveis por inserir e alterar valores contidos nos atributos.

- ***Getters***: métodos responsáveis por retornar o valor de um dado atributo.



Encapsulamento de Atributos

- ***Setters***: métodos responsáveis por inserir e alterar valores contidos nos atributos.

- ***Getters***: métodos responsáveis por retornar o valor de um dado atributo.

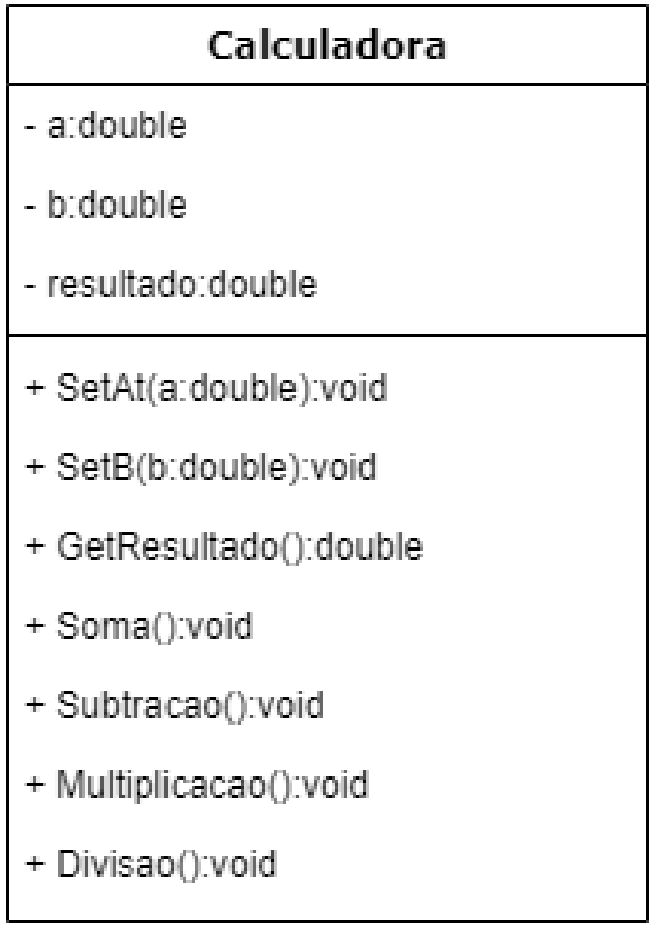
```
public class Classe
{
    private String atributo1;
    private int atributo2;
    private double atributo3;

    public void SetAtributo1(String atributo1)
    {
        this.atributo1 = atributo1;
    }
    public void SetAtributo2(int atributo2)
    {
        this.atributo2 = atributo2;
    }
    public void SetAtributo3(double atributo3)
    {
        this.atributo3 = atributo3;
    }

    public String GetAtributo1()
    {
        return this.atributo1;
    }
    public int GetAtributo2()
    {
        return this.atributo2;
    }
    public double GetAtributo3()
    {
        return this.atributo3;
    }
}
```

Encapsulamento

Então para ter um método encapsulado utilizamos um modificador de acesso que geralmente é public, além do tipo de retorno dele. Para se ter acesso a algum atributo ou método que esteja encapsulado utiliza-se o conceito de get e set. Por definição, com SET é feita uma atribuição a algum atributo, ou seja, define, diz o valor que algum atributo deve ter. E com GET é possível recuperar esse valor.

Encapsulamento de código

- Implica em dividir um código em trechos menores;
- Encapsular estes trechos de códigos em estruturas (funções e/ou procedimentos) que podem ser chamadas ao longo da execução;
- Estas estruturas devem executar uma única tarefa com base em um conjunto de entrada e prover um conjunto de saídas após o processamento;
- Propicia a reutilização de código;
- Otimiza o desenvolvimento;



Encapsulamento de código

- Implica em dividir um código em trechos menores;
- Encapsular estes trechos de códigos em estruturas (funções e/ou procedimentos) que podem ser chamadas ao longo da execução;
- Estas estruturas devem executar uma única tarefa com base em um conjunto de entrada e prover um conjunto de saídas após o processamento;
- Propicia a reutilização de código;
- Otimiza o desenvolvimento;

```
public class Calculadora
{
    private double a;
    private double b;
    private double resultado;

    public void SetA(double a)
    {
        this.a = a;
    }
    public void SetB(double b)
    {
        this.b = b;
    }

    public double GetResultado()
    {
        return this.resultado;
    }

    public void Soma()
    {
        this.resultado = this.a + this.b;
    }

    public void Subtracao()
    {
        this.resultado = this.a - this.b;
    }

    public void Multiplicacao()
    {
        this.resultado = this.a * this.b;
    }

    public void Divisao()
    {
        this.resultado = this.a / this.b;
    }
}
```

OBRIGADO