3 – Encapsulamento

Entenda como encapsulamento o ato de proteger características de nossos objetos, isto quer dizer que o que não for público estará protegido de ação externa.

Como exemplo iremos pegar um motor. Não precisamos saber o funcionamento dele, apenas temos que saber que o método Ligar() da nossa classe Motor irá fazer com que o nosso motor entre em funcionamento. Mas não precisamos saber quais componentes o método irá acessar, modificar ou criar para que o nosso motor entre em funcionamento.

O que o método ligar faz, não nos diz respeito, apenas usamos e sabemos que vai funcionar.

No conceito de orientação a objetos podemos proteger aquilo que não queremos que sofram intervenção, nossas variáveis locais, nossos métodos e atributos.

Os dados protegidos só podem ser alterados dentro dos métodos em que foram declarados ou dentro do objeto a que eles pertencem.

Exemplo:

```
public class Programa
{
    public class Motor
    {
        public bool Ligado { get; private set; }
        public int gasolina { get; set; }

        public void Ligar()
        {
            bool temGasolina = TemGasolina();

            if (temGasolina) //a mesma coisa que if(temGasolina == true)
            {
                this.Ligado = true;
                Console.WriteLine("O motor foi ligado.");
            }
            else
                Console.WriteLine("Não temos gasolina para ligar o motor.");
        }

        private bool TemGasolina()
        {
            if (this.gasolina > 0)
                return true;
            else
                return false;
        }
    }

    public static void Main()
    {
```

```
            Motor meumotor = new Motor();
            meumotor.gasolina = 1;
            meumotor.Ligar();
            //Console.WriteLine("Tem gasolina? {0}",meumotor.TemGasolina()); //para isso tire o private
do método
            Console.WriteLine("O estado do motor é {0}", meumotor.Ligado);
            Console.ReadKey();
        }
    }
```

Exercícios de Encapsulamento

1 – Crie uma classe chamada Retângulo. Esta classe tem como atributos encapsulados altura e largura (ou seja, estas propriedades não serão visíveis pelo objeto). Crie dois métodos da seguinte forma: o primeiro para pedir e atribuir valor para altura e o segundo para pedir e atribuir valor para largura.

2 – Crie mais dois métodos chamados area e perimetro, onde estes, quando chamados, retornam a área e o perímetro do retângulo (área do retângulo = largura * altura e o perímetro do retângulo é dado por (largura * 2) + (altura * 2)).

3- Crie uma classe chamada trianguloretangulo. Esta classe tem como atributos encapsulados a catetobase e a catetoaltura. Crie dois métodos para pedir e atribuir valores separadamente para base e altura como no exercício 1.

4 - Crie mais um atributo encapsulado chamado hipotenusa. Este atributo não será informado pelo usuário. Ele será calculado através de um método que utiliza os valores dos catetos lidos no exercício 3 e determinados através da fórmula $a^2 = b^2 + c^2$ - Obs. Raiz quadrada em C#: Math.Sqrt(número). Crie o método e calcule a hipotenusa.

5- Crie um método que mostra o valor da hipotenusa calculado no exercício 4. Crie também mais dois métodos: um para retornar a área do triângulo (catetobase * catetoaltura) / 2 e perímetro do triângulo (catetobase + catetoaltura + hipotenusa).