```
//GETTERS E SETTERS:
//Os métodos getters e setters são usados para encapsularmos os nossos atributos e métodos dentro do conceito de POO...
//Porém, devemos lembrar que o Javascript não é uma linguagem voltada totalmente para o contexto O.O., ela é uma linguagem multiparadigma que permite a usabilidade do O.O. porém do seu próprio jeito.
//Método Getter: Permitem apenas a visualização de atributos e métodos privados...
//Método Setter: Permite a alteração de atributos e métodos privados...

//CONVENÇÃO DOS ELEMENTOS PRIVADOS NO JAVASCRIPT:
//No Javascript, por convenção os desenvolvedores colocam o underline antes dos nomes dos elementos privados, da seguinte forma:
const sequencia = { //Estamos usando objeto por que como havemos de lembrar o POO lida com objetos, conjuntos de par chave e  valor...

    _valor: 1, //Perceba que colocamos um underline antes do nome, isso por si só não impede que a chave seja acessada diretamente, isso é apenas uma convenção da linguagem, onde os desenvolvedores sabem aque ao ver um elemento como este, ele é um elemento privado de um objeto...
    get valor() {return this._valor}, //O método "get" é reconhecido automáticamente pelo javascript, quando o get é seguido por uma variável ou método o javascript já sabe que se trata de um método de visualização...
    set valor(valor) { //O método "set" também é reconhecido imediatamente pelo javascript, sabemos que quando estamos usando set, queremos fazer alguma alteração em atributos ou métodos...
        if (valor > this._valor){ //Aqui temos um condicional no nosso métodos "set", ele não permite que valores abaixo do valor já exitente no atributo valor seja atribuído ao atributo do objeto valor...
            this._valor = valor
        } else {
            console.log('O valor está abaixo do valor atual, use um valor mais alto')
        }
    }

}



// ACESSANDO GETTERS E SETTERS:
// Aqui está o pulo do gato!! O javascript já reconhece automáticamente a hora de usar um get e um set, não precisamos colocar os nomes get e set quando chamamos por eles, no exemplo abaixo temos um get chamado duas vezes, lembre-se que o nosso
console.log('\n1)', sequencia.valor) //Aqui não estamos atribuíndo nada ao valor, estamos apenas imprimindo, o javascript já sabe que deve usar um get...
```

```
sequencia.valor = 0 //Os métodos set são ativados sempre que fazemos uma atribuição, eles não são ativados como as funções comuns onde colocávamos o campo de parâmetros e aos valores nos argumentos deles...
console.log(sequencia.valor) //Veja que a atribuição acima foi feita com um valor abaixo do valor atual, ocasionando que o set entrasse na condicional "else"...

sequencia.valor = 1000
console.log('\n2)', sequencia.valor) //Quando o set usa um valor acima do valor atual do atributo, o set entra na condicional "if"...

sequencia.valor = 900 //Mas quando usamos set com um número abaixo do atual novamente, o set entra na condicional "set"...
console.log('\n3)', sequencia.valor) //Como resultado ele mostra o último número que deveria ser...
```

RESULTADO NO CONSOLE...

```
[Running] node "c:\Users\Almoxarifado\Documents\javascript\arquivos_das_aulas\078-Getters_e_Setters.js"

1) 1
O valor está abaixo do valor atual, use um valor mais alto
1

2) 1000
O valor está abaixo do valor atual, use um valor mais alto

3) 1000

[Done] exited with code=0 in 0.1 seconds
```