**Exercício 1: Classe Encapsulada Produto**

Escreva uma classe encapsulada chamada Produto. Ela deve ter os seguintes atributos privados:

- _codigo (number)
- _nome (string)
- _preco (number)

A classe deve ter um **construtor** que inicializa todos os três atributos. Além disso, implemente os seguintes métodos públicos:

- Um método get para cada atributo (getCodigo, getNome, getPreco).
- Um método setPreco(novoPreco: number) que altera o preço. Este método deve impedir que um valor negativo seja atribuído; caso isso ocorra, o preço deve ser definido como 0.

---

**Exercício 2: Classe com Composição Estoque**

Crie uma classe chamada Estoque. Ela deve possuir um atributo privado:

- _produtos: um array (lista) de objetos da classe Produto, inicializado como um array vazio[3].

A classe deve ter os seguintes métodos públicos:

- adicionarProduto(produto: Produto): insere um objeto Produto no array _produtos[4].
- removerProduto(codigo: number): remove um produto do array com base em seu código.
- listarProdutos(): imprime no console os detalhes de todos os produtos no estoque.
- valorTotalEstoque(): retorna o valor total da soma dos preços de todos os produtos no estoque.

---

**Exercício 3: Herança com ContaBancaria**

1. Crie uma classe base chamada ContaBancaria com os seguintes atributos protected:
    - _numeroConta (string)
    - _saldo (number)
2. A classe deve ter um construtor para inicializar o número da conta e o saldo.
3. Crie um método público depositar(valor: number) que adiciona ao saldo e um método sacar(valor: number) que subtrai do saldo (se houver fundos suficientes).

4. Agora, crie uma classe ContaPoupanca que **herda** de ContaBancaria. A ContaPoupanca deve ter um atributo privado adicional:
    - _taxaJuros (number)
5. O construtor de ContaPoupanca deve receber o número da conta, o saldo inicial e a taxa de juros, e deve chamar o construtor da classe pai (super) para inicializar os atributos herdados.
6. Adicione um método calcularJuros() em ContaPoupanca que aumenta o saldo com base na taxa de juros.

---

**Exercício 4: Sobrescrita de Métodos**

1. Utilizando a classe ContaBancaria do exercício anterior, adicione um método público chamado getDescricao() que retorna uma string como: "Conta: 123-4, Saldo: R$ 1000.00".
2. Na classe ContaPoupanca, **sobrescreva** o método getDescricao(). A nova versão deve incluir a taxa de juros na descrição, retornando uma string como: "Conta Poupança: 567-8, Saldo: R$ 2500.00, Juros: 0.5%".

---

**Exercício 5: Atributos readonly e static**

Crie uma classe Matematica que não precisa ser instanciada para ser usada. Ela deve ter:

1. Um atributo readonly e public chamado PI, inicializado com o valor 3.14159. Como é

readonly, seu valor não poderá ser alterado após a inicialização[8].

2. Um método static e public chamado isPar(numero: number) que retorna true se um número for par e false caso contrário.
3. Um método static e public chamado calcularAreaCirculo(raio: number) que utiliza o atributo PI para retornar a área de um círculo ($A=\pi \cdot r2$).

**Como testar:**

- Você deve conseguir acessar

Matematica.PI e Matematica.isPar(2), por exemplo, sem precisar criar um objeto com new Matematica()[9].

- Qualquer tentativa de alterar o valor de PI (ex: Matematica.PI = 3;) deverá resultar em um erro do compilador.