Faculdade IMPACTA	**Programação Orientada a Objetos** **Encapsulamento** **Exercícios**

Exercício 01

Altere a classe Funcionario do programa abaixo:

- Transforme os seus atributos em atributos privados.
- Crie os métodos get e set para todos os atributos.
- Faça as alterações necessárias para que o programa principal funcione corretamente.

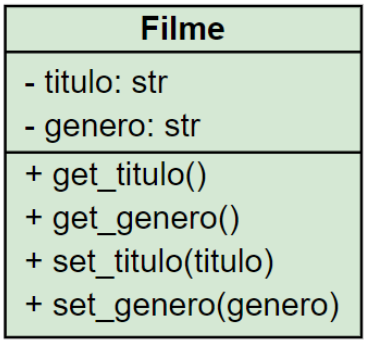
```
class Funcionario:
    def __init__(self, nome, cpf, salario):
        self.nome = nome
        self.cpf = cpf
        self.salario = salario


func1 = Funcionario("Pedro", "111222333-22", 1500.0)
func1.salario = 2000.0              # Altera salário
print("Nome:", func1.nome)
print("CPF:", func1.cpf)
print("Salário:", func1.salario)
```

Exercício 02

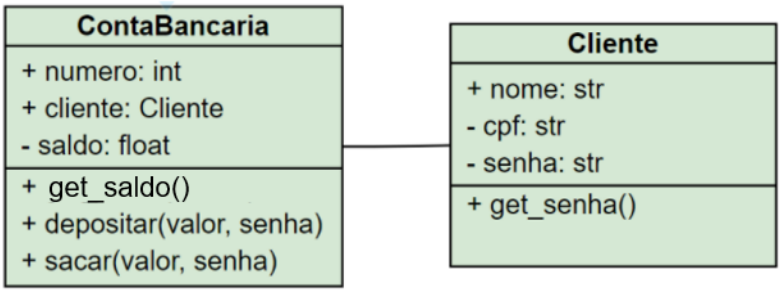
- Implemente a classe Filme, conforme o diagrama de classes abaixo
  - Todos os atributos devem ser *privados*
  - Crie os métodos *get* e *set* para todos os atributos

- No seu programa principal, faça a seguinte implementação:
  - Criar uma lista de filmes vazia
  - Cadastrar 3 filmes (com os dados informados pelo usuário)
  - Armazenar os dados de cada filme em um objeto da classe Filme
  - Armazenar os objetos na lista de filmes
  - Exibir um relatório com os dados de todos os filmes cadastrados



Exercício 03

Implemente as classes ContaBancaria e Cliente, conforme o diagrama de classes abaixo.



Classe Cliente

Atributos (todos definidos no construtor):

- **nome**: nome do cliente (público)
- **cpf**: cpf do cliente (privado)
- **senha**: senha do cliente (privado)

Métodos:

- **get_senha**: retorna a senha do cliente

Classe ContaBancaria

Atributos:

- **numero**: numero da conta (público)
- **cliente**: objeto Cliente associado à conta (público)
- **saldo**: saldo da conta (privado). Deve começar sempre com zero.

Métodos:

- **get_saldo**: retorna o saldo da conta.
- **depositar**: recebe como parâmetros de entrada um valor e uma senha. Acrescenta esse valor ao saldo da conta apenas se a senha for igual à senha do cliente.
- **sacar**: recebe como parâmetros de entrada um valor e uma senha. Subtrai esse valor do saldo da conta, apenas se a senha for igual à senha do cliente.

Utilize o código abaixo para testar as suas classes

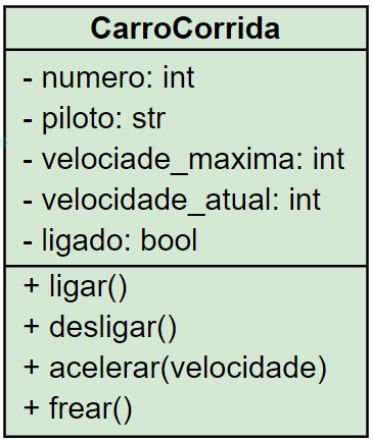
```
cliente1 = Cliente("João", "111111111", "123")
conta = ContaBancaria(1111, cliente1)

conta.depositar(200, "123")
print(conta.get_saldo())            # Imprime 200
conta.sacar(50, "123")
print(conta.get_saldo())            # Imprime 150

conta.depositar(100, "111")         # senha inválida
print(conta.get_saldo())            # Imprime 150
conta.sacar(50, "111")              # senha inválida
print(conta.get_saldo())            # Imprime 150
```

Exercício 04

Implemente a classe CarroCorrida



Classe CarroCorrida

Atributos (*todos privados*):

- **numero**: número de identificação do carro
- **piloto**: nome do piloto do carro ao carro
- **velocidade_maxima**: velocidade máxima do carro em km/h
- **velocidade_atual**: velocidade atual do carro em km/h (valor inicial deve ser zero)
- **ligado**: informa se o carro está ligado (valor inicial deve ser False)

Métodos:

- **ligar**: define o carro como ligado
- **desligar**: define o carro como desligado
- **acelerar**: aumenta velocidade atual do carro em *N* km/h. O carro só pode acelerar se estiver ligado. Ao acelerar, o carro nunca pode ultrapassar a sua velocidade máxima
- **frear**: define a velocidade atual do carro em 0 km/h.
- Criar os métodos ***get*** e ***set***, quando for necessário.

Utilize o trecho de programa abaixo para testar a sua classe

```
carro = CarroCorrida(1, "Paulo", 150)
carro.acelerar(20)
print(carro.get_velocidade_atual())         # imprime 0, porque o carro está desligado
carro.ligar()
carro.acelerar(20)
print(carro.get_velocidade_atual())         # imprime 20
carro.acelerar(80)
print(carro.get_velocidade_atual())         # imprime 100
carro.acelerar(70)
print(carro.get_velocidade_atual())         # imprime 150, não ultrapassar a vel. max.
carro.frear()
print(carro.get_velocidade_atual())         # imprime 0
```
