






Exercícios

 Comentários	Exercícios para fixar e avaliar os conceitos estudados.
 Link	https://www.notion.so/starters/Exerc-cios-8e77666e51c48ffae82a2d5bf8b323e
 Status	

Todas as resoluções devem ser submetidas no seu gitlab da GFT.

Exercício 1

Seguindo o diagrama UML a seguir, implemente a classe **Veiculo** e seus métodos. Após a implementação, testar cada um dos métodos via console.

- `acelerar()`: este método acrescenta um valor de **+20** no atributo velocidade.
- `abastecer(int combustivel)`: recebe como parâmetro uma quantidade de combustível e atribui a **listrosCombustivel**. **OBS: O limite do tanque de combustível é de 60 litros, validar para não ultrapassar.**
- `frear()`: a cada chamada do método diminui a velocidade em 20. Não aceitar a chamada do método se o veiculo estiver parado.
- `pintar(String cor)`: recebe uma cor como parâmetro e altera o atributo.
- `ligar()`: Verifica se o veículo já se encontra ligado, caso não, liga o carro.
- `desligar()`: Verifica se o veículo já se encontra desligado, caso não, desliga o carro. Não permitir que desligue o veículo com **(velocidade > 0)**.

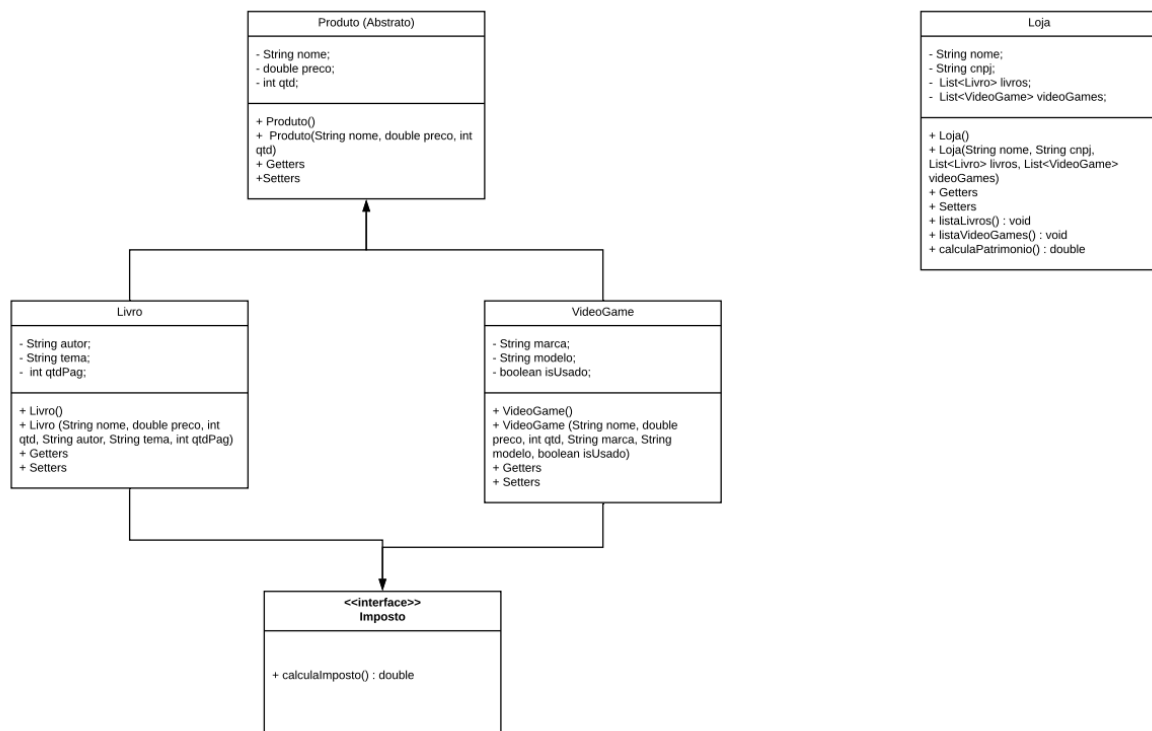
<i>Veículo</i>
<p><i>-Marca: String</i></p> <p><i>-Modelo: String</i></p> <p><i>-Placa: String</i></p> <p><i>-Cor : String</i></p> <p><i>-Km: float</i></p> <p><i>-isLigado: boolean</i></p> <p><i>-litrosCombustivel: int</i></p> <p><i>-Velocidade: int</i></p> <p><i>-Preco: Double</i></p>
<p><i>+getters / setters</i></p> <p><i>+ acelerar()</i></p> <p><i>+ abastecer()</i></p> <p><i>+ frear()</i></p> <p><i>+ pintar()</i></p> <p><i>+ ligar()</i></p> <p><i>+ desligar()</i></p>

Exercício 2

Seguindo o diagrama UML a seguir, implemente as classes, interfaces e as saídas do programa. Após a implementação, testar cada um dos métodos via console.

Descrição dos métodos em anexo:

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/aa0c4154-2871-4974-ad64-f8acdabcbe34/Ex02_-_descricao_dos_mtodos.pdf



Exercício 3

Seguindo o diagrama UML, implemente as classes e exiba no console conforme a imagem a seguir:

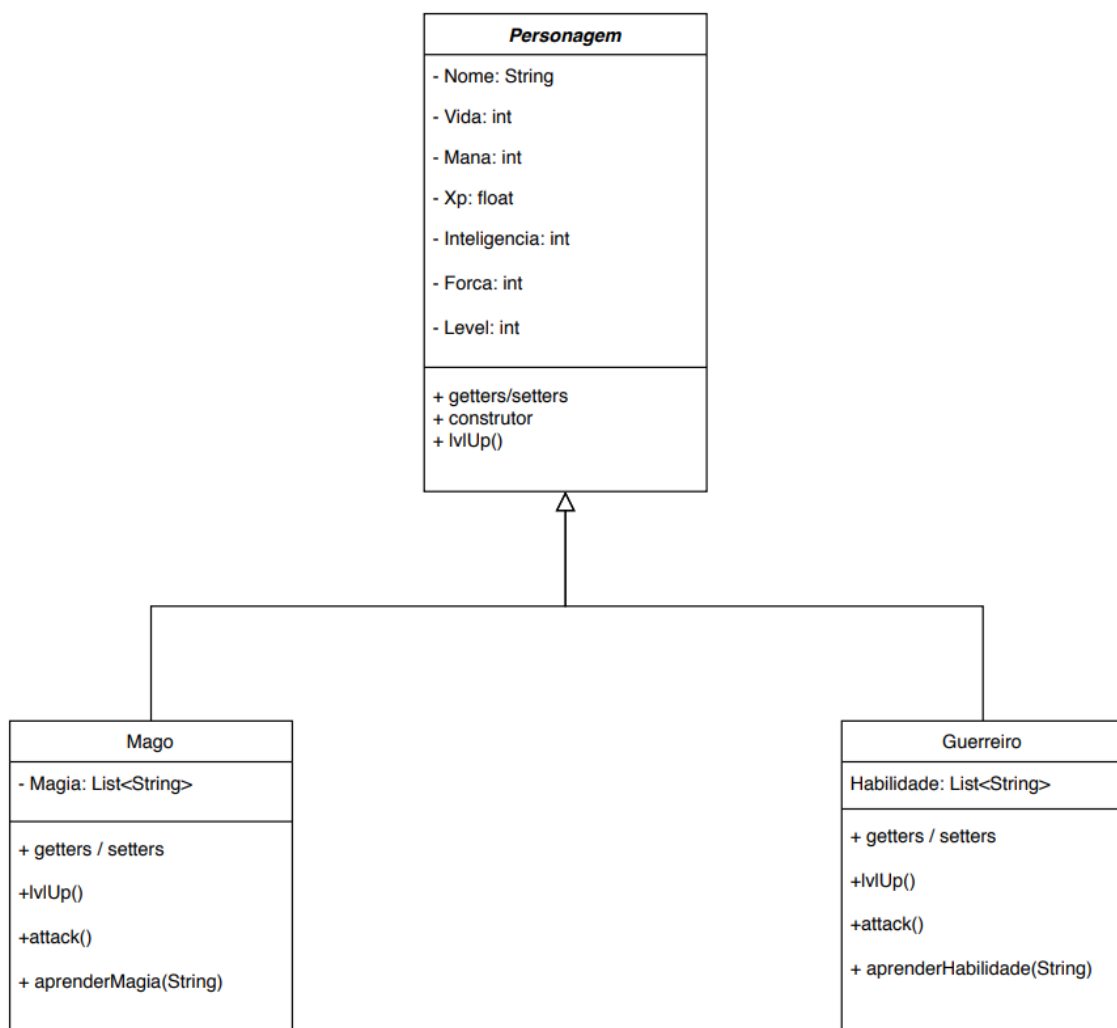
- Usando o conceito de polimorfismo, implemente o método **lvUp()**, de forma que o **Mago** ao subir de nível possua um aumento maior nos atributos **Mana e Inteligência** e o **Guerreiro** possua um aumento maior nos atributos **Vida e Força**.

- Implemente o método **attack()** de forma que siga a seguinte regra:

- Mago 🪄 : (**Inteligência * Level**) + numeroRandomico(0 até 300).
- Guerreiro ⚔️ : (**Força * Level**) + numeroRandomico(0 até 300).

2. Exiba a quantidade de Personagens criados, utilize atributo *static* para implementar a solução.

- <https://www.caelum.com.br/apostila-csharp-orientacao-objetos/metodos-e-atributos-estaticos#para-saber-mais-classes-estaticas>



Exercício 4

Instancie o objeto e a Lista<Pessoa>, adicione os dados conforme a tabela abaixo e por fim imprima o nome da pessoa mais velha.

Pessoas

<u>Aa</u> Nome	# Idade
<u>João</u>	15
<u>Leandro</u>	21
<u>Paulo</u>	17
<u>Jessica</u>	18

Exercício 5

Aproveitando a questão anterior (4) - Exclua da Lista as pessoas com idade inferior a 18 anos. E exiba a quantidade da lista antes e depois da exclusão. (Não reescreva o código do item 01).

Exercício 6

Aproveitando – o seu código já escrito na questão (4) e na Questão (6) – Consulte se o objeto Jessica existe na lista e exiba a sua idade.

<u>Aa</u> Nome	# Idade
<u>Leandro</u>	21
<u>Jessica</u>	18

Exercício 7

Considerando os conceitos de Orientação a Objetos, crie uma **classe Pai** de nome **Funcionário** com os seguintes atributos (nome, idade e salário) e mais três classes Filhas (Gerente, Supervisor e Vendedor). Na classe **Funcionário** deve existir um **método** de nome **bonificação** que retorna o salário, nas classes filhas deve existir o mesmo método **bonificação** porem com as seguintes regras:

- Para Gerente, o método **bonificação** deve retornar o salário + 10000.00;
- Para Supervisor, o método **bonificação** deve retornar o salário + 5000.00;
- Para Vendedor, o método **bonificação** deve retornar o salário + 3000.00;

Por fim, criar uma classe principal que instancie objetos de Gerente, Supervisor e Vendedor e adicione no mínimo um valor para cada atributo e imprima cada funcionário (Gerente, Supervisor e Vendedor) com suas devidas bonificações.