

Exercício 1

Crie uma classe `Veiculo` com um atributo `ligado` (privado), que indica se o carro está ligado ou não. Esta classe deve ter também os métodos `ligar()` e `desligar()`, que definem o valor para este atributo, e um método getter (`isLigado()`).

Depois crie três subclasses de `Veiculo`: `Automovel`, `Motocicleta` e `Onibus`. Cada classe destas deve sobrescrever os métodos `ligar()` e `desligar()` e deve imprimir mensagens como “Automóvel ligado”, “Motocicleta desligada”, etc. Para manter a consistência do modelo, descubra como fazer para que o atributo `ligado` de `Veiculo` tenha o valor correto quando os métodos são chamados.

Crie uma aplicação que instancia três veículos, um de cada tipo (`Automovel`, `Motocicleta` e `Onibus`), e chama os métodos `ligar()`, `desligar()` e `isLigado()`. O resultado obtido deve ser consistente com o que o modelo representa. Por exemplo, ao chamar o método `ligar()` de um `Automovel`, é esperado que o método `isLigado()` retorne `true`.

Exercício 1

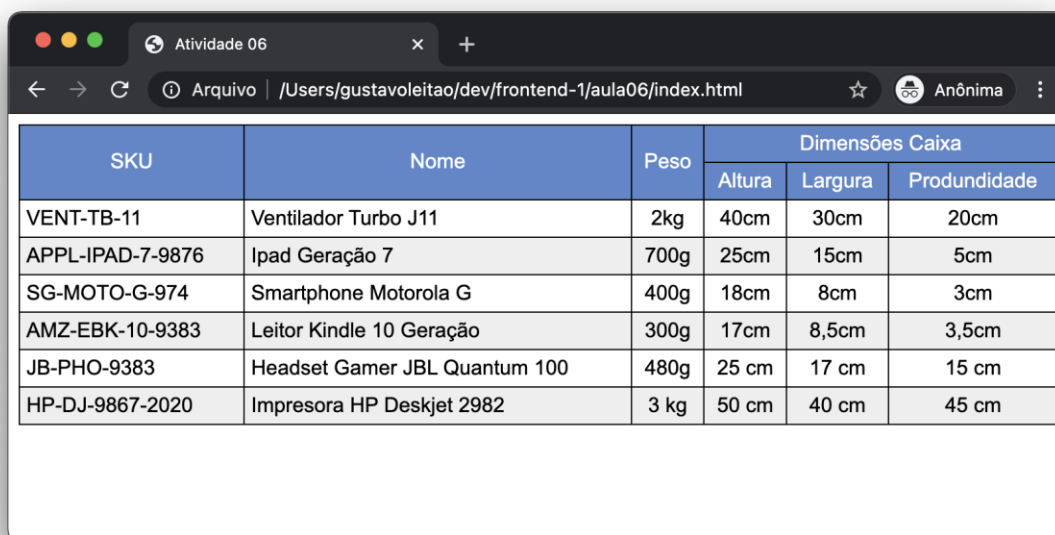
Crie uma classe `ContaBancaria`, que representa uma conta bancária genérica e não pode ser instanciada. Esta classe deve ter um atributo `saldo` (visível apenas para ela e para as suas subclasses) e os métodos `depositar(double)`, `sacar(double)` e `transferir(double, ContaBancaria)`. Estes métodos devem depositar um valor na conta, sacar um valor da conta e transferir um valor da conta de origem para uma conta de destino, respectivamente.

Além destes, `ContaBancaria` deve ter um método `calcularSaldo()`. Este método possui a regra do cálculo do saldo final (que pode ser diferente do saldo armazenado no atributo `saldo`) e deve ser obrigatoriamente implementado pelas subclasses de `ContaBancaria`, pois cada classe possui suas próprias regras de cálculo.

Crie duas subclasses de `ContaBancaria`: `ContaCorrente` e `ContaInvestimento`. Cada uma deverá implementar suas regras para calcular o saldo (método `calcularSaldo()`). No caso de `ContaCorrente`, o saldo final é o saldo atual subtraído de 10%, referente a impostos que devem ser pagos. Já para a `ContaInvestimento`, o saldo final é o saldo atual acrescido de 5%, referente aos rendimentos do dinheiro investido.

Crie uma aplicação que instancia uma conta corrente e uma conta investimento e executa as operações de depósito, saque, transferência e cálculo de saldo. Verifique se os resultados obtidos são consistentes com a proposta do modelo e com as regras de cálculo estabelecidas.

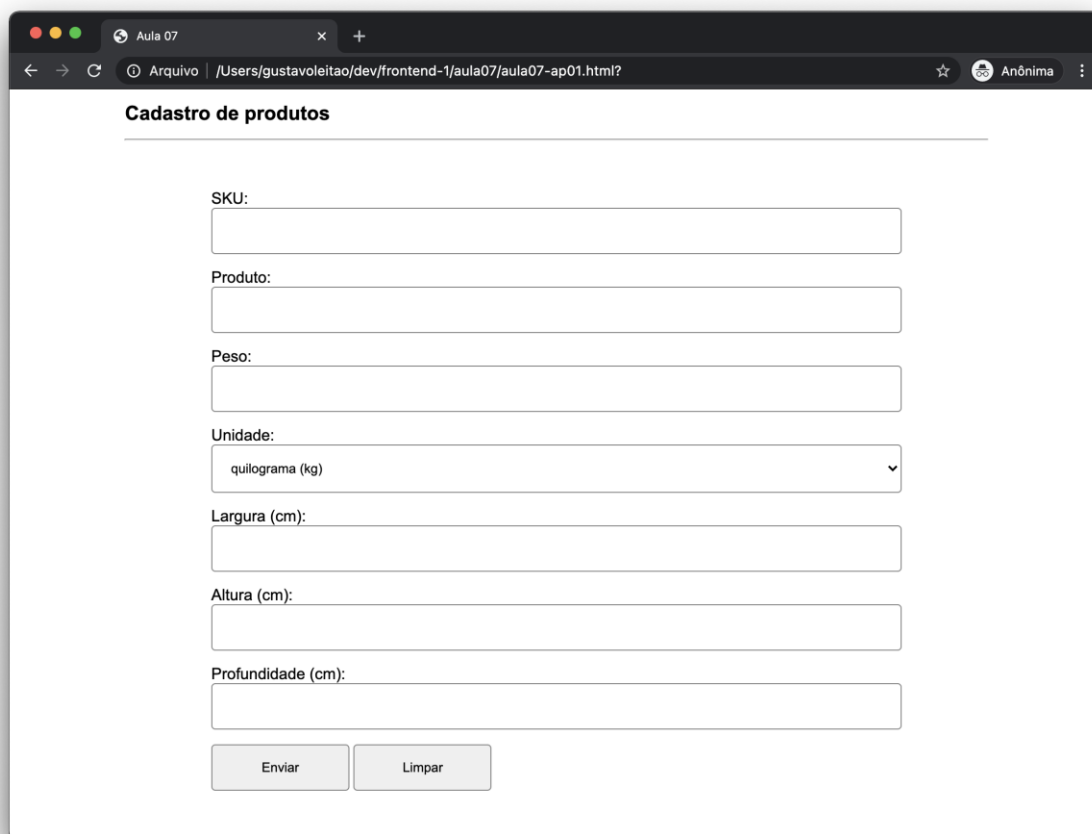
Crie um documento HTML com uma Tabela e seu CSS, conforme imagem abaixo, que satisfaça os seguintes critérios:



SKU	Nome	Peso	Dimensões Caixa		
			Altura	Largura	Profundidade
VENT-TB-11	Ventilador Turbo J11	2kg	40cm	30cm	20cm
APPL-IPAD-7-9876	Ipad Geração 7	700g	25cm	15cm	5cm
SG-MOTO-G-974	Smartphone Motorola G	400g	18cm	8cm	3cm
AMZ-EBK-10-9383	Leitor Kindle 10 Geração	300g	17cm	8,5cm	3,5cm
JB-PHO-9383	Headset Gamer JBL Quantum 100	480g	25 cm	17 cm	15 cm
HP-DJ-9867-2020	Impresora HP Deskjet 2982	3 kg	50 cm	40 cm	45 cm

- Tabela com bordas colapsadas e visíveis
- Família de fontes “Arial, Helvetica, sans-serif;”
- Tabela ocupando 100% da página
- Espaçamento interno de 5px nas células (*padding*)
- Cabeçalho da tabela organizado conforme imagem e com seguinte regras de estilo:
 - Fonte branca
 - Fonte sem negrito
 - Cor de fundo azul claro ou similar
- Linhas pares com cor de fundo cinza claro
- Conteúdo das colunas “Peso”, “Altura”, “Largura” e “Profundidade” centralizado. Demais colunas deve-se manter alinhado à esquerda.

Crie um documento HTML com um formulário para cadastro de produtos, conforme imagem abaixo, que satisfaça os seguintes critérios:



The screenshot shows a web browser window with the title 'Aula 07'. The address bar shows the URL: `/Users/gustavoleitao/dev/frontend-1/aula07/aula07-ap01.html?`. The page content is a form titled 'Cadastro de produtos'. The form contains the following fields and controls:

- SKU:
- Produto:
- Peso:
- Unidade:
- Largura (cm):
- Altura (cm):
- Profundidade (cm):
- Buttons:

Para a estruturação do HTML, considerar:

- O campo SKU deve ser do tipo texto;
- O campo Produto deve ser do tipo texto;
- O campo peso deve permitir apenas números;
- O campo de unidade deve possuir as seguintes opções:
 - quilograma (kg)
 - grama (g)
 - tonelada (t)
- Os campos largura, altura e profundidade devem permitir apenas números;
- Deve possuir um botão de submissão e um botão para limpar o formulário;
- A ação do formulário deve levar para o seguinte endereço: <https://autoria-form.herokuapp.com/>
- Todos os campos devem possuir os atributos "name" e "id" definidos.

Para o estilo do CSS, considerar:

- a) A fonte do documento deve ser da família “Arial, Helvetica, sans-serif”;
- b) Os campos devem possuir 80% da largura da página;
- c) Os campos devem ficar centralizados;
- d) Os campos de entrada devem ter 15px de *padding*, border-radius de 4px e margin-bottom de 15px.
- e) Os botões devem ocupar 20% do tamanho da tela, cada.