Exercício 03

Carrinho de um e-commerce

Nessa sequência de exercícios vamos simular a parte da lógica de um processo simples de checkout num e-commerce.

a) Declare uma função de nome `imprimirResumoDoCarrinho` que tenha apenas um parâmetro chamado `carrinho`.

b) Sabendo que o parâmetro `carrinho` será sempre um objeto com o seguinte formato:

```
const carrinho = {
    nomeDoCliente: "Guido Bernal",
    produtos: [
        {
            id: 1,
            nome: "Camisa",
            qtd: 3,
            precoUnit: 3000
        },
        {
            id: 2,
            nome: "Bermuda",
            qtd: 2,
            precoUnit: 5000
        }
    ]
}
```

Implemente a função `imprimirResumoDoCarrinho` de modo que, ao receber um objeto como este como parâmetro, o resultado seja o seguinte.

```
Cliente: Guido Bernal
Total de itens: 5 itens
Total a pagar: R$ 190,00
```

Para testar sua implementação, chame a função `imprimirResumoDoCarrinho` passando o objeto `carrinho` exemplificado acimam como argumento.

c) Modifique a questão anterior para que a função `imprimirResumoDoCarrinho` seja um método do objeto carrinho (sendo assim, cabe mudar o nome do método para apenas `imprimirResumo`) ao invés de uma função isolada.

d) Declare uma nova função isolada chamada `addProdutoAoCarrinho` que tenha dois argumentos: `carrinho` e `produto`.

e) Implemente a função `addProdutoAoCarrinho` de modo que o produto passado com argumento seja adicionado ao array de produtos da compra.

Lembre-se que, antes de simplesmente adicionar o item ao array, é necessário verificar se já existe um item igual no carrinho. Caso exista, basta incrementar a propriedade qtd do item adequadamente. Caso não exista, aí sim é necessário adicionar um novo item ao array. Use o id dos produtos para fazer esta comparação.

Para testar sua implementação faça as seguintes chamadas, nesta ordem:

```
const novaBermuda = {
    id: 2,
    nome: "Bermuda",
    qtd: 3,
    precoUnit: 5000
}

addProdutoAoCarrinho(carrinho, novaBermuda);
carrinho.imprimirResumo();
```

Isso deve imprimir o seguinte resultado:

```
Cliente: Guido Bernal
Total de itens: 8 itens
Total a pagar: R$ 340,00
```

Em seguida, faça as seguintes chamadas:

```
const novoTenis = {
    id: 3,
    nome: "Tenis",
    qtd: 1,
    precoUnit: 10000
}

addProdutoAoCarrinho(carrinho, novoTenis);
carrinho.imprimirResumoDoCarrinho();
```

Isso deve imprimir o seguinte resultado:

```
Cliente: Guido Bernal
Total de itens: 9 itens
Total a pagar: R$ 440,00
```

f) Modifique a questão anterior para que a função `addProdutoAoCarrinho` seja um método do objeto carrinho (sendo assim, cabe mudar o nome do método para apenas `addProduto`. Faça os mesmos testes anteriores (adaptando a chamada, que não precisará mais do argumento carrinho) e o resultado obtido deve ser o mesmo.

g) Declare um novo método para o objeto carrinho chamado `imprimirDetalhes`

Depois, implemente-o de modo que, ao chamá-lo para o objeto exemplo da questão dois, tenha-se o seguinte resultado:

```
Cliente: Guido Bernal

Item 1 - Bermuda - 2 und - R$ 100,00
Item 2 - Camisa - 3 und - R$ 90,00

Total de itens: 5 itens
Total a pagar: R$ 190,00
```

h) Repare que, nas questão anterior, você fez uma função em que parte da lógica é muito parecida com o do método `imprimirResumo`, pois ambas precisam calcular o total de itens e calcular o total a pagar. Se algum dia o formato dos objetos forem modificados, de modo que você precise modificar a lógica do cálculo desses totais, você terá que lembrar de modificar isso em ambas as funções.

Por isso, muitas das vezes em que você for implementar alguma lógica repetidamente, será melhor criar uma nova função para centralizar essa lógica.

Para corrigiir isso, vamos fazer dois novos métodos `calcularTotalDeItens` e `calcularTotalAPagar`, que devem retornar um número inteiro (lembre-se que você pode acessar os dados do carrinho através da palavra reservada `this`).

Depois, altere os métodos `imprimirResumo` e `imprimirDetalhes` para que elas chamem essas duas novas funções.

i) Declare e implemente um novo método chamado `calcularDesconto`, sem nenhum parâmetro.

Este método deverá retornar um valor inteiro `desconto`, que é o valor de desconto que deve ser aplicado neste carrinho.

Para calcular o valor do desconto, utiliza-se a seguinte lógica:

Existem dois descontos possíveis, **não cumulativos**.

O primeiro é um desconto em que, para compras acima de 4 itens, o item mais barato sai de graça.

O segundo é um desconto de 10% para compras acima de 100 reais.

Sempre **no máximo** um deles será aplicado - o que for mais vantajoso para o cliente.

Para o exemplo da **letra B**, o desconto deverá ser de R$ 30,00.

Para o exemplo da **letra E**, com 8 itens, o desconto deverá ser de R$ 44,00.

Preencha a checklist para finalizar o exercício:

- [ ] Resolver o exercício revendo a aula se necessário
- [ ] Adicionar as mudanças aos commits (`git add .` para adicionar todos os arquivos, ou `git add nome_do_arquivo` para adicionar um arquivo específico)
- [ ] Commitar a cada mudança significativa ou na finalização do exercício (`git commit -m "Mensagem do commit"`)
- [ ] Pushar os commits na sua branch na origem (`git push origin nome-da-branch`)

TAGS: `LÓGICA` `MÓDULO 1` `EXERCÍCIO DE CLASSE` `NODEJS` `FUNCAO` `OBJETOS`