02

O desenvolvedor inspirado

Paulo é o novo estagiário da Caelum. Em um dia de inspiração, encontrou o seguinte código em um projeto que, embora crítico, está funcionando sem nenhum bug relatado há anos:

```
class Conta:

    def __init__(self, numero, saldo, limite ):
        self.__numero = numero
        self.__saldo = saldo
        self.__limite = limite
        self.__tarifaTransferencia = 8.0


    def saca(self,valor):

        if valor < (self.__saldo + self.__limite):
                self.__saldo -= valor
                print("Saque efetuado.")
        else:
                print("Saldo insuficiente.")



    def transfere(self, valor, destino):


            valorTotal = valor + self.__tarifaTransferencia

           if valorTotal < (self.__saldo + self.__limite):

                self.__saldo -= valorTotal
                destino.deposita(valor)

                print("Transferência efetuada.")
            else:
                print("Saldo insuficiente.")


    # outros métodos omitidos...
```

Como líder técnico da empresa, você precisa inspecionar as modificações que Paulo fez na classe anterior e ajudá-lo no desenvolvimento profissional. Analise o código do Paulo:

```
class Conta:
    def __init__(self, numero, saldo, limite):
        self.__numero = numero
        self.__saldo = saldo
        self.__limite = limite
        self.__tarifaTransferencia = 8.0

    #NOVO MÉTODO AQUI
```

```
def temSaldoDisponivelPara(self, valor):
    return valor < (self.__saldo + self.__limite)


def saca(self, valor):

    if self.temSaldoDisponivelPara(valor):
        self.__saldo -= valor
        print("Saque efetuado.")
    else:
        print("Saldo insuficiente.")


def transfere(self, valor, destino):


    valorTotal = valor + self.__tarifaTransferencia

    if self.temSaldoDisponivelPara(valorTotal):

        self.__saldo -= valorTotal
        destino.deposita(valor)

        print("Transferência efetuada.")
    else:
        print("Saldo insuficiente.")
```

A Paulo criou mais complexidade ao projeto ao adicionar mais métodos na classe.

✓ Paulo pode melhorar ainda mais o código.

Correto! Embora o método criado evite a duplicação de códigos, ele não será chamado por nenhuma outra classe do sistema. Logo, não faz sentido deixá-lo público e expô-lo na interface pública da classe.

C Paulo não deveria mexer em código que já funciona.

D Paulo fez uma modificação que não influenciou nem positivamente nem negativamente no projeto.

PRÓXIMA ATIVIDADE