Descrição do Exercício

Classes, Objetos, Atributos e Métodos

Material de apoio: slides "3_OOBasico.pdf "

Objetivo: Este exercício tem como objetivo a familiarização com a criação de classes, atributos e métodos, através de uma aplicação bancária simplificada.

Passos para execução do exercício

Parte I

1. Criar uma nova classe, chamada `Cliente`.
2. Acrescentar à classe `Cliente` atributos que correspondam ao **nome** e **cpf** do cliente. Qual o tipo de dados adequado para estes atributos (**`int`**, **`boolean`**, **`String`**,...)? Quais os modificadores adequados para estes atributos (**`public`**, **`private`**, ...)?
3. Adicionar um Construtor à classe `Cliente`. Este construtor deve receber todos os valores necessários para inicializar os atributos de `Cliente`. Você pode também adicionar um construtor *default.*
4. Acrescentar à classe `Cliente` métodos para acessar cada um de seus atributos, isto é, métodos para obter (`get`) e modificar (`set`) atributos. Qual a importância desses métodos?
5. Criar uma nova classe intitulada `Conta`.
6. Adicionar os seguintes atributos para a classe `Conta`: seu **número**, seu **saldo** e seu dono (um **cliente**).
7. Adicionar dois construtores à classe `Conta`: um recebendo somente o **numero** e o **cliente** da conta para inicialização e outro recebendo todos valores necessários para a inicialização de **todos** os atributos de Conta. Você também pode definir o construtor *default*, se desejar.
8. Criar métodos de acesso (`get` e `set`) para cada um dos atributos de `Conta`.
9. Implementar o método `creditar` na classe conta, que incrementa o valor do saldo da conta. Que atributos este método deve receber?
10. Implementar o método `debitar`, que decrementa o valor do saldo da conta. Que atributos este método deve receber?
11. Criar uma classe de teste com o método `main()`. Neste método, declare dois objetos do tipo `Conta`, e atribua a cada um valores de saldo 100 e 0 (Dica: utilize método `setSaldo()` em cada objeto para setar o valor do saldo correspondente).
12. Ainda no método `main()`, imprimir na tela o saldo das contas criadas. Em seguida, testar a execução dos métodos `creditar` e `debitar` e imprimir novamente o valor dos saldos das contas.

Parte II

1. Voltar à classe Conta e implementar o método `transferir`, que transfere dinheiro de uma conta para outra. É necessário que este método receba alguma `Conta` como parâmetro? Uma? Duas? Nenhuma?

2. **Crie uma outra classe de teste com um método `main()` ou utilize a mesma já criada na Parte I, para testar a execução do método transferir. No método `main()`, você deve realizar uma transferência entre as contas e imprimir seus saldos antes e depois da transferência. Além disso, imprima os dados do cliente da conta e origem e os dados do cliente da conta destino a partir dos objetos conta.**

Parte III

1. Voltar novamente à classe `Conta` e modificar o método `debitar` de modo a evitar que o débito seja realizado caso o saldo seja insuficiente. Imprimir uma mensagem na tela informando se a operação foi ou não realizada com sucesso. Reflita: por que a impressão de uma mensagem na tela não é uma abordagem interessante pala lidar com a falha da operação?
2. Tentar realizar uma transferência cuja conta de origem possui saldo insuficiente. Observe a mensagem de erro. Verifique se a transferência cancela com sucesso a operação caso o saldo da conta a ser debitada seja insuficiente. O que seria necessário para que isso acontecesse?
3. Modifique o método `transferir` de modo a imprimir uma mensagem na tela para sinalizar que o saldo da conta de origem é insuficiente para o valor a ser transferido.