UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
DEPARTAMENTO DE SISTEMAS E COMPUTAÇÃO
PROFESSOR ANDRÉ FELIPE BURGER
PROGRAMAÇÃO ORIENTADA A OBJETOS

Lista de Exercícios 02

Crie um projeto no Netbeans. Para cada questão abaixo, crie um pacote.

Questão 1

Partindo da solução da questão 4 da lista de exercícios 01, redesenhe o diagrama de classes adaptando a classe `Pessoa` para que utilize o conceito de encapsulamento. Em seguida, solucione novamente a questão 4 (da lista de exercícios 01) para que seja utilizada a nova classe `Pessoa`.

Questão 2

Implemente uma classe Produto com os seguintes requisitos:
Atributos privados: nome (String), preco (double) e estoque (int).

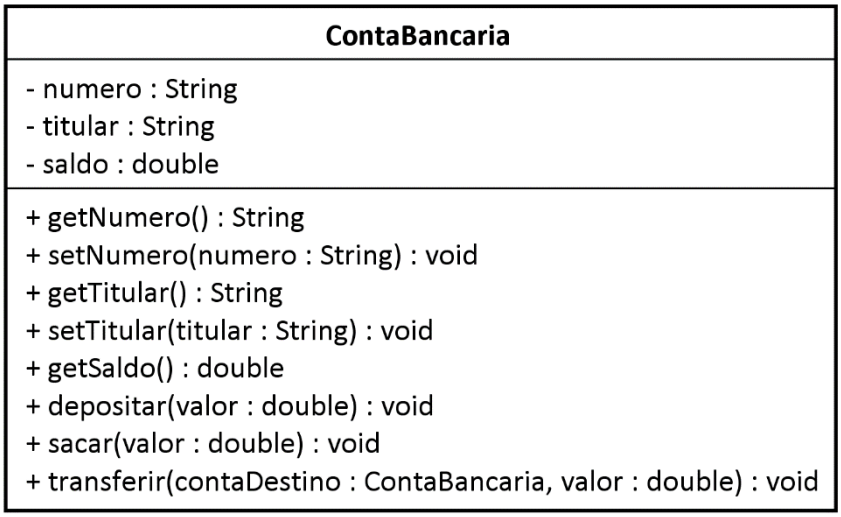
Métodos públicos para:

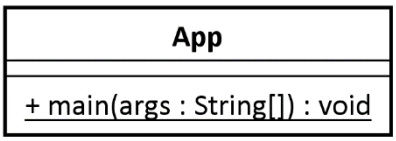
vender(int quantidade) – reduz o estoque se houver quantidade suficiente.
repor(int quantidade) – aumenta o estoque se a quantidade for positiva.
getPreco() e setPreco(double preco) – o preço só pode ser alterado se o valor for positivo.
getNome() – retorna o nome do produto (o nome não pode ser alterado após a criação do objeto).
getEstoque() – retorna a quantidade atual no estoque.

No main, crie um produto, faça vendas, reposições e tente alterar o preço com valores inválidos para verificar a proteção dos atributos.

Questão 3

Com intuito de representar contas bancárias, implemente o diagrama de classes abaixo:





1. O método `getNumero()` deve ser o método *getter* da variável de instância `numero`.
2. O método `setNumero()` deve ser o método *setter* da variável de instância `numero`.
3. O método `getTitular()` deve ser o método *getter* da variável de instância `titular`.
4. O método `setTitular()` deve ser o método *setter* da variável de instância `titular`.
5. O método `getSaldo()` deve ser o método *getter* da variável de instância `saldo`.
6. O método `depositar()` deve acrescentar valores ao saldo da conta bancária. O método `depositar()` deve recusar tentativas de depósito com valor negativo.
7. O método `sacar()` deve subtrair valores do saldo da conta bancária. O método `sacar()` deve recusar tentativas de saque com valor negativo. Também deve recusar tentativas de saque que causem o saldo ficar negativo.

8. O método `transferir()` deve ser implementado para transferir valores de uma conta bancária para outra. Considerar que a conta de origem é a conta onde será invocado o método `transferir()`, enquanto que a conta de destino será fornecida como argumento para o método `transferir()`. O valor a ser transferido também é fornecido como argumento.
9. Crie um programa (classe `App`) que solicite ao usuário o número e titular de duas contas bancárias. Em seguida, efetue as seguintes operações:
    a. Realize depósitos na primeira conta nos valores de R$ 1.000,00 e R$ 700,00.
    b. Realize depósitos na segunda conta nos valores de R$ 5.000,00.
    c. Faça um saque na 2ª conta no valor de R$ 3.000,00.
    d. Transfira R$ 1.800,00 da 2ª conta para a 1ª conta.
    e. Exiba o titular de cada uma conta com o respectivo saldo da conta.