Lista de Exercícios 02

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

Questão 1

Partindo da solução da questão 5 da lista de exercícios 01, redesenhe o diagrama de classes adaptando a classe `Pessoa` para que utilize o conceito de encapsulamento e construtores. Em seguida, solucione novamente a questão 5 (da lista de exercícios 01) para que seja utilizada a nova classe `Pessoa`.

Questão 2

Partindo da solução da questão 1 da lista de exercícios 01, adapte a classe `Retangulo` para possuir um construtor que receba os valores de altura e lado, além de aplicar o encapsulamento aos atributos. Em seguida, faça um programa que solicite a altura e o lado de três retângulos e imprima a área deles. Os retângulos devem ser criados usando o método construtor.

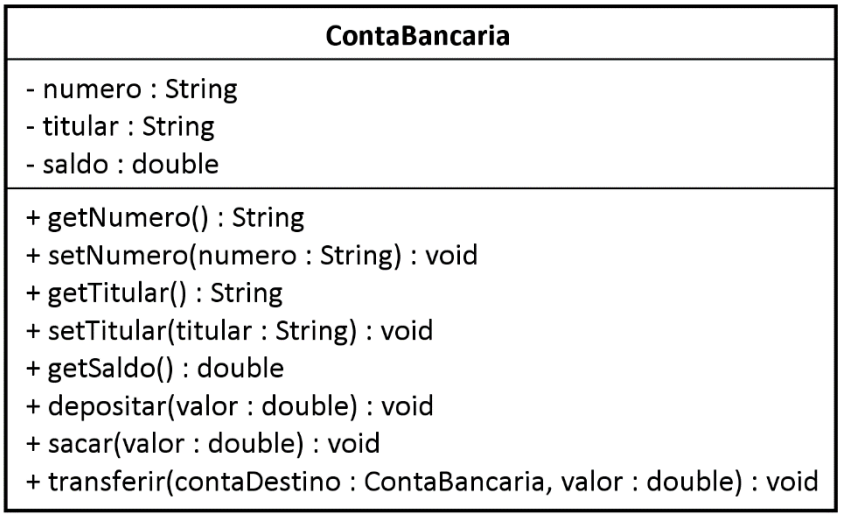
Questão 3

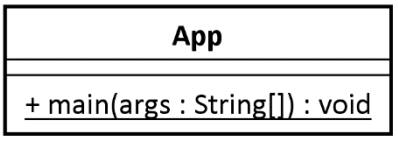
Partindo da solução da questão 6 da lista de exercícios 01, adapte a classe Ponto para possuir um construtor que receba os valores de X de Y, além de aplicar o encapsulamento aos atributos. Depois de realizar as modificações, solucione novamente a questão 6 da lista 01, mas utilizando a nova classe de Ponto.

Ainda tirando proveito da classe ponto, faça um método chamado `estaIncidindoSobreX()` e outro chamado `estaIncidindoSobreY()`, que devem respectivamente retornar um `booleano` indicando se o ponto está sobre o eixo X e se ele está sobre o eixo Y. Modifique o diagrama de classe para adicionar esses novos métodos.

Questão 4

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.

Questão 5

Crie um diagrama de classes para representar um dado. O dado deve permitir definir o número de faces que ele terá, sendo que o número de faces deve ser maior ou igual a 4 e sempre deve ser um número par. A classe de dado não deve permitir alterar o número de faces definido na sua criação. A classe de dado deve ter um método `jogarDado(),` que irá retornar um valor entre 1 e o número de faces do dado (pode-se usar o método nextInt da classe Random para isso). A classe Dado também deve possuir um método estático chamado `jogarDados(conjuntoDados: Dado[]): int[]` que irá receber um vetor de dados e irá retornar um vetor com a jogada de cada dado em ordem. Crie uma classe `App` com um método `main()`, instancie um conjunto de dados e chame o método `jogarDados()` para testar o lançamento deles. Depois disso crie um vetor com 5 dados (um de 4, um de 6, um de 8, um de 12 e um de 20 faces) e invoque o método estático de `jogarDados(conjuntoDados: Dado[]): int[]` e imprima o resultado das jogadas.

Questão 6

Crie uma classe para representar uma fração, a classe deve possuir um atributo representando um numerador, e um atributo representando um denominador, além de possuir um método construtor que permita informar os atributos. A classe fração não deve permitir alterações externas nos seus atributos depois de sua criação. É importante lembrar que o denominador não pode ser zero. O numerador deve sempre conter o sinal da fração, caso ela seja negativa. A classe fração deve possuir um método que permita somar o seu valor com o de outro objeto de fração, retornando uma nova fração. A classe de fração também deve possuir um outro método que permita com que seu valor seja somado ao valor de um número inteiro, retornando uma nova fração. Crie uma classe `App` com um método `main()`, onde você irá criar a fração ¼ e somar ela com a fração ½ e o número 1, mostrando o resultado de cada operação. Antes de realizar a implementação, faça um diagrama de classe para representar a classe fração e seus métodos.

Obs: para a soma de fração considere: $\frac{a}{b} + \frac{c}{d} = \frac{(a \cdot d) + (c \cdot b)}{(b \cdot d)}$