Universidade Federal do Rio Grande do Norte
Instituto Metropole Digital
IMD0040 - Linguagem de Programação 2
Aula17 - Lista de exercícios

- Esta lista de exercício é composta por 3 questões: uma fácil, uma média e uma difícil.
- As questões valem, respectivamente, 20%, 30% e 50%.
- Não se assustem com o tamanho dos textos.
- Serão aceitos apenas arquivos **.zip**.
- Não será aceita nenhuma questão feita no BlueJ.
- Não serão aceitos arquivos enviados por e-mail.

Questão Fácil (20%) - Transformar atividade média da aula 09 em classe abstrata
Fique atento: para fazer o exercício difícil de hoje, você precisará fazer o exercício fácil e médio primeiro.

A tarefa

Utilizando o exercício da lista de exercícios sobre Herança, transforme as classes descritas abaixo em abstratas e insira os métodos novos sublinhados:

Classe Abstrata Produto:
Atributos: id (NOVO ATRIBUTO), nome, preço, marca, descrição, data de fabricação
Métodos: sets e gets de cada atributo, impressão dos atributos.
Métodos abstratos (somente a assinatura): diponível para venda?

Classe Abstrata Produto Durável (herdando da classe produto):
Atributos: material predominante, durabilidade, avaria.
Métodos: sets e gets de cada atributo, impressão dos atributos.
Métodos abstratos (somente a assinatura): é eletro eletrônico?

Classe Abstrata Produto não Durável (herdando da classe Produto):
Atributos: data de validade, gênero (alimentício, produto de limpeza, roupa, produto de uso pessoal).
Métodos: sets e gets de cada atributo, impressão dos atributos,
Métodos: está na validade(recebe data)
Obs: Os métodos de impressão deverão ser sobrescritos.

Reaproveite o código da atividade passada:

Crie, pelo menos, mais 2 classes concretas que herdam da classe Produto Durável (carro, livro, celular) e 2 da classe concretas Produto não Durável (pizza, refrigerante, chocolate) e que possuem atributos e métodos relevantes para essa classe, sobrescrevendo os métodos necessários. **As classes devem implementar os métodos abstratos.**

Observe que um produto só poderá ser vendido se ele estiver no estoque e disponível para venda.
Um produto está disponível para venda se estiver na validade e se não possuir avarias.

Questão Média (30%) - Formas

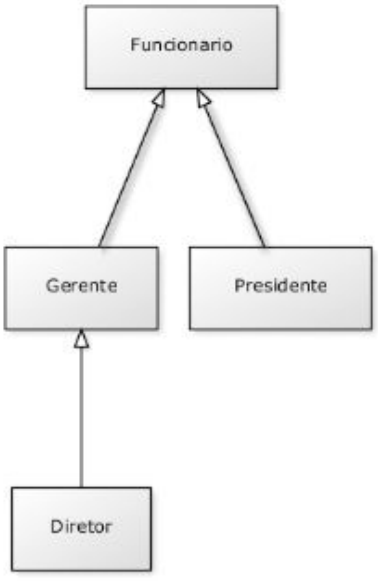
Implemente uma classe abstracta de nome Forma onde são declarados dois métodos abstratos:

• float calcularArea();
• float cacularPerimetro();

Crie, como subclasse de Forma, uma classe de nome Retângulo cujas instâncias são caracterizadas pelos atributos lado e altura ambos do tipo float. Implemente na classe Retângulo os métodos herdados de Forma e outros que ache necessários.
Crie, como subclasse de Forma, uma classe de nome Círculo cujas instâncias são caracterizadas pelo atributo raio do tipo float. Implemente na classe Circulo os métodos herdados de Forma e outros que ache necessários. Nota: considere o valor de Pi utilizando o valor fornecido pela API: Math.Pi.
Crie, como subclasse de Rectângulo, uma classe de nome Quadrado cujas instâncias são caracterizadas por terem os atributos lado e altura com o mesmo valor.
Elabore uma classe repositórioFormas onde é declarado um arrayList e adicionado aleatoriamente as formas criadas. Nota: para gerar números aleatórios crie primeiro uma instância da classe Random (presente na biblioteca java.util) e para extrair um inteiro entre 0 e n utilize nextInt(n). Depois implemente um laço que percorre para criação e acesso da lista de formas, onde em cada interação o objeto atual deve efetuar o cálculo da área. e apresentar o resultado.

Questão Difícil (50%) - Registro de Funcionários

Você foi contratado para desenvolver um controle de bonificação de uma empresa, onde o quadro de funcionários é representado pelo seguinte diagrama:



Ao modelar as entidades acima foram identificados os seguintes atributos pertencentes a cada uma das classes:

- Nome
- Data de Nascimento
- CPF
- Salario

Como também foi decidido que a classe Funcionário não pode ser instanciada.

Sabendo que os campos se repetem, utilizem a estratégia de classe abstrata para evitar a repetição de código.
Crie uma classe ControleDeBonificacoes para criar as funcionalidades do sistema, a classe deverá conter uma função chamada RegistraBonificacao que recebe um funcionário e adiciona uma bonificação no salário de:

- 10% - Gerente
- 20% - Diretor
- 30% - Presidente

O limite máximo de bonificação é 5 por funcionário individualmente ou 10 no geral.
O seu sistema deve ser capaz de cadastrar funcionários, listar funcionários não bonificados e listar funcionários bonificados(informando a quantidade de vezes que o mesmo recebeu o bônus), em ambas as listagens deve ser apresentado o tipo do funcionário
Adicione na classe Funcionário o método com a seguinte assinatura:
abstract double getBonificacao();
Este método deve armazenar a quantidade da bonificação recebida pelo funcionário, e será muito útil na função RegistraBonificacao