Exercícios de programação orientada a objetos com Java – lista 3

Orientações: utilize os padrões de encapsulamento (construtores e métodos acessores para atributos privados) e mutabilidade dos atributos vistos em aula (se um atributo não varia ao longo do tempo pode ser tratado como imutável – final). Para cada projeto crie um classe de teste com o main onde objetos das classes de cada projeto são criadas e seus métodos testados.

1) Pegue os códigos das classes criadas na lista 2 (sobre encapsulamento) e faça as devidas alterações definindo corretamente os atributos imutáveis (realizando as alterações necessárias: definição de valor inicial no construtor e retirada do setter desse atributo). Após, implemente o toString() e equals() nas classes e faça os devido testes.

2) Para a classe Livro do exercício 1 da lista anterior, faça com que seja possível comparar dois livros para posterior ordenação pelo número de páginas (crescente), caso tenha o mesmo número de páginas ordene pelo título do Livro. Após, em uma classe de testes crie uma lista de livros e os ordene.

3) Projeto: máquina de café:
   Classe: Cafe (representa um tipo de café)
   Atributos: imutáveis: tipo (char T ou F para tradicional ou forte respectivamente) e doce (boolean que diz se tem açucar)
   Construtor: Crie o construtor com os atributos.
   Métodos:
   getTipo() retorna as strings “tradicional” ou “forte”
   isDoce() retorna se é ou não doce
   equals (verifique se o café é do mesmo tipo e doce como o café recebido como parâmetro)

   Classe: CopoCafe (representa um copo de café)
   Atributos: quantidade em mL (atributo mutável) e capacidade em mL (atributo imutável), tipoCafe (objeto de café)
   Construtor: que cria um copo com uma capavidade e vazio.
   Métodos: encher(cafe tipoCafe) –> que enche todo copo se o copo estiver vazio ou se esta com o mesmo tipo de café (compare a igualdade dos tipos de café do parâmetro e atributo). Retorna True ou false se conseguiu adicionar café no copo;
   encher(int qtde, cafe tipoCafe) –> preenche o copo com o valor passado que deve ser positivo (não transborde o copo) se o copo estiver vazio ou se esta com o mesmo tipo de café. Retorna True ou false se conseguiu adicionar café no copo;
   esvazia(int) –> retira do copo o valor passado (a quantidade não pode ser negativa)
   getters –> para os três atributos;
   equals(object) –> verifica se dois copos de café são iguais (três atributos iguais), dica: você pode utilizar o equals do tipoCafé para comparar os tipos
   toString –> que retorna uma string. Exemplo do formato: “copo de 300 mL com 100 mL de café tradicional adoçado ”.

   Classe: MaquinaCafe (representa uma maquina de café)

Atributos: voltagem, marca, quantidade de água no repositório, quantidade de açúcar (em gramas) e quantidade de café (em gramas). Analise quais atributos podem ser imutáveis e os implemente assim.
Construtor: que cria uma máquina de café recebendo os atributos imutáveis como parâmetro. Considere que a máquina está vazia quando criada (sem café, água e açúcar)
Métodos: abasteceAgua(int X) –> enche o reservatório com X mL de água;
abasteceCafe(int X) –> enche o reservatório com X gramas de café;
abasteceAcucar(int X) –> enche o reservatório com X gramas de açúcar;
preparaCafeTradional(copo c, boolean duplo, boolean adoçado) –> adiciona café tradicional ao copo, retorna boolean se conseguiu preparar e encher o copo. Regras para criar um café tradicional simples:

- É utilizado 50 mL de água, 5 gramas de café e caso adoçado, 10 gramas de açúcar.
- Para um café duplo considere o preparo de dois simples (ingredientes dobrados).
- Caso não tenha água ou café suficiente nos reservatórios retorne falso;
- Caso não tenha açúcar suficiente para receita, tente encher com o café sem açúcar;
- Caso o copo tenha outro tipo de café retorne false;
- Caso consiga encher o copo com algum tipo de café retorne true.

preparaCafeForte(copo c, boolean duplo, boolean adoçado) –> mesma lógica do preparaCaféTradicional(), porém utilizando o dobro de café. Você consegue perceber uma lógica repetida entre os métodos que preparam café? Extraia um ou mais métodos privados que são chamados por preparaCafeTradional() e preparaCafeForte() evitando o código duplicado...
Getters para todos atributos

Classe: testaCafeteria
Implemente o método main que:
- cria uma máquina de café 220 volts, marca “TabajaraPresso”;
- adiciona 500 mL de água;
- adiciona 50 gramas de café e açúcar;
- crie 5 copos (c1, c2, c4 e cf de 200mL e c3 com 300mL)
- prepara um café forte (cf) adoçado duplo;
- prepara quatro cafés (c1, c2, c3 e c4) tradicional simples adoçado;
- print os 5 cafés (utilizando o toString de CopoCafe)
- compare os cafés –> c1 e c2 devem ser iguais (teste o equals) porém o c4 e c3 devem ser diferentes entre si e de c1, c2 e cf pois o açúcar acabou, a capacidade dos copos é diferente ou o tipo de café (tradicional ou forte);
- tome 10 mL do café c1 e teste novamente se ele é igual ao objeto c2 (agora o quantidade nos copos será diferente);
- tente servir café tradicional sem açúcar simples no copo c1 (a máquina não deve permitir – retorna false pois tipo de café diferente);
- tente servir um café tradicional duplo adoçado no copo c1 (agora a máquina deve permitir – retorna true)

4) Altere o projeto do item 3) –> É desejável saber quantos copos já foram utilizados. Para controlar quantos copos já foram utilizados coloque um atributo estático na classe CopoCafé, crie um getter para o atributo. Faça os devidos testes.