Lista 2 – Programação Orientada a Objetos

1) Crie uma classe denominada *Equacao* que possua três atributos do tipo *double*: a, b e c. Tais atributos devem ser encapsulados, por isso, crie métodos *get* e *set* para cada um. Tal classe também deve possuir mais três métodos. O primeiro denominado *delta*, que calcula o delta de uma equação do segundo grau pela fórmula:

$$delta = b2 - 4*a*c$$

O segundo, denominado *retornaX1*, que retorna um dos resultados da equação, calculado pela fórmula:

$$x1 = \frac{-b + Math.sqrt(delta)}{2*a}$$

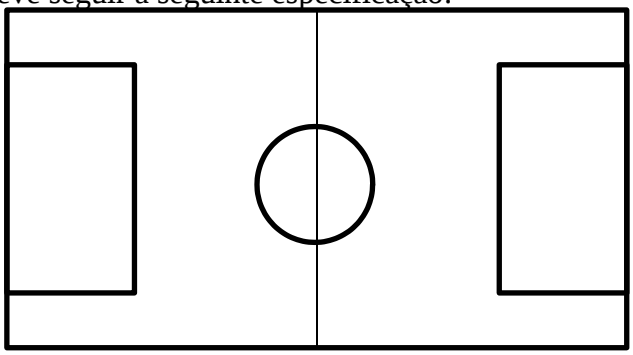
O terceiro, denominado *retornaX2*, que retorna o outro resultado da equação, calculado pela fórmula:

$$x2 = \frac{-b - Math.sqrt(delta)}{2*a}$$

Crie um *applet* que possua um objeto do tipo *Equacao*, representando uma equação real. Tal *applet* também deve receber os valores de a, b e c do usuário e exibir os dois resultados da equação. Após calcular e exibir os valores de uma equação, o *applet* deve perguntar se o usuário deseja passar outros valores ou sair do programa.

2) Crie um *applet* que represente a posição geográfica de duas sondas submarinas, utilizadas para exploração de petróleo. Os dados necessários para cada sonda são: nome, longitude (podendo ser representada pela sua posição no eixo X entre 0 e 180) e latitude (podendo ser representada pela sua posição no eixo Y entre 0 e 90). Toda vez que o *applet* for executado o usuário deverá informar os três dados de cada uma das sondas. Deverá então ser pintado na tela 2 círculos, de raio 3, para representar a posição de cada uma das sondas.
Importante: crie uma classe sonda; cada uma das duas sondas deve ser um objeto desta classe sonda. O encapsulamento deve utilizado.

3) Crie um *applet* que simule um jogo de futsal contendo 10 jogadores. A quadra deve ser formada por um retângulo de 500x250 *pixels*. Na tela, cada jogador deve ser representado por um círculo de raio 3, contudo para cada jogador deve haver um objeto da classe Jogador. Tal classe deve seguir a seguinte especificação:



Os dados dos jogadores deverão ser preenchidos pelo usuário. O nome do time de cada jogador deverá aparecer na tela junto com sua posição. O campo também deverá ser desenhado.

Obs1: crie um vetor do tipo Jogador.

4) Crie uma classe com métodos estáticos para solucionar problemas matemáticos. Tais métodos e suas características são apresentados a seguir:

- primo: recebe um inteiro e retorna se ele é primo ou não
- perfeito: recebe um inteiro e retorna se ele é um número perfeito ou não. Números perfeitos são aquele que são iguais à soma de seus divisores. O 6 é perfeito pois 1+2+3 = 6.
- fatorial: recebe um inteiro e retorna seu fatorial
- fibonacci: recebe um inteiro N e retorna o N-ésimo número da Série de Fibonacci. Tal série é uma seqüência de números onde o primeiro é 0, o segundo é 1 e a partir do terceiro, cada número é igual à soma de seus antecessores. Ou seja, o terceiro é 1, pois 0+1=1, o quarto é 2 (1+1 = 2), e assim sucessivamente.
- x1: recebe os valores a, b e c de uma equação do segundo grau e retorna a primeira raiz.
- x2: recebe os valores a, b e c de uma equação do segundo grau e retorna a segunda raiz.

Crie um programa que dê ao usuário a opção (por meio de um menu) de utilizar os métodos criados na classe descrita anteriormente. Após escolher uma das opções, o programa deve pedir os dados necessários e exibir o resultado. Coloque nesse menu uma opção para terminar o programa.

5) Crie uma classe denominada Elemento, com os atributos indice(int), dado (String) e proximo(Elemento). Crie uma classe denominada Pilha, que possui a implementação de uma pilha de objetos da classe Elemento. Em Pilha, crie os métodos: empilha (recebe um objeto do tipo Elemento), desempilha (retorna um objeto do tipo Elemento), pilhaVazia (retorna *true* se a pilha estiver vazia e *false* caso contrário) e imprime (retorna uma String com todos os dados de todos os elementos contidos na pilha). Crie um aplicativo que possua um objeto do tipo pilha. Crie um menu que dá as seguintes opções para o usuário:
1-Empilhar
2-Desempilhar
3-Exibir a pilha
4-Sair

6) Crie uma classe denominada Cargo que possui os campos: titulo (String) e salario (double). Crie também uma classe denominada Funcionario, a qual possui os seguintes atributos e seus tipos: nome (String), CPF (String) e Funcao (Cargo, descrito acima). Crie um vetor do tipo Funcionário. Crie um menu em um JOptionPane com as seguintes opções:
1-Cadastrar um funcionário
2-Listar todos os funcionários
3-Buscar funcionário por nome
4-Sair
Se o usuário escolher a opção 1, apenas um funcionário deve ser inserido no vetor. Se ele escolher a opção 2, todos os dados de todos os funcionários devem ser exibidos na tela. Se

ele escolher a opção 3, o programa deve pedir um nome e exibir na tela todos os dados do funcionário com aquele nome. Exiba “Funcionário não encontrado” se tal funcionário não existir no vetor. A opção 4 finaliza o programa.

7) Escreva uma classe em Java chamada Estoque. Ela deverá possuir:

a) os atributos nome (String), qtdatual (int) e qtdminima (int).

b) um construtor sem parâmetros e outro contendo os parâmetros nome, qtdatual e qtdminima.

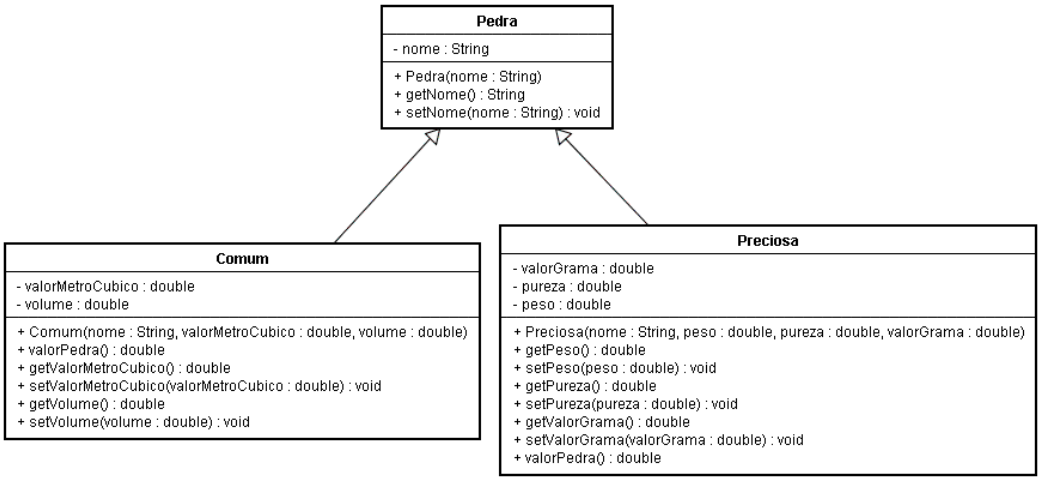
c) crei o método precisaRepor(), o qual retorna true caso a quantidade atua esteja menor ou igual à quantidade mínima e false, caso contrário.

d)Crie também uma classe denominada Frio que é filha de Estoque. Tal classe deve possuir os atributos: validade (String) e tempomaxcongelado(double). Crie agora uma classe denominada Bebida, filha de Estoque, com os campos: volume (double) e alcoolico (boolean).

e) Crie um aplicativo que possua dois vetores, um do tipo Frio e outro do tipo Bebida. Faça um menu com opções:

- Inserir um objeto no vetor do tipo Frio: pedir para o usuário todos os dados necessários.
- Inserir um objeto no vetor do tipo Bebida: idem.
- Buscar um produto pelo nome: deve pedir o nome e procurar nos dois vetores; se encontrar todos os dados do objeto devem ser exibidos na tela.
-Aumentar estoque: deve pedir o nome do produto para o usuário e aumentar o estoque.
-Reduzir estoque: deve pedir o nome do produto para o usuário e diminuir o estoque.
-Listar os nomes de todos os produtos abaixo do estoque mínimo.
- Sair

8) Crie a seguinte estrutura de classes apresentada na figura. Todos os atributos de todas as classes devem ser encapsulados, ou seja, devem ser *private* com métodos *getters* e *setters*. Essa estrutura de classes será utilizada por uma loja de pedras. As pedras são classificadas em “pedras comuns” e “pedras preciosas”. Toda pedra possui um nome, que será salvo no atributo *nome* da classe *Pedra*. Sobre as pedras comuns (classe *Comum*), deseja-se saber o volume da pedra em metros cúbicos (atributo *volume*) e o valor de um metro cúbico (atributo *valorMetroCubico*). Sobre as pedras preciosas, deseja-se saber o valor de um grama (atributo *valorGrama*), o peso da pedra em gramas (atributo *peso*) e a porcentagem de pureza (atributo *pureza*).

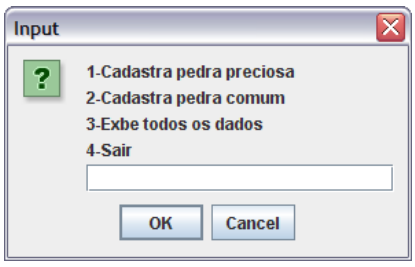


O método *valorPedra()* deve calcular o valor da pedra cadastrada. Na classe *Comum*, o valor da pedra é calculado multiplicando-se o *volume* pelo *valorMetroCubico.* Na classe *Preciosa*, o valor da pedra é calculado multiplicando-se o *peso* da pedra pelo *valorGrama*, tudo isso multiplicado pela porcentagem de *pureza* da pedra.

Exemplo: se *valorGrama*=10.0, *peso* = 5.0 e *pureza* = 80.0, *valorPedra()* retornara 40.0, pois (10.0 * 5.0 * (80.0/100)) = 40.

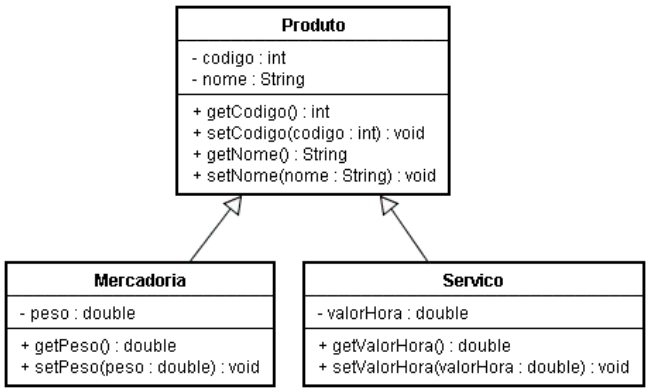
Importante: Os construtores das três classes recebem valores para todos os parâmetros contidos na classe.

Crie um *applet* para manipular a estrutura de classes descrita acima. Nele, devem ser criados dois vetores, um com objetos do tipo *Preciosa* e outro com objetos da classe *Comum* (todos de tamanho 1000). Crie o seguinte menu:

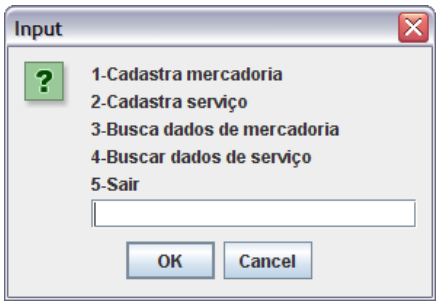


A opção 1 cadastra um único objeto da classe *Preciosa*, de forma que o usuário preencha todos os atributos do objeto criado. A opção 2 cadastra um único objeto da classe *Comum* da mesma forma. A opção 3 exibe em um *JOptionPane* o nome e o valor da pedra de todas a pedras cadastradas nos vetores (independente se os vetores estão cheios ou não). A opção 4 deve finalizar o *applet*.

9) Crie a seguinte estrutura de classes



Essa estrutura de classes será utilizada por uma loja que vende dois tipos de produtos: mercadorias e serviço. Crie um *applet* para manipular a estrutura de classes criada no exercício anterior. Nele, devem ser criados dois vetores, um com objetos do tipo *Mercadoria* e outro com objetos do tipo *Servico*. Implemente o seguinte menu:



Esse menu deve aparecer várias vezes, até que o usuário escolha a opção 5 (sair). A opção 1 cadastra um objeto no vetor mercadoria. Para isso, ela chama um chama um método denominado *cadastraMercadoria*, que recebe como parâmetro todos os dados para todos os atributos de mercadoria. O mesmo ocorre com a opção 2 e o vetor de objetos do tipo *Serviço*.

A opção 3, ao ser escolhida, pede para o usuário informar o nome de uma mercadoria. O *applet* deve então buscar no vetor e exibir todos os dados do objeto que possui o nome igual ao informado pelo usuário. O mesmo deve ocorrer com a opção 4 e o vetor de *Serviço*.

Com o intuito de economizar tempo para resolução da prova, o método *cadastraServico*, a opção 2 e a opção 4 não precisam ser implementados, uma vez que são muito parecidos com o método *cadastraMercadoria*, opção 1 e opção 3, respectivamente.