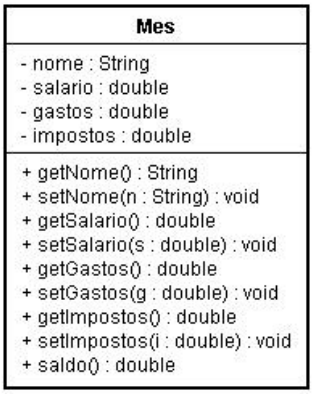
Universidade Federal de São João del-Rei – Campus Alto Paraopeba

DTECH – Departamento de Tecnologia em Engenharia, Computação e Humanidades

Exercícios de Programação Orientada a Objetos - JAVA

Prof.: Rone Ilídio / Thiago Oliveira

1) Crie a seguinte classe em Java:



Essa classe faz parte de um sistema que controla os gastos de uma pessoa. Cada instância dessa classe representa um mês desta pessoa. Seus atributos são:

- Nome: nome do mês que o objeto corresponde (janeiro, fevereiro, etc);
- Salário: salário ganho pela pessoa naquele mês;
- Gastos: gastos no mês;
- Impostos: valor total de impostos pagos.

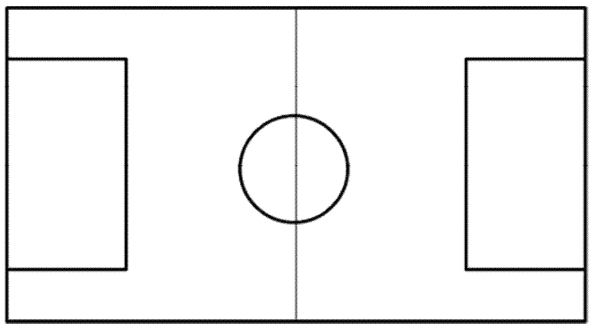
Implemente essa classe. Todos os atributos devem ser encapsulados. O método Saldo retorna o saldo do mês, ou seja, o salário menos os gastos e impostos.

Crie um *applet* que possua um vetor de meses (12 posições), que deve solicitar ao usuário que realize o cadastro de meses no vetor. Crie também um menu com as seguintes funções:

- exibir todos os dados do mês que teve o maior saldo;
- exibir a média dos salários recebidos;
- exibir o nome e o saldo de todos os meses em um *JOptionPane.*

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 dessa classe Sonda. Utilize encapsulamento.

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. Sugestão: 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:

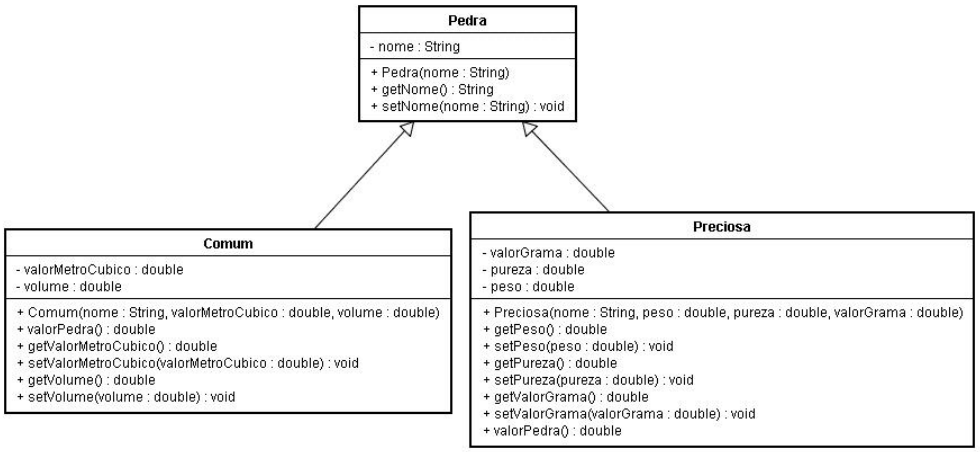
- primo: recebe um inteiro e retorna se ele é primo ou não
- perfeito: recebe um inteiro e retorna se ele é um número perfeito ou não. Números perfeitos são aquele que são iguais à soma de seus divisores. O 6 é perfeito pois 1+2+3 = 6.
- fatorial: recebe um inteiro e retorna seu fatorial
- fibonacci: 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 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. Utilize um menu com opção para terminar o programa.

5) Escreva uma classe 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) o método precisaRepor(), o qual retorna *true* caso a quantidade atual esteja menor ou igual à quantidade mínima e *false*, caso contrário.

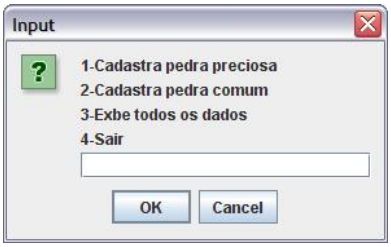
d) Crie também uma classe denominada Frio que é filha de Estoque. Tal classe deve possuir os atributos: validade (String) e tempomaxcongelado(double). Crie agora uma classe denominada Bebida, filha de Estoque, com os campos: volume (double) e alcoolico (boolean).
e) Crie um aplicativo que possua duas coleções, uma do tipo Frio e outra do tipo Bebida. Faça um menu com opções:
- Inserir um objeto Frio: pedir para o usuário todos os dados necessários.
- Inserir uma Bebida: idem para os dados de objeto Bebida.
- Buscar um produto pelo nome: deve pedir o nome e procurar nas duas coleções; 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

6) 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*).



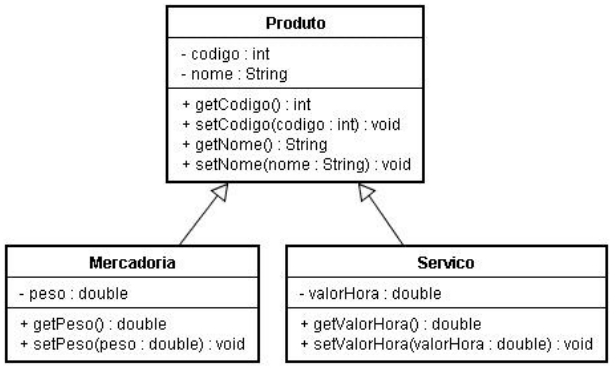
O método *valorPedra()* deve calcular o valor da pedra cadastrada. Na classe *Comum,* o valor da pedra é calculado multiplicando-se o *volume* pelo *valorMetroCubico.* Na classe *Preciosa,* o valor da pedra é calculado multiplicando-se o *peso* da pedra pelo *valorGrama*, tudo isso multiplicado pela 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, que deve conter uma coleção para armazenar objetos do tipo *Preciosa* e objetos da classe *Comum*. Crie o seguinte menu:

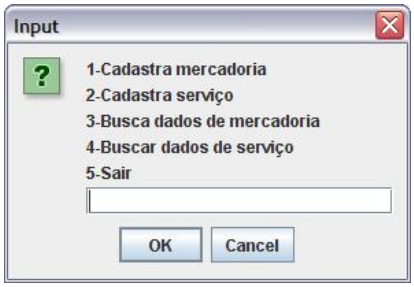


A opção 1 cadastra um único objeto da classe *Preciosa*, de forma que o usuário preencha todos os atributos do objeto criado. A opção 2 cadastra um único objeto da classe *Comum* da mesma forma. A opção 3 exibe em um *JOptionPane* o nome e o valor da pedra de todas a pedras cadastradas (independente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

7) Crie a seguinte estrutura de classes:



Crie um *applet* para manipular essas classes por uma loja que vende dois tipos de produtos: mercadorias e serviço. Devem ser criadas duas coleções, uma com objetos do tipo *Mercadoria* e outra com objetos do tipo *Servico*. Implemente o seguinte menu:



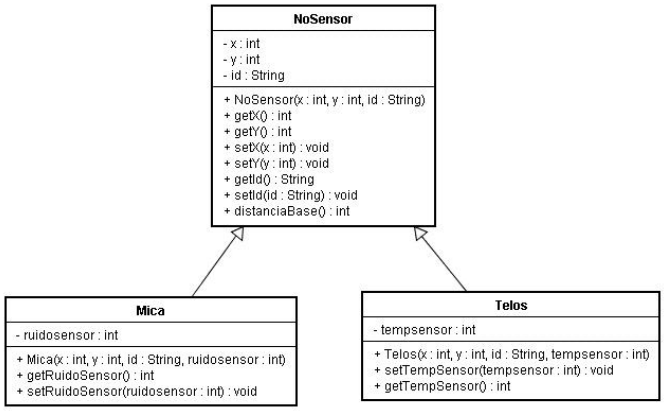
Esse menu deve aparecer várias vezes, até que o usuário escolha a opção 5 (sair). A opção 1 cadastra um objeto mercadoria. Para isso, chama o método denominado *cadastraMercadoria,* que recebe como parâmetro todos os dados para todos os atributos de mercadoria. O mesmo ocorre com a opção 2 e o vetor 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 e exibir todos os dados do objeto que possui o nome igual ao informado pelo usuário. O mesmo deve ocorrer com a opção 4 para *Serviço*.

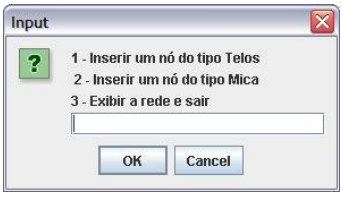
8) Crie uma classe denominada Cargo que possui os campos: titulo (String) e salario (double). Crie também uma classe denominada Funcionario, a qual possui os seguintes atributos e seus tipos: nome (String), CPF (String) e Funcao (Cargo, descrito acima). Crie uma coleção do tipo Funcionário. Crie um menu 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. 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. A opção 4 finaliza o programa.

9) Crie uma classe denominada Veiculo, com nome e ano de fabricação. Faça um *applet* que crie dois objetos da classe Veículo. O usuário deve preencher os dados de cada um dos objetos para utilização do construtor. Após isto, tais dados devem ser impressos na tela.

10) Veja a estrutura exibida no diagrama de classes UML a seguir. Crie uma classe denominada *Mica* e outra *Telos*. Como é mostrado no diagrama, tais classes são filhas da classe *NoSensor*. Os construtores das classes filhas devem receber valores para serem colocados em todos os atributos. Para colocar valores nos atributos vindos por herança, faça uma chamada ao construtor da superclasse. Todos os atributos devem ser encapsulados.



Crie um *applet* que utilize as classes acima. Ele deve possuir um *menu* como o exibido abaixo:



Tal *applet* deve possuir um vetor do tipo *NoSensor* de 20 posições. Todos os atributos de todos os objetos devem ser preenchidos com dados fornecidos pelo usuário. Porém, se o

usuário escolher a opção 3, o *applet* deve parar de exibir o menu acima e desenhar um círculo na tela para cada objeto já criado em cada vetor. As posições de cada círculo devem ser obtidas a partir das coordenadas x e y contidas em cada objeto. Garanta que o vetor receberá no máximo 20 elementos, podendo receber menos.

11) Crie um aplicativo para salvar os nomes dos alunos que entram em uma academia. Tal programa deve ter o seguinte *menu*:

Digite:
1-Cadastrar o nome de um aluno que está entrando
2-Listar todos os alunos que entraram
3-Procurar por um aluno
4-Sair

Tal *menu* deve ser exibido várias vezes, ou seja, o *menu* aparece, o usuário escolhe uma das opções, o programa faz as operações solicitadas e o *menu* aparece novamente. O programa deve conter uma coleção para salvar os nomes, mas o número de alunos que entram na academia é indefinido. Se o usuário escolher a opção 1, o programa deve pedir um nome e colocá-lo na lista. Se escolher a opção 2, o programa deve exibir em um único *JOptionPane* os nomes de todos os alunos. Se escolher a opção 3, o programa deve pedir um nome e procurar tal nome. Se o nome existir, apenas a frase "aluno presente" deve aparecer uma vez, se o nome não for encontrado, a frase "aluno ausente" deve ser mostrada.

12) Crie um *applet* que possua uma lista do tipo Pratico e outra do tipo teórico, de acordo com o diagrama abaixo. Tal programa deve exibir o menu apresentado abaixo:
A opção 1 insere um (SOMENTE UM) trabalho prático na lista correspondente e a opção 2 insere um trabalho teórico. A opção 3 pede para o usuário um nome de aluno e exibe na tela todos os dados de todos os trabalhos (práticos ou teóricos) daquele aluno. Exibe também a soma de todas as notas desse aluno.