Universidade Federal do Rio Grande do Norte
Escola Agrícola de Jundiaí
Tecnologia em Análise e Desenvolvimento de Sistemas
Programação Orientada a Objetos
Profa. Alessandra Mendes

LISTA 3 – 01/10/2023

1. Elabore um programa em Java que implemente a classe Calculadora com as seguintes funções: soma, subtração, divisão e multiplicação. Considere que a calculadora tenha como atributos operador, operando 1 e operando 2 e um construtor que os inicialize com 0. Considere ainda que só exista um único método público (calcular) e que os demais métodos (soma, subtração, divisão e multiplicação) sejam privados.
   Implemente a classe CalculadoraMelhorada, que herda os métodos de calculadora e implementa um construtor, o atributo memoria (que guarda o nome da última operação realizada pela calculadora), o método privado potência() e o método público verUltimaOperacao(). Sobrecarregue o método calcular para que a potência possa ser calculada.
   Implemente, finalmente, a classe Principal que instancie uma Calculadora e uma CalculadoraMelhorada e execute todas as suas funcionalidades a critério do usuário.

2. Elabore um programa em Java que implemente a classe Data, cuja instância (objeto) represente uma data. Esta classe deverá dispor dos seguintes métodos:

construtor	define a data do objeto (através de parâmetro). Este método verifica se a data está correta, caso não esteja a data é configurada como 01/01/0001
compara	recebe como parâmetro um outro objeto da Classe data, compara com a data corrente e retorna: • 0 se as datas forem iguais; • 1 se a data corrente for maior que a do parâmetro; • -1 se a data do parâmetro for maior que a corrente.
getDia	retorna o dia da data
getMes	retorna o mês da data
getMesExtenso	retorna o mês da data corrente por extenso
getAno	retorna o ano da data
clone	o objeto clona a si próprio (cria um novo objeto da classe Data com os mesmos valores de atributos e retorna sua referência pelo método)

   Implemente a classe Voo em que cada objeto representa um vôo que acontece em determinada data e em determinado horário. Cada vôo possui no máximo 100 passageiros, e a classe permite controlar a ocupação das vagas. A classe deve ter os seguintes métodos:

construtor	configura os dados do vôo (número do vôo e data – objeto da classe data)
proximoLivre	retorna o número da próxima cadeira livre
verifica	verifica se a cadeira, cujo número foi recebido como parâmetro, está ocupada

ocupa	ocupa determinada cadeira do vôo, cujo número é recebido como parâmetro, e retorna verdadeiro se a operação foi bem sucedida e falso caso contrário
vagas	retorna o número de cadeiras vagas disponíveis (não ocupadas) no vôo
getVoo	retorna o número do vôo
getData	retorna a data do vôo (na forma de objeto)
clone	o objeto clona a si próprio (idem método clone da classe Data)

Implemente a classe Principal para testar todos os métodos da classe Voo.

3. Escreva uma classe VooMarcado herdeira da classe Voo (questão anterior), que permita definir quantas cadeiras existem no máximo no voo e permita também dividir o avião em ala de fumantes e não fumantes. Para isto, esta classe deve acrescentar os atributos necessários e adicionar os seguintes métodos:

construtor	além dos parâmetros recebidos pelo construtor da superclasse, receberá também como parâmetros o número de vagas do voo e quantas cadeiras serão destinadas para fumantes
maxVagas	determina o número máximo de cadeiras no voo
cadeirasFumantes	determina quantas cadeiras estão destinadas aos fumantes (as demais serão automaticamente destinadas aos não fumantes); as cadeiras dos fumantes serão sempre as últimas do avião
tipo	recebe como parâmetro o número da cadeira e retorna 'F' se for uma cadeira para fumantes e 'N' se for para não fumantes

Os métodos *proximoLivre*, *verifica* e *ocupa* da superclasse devem ser adaptados para tratar o número máximo de vagas informado, ao invés do número fixo de 100.

Implemente a classe Principal para testar todos os métodos das classes Voo e VooMarcado.

4. Crie uma Classe Pessoa, contendo os atributos encapsulados, com seus respectivos seletores (getters) e modificadores (setters). Atributos: nome, endereço e telefone.
   Crie como subclasse da classe Pessoa a classe Fornecedor. Considere que cada instância da classe Fornecedor tem, além dos atributos que caracterizam a classe Pessoa, os atributos valorCredito (correspondente ao crédito máximo atribuído ao fornecedor) e valorDivida (montante da dívida para com o fornecedor). Implemente na classe Fornecedor, além dos usuais métodos sets e gets, um método obterSaldo() que devolve a diferença entre os valores dos atributos valorCredito e valorDivida.
   Implemente a classe Principal para testar todos os métodos das classes anteriores (Pessoa e Fornecedor).

5. A empresa XPTO necessita desenvolver um sistema para catalogar itens colecionáveis (livros, CDs, DVDs e revistas). O objetivo deste sistema é manter os itens colecionáveis, organizados por tipo. O sistema deve permitir cadastrar os dados comuns e os específicos de cada tipo de item. Os dados comuns são: identificação única, título e data de aquisição. Para os livros é importante

manter também, o nome da editora e o ano de publicação. Já para os CDs, é interessante manter o gênero. Para os DVDs é importante armazenar o tipo (musical, filme ou dados). Por fim, das revistas é interessante manter o ano de publicação, o volume e a editora. Desenvolva um modelo de classes que melhor represente as informações acima, utilizando os conceitos da programação orientada a objetos.

6. Elaborar um programa OO que:
    a. Implemente uma classe abstrata C1;
    b. Implemente duas classes concretas C2 e C3, ambas herdando de C1;
    c. Implemente duas classes concretas C4 e C5, ambas herdando de C2;
    d. Implemente duas interfaces, I1 (com 1 método, no mínimo) e I2 (com dois métodos, no mínimo);
    e. Considere que a classe C3 implementa as interfaces I1 e I2;
    f. Considere que a classe C5 implementa a interface I1;
    g. Considere que todas as classes devem ter pelo menos um método e um atributo próprios;
    h. Demonstre no exercício:
    i. Sobrecarga de construtores;
        ii. Sobrecarga de métodos;
        iii. Sobreposição de métodos.
    i. Implemente a classe Principal, para testar todos os métodos das demais classes, contendo pelo menos um objeto de cada classe concreta.

7. Implemente o seguinte conjunto de classes em um programa Java. Obs: Não são permitidas chamadas a System.in, System.out ou similares de dentro das classes criadas. a) Classe: Porta
   Atributos: aberta, cor, dimensaoX, dimensaoY, dimensaoZ.
   Métodos: void abre(), void fecha(), void pinta(String s), boolean estaAberta().
   Para testar, crie uma porta, abra e feche a mesma, pinte-a de diversas cores, altere suas dimensões e use o método estaAberta para verificar se ela está aberta. b) Classe: Casa
   Atributos: cor, porta1, porta2, porta3;
   Método: void pinta(String s), int quantasPortasEstaoAbertas(), int totalDePortas().
   Para testar, crie uma casa e pinte-a. Crie três portas e coloque-as na casa; abra e feche as mesmas como desejar. Utilize o método quantasPortasEstaoAbertas() para imprimir o número de portas abertas. c) Classe: Edificio
   Atributos: cor, totalDePortas, totalDeAndares, portas[];
   Métodos: void pinta(String s), int quantasPortasEstaoAbertas(), void adicionaPorta(Porta p), int totalDePortas(), void adicionarAndar(), int totalDeAndares().
   Para testar, crie um edifício, pinte-o. Crie seis portas e coloque-as no edifício através do método adicionaPorta, abra e feche-as como desejar. Utilize o método quantasPortasEstaoAbertas para imprimir o número de portas abertas e o método totalDePortas para imprimir o total de portas em seu edifício. Crie alguns andares utilizado o método adicionarAndar e retorne o número total de andares utilizando o método totalDeAndares.
   d) As classes Casa e Edifício ficaram muito parecidas. Crie a classe Imovel e coloque nela tudo o Casa e Edificio tem em comum. Faça Imovel superclasse de Casa e Edificio. Note que alguns métodos em comum não poderão ser implementados por Imovel. Logo, esses deverão ser abstratos.

8. Implemente o seguinte conjunto de classes em um programa Java: Veículo, carro, motocicleta, caminhão, brinquedo. Estabeleça as relações entre elas corretamente. Utilize os conceitos de abstração, herança, polimorfismo, encapsulamento e interface.

9. Dada as seguintes interfaces:

Pessoa – representa genericamente uma pessoa	

`public interface Pessoa {`
`public String getCPF();`
`public String getNome();`
`}`	

getCPF	retorna o CPF da pessoa
getNome	retorna o nome da pessoa

Repositorio – representa genericamente um repositório	

`public interface Repositorio {`
`public guarda(Pessoa nova);  public Pessoa recupera(String cpf);  public Pessoa primeiro();`
`public Pessoa proximo();`
`}`	

guarda	guarda uma pessoa
recupera	recupera pessoa com o CPF informado
primeiro	se desloca para a primeira pessoa e a retorna
proximo	se desloca para a próxima pessoa e a retorna

Escreva uma classe, denominada utilitários, que possua os seguintes métodos:

duplica	Recebe como parâmetro dois objetos que implementam a interface Repositorio, A e B, e copia todas as pessoas do repositório A para o repositório B.
diferenca	Recebe como parâmetro três objetos que implementam a interface Repositorio, A, B e C, e coloca no repositório C todas as pessoas de A que não estiverem em B.

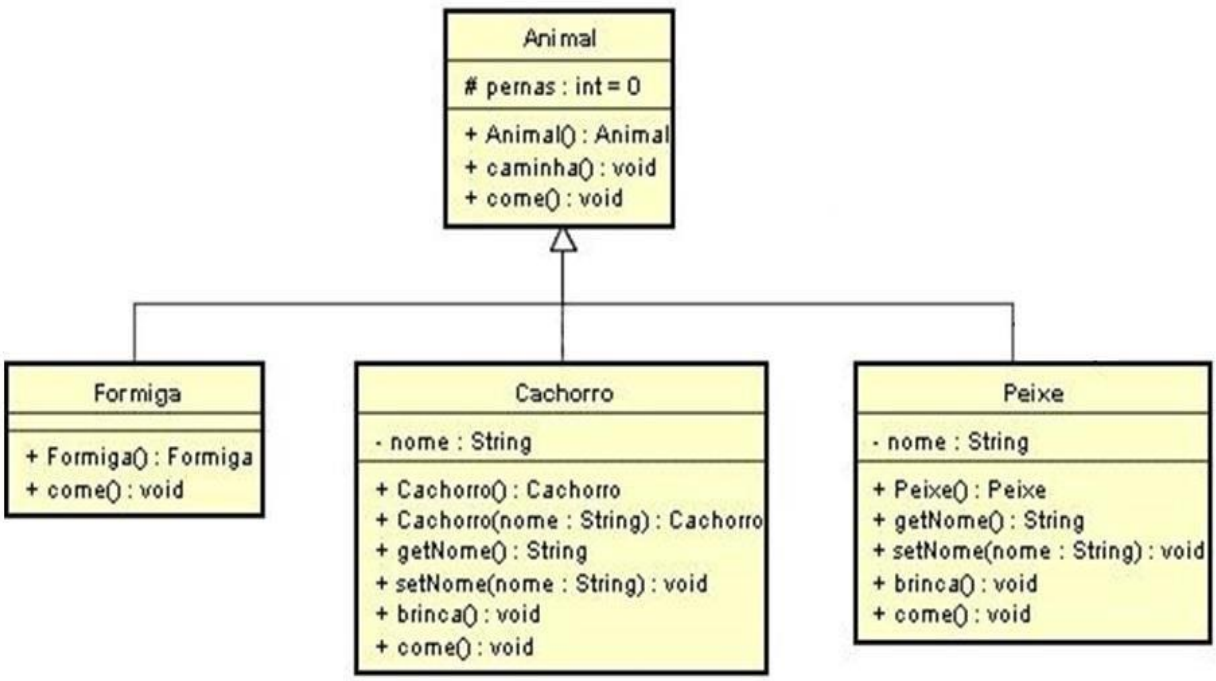
Teste todos os métodos em uma classe Principal.

10. Crie três classes pequenas não relacionadas por meio de herança — as classes Predio, Carro e Aviao. Dê a cada classe alguns atributos e comportamentos adequados únicos que ela não tem em comum com outras classes. Escreva uma interface de AlarmeIncendio com um método *desocupar*. Faça com que cada uma das suas classes implemente essa interface para que o método *desocupar* retorne uma String informando como deve ser a desocupação do objeto em caso de incêndio. Escreva um aplicativo que cria 3 objetos, um de cada uma das três classes, insere as referências a esses objetos em um ArrayList<AlarmeIncendio> e itera pelo ArrayList

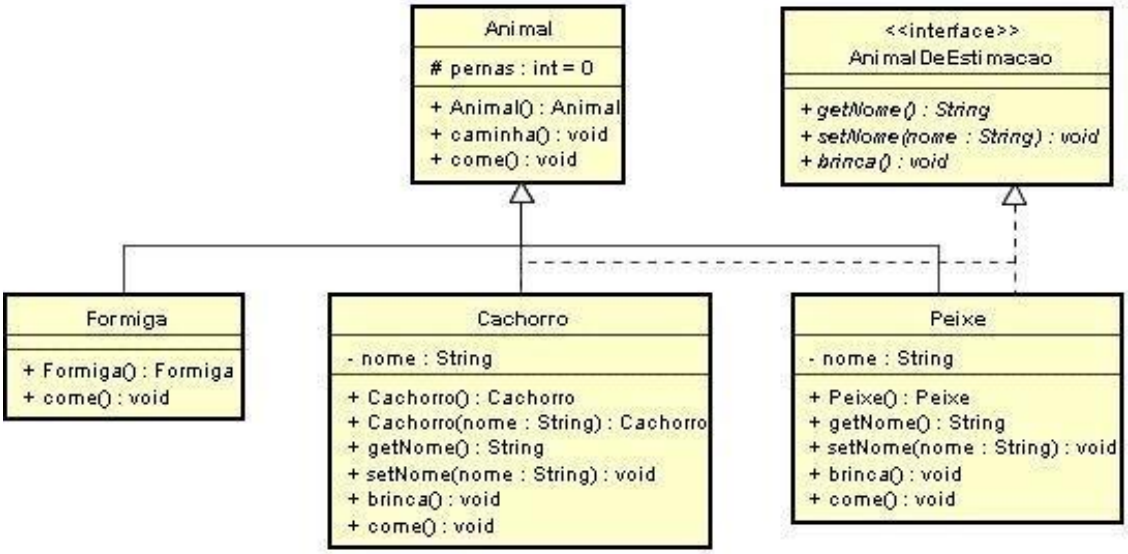
polimorficamente invocando o método *desocupar* de cada objeto (em cada posição). Para cada objeto, escreva suas informações particulares (comportamentos únicos) e as instruções de desocupação em caso de incêndio do objeto.

11. Elabore um programa em Java que:
    - Declare uma classe abstrata A com um atributo encapsulado, dois construtores sobrecarregados (um que não recebe parâmetros e outro que inicializa o valor do atributo), e seus métodos gets e sets;
    - Declare em A dois métodos abstratos;
    - Declare três subclasses, B, C e D, que herdam de A;
    - Em cada subclasse declare os respectivos atributos próprios b1, c1 e d1, seus métodos próprios e construtores que inicializem todos os atributos;
    - Declare uma interface I com dois métodos;
    - Faça com que C e D implementem I;
    - Declare uma classe E que tenha um atributo ArrayList que seja capaz de guardar objetos de B, C e D e dois métodos estáticos: um para retornar o ArrayList e outro para retornar um elemento específico do ArrayList a partir de uma busca;
    - Declare a classe Principal contendo objetos das classes B, C e D;
    - Insira estes objetos no ArrayList da classe E;
    - Teste todos os métodos, inclusive os da interface I e os da classe E.

12. Escreva as classes em Java de acordo com o diagrama abaixo e uma classe Principal que teste seus métodos (não é necessário fazer um menu na classe principal; basta instanciar um objeto de cada classe e chamar todos os seus métodos).



13. Refaça o exercício anterior de acordo com o diagrama abaixo.



OBS: Implemente outros métodos que julgar necessários nas questões.