1) (**Sobrecarga**) Escreva uma classe Ponto2D para representar um ponto no espaço cartesiano de duas dimensões. Esta classe deve fornecer dois construtores, sendo o primeiro um construtor padrão, para inicializar as coordenadas x e y do ponto em questão, e um outro, que permita criar pontos com coordenadas na origem. Esta classe deve ainda implementar os seguintes métodos:
Um método chamado distância,que recebe uma outra instância da classe Ponto2D e retorna um valor do tipo double correspondente à distância euclidiana entre o Ponto2D encapsulado e o passado como argumento.
Um método distância sobrecarregado, que não recebe nenhum argumento, mas calcula a distância euclidiana entre as coordenadas encapsuladas e a origem do sistema de coordenadas. Para isso, dentro do método, crie uma instância de Ponto2D correspondente à origem e passe-a como argumento para o método distância, implementado no item anterior. Crie uma classe teste, que permita testar as funcionalidades da classe definida.

2) (**Sobrecarga**) Utilizando a classe definida no exercício anterior, crie uma classe Reta, para representar uma reta, unida por dois pontos no espaço cartesiano de duas dimensões. Implemente quatro construtores para esta classe: um sem argumentos que considere que a reta comece e termine no ponto (0,0); um que receba um argumento do tipo Ponto2D e que considere que a reta comece na origem e termine no ponto passado como argumento; um que receba duas instâncias da classe Ponto2D como argumentos e um que receba quatro valores de ponto flutuante, correspondentes às duas coordenadas. Crie uma classe teste, que permita testar as funcionalidades da classe Reta definida.

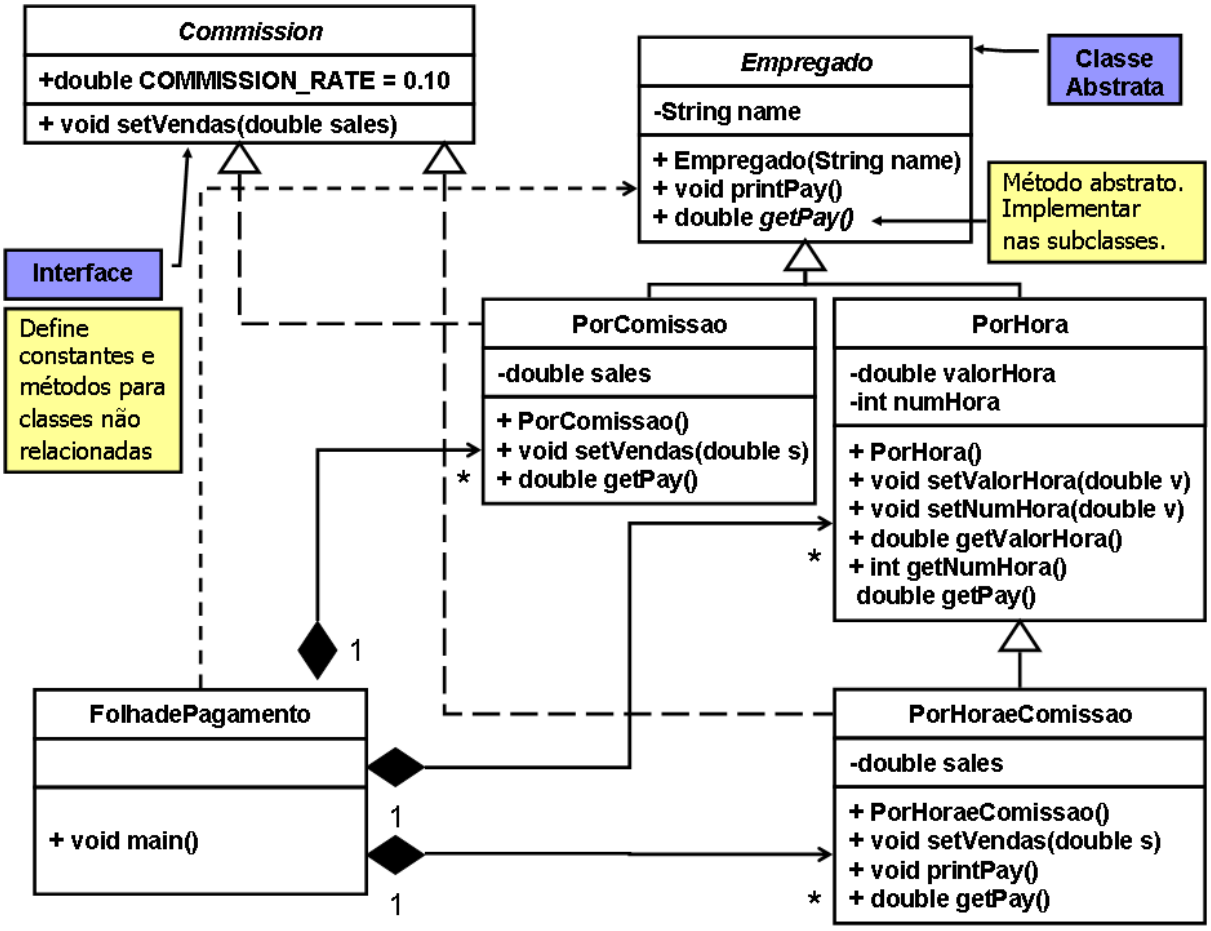
3) (**Sobrecarga**) Escreva três construtores para a classe NumeroComplexo. Um construtor deverá receber os dois valores (real e imaginário) como argumentos, o outro somente o valor real, considerando o imaginário como sendo zero, e o terceiro construtor não recebe argumentos, considerando as partes real e imaginária do número complexo como sendo iguais a zero. Implemente também uma classe teste, para testar os construtores definidos na classe NumeroComplexo.

4) (**Interface**) Escreva uma interface ObjetoGeometrico que representa um objeto geométrico. Essa interface deve ter métodos para mostrar os dados do objeto geométrico e para calcular e retornar sua área e perímetro. Usando essa interface como base, escreva as classes Circulo (contendo um raio), Retangulo (contendo dois valores para os lados) e Triangulo (contendo três valores para os lados).
Dicas: A área de um círculo pode ser calculada com Math.PI*r*r, onde r é o raio do círculo. O perímetro de um círculo é dado por 2*Math.PI*r. A área do retângulo é dada por b*h, onde b é um dos lados e h é o outro lado. Seu perímetro é dado por 2*b+2*h. A área de um triângulo é dada por Math.sqrt(s*(s-a)*(s-b)*(s-c)), onde Math.sqrt é a função que calcula a raiz quadrada, a, b e c são os lados do triângulo, e s é a metade do perímetro do triângulo. O perímetro do triângulo é calculado como (a+b+c).

5) (**Herança**) Usando o exercício anterior, escreva a classe Quadrado, que herda da classe Retangulo, mas somente precisa inicializar um dos lados, e as classes TrianguloEquilatero,

TrianguloIsosceles e TrianguloEscaleno, que precisam inicializar somente um, dois ou três lados do triângulo. Para cada uma dessas classes, quais métodos devem ser sobrepostos e quais podem ser aproveitados?

6)(**Classe Abstrata, Interface**) Dado o diagrama UML abaixo, construir um programa capaz de simular o funcionamento de folha de pagamento com quatro classes de trabalhadores: Empregado, PorHora, PorComissao e PorHoraeComissao. A classe Empregado deve ser abstrata, pois o método getPay(), que retorna o quanto cada tipo de empregado deve ganhar, só poderá ser definido nas subclasses. Desse modo, a classe Empregado deve ser declarada abstrata. Para todas as classes cujo ganho dos trabalhadores está relacionado com a comissão relativa ao montante de vendas (PorComissao e PorHoraeComissao), deve-se empregar o método setVendas e a informação contida no campo COMMISSION_RATE. Por último, a classe FolhadePagamento emprega objetos de todas as classes.



7) (**Herança**) Utilize o conceito de herança para implementar as classes descritas abaixo. Seja a classe Pessoa (atributos nome, cpf e email). A classe Fornecedor é subclasse de Pessoa e cada instância da classe Fornecedor tem, além dos atributos da classe Pessoa, os atributos cred correspondente ao crédito máximo e valorEmDivida que é o montante da dívida com o fornecedor. A classe Fornecedor deve ter um método obterSaldo que devolve a diferença entre os valores dos atributos cred e valorEmDivida.

Seja a classe Empregado, subclasse da classe Pessoa. Cada instância da classe Empregado tem, além dos atributos de Pessoa, os atributos numeroSeccao, salarioBase (vencimento base) e INSS (percentagem retida para INSS). Escreva a classe Empregado com métodos get e set e um método calcularSalario.
Seja a classe Administrador como subclasse da classe Empregado. Um determinado administrador tem como atributos, além dos atributos da classe Pessoa e da classe Empregado, o atributo ajudasDeCusto (ajudas referentes a viagens, estadias, ...). O salário de um administrador é equivalente ao salário de um empregado usual acrescido das ajudas de custo.
Escreva a classe Operario como subclasse da classe Empregado. Um determinado operário tem como atributos, além dos atributos da classe Pessoa e da classe Empregado, o atributo valorProducao (que corresponde ao valor monetário dos artigos produzidos pelo operário) e comissão (que corresponde à percentagem do valorProducao que será adicionado ao salário do operário). O salário de um operário é equivalente ao salário de um empregado usual acrescido da referida comissão.

8) (**Herança, Classe e Métodos Abstratos**) Implemente a hierarquia de classes a seguir.
Classe ContaBancaria. Uma conta bancária possui como atributos um número e um saldo. Como métodos, devem estar presentes os métodos abstratos sacar(double valor), depositar(doble valor) e tirarExtrato().
Classe ContaCorrente. Uma conta corrente possui como atributo um limite de cheque especial, além dos atributos número e saldo, definidos na classe ContaBancaria. O método sacar deve ser implementado nesta classe considerando-se o atributo limite de cheque especial. O método tirarExtrato deve retornar o valor do saldo em conta, acrescido do limite de cheque especial.
Classe ContaPoupanca. Uma conta poupança possui como atributo um valor que indica a taxa de rendimento da conta, além dos atributos número e saldo, definidos na classe ContaBancaria. O método tirarExtrato deve retornar o valor de saldo em conta multiplicado pela taxa de rendimento da conta.
Classe Agencia. Esta classe deve permitir armazenar contas em um array, sejam elas contas corrente ou contas poupança. O construtor da classe Agencia deve receber como parâmetro a quantidade máxima de contas que a agência pode armazenar. Crie, para esta classe, um método addConta, que permita incluir contas na agência; um método montanteContaCorrente, que retorna o montante de saldo acumulado em todas as contas correntes da agência; um método montanteContaPoupança, que retorna o montante de saldo acumulado em todas as contas poupanças da agência e um método imprimeContas, que permita visualizar as informações referentes às contas armazenadas.
ClasseAgenciaTeste. Instanciar nesta classe um objeto da classe Agencia, dois objetos da classe ContaCorrente e um objeto da classe ContaPoupanca. Adicionar à agência as contas criadas. Em seguida, imprimir as informações referentes às contas, bem como o montante total de saldo das contas correntes e o montante total de saldo das contas poupanças.

9) (**Classes e Métodos Abstratos**) Escreva uma classe abstrata chamada CartaoWeb. Essa classe representa todos os tipos de cartões web e conterá apenas um atributo: destinatário (tipo String). Nessa classe você deverá também declarar o método public

abstract void showMessage(). Crie classes filhas da classe CartaoWeb: DiaDosNamorados, Natal, Aniversario. Cada uma dessas classes deve conter um método construtor que receba o nome do destinatário do cartão. Cada classe também deve implementar o método showMessage(), mostrando uma mensagem ao usuário com seu nome e que seja específica para a data comemorativa do cartão. Escreva um programa e no método main crie um array de CartaoWeb. Insira instâncias dos 3 tipos de cartões neste array. Após, use um laço para exibir as mensagens deste cartão chamando o método showMessage().

10) (**Exceção**) Dada a classe Quadrado a seguir:

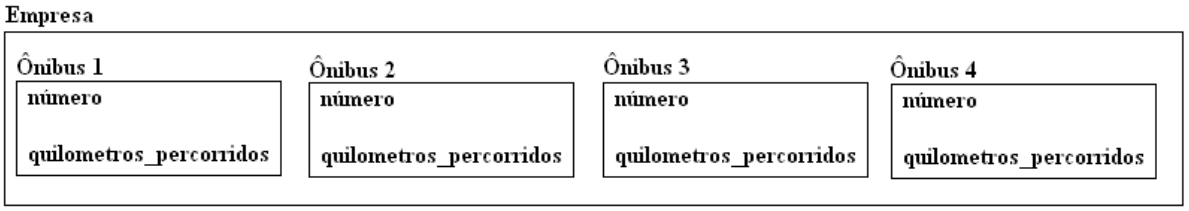
```
public class Quadrado{
        private double lado;
        public setLado(double l){
                lado = l; }
public double área() {
        return lado*lado; }
}
```

Modifique esta classe de forma que uma exceção específica seja lançada pelo método setLado caso o valor informado para o lado do quadrado seja negativo. Essa exceção deve ser tratada no método main, permitindo que o usuário tente novamente fazer a entrada de dados, digitando um número válido (inteiro e positivo).

11) (**Exceção**) Desenvolva um programa Java que crie um vetor de 5 posições de Strings. O programa deve solicitar ao usuário que preencha cada posição do vetor com valores que serão lidos. Em seguida, o programa deve permitir ao usuário realizar uma busca no vetor citado, informando a posição do vetor que deseja pesquisar. O programa então deve retornar a String localizada na posição informada e, também, qual o caracter que se encontra na quinta posição dessa String. Os resultados também devem ser informados utilizando-se caixas de diálogos. O programa deve ainda tratar todos os tipos de exceções que possam ser geradas em tempo de execução do mesmo, informando ao usuário qual o erro cometido.

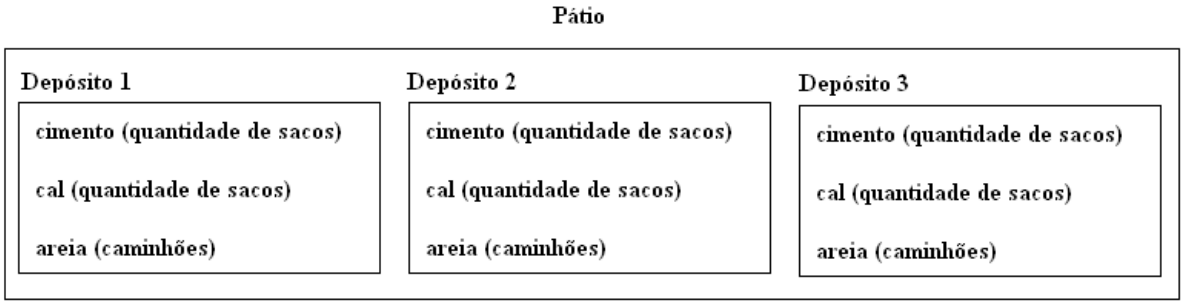
12) (**Exceção**) Desenvolva um programa em Java que contenha um método que recebe uma String como parâmetro e verifica se a mesma é composta apenas por caracteres maiúsculas. O método deve gerar dois tipos de exceções específicas: uma para indicar se existe algum caracter que não é uma letra e outra para indicar que uma das letras não é maiúscula. Para verificar o tipo dos caracteres use os métodos isLetter e isUpperCase, da classe Character, ambos static. Crie também uma classe teste para validar o método descrito acima e tratar as exceções que eventualmente são geradas pelo mesmo.

13) (**ArrayList**) Considere uma empresa de ônibus,onde, para todos os ônibus deseja-se armazenar informações como número do ônibus e quilometragem total percorrida durante uma semana.



Escreva um projeto em Java que implemente a(s) classe(s) necessária(s) para modelar a situação descrita acima. Considere ainda que a empresa deve ser capaz de cadastrar novos ônibus em sua frota, bem como verificar a quilometragem total percorrida por todos os ônibus, juntamente com o maior e menor percurso realizado. Em seguida, crie uma classe para testar a(s) classe(s) implementadas.

14) (**ArrayList**) Considere uma loja de materiais de construção especializada na venda de três tipos de produtos: cimento, cal e areia. Esta loja possui um pátio, composto por vários depósitos, nos quais estão armazenados seus estoques disponíveis para venda, conforme ilustrado em figura abaixo:



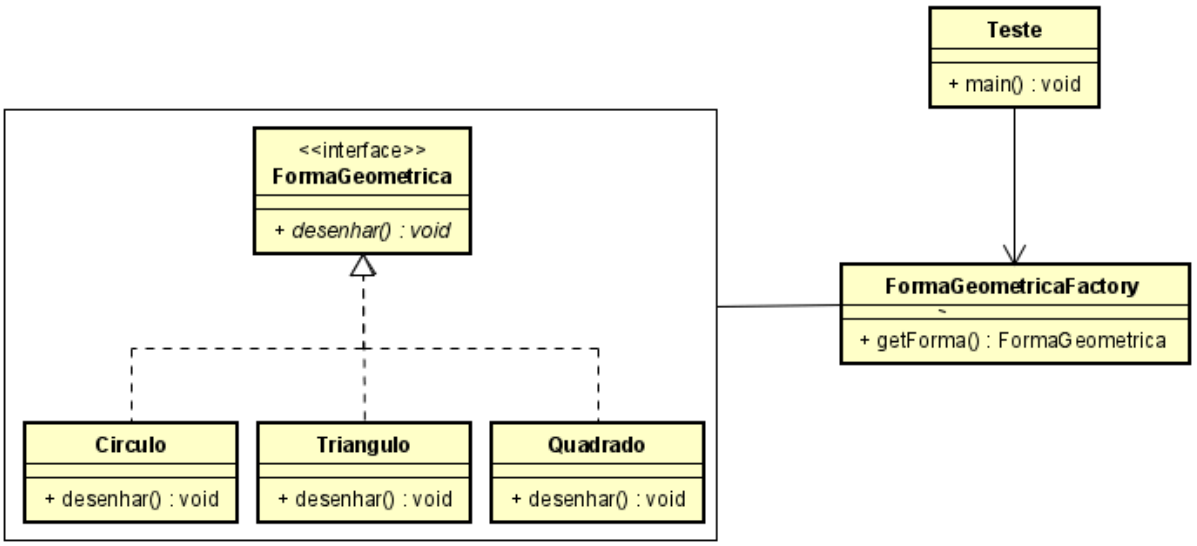
Deseja-se desenvolver um projeto em Java capaz de modelar a situação descrita acima. Considere na implementação que mais depósitos possam ser cadastrados no pátio. Deseja-se ainda verificar o total de cada material (areia, cal e cimento) disponível em estoque, considerando-se os depósitos disponíveis. Uma classe Teste deve estar presente para testar as funcionalidades das demais classes.

15) (**Padrões de projeto**) Escreva, compile e execute o programa abaixo. Em seguida, troque sua implementação para que a classe Incremental seja Singleton. Execute novamente e veja os resultados.

```
class Incremental {
	private static int count = 0;
	private int numero;
	public Incremental() {
		numero = ++count;
	}
	public String toString() {
	return "Incremental " + numero;
	}
}
```

```
public class TesteIncremental {
	public static void main(String[] args) {
		for (int i = 0; i < 10; i++) {
			Incremental inc = new Incremental();
			System.out.println(inc);
		}
	}
}
```

16) (**Padrões de projeto**) Utilize o padrão Factory Method para a criação de formas geométricas. O método desenhar deve imprimir na tela o nome da forma geométrica.



17) (**Padrões de projeto**) Utilize o padrão Abstract Factory para a criação de uma fábrica de carros. O sistema pode construir carros de dois tipos: Sedan ou Popular.
A) Sedan
i - Siena - Fiat;
ii - Fiesta Sedan - Ford;

A) Popular
i - Palio - Fiat;
ii - Fiesta - Ford;

Dica: Utilize o seguinte diagrama de classes como referência.