Lista de Exercícios – Orientação a Objetos – 2018.3

1) Considerando as características e propriedades de POO, aponte 5 (cinco) vantagens do paradigma de orientação a objetos para programação.
2) Qual o conceito de máquina virtual? Faça um comparativo entre as linguagens que não usam máquina virtual e as que usam. Quais são os pontos positivos e negativos?
3) Descreva o que são objetos e classes. Há uma relação entre eles?
4) O que são métodos e atributos?
5) Quais as diferenças dos modificadores de acesso `private`, `protected`, `public` e `default`(sem indicador)?
6) Qual o propósito do operador `new` ? O que acontece quando você o executa?
7) O que é a referência `this` e para que ele é utilizado?
8) O que é sobrecarga? Para que é utilizado?
9) Descreva a finalidade de `package`.

10) Escreva uma classe para representar uma lâmpada que está à venda em um supermercado. Imagine que essa lâmpada possa ter três estados: apagada, acesa e meia-luz. Além disso, ela tem uma operação "`estaLigada`" que retorne verdadeiro se a lâmpada estiver ligada e falso, caso contrário. Crie uma classe `TesteLampada` que implementa o método `main` com as seguintes operações:
   a. Instancie 2 objetos dessa classe.
   b. Ligar um dos objetos de `Lampada` e desligar o outro.
   c. Mostrar na tela se as lâmpadas dos objetos estão ligadas ou desligadas.

11) Escreva a classe `ConversaoDeUnidadesDeArea` com métodos estáticos para conversão das unidades de área segundo a lista abaixo.
   a. 1 metro quadrado = 10.76 pés quadrados
   b. 1 pé quadrado = 929 centímetros quadrados
   c. 1 milha quadrada = 640 acres
   d. 1 acre = 43.560 pés quadrados

12) A área de um campo de futebol é de 8.250 metros quadrados. Usando a classe `ConversaoDeUnidadesDeArea`, escreva um programa em Java que mostre qual é a área de um campo de futebol em pés quadrados, acres e centímetros quadrados. Escreva métodos adicionais para a classe `ConversaoDeUnidadesDeArea`, se necessário.

13) Escreva uma classe em Java que represente um país. Um país tem como atributos o seu nome, o nome da capital, sua dimensão em Km2 e uma lista de países com os quais ele faz fronteira. Forneça os seguintes construtores e métodos:
   a. Construtor que inicialize o nome, capital e a dimensão do país;
   b. Métodos de acesso (getters/setters) para as propriedades indicadas no item (a);
   c. Um método que permita verificar se dois países são iguais. Dois países são iguais se tiverem o mesmo nome e a mesma capital. A assinatura deste método deve ser: `public boolean equals(Pais outro);`
   d. Um método que define quais outros países fazem fronteira (note que um país não pode fazer fronteira com ele mesmo);
   e. Um método que retorne a lista de países que fazem fronteira;
   f. Um método que receba um outro país como parâmetro e retorne uma lista de vizinhos comuns aos dois países.

14) Escreva uma classe "`Funcionario`" com os atributos `matricula (int), nome (String), departamento (int), salário (float) e função (String).` Adicione na classe um construtor que receba todos os parâmetros para inicializar os dados de um funcionário.

15) Escreva uma classe "`SetorPessoal`" que tenha como atributo um vetor da classe "`Funcionario`" e uma variável inteira para ser usada como índice do vetor. Crie um construtor que receba como parâmetro o número de funcionários de uma empresa para instanciar o vetor e inicialize o índice do vetor com zero. Acrescente os seguintes métodos à classe:

a. um método para adicionar funcionários no vetor definido na classe;
b. um método que possa imprimir a folha de pagamento informando o nome dos funcionários e os seus respectivos salários.
c. um método que possa retornar o valor total da folha de pagamento.
d. um método que possa retornar o nome do funcionário que recebe o maior salário.
e. um método que possa receber como parâmetro o número de um determinado departamento e mostrar o nome e o cargo de todos os funcionários daquele departamento.
f. um método que possa receber como parâmetro o nome de uma determinada função e posteriormente imprimir o nome de todas as pessoas que exercem essa função.
g. um método que possa imprimir a folha de pagamento informando o nome dos funcionários e os seus respectivos salários em ordem crescente de salário.
   Dica: É necessário criar um outro vetor que conterá o vetor original e posteriormente ordena-lo.

16) Escreva uma classe chamada `Quadrado` com atributo lado iniciado no método contrutor da classe. A classe deve conter também os métodos `calcularArea` e `calcularPerimetro` que retornarão respectivamente a área e o perímetro do quadrado. Escreva também uma classe `UsaQuadrado` que implementa o método `main()` que instancia três objetos da classe `Quadrado`. O método `main` deverá apresentar a área e o perímetro dos três quadrados.

17) Crie uma classe `Retangulo` com atributos altura e largura, cada um dos quais assume o padrão de 1. Forneça métodos que calculem o perímetro e a área do retângulo. A classe deve conter os métodos `get` e `set` para comprimento e largura. Os métodos `set` e get devem verificar se comprimento e largura são cada um números de ponto flutuante maiores que 0,0 e menores que 20,0. Escreva também um programa para testar sua classe `Retangulo`.

18) Escreva um programa completo para jogar o jogo da velha. Para tanto crie uma classe `JogoDaVelha`:

- a classe deve conter como dados privados um array bidimensional 3x3 para representar a grade do jogo.
- crie uma enumeração para representar as possibilidades de ocupação de uma casa na grade (vazia, jogador 1 ou jogador 2).

- o construtor deve inicializar a grade como vazia.
- forneça um método para exibir a grade.
- permita dois jogadores humanos.
- forneça um método para jogar o jogo; todo movimento deve ocorrer em uma casa vazia; depois de cada movimento, determine se houve uma derrota ou um empate.

19) Uma clínica te contrata para informatizar o prontuário de um cliente. Para isso defina a classe `Prontuario`, que deverá ter como informações:

- o nome do paciente, a data de nascimento, além do peso e da altura. Construa os métodos `get/set` para essas características. Crie um método que imprime todas as informações do prontuário. Construa agora um método que calcule o IMC do paciente, baseado no seu peso e altura, retornando esse valor. IMC = peso/altura²

Crie um método chamado `preDiagnosticar` que informará a situação do paciente, com base no valor do IMC, de acordo com a tabela abaixo:

IMC	**Situação**
Abaixo de 18.5	Você está abaixo peso ideal
Entre 18.5 e 24.9	Você está no seu peso ideal
Entre 25 e 29.9	Você está com sobrepeso
Entre 30 e 34.	Obesidade Grau I
Entre 35 e 39.9	Obesidade Grau II
Acima de 40	Obesidade Mórbida

Agora construa a classe `Principal`, com o método `main` para testar as classes recém-criadas. Crie pelo menos 4 prontuários. Execute os métodos que imprime todas as informações do prontuário e também execute o método `preDiagnosticar`, informando na tela o retorno do método.

20) Crie uma classe `Data` com as seguintes capacidades:

- Gerar saída da data em múltiplos formatos, como: DD/MM/AAAA; Agosto 08, 2017; DDD AAAA;
- Utilize construtores sobrecarregados para criar objetos `Data` inicializados com datas no formato definido em cima.

- No primeiro caso, o construtor deve receber 3 valores inteiros.
- No segundo caso, deve receber uma String e 2 valores inteiros.
- No terceiro caso deve receber 2 valores inteiros, o primeiro representando o número de dias no ano.
- Faça uma classe principal que implemente o método `main` que deverá instanciar 2 objetos de cada formato e depois imprimir cada um deles.

21) Escreva uma classe em Java chamada `Estoque`:

Atributos: `nome; quantidadeAtual; quantidadeMinima`. (Observação: os atributos `quantidadeAtual e quantidadeMinima` não poderão ser negativos)

Métodos:

a. Construtor que inicializa todos os atributos da classe;
b. Um método para alterar o nome do produto;
c. Um método para alterar a quantidade mínima do produto;
d. Um método para repor estoque do produto;
e. Um método para dar baixa no estoque do produto;
f. Um método para mostrar os dados de um produto no estoque;
g. Um método que verifica se é preciso repor um determinado produto.

Escreva uma classe `TesteEstoque` que implementa o método `main` com as seguintes ações:

1. Instanciar 4 objetos da classe Estoque:
    a. produto1:
        i. Nome: Impressora Jato de tinta;
        ii. Quantidade atual: 13;
        iii. Quantidade mínima: 6.
    b. produto2:
        i. Nome: Monitor LCD 17 polegadas;
        ii. Quantidade atual: 11;
        iii. Quantidade mínima: 13.
    c. produto3:
        i. Nome: Mouse óptico;

ii. Quantidade atual: 6;

iii. Quantidade mínima: 2.

d. produto4: Você deverá especificar o nome, quantidade atual, quantidade mínima.

2. Dar baixa de 5 unidades do produto1;
3. Fazer a reposição de 7 unidades do produto2;
4. Dar baixa em 4 unidades do produto3;
5. Executar o método `precisaRepor` dos três produtos;
6. Mostrar todas as informações dos 3 produtos

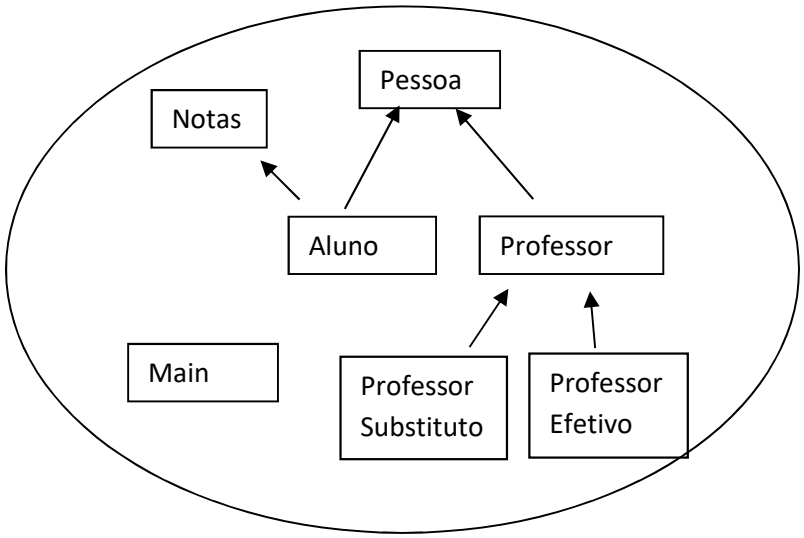
22) Crie uma classe que representará um Funcionário de uma Empresa.

- Atributos:
  - matricula;
  - nome;
  - idade;
  - cargo;
  - salario;
- Faça o encapsulamento dos atributos desta classe.
- Criar os métodos `get e set`.
- Implementar três métodos construtores para a classe:
- O 1º não deve receber parâmetros, mas deve inicializar o valor do atributo salário com R$1.500,00.
- O 2º deve receber o nome e a idade do funcionário como parâmetros e inicializar estes dois atributos com os respectivos valores recebidos.
- O 3º deve receber a matricula, o nome e o salário do funcionário como parâmetros e inicializar estes três atributos com os respectivos valores recebidos.
- Cria uma classe de nome `Main` que deverá implementar o método `main`(). No método `main`():
  - Instanciar um objeto da classe `Funcionario` utilizando o construtor sem parâmetros.
  - Imprimir os valores dos atributos do objeto.

- Instanciar um outro objeto da classe `Funcionario` utilizando o método construtor com os parâmetros nome e idade.
- Imprimir os valores dos atributos do segundo objeto.
- Instanciar um outro objeto da classe `Funcionario` utilizando o método construtor com os parâmetros matricula, nome e salário.
- Imprimir os valores dos atributos do terceiro objeto.
- Modificar a classe `Funcionario` criando um atributo estático `numeroFuncionarios` que deverá manter o número total de objetos instanciados da classe para controlar o número de funcionários.

23) Desenvolver um sistema de cadastro de Pessoas, Alunos e Professores.

Classes envolvidas



Classe pessoa:

Atributos: String nome, telefone, endereco, cpf;

int identidade, idade;

Métodos: `public pessoa(String n, String t, String e, String c, int id, int i)`

```
public pessoa(String n, int i)
public String getNome()
public String getTelefone()
```

```
public String getEndereco()
public String getCpf()
public int getIdentidade()
public int getIdade()
public int getIdade(int ano_atual, int ano_futuro)
```

Classe Notas:

Atributos: `float nota1, nota2, nota3, media;`

Métodos: `public notas(float n1, float n2, float n3)`

```
public notas()
public void cal_media()
```

Classe Aluno extends Pessoa:

Atributos: int matricula;

notas notaAluno = new notas();

Métodos:
```
public aluno(String n, String t, String e, String c, int id,
             int i, int m, float n1, float n2, float n3)
```

Abstract classe Professor extends Pessoa:

Atributos: float salario;

int horas;

Métodos: `public professor(String n, String t, String e, String c, int id, int i, int h)`

```
abstract void calculo_salario();
public float mostra_salario()
```

Classe ProfessorEfetivo extends Professor:

Atributos: int horas_pesquisa;

Métodos: public ProfessorEfetivo(String n, String t, String e, String c, int id, int i, int h, int hp)

```
public void calculo_salario()
```

Classe ProfessorSubstituto extends Professor:

Atributos: int horas_atendimento;

Métodos: public ProfessorSubstituto(String n, String t, String e, String c, int id, int i, int h, int ha)

```
public void calculo_salario()
```

Classe Main:

a) Responsável em ler todas as informações para o cadastramento de Pessoas, Alunos, Professores Substituto e Efetivos e colocar no vetor adequado.
b) Responsável pela leitura de um nome e a pesquisa no vetor adequado.
c) Responsável em mostrar as informações existentes no objeto com o nome solicitado.

24) De forma incremental, traduza o seguinte conjunto de classes em um programa Java.

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. Utilize uma classe principal.

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. Utilize uma classe principal.

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. Cria alguns andares

utilizado o método `adicionarAndar` e retorne o número total de andares utilizando o método `totalDeAndares`. Utilize uma classe principal.

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`.

25) Há uma preocupação atual com as pegadas de carbono (*carbon footprints*, emissões anuais de gás carbônico na atmosfera) a partir de instalações que queimam vários tipos de combustíveis para aquecimento, veículos que queimam combustíveis para se mover, e assim por diante. Nesse cenário:

Crie três pequenas classes não relacionadas por herança - classes `Building`, `Car` e `Bicycle`. Dê a cada classe alguns atributos e comportamentos (métodos) únicos que ela não tem em comum com as outras classes. Sugestões:

-`Building`: número de pessoas (int), uso de energia renovável (boolean), número de lâmpadas (int), uso de ar-condicionado (boolean).

-`Car`: combustível (string), cilindrada (float).

Escreva uma interface `CarbonFootprint` com um método `getCarbonFootprint`. Faça cada uma das suas classes implementar essa interface, para que seu método `getCarbonFootprint` calcule uma pegada de carbono apropriada a cada classe (usando os atributos sugeridos ou outros – consulte na internet como calcular).

Escreve um aplicativo que crie 2 objetos de cada uma das três classes. Crie um objeto `ArrayList<CarbonFootprint>` e insira as referências dos objetos instanciados nessa coleção. Finalmente, itere pela coleção, chamando polimorficamente o método `getCarbonFootprint` de cada objeto. Para cada objeto, imprima algumas informações de identificação e as pegadas de carbono do objeto.

26) Modifique o exercício anterior, tornando `Building` uma classe abstrata e implementando as duas novas subclasses concretas `House` e `School`. O aplicativo que cria a coleção de objetos vai continuar funcionando após a modificação na estrutura de classes? Modifique o aplicativo para que passe a instanciar diretamente objetos `House` e `School`, incluindo-os na coleção

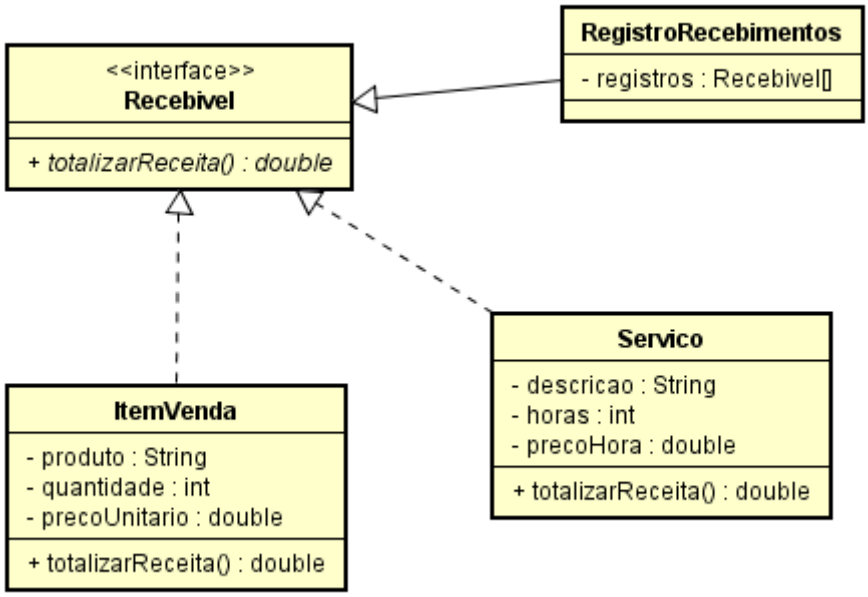
27) Uma empresa paga seus funcionários semanalmente. Os funcionários são de quatro tipos: Funcionários assalariados recebem salários fixos semanais independentemente

do número de horas trabalhadas, funcionários que trabalham por hora são pagos da mesma forma e recebem horas extras (isto é, 1,5 vezes sua taxa de salário por hora) por todas as horas trabalhadas além das 40 horas normais, funcionários comissionados recebem uma porcentagem sobre suas vendas e funcionários assalariados/ comissionados recebem um salário-base mais uma porcentagem sobre suas vendas. Para o período salarial atual, a empresa decidiu recompensar os funcionários assalariados/comissionados adicionando 10% aos seus salários-base. A empresa quer escrever um aplicativo Java que realiza os cálculos da folha de pagamento polimorficamente.

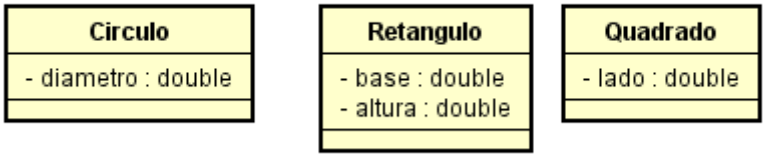
28) Suponha agora que a empresa deseje realizar várias operações de contabilidade em único aplicativo de contas a pagar – além de calcular os vencimentos que devem ser pagos para cada funcionário, a empresa também deve calcular o pagamento devido de cada uma das várias faturas (isto é, contas de mercadorias adquiridas). Embora aplicadas a coisas não relacionadas (isto é, funcionários e faturas), as duas operações têm a ver com a obtenção de algum tipo de quantia de pagamento. Para um funcionário, o pagamento se refere aos vencimentos do funcionário. Para uma fatura, o pagamento se refere ao custo total das mercadorias listadas na fatura.

29) Você deve desenvolver um sistema para empresas que prestam serviços de manutenção de computadores e que comercializam peças e equipamentos de informática. O sistema precisa permitir que sejam registrados todos os valores recebidos dos clientes, tanto os relativos à prestação de serviços quanto os oriundos de vendas de mercadorias. Construa este sistema em conformidade com a estrutura apresentada na figura abaixo.

OBS: A classe `RegistroRecebimentos` é a classe executável do sistema e deverá permitir o registro de recebimentos e a impressão da lista desses registros (coloque como opções no menu para o usuário decidir). A classe `RegistroRecebimento` deverá criar instâncias das classes `ItemVenda` e `Servico` para representarem cada recebimento realizado e deverá gravá-las em um atributo registros. A interface `Recebivel` deve ser utilizada como tipo na declaração desse atributo.



30) Considere as classes abaixo:



É necessário criar um método que calcule a área para as figuras geométricas. Como você implementaria? Um método específico em cada uma das classes? Utilizaria herança? Utilizaria interfaces? Implemente e justifique sua escolha de implementação. Escreva como comentário no método `main`().

31) Pesquise na API Java os tipos de Exceção `IllegalArgumentException` e `NumberFormatException`. O que causa essas exceções? (escreva como comentário em seu código). Implemente um programa em Java que faça o tratamento dessas exceções

32) Implemente a classe "`ContaCorrente`":

Atributos (todos float):

- "`limite`" que armazena a quantidade atual disponível do limite da conta que o usuário possui;
- "`saldo`" que é o valor que realmente é pertencente ao usuário;
- "`valorLimite`" que consiste no valor máximo que o banco lhe fornece como valor de limite.

Métodos:

- `public void sacar(float valor)`
- `depositar(float valor)`
- `setValorLimite(float valor).`

Na construção dos três métodos faça com que eles lancem exceções relacionados aos argumentos, por exemplo, sacar, depositar ou setar um valor negativo para esses eventos. Lance também uma exceção no caso de tentar sacar um valor maior que o possível.

Nos métodos lancem a exceção "`IllegalArgumentException`()" (Java)com comando "`throw`", passe o motivo da exceção (uma String) em seu construtor.

Implemente a classe `Main`, onde se deve instanciar um objeto da classe "`ContaCorrente`", usar os três métodos construídos acima e tratar as exceções propostas com os comandos "`try`" e "`catch`", no catch imprimir na tela o método "`getMessage`()" da exceção declarada no catch. Cause condições apropriadas para a execução das exceções.

33) Implemente a Classe `Cliente`:

a. Identifique os atributos e métodos dessa classe (incluir getters/setters). Atributo obrigatório: CPF
b. Os métodos set devem lançar exceções do tipo `DadoInvalidoException` quando o dado passado como parâmetro não for válido.

Implemente a classe `CadastroCliente`:

c. Deve possuir um conjunto de `Cliente` (utilizar `array` de tamanho 10)
d. Deve conter os métodos `inserir` (que adiciona um novo cliente ao conjunto de `Cliente`) e o `buscar` (que busca um cliente no conjunto)
e. O método `inserir` deve lançar a `RepositorioException` quando não puder mais inserir clientes no array e a exceção `ElementoJaExistenteException` quando um cliente com o mesmo CPF já estiver cadastrado.
f. O método `buscar` deve lançar a exceção `ElementoInexistenteException` quando o cliente que deseja buscar não estiver cadastrado.