

LISTA L1

PROF. ME. ALEXANDRE GARCIA

1. ORIENTAÇÃO A OBJETOS - BÁSICO

Exercício 1.1. *Dada a classe Pessoa, que possui os atributos: String nome, String sexo, int idade, boolean vegetariana. Faça agora uma classe Churrasco que possua:*

Atributos: qtdCarne(double);

Método: verificarConsumo(): Recebe via parâmetro uma Pessoa, e com isto define a consumação média de carne(quantidade de carne consumida), pessoas de 0 a 3 anos não consomem, vegetarianos também não. Pessoas de 4 a 12 anos consomem 1 kilo de carne e de 13 anos em diante 2 kilos de carne.

Exercício 1.2. *Implemente uma classe Lâmpada com os seguintes componentes, e faça um teste ao final:*

Atributos: estado(String)

Métodos:

- *click(): ao chamar este método a lâmpada é colocada no estado "apagada", caso esteja "acesa", e é colocado no estado "acesa" caso esteja "apagada".*
- *qtdAcendimentos(): retorna quantas vezes a lâmpada foi acesa (DICA: este método deve ser chamada no método acima).*
- *checaEstado(): retorna o estado atual da Lâmpada.*

Exercício 1.3. *(**) Implemente uma classe chamada Complexo para representar números imaginários e esta deve possuir:*

Atributos: dois doubles a(Parte real) e b(Parte imaginária).

Métodos:

- (1) *Construtor;*
- (2) *soma(): recebe via parâmetro outro número complexo(objeto desta classe) e efetua sua soma, ou seja, parte real será somada com parte real, e parte imaginária com parte imaginária.*

- (3) *multiplica()*: recebe via parâmetro outro complexo(objeto desta classe) e efetue a formula $(a+bi)*(c+di) = (ac-bd)+(ad+bc)i$
- (4) *toString()*: Mostra uma string na tela com os atributos a e b na notação Complexa $a+bi$;
- (5) *modulo()*: retorna o modulo do número complexo que é dado por $|a+bi|=\sqrt{a^2+b^2}$
- (6) *argumentoPrincipal()*: retorna o ângulo formado pelo número complexo no plano de argand-gauss que é dado pela fórmula $\theta = \tan^{-1}\left(\frac{b}{a}\right)$

Exercício 1.4. Implemente a classe *Cliente* que possua os atributos *nome*, *saldo* e *limite*. Esta deve possuir também os métodos *sacar()*, *depositar()*, *checarSaldo()* e *obterNome()*. Sabe-se que só é possível sacar se o *saldo+limite* forem superiores a *quantia*. Os métodos *sacar()* e *depositar()* necessitam de parâmetros. O método *checar saldo* deve retornar o valor *saldo+limite*. O método *obterNome()* deve retornar o nome do *Cliente*.

Exercício 1.5. Implemente a classe *Doc* que possua o método *transferir()* que recebe via parâmetro dois *Clientes* *c1* e *c2*(ver exercício acima) e a *quantia* (necessita uma verificação de saldo). Deve ser tirado da conta de *c2* e colocado na conta de *c1*, exiba também uma mensagem de conclusão de transferência explicitando os nomes dos envolvidos.

Exercício 1.6. Implemente uma classe que modele um triângulo equilátero(lados iguais)

Atributos: *lado*, *perímetro*, *área*.

Métodos: *calcArea()*, *calcPerímetro()* e seus gets. O *lado* deverá ser o único atributo inicializado via construtor.

Fórmulas:

$$\text{Área} = \text{lado} * \frac{\sqrt{3}}{2}$$

$$\text{Perímetro} = 3 * \text{lado}$$

Exercício 1.7. Implemente a classe *Cliente* que contenha os atributos: *nome*, *cpf* (*Strings*) e *telefone* (*Telefone*). E que contenha os métodos: *mostrarDados()* e *adicionarTelefone()*, o primeiro deve mostrar todos os dados do cliente, incluindo o telefone e o último deve associar um novo telefone ao cliente. Implemente a classe *Telefone* que possua os atributos: *ddd* e *número* (*Strings*) e os métodos: *obterNumero()* e *obterDDD()*.

Exercício 1.8. *Implemente uma classe que modele um jogo de adivinhação de números de 0 a 99.*

Atributos: um número inteiro sorteado.

Métodos: Sortear(), Advinhar().

OBS: O objeto para gerar número aleatórios no java é o Random, você deve instanciá-lo e chamar seu método nextInt() que deve possuir um argumento inteiro no caso aqui 100.

Exercício 1.9. *Implemente a classe Eq2Grau que possua:*

- *Atributos: a, b e c (doubles);*
- *Métodos: delta(): retorna o delta da equação;*
- *raiz1(): retorna a primeira raiz se $\Delta \geq 0$, se não retorna NaN;*
- *raiz2(): retorna a segunda raiz se $\Delta \geq 0$, se não retorna NaN.*

Exercício 1.10. *Implemente a classe Porta que possua:*

- *Atributos: isOpen(boolean), numAberturas(int);*
- *Métodos: abrir(): abre a porta e conta 1 na contagem de aberturas;*
- *fechar(): fecha a porta. OBS: O atributo numAberturas deve contar o total de aberturas de todas as portas possíveis.*

Exercício 1.11. *Faça casos de teste usando o JUnit para cada exercício acima.*