Exercícios de Java e OO

Lógica estruturada

Repetições

1) Fazer uma classe Ex1Primos para:
   - Receber um inteiro N do usuário
   - Testar se este inteiro é primo ou não e informar

2) Fazer uma classe Ex2Sorteio para:
   - Sortear um número de 0 a 1000 (dica: usar *Math.random()*)
   - Pedir um palpite ao usuário. Se ele errar, informar se o palpite é maior ou menor do que o número sorteado.
   - Pedir novos palpites até que o usuário acerte e, depois disso, mostrar em quantas tentativas ele acertou.

3) Fazer um programa para receber dois números do tipo *int* do usuário e determinar se um número é permutação do outro ou não. Ex: 431 é permutação de 143, 42 é permutação de 204, 1211 é permutação de 1112, etc. O programa só deve aceitar números positivos.

4) Fazer um programa para medir os reflexos do usuário. O programa deve:
   - Mostrar a palavra "Agora!" após um tempo aleatório e um número, também aleatório
   - Contar o tempo até que o usuário digite o número pedido e mostrar esse tempo.
   - Dicas: usar o método *getTimeInMillis* da classe *Calendar* ou o método *nanoTime* da classe *System.*

String

5) Fazer uma classe *ExecString* que:
   - Recebe duas strings do usuário (usar *TextConsole.getString())*
   - Conta e informa quantas vezes a segunda string ocorre dentro da primeira
   - Informa uma estatística dos caracteres contidos nas 2 strings.

6) Faça um programa que receba um nome completo na forma de uma *String* e mostre a abreviatura deste nome. Não se devem abreviar as palavras com 2 ou menos letras. A abreviatura deve vir separada por pontos. Ex: Paulo Jose de Almeida Prado. Abreviatura: P. J. de A. P.

7) Fazer um programa que receba uma string do usuário e mostre o conteúdo desta string de forma invertida.

8) Um dos sistemas de encriptação mais antigos é atribuído a Júlio César: se uma letra a ser encriptada é a letra de número N do alfabeto, substitua-a com a letra (N+K), onde K é um número inteiro constante (César utilizava K = 3). Usualmente consideramos o espaço como zero e todos os cálculos são realizados com módulo-27. Dessa forma, para K = 1 a mensagem "Ataque ao amanhecer" se torna "bubrfabpabnboifdfs". Faça um programa que receba como entrada uma mensagem e um valor de J e retorne a mensagem criptografada pelo código de César. Fraquezas: apenas 26 chaves possíveis. É possível utilizar conhecimento da linguagem para facilitar a busca.

Array

9) Fazer uma classe Ex3Array com as seguintes características:

- Atributos: array de inteiros e duas variáveis para controlar o número atual e máximo de elementos inseridos no array
- Métodos:
  - Construtor que recebe o tamanho do array como parâmetro e inicializa o objeto (cria array, etc.)
  - *public boolean adicionar(int n)* – adiciona elemento ao final do array, retornando true se bem sucedido e false caso contrário
  - *public int calculaMedia()* – retorna a média aritmética dos números armazenados no array

Fazer a classe Ex3ArrayControle que:

- Pergunta, via console, qual o tamanho N do array que o usuário quer instanciar
- Instancia um objeto da classe Ex1Array, passando N como argumento
- Pergunta N números ao usuário e armazena no objeto instanciado
- Chama o método para calcular a média aritmética e mostra o resultado

10) Adaptar o exercício 3 para utilizar um objeto da classe *ArrayList* ao invés de um array comum.

11) Fazer um programa para receber um número do usuário e decompô-lo em fatores primos. Os fatores primos devem ser armazenados em um *array* com o tamanho exato (dica: primeiro determinar quantos são os fatores primos, depois criar o *array* para armazená-los).

12) A distância entre várias cidades é dada pela tabela abaixo (em km):

	1	2	3	4	5
1	00	15	30	05	12
2	15	00	10	17	28
3	30	10	00	03	11
4	05	17	03	00	80
5	12	28	11	80	00

Implemente um programa que:

- leia a tabela acima em um *array* bidimensional. O programa não deve perguntar distâncias já informadas (por exemplo, se o usuário já forneceu a distância entre 1 e 3 não é necessário informar a distância entre 3 e 1, que é a mesma) e também não deve perguntar a distância de uma cidade para ela mesma, que é 0.
- leia um percurso fornecido pelo usuário em um *array* unidimensional. Calcule e mostre a distância percorrida. Por exemplo: dado o percurso 1, 2, 3, 2, 5, 1, 4, para a tabela mostrada como exemplo teremos: 15 + 10 + 10 + 28 + 12 + 5 = 80 km.

Exceções

13) Adaptar o exercício 2 para:

- lançar a exceção *MenorQueException* caso o número arriscado seja menor do que o sorteado
- lançar a exceção *MaiorQueException* caso o número arriscado seja maior do que o sorteado
- capturar essas exceções e tratá-las, mantendo a lógica original.

Fluxos

14) Fazer um programa em Java que:

- Receba o nome, o código e as duas notas bimestrais de 3 alunos para uma determinada matéria.
- Salve estes dados em um arquivo. Os dados devem ser salvos registro a registro, obedecendo o seguinte formato:
    i. número inteiro contendo o tamanho em char do nome do aluno.
    ii. sequência de chars correspondente à string que contém o nome do aluno.
    iii. duas notas na forma de números inteiros.

Fazer um programa em Java para:

- ler os dados contidos no arquivo gerado pelo programa anterior
- calcular e mostrar: quais alunos foram aprovados, quais foram para exame, quais foram reprovados e a média da turma.

Dica: utilizar *FileOutputStream* encapsulado por um *DataOutputStream.* Utilizar métodos de *String* para convertê-la para um array de bytes.

Api Gráfica Java

1) Fazer um programa para medir os reflexos do usuário:

    a. Programa deve instanciar um JFrame com uma área de desenho
    b. programa deve, depois de um tempo aleatório, exibir um alvo (por exemplo, um círculo) em uma posição aleatória da área de desenho
    c. usuário deve clicar com o ponteiro do mouse sobre o alvo.
    d. Caso o usuário acerte, o programa deve mostrar o tempo decorrido entre a exibição do alvo e o clique do usuário
    e. Caso o usuário erre, o programa mostra uma mensagem de erro

2) Implementar um conversor binário- decimal com as seguintes características:
    - Interface: duas caixas de texto para receber valores numéricos na base binária e na base decimal e botões para a conversão em ambos os sentidos
    - Funcionalidade: clicando-se um botão deve-se fazer o cálculo no sentido correspondente. Considerar que o usuário SEMPRE insere números válidos e que caixa vazia significa 0.

3) Implementar o jogo da memória
    - Jogo apresenta um frame com 5 botões, contendo números ou figuras
    - Ao ser iniciado, o jogo sorteia um dos botões e o indica (por exemplo, exibindo em outro lugar do frame o número ou figura sorteada). O jogador deve então clicar no botão correspondente.
    - O jogo prossegue, exibindo uma sequência de sorteios que vai aumentando progressivamente. O jogador, por sua vez, ao final da exibição deve repetir aquela sequência clicando nos botões.
    - O jogo termina quando o usuário erra a sequência.

4) Implementar o layout de uma calculadora
    - Caixa de texto com o número no alto do frame
    - Números de 0 a 9 e sinais de . e = em grid de 4x3 no centro do frame
    - Botões de + e – à direita do frame
    - Botões de * e / na parte de baixo do frame

5) Fazer uma aplicação de relógio com as seguintes características:
    - Interface: um frame com quatro textos estáticos (JLabel) dispostos em duas linhas e duas colunas. Na primeira linha, um label com o texto “Data:” e um label vazio e, na segunda linha, um label com o texto “Hora: “ e um label vazio.

- Funcionalidade: ao ser executado o programa deve obter data e hora atuais e exibir a data no segundo label da primeira linha e a hora no segundo label da segunda linha.
  Dica: utilizar objetos das classes *Calendar* e *DateFormat* para obter e formatar as informações de data e hora.

6) Fazer uma aplicação Java com as seguintes características:

- barra de menu com o menu "Desenhar" e os itens de menu "Círculo" e "Quadrado".
- Ao clicar-se em um item de menu, o aplicativo deve desenhar a figura correspondente no painel de conteúdo do frame.

Orientação a Objetos

Básico

1) Criar a classe *Pessoa* com as seguintes características:

- atributos: idade e dia, mês e ano de nascimento, nome da pessoa
- métodos:
  - *calculaIdade()*, que recebe a data atual em dias, mês e anos e calcula e armazena no atributo *idade* a idade atual da pessoa
  - *informaIdade()*, que retorna o valor da idade
  - *informaNome()*, que retorna o nome da pessoa
  - *ajustaDataDeNascimento()*, que recebe dia, mês e ano de nascimento como parâmetros e preenche nos atributos correspondentes do objeto.
- Criar dois objetos da classe *Pessoa*, um representando Albert Einstein (nascido em 14/3/1879) e o outro representando Isaac Newton (nascido em 4/1/1643)
- Fazer uma classe principal que instancie os objetos, inicialize e mostre quais seriam as idades de Einstein e Newton caso estivessem vivos.

2) Alterar o programa do exercício 1) para substituir o método *ajustaDataDeNascimento* por uma construtora

Relacionamentos entre classes

3) Fazer um programa com as seguintes características:

- Uma classe chamada Universidade que terá como atributo um nome e terá um método para informar o seu nome.
- Relacionar a classe Pessoa para com a classe Universidade. Cada pessoa poderá ser associada a uma Universidade.
- A classe Pessoa, por sua vez, terá um método que dirá seu nome e em que universidade trabalha.

- Criar dois objetos da classe Pessoa, um representando Albert Einstein (nascido em 14/3/1879) e o outro representando Isaac Newton (nascido em 4/1/1643)
- Criar dois objetos de Universidade, associando um para Einstein e outro para Newton.

    - Einstein trabalhou como professor de física em Princeton (Nova Jersey - Estados Unidos da América).
    - Newton trabalhou como professor de matemática em Cambridge (Inglaterra).

4) Fazer um programa para:

- Criar uma classe Departamento que permita relacionar um objeto (Departamento) à classe Universidade por composição (Universidade "contém" Departamento)
- Adaptar a classe Pessoa para que ela possua uma referência ao departamento que trabalha, ou seja, ela deve possuir uma associação com a classe Departamento, permitindo que cada objeto Pessoa tenha a referência de um objeto Departamento.

Adaptar o enunciado anterior para:

- alterar a relação entre Universidade e Departamento para uma agregação, ou seja, uma Universidade pode inicialmente não ter Departamento porém eles podem ser criados posteriormente

Adaptar o primeiro enunciado para:

- Fazer com que uma Universidade possa ter 50 Departamentos.
- Fazer com que um Departamento referencie a Universidade a qual está filiada.
- Criar mais Departamentos filiando-os às Universidades.

5) Fazer uma classe Aluno que possua as seguintes características:

- dois atributos do tipo inteiro: primeira nota parcial (de 0 a 100) e Segunda nota parcial (de 0 a 100)
- um atributo *String* representando o nome do aluno
- possua métodos para ler e escrever os atributos (ou uma construtora)

Fazer uma classe Controle que:

- pergunte ao usuário o nome e as duas notas parciais de um aluno. Caso o nome entrado seja "fim" isso significa que o usuário não quer inserir mais nenhum aluno, do contrário deve ser instanciado um objeto da classe Aluno e armazenados os dados digitados.
  Dicas: usar um objeto da classe *ArrayList* de Java para armazenar as referências para os objetos instanciados). Usar o método *equals* da classe *String* para verificar se o valor do nome entrado é igual a "fim".

Para encontrar a documentação destas classes: http://java.sun.com/j2se/1.5.0/docs/api/

- Calcular, ao final da inserção de todos os alunos, a média da turma, quantos alunos foram aprovados, quantos foram para a final e quantos foram reprovados e mostrar os códigos de todos os alunos cujas notas ficaram abaixo da média da turma.

6) Fazer um sistema de calculadora simples, composto das seguintes classes:

CalcControle: controle da calculadora ("processador"), com os seguintes métodos:

- *public void executar()* – faz a calculadora funcionar através dos seguintes passos:
  - Recebe primeiro operando do usuário através de *CalcInterface* e armazena no objeto de *CalcDados*
  - Recebe segundo operando do usuário através de *CalcInterface* e armazena no objeto de *CalcDados*
  - Recebe operação do usuário através de *CalcInterface* e armazena no objeto de *CalcDados*. Se a operação for igual a 's', finaliza o programa (*System.exit(0)*).
  - Executa a operação (método *opera*) e mostra o resultado através de *CalcInterface.*
  - Armazena resultado como primeiro operando para a próxima operação e volta para o segundo passo
- *private double opera(double opn1, double opn2, char op)* - executa a operação desejada e retorna o resultado.

CalcDados: implementa a parte da calculadora que armazena os dados (operandos e operação) para o seu funcionamento ("memória"). Possui as seguintes características:

Atributos: dois números do tipo *double* para armazenar os operandos e um dado do tipo *char* para armazenar a operação.

Métodos:

- *public void setOperando(int i, double valor)* – armazena o i-ésimo operando com o valor expresso em *valor*
- *public double getOperando(int i)* – retorna o valor do i-ésimo operando
- *public void setOperacao(char op)* – armazena o caracter *op* como sendo a operação atual
- *public char getOperacao()* – retorna o valor da operação atual

CalcInterface: implementa a parte da calculadora que coleta e exibe informações ao usuário (display e teclado da calculadora). Possui os métodos:

- *public double recebeOperando(int i)* – recebe o operando i da operação (i vale 1 ou 2) e retorna.
- *public char recebeOperacao()* – recebe um char representando uma operação válida (+, -, * ou /) e retorna.

- *public void mostraResultado(double res)* – mostra o resultado recebido como parâmetro.

Criar a classe *Principal*, cujo único objetivo é instanciar os objetos de controle, dados e interface e criar os vínculos (associações) entre eles. Todas as classes citadas devem possuir, além dos atributos citados, outros atributos que representem as referências para os outros objetos (criando as associações entre eles).

Herança e polimorfismo

7) Implementar a classe PolReg, que define um polígono regular

- Atributos: número de lados, tamanho do lado
- Métodos: cálculo do perímetro, cálculo do ângulo interno e cálculo de área. Este último deve retornar o valor zero, dado que não é possível calcular a área de um polígono regular genérico
- Construtora que inicializa os valores dos atributos

Fazer a classe *TrianguloEq*derivada de *PolReg*, que:

- redefine o método de cálculo de área para calcular e retornar a área do triângulo equilátero

Fazer a classe *Quadrado*, também derivada de *PolReg*, que:

- redefine o método de cálculo de área para calcular e retornar a área do quadrado

Fazer um programa que:

- pergunte ao usuário que tipo de objeto ele quer criar (polígono regular, triângulo equilátero ou quadrado)
- pergunte número de lados (se for polígono regular) e tamanho dos lados
- instancie o objeto e mostre o valor do perímetro, ângulo interno e área calculados

8) Escrever a classe *Complexo*, que pretende representar um modelo para a criação de números complexos. A classe *Complexo* deve possuir:

- 2 atributos float: real e imag
- 1 construtora que recebe a parte real e a parte imaginária do número e inicializa os atributos
- 1 método Modulo() que retorna o módulo do número complexo.
- 1 método Angulo() que retorna o ângulo do número complexo em graus.

Escrever a classe *Real*, derivada da classe *Complexo*, que pretende representar um modelo para a criação de números reais. A classe *Real* deve:

- possuir uma construtora que recebe o valor do número como parâmetro e chama a construtora da base.
- adicionar o método Sinal(), que retorna 1 se o número for positivo e -1 se for negativo.

Escrever um programa em Java para testar estas classes (instanciar objetos, chamar métodos, etc.)

9) Escrever a classe *Pessoa* com as seguintes características:

- atributos: nome e idade
- métodos: construtora para inicializar os parâmetros e *mostraDados()* que exibe os dados da pessoa no console na forma:

Nome da pessoa: xxx
Idade da pessoa: yyy

Escrever a classe *Aluno*, derivada de *Pessoa*, com as seguintes características:

- atributos: nome do curso que está cursando
- métodos: construtora para inicializar os atributos e redefinição do método *mostraDados()* para exibir as seguintes mensagens:

Nome do aluno: xxx
Idade do aluno: yyy
Curso do aluno: zzz

Elaborar um programa em Java que:

- declare uma referência para objeto da classe *Pessoa*
- pergunte ao usuário, via console, se ele deseja instanciar um aluno ou uma pessoa
- crie o objeto correspondente, referencie com a referência já criada e chame o método *mostraDados()* para exibir os dados da pessoa ou do aluno

10) Alterar o exercício sobre polígonos regulares, triângulos e quadrados para que a classe *PolReg* não apresente uma definição para o método de cálculo de área; ou seja, a classe *PolReg* deve ser transformada em uma classe abstrata

Estruturas de dados

1) Fazer uma classe Empregado , com atributos nome, cpf, e salario. Escrever uma classe principal que:
   - instancia um array de 5 empregados
   - pede os dados, instancia os objetos e armazena no array
   - classifica pelo salário utilizando bubble-sort.
   - pede ao usuário um cpf a ser removido, efetua a operação e mostra o array resultante.

2) Fazer uma classe ListNode que represente os nós de uma lista encadeada. Esta classe deve conter o campo nome e uma referência para o próximo nó da lista chamada prox.
   Fazer então a classe ListOperation que implementa os seguintes métodos:
   - add(ListNode n, ListNode ant) – adiciona o nó n após o nó ant (no início se ant for 0).
   - remove(ListNode n) – remove o nó n da lista.

- print() – percorre a lista e mostra o valor do campo nome de cada nó.

Fazer uma classe principal que exercita as operações acima implementadas. Tentar efetuar as mesmas operações utilizando LinkedList e comparar o desempenho.

3) Escrever um programa em Java que:
   - receba uma string representando uma expressão aritmética na forma infixa. Os operadores, operandos e símbolos de precedência devem ser separados por espaços
   - transforme a expressão infixa em uma expressão posfixa, utilizando uma pilha de operadores.
   - avalie a expressão posfixa utilizando uma pilha de operandos

Exercícios de integração

Matemática

1) Fazer um programa para:

- Instanciar um Jframe com uma área de desenho, uma combo box, cinco caixas de texto e um botão "desenhar".
- Implementar a classe *Funcoes*, a qual contém diversos métodos que implementam funções polinomiais (*f1, f2* e *f3).* Exemplos para estas funções:

  double f1(double x) - retorna o valor de x^2
  double f2(double x) - retorna o valor de x^3+ 3x^2

  A classe *Funcoes* deve também implementar a função *polinomio*, que recebe os coeficientes dos diversos graus do polinômio (c1, c2, etc.) e retorna o resultado da função:

  double polinomio(float c0, float c1, float c2, float c3, float c4, float x) – retorna o valor de c0 + c1*x + c2*x^2 + c3*x^3 + c4*x^4

  - O combo box do frame deve conter as opções para se selecionar *f1*, *f2, f3* e *polinomio*. Caso se selecione *polinomio,* as caixas de texto devem ser habilitadas para receber os valores de *c0* a *c4.*
  - Ao clicar-se no botão "desenhar", o programa simula valores de *x* de -100 a 100 e plota o gráfico correspondente a partir do retorno da função selecionada.
  - (Opcional) incluir caixas de texto extra para inserir o valor inicial, o valor final de *x* e o incremento entre valores. Calcular a escala do gráfico automaticamente.

2) Fazer um programa para efetuar operações comuns de vetor. O programa possui as seguintes características:

- item de menu “adicionar vetor”, no qual o usuário insere um nome para o vetor e dimensões x e y (vetores bidimensionais). Os vetores adicionados devem ser exibidos numa área de desenho.
- item de menu “operações”, no qual devem constar “produto vetorial”, “produto escalar” e “projeção” como sub-itens. Ao ativar uma operação o usuário deve fornecer os nomes dos vetores envolvidos. Para as operações de produto vetorial e projeção o programa deve desenhar o vetor resultado, para o produto escalar ele deve mostrar o resultado como uma string na área de desenho.

3) Fazer um programa para resolver sistemas lineares de equações. O programa possui as seguintes características:
   - o usuário deve editar um arquivo contendo os coeficientes das equações, separados por espaço, e contendo uma equação por linha. Por exemplo, para o sistema:

     $2x + 3y + z = 5$
     $x - 2y + 8z = 9$
     $4x + y + 3z = -2$

     O arquivo deveria possuir o seguinte conteúdo:

     2 3 1 5
     1 -2 8 9
     4 1 3 -2
   - o programa deve abrir o arquivo, obter os coeficientes e utilizar as técnicas de álgebra (cálculo de determinante, etc.) para solucionar os valores das incógnitas. Deve ser possível solucionar sistemas de até 10 incógnitas.

Cálculo

4) Fazer um programa em Java para calcular, simulando valores de *x*, o limite de uma determinada função *f(x)*. Os valores de *x* devem se aproximar progressivamente do valor de tendência, à direita e à esquerda, de tal forma que se possa determinar o valor do limite através da análise da variação dos valores.

Física

5) Fazer um programa para mostrar a variação da posição de um corpo qualquer em queda livre na atmosfera, considerando a resistência do ar. Esta variação deve ser calculada de 2 formas:

   - a partir do resultado da integração da fórmula que determina a velocidade em função do tempo

$$y = v_t \bullet (t - \frac{v_t}{g} \bullet (1 - e^{\frac{-g}{v_t} \bullet t}))$$

$$v_t = (\frac{m \bullet g}{b})^{\frac{1}{n}}$$

$$a = g - (\frac{b}{m}) \bullet v^n$$

- a partir do método de Euler, ou seja, do cálculo da aceleração, depois velocidade e deslocamento em vários instantes de tempo consecutivos, utilizando um valor delta de incremento de tempo que pode ser variado para se aproximar de zero.

$$a_0 = g - (\frac{b}{m}) \bullet v_0^{\,n}$$

$$v_1 = v_0 + a_0 \bullet t$$

$$y_1 = y_0 + v_0 \bullet t + \frac{a_0 \bullet t^2}{2}$$

Sugestões: m = 70 kg, b = 0,251 kg/m, n = 1 e tfinal = 10 s.

6) Fazer um programa para simular a atração gravitacional entre dois corpos:
   - programa instancia um frame, o qual possui uma área de desenho na qual é desenhada uma porção bidimensional do espaço com dimensões em escala planetária (por exemplo, 100e6 km de largura por 100e6 km de altura)
   - dois corpos são posicionados aleatoriamente (ou por entrada do usuário) no espaço bidimensional. Os corpos possuem massas em escala planetária (por exemplo, 6e27 kg), também definida pelo usuário
   - os corpos possuem velocidades nos eixos x e y, também em escala planetária (por exemplo, 30 km/s) e também definida pelo usuário
   - ao iniciar-se o funcionamento do programa, os corpos devem ter sua interação gravitacional calculada (terceira lei de Newton), o que permite calcular a aceleração, variação da velocidade e variação da posição para um intervalo de tempo (também definível pelo usuário). Os corpos devem então ser redesenhados e o processo deve ser repetido, simulando a trajetória dos corpos sujeita à atração gravitacional.

7) Um passageiro corre à sua velocidade máxima de *8 m/s* para pegar um trem. Quando está à distância *d* da porta de entrada, o trem principia a rodar com aceleração constante *a = 1 m/(s*s)*, afastando-se.
    a. Se *d = 30 m* e se o passageiro continua a correr, conseguirá ou não pegar o trem? Responda a essa pergunta elaborando um programa em Java que simula os deslocamentos do passageiro e do trem, com intervalos de tempo variando entre *1 s e 0,1 s*. Avalie os resultados obtidos frente ao resultado analítico (obtido através da resolução da equação).
    b. A distância crítica de separação inicial é chamada de *dc*. Adaptar o programa anterior para variar a distância inicial *d* e obter a distância crítica por comparação. Com a distância crítica de separação *dc*, qual a velocidade do trem quando o passageiro consegue pegá-lo? Qual é a velocidade média do trem no intervalo de tempo *t = 0* até este instante? Qual é o valor de *dc*?
    c. Fazer um programa em Java para desenhar a função posição *x(t)* do trem, com *x = 0* em *t = 0*. No mesmo gráfico, desenhar a função *x(t)* do passageiro, com diversas distâncias de separação inicial *d*, incluindo a distância *d = 30 m* e a distância crítica de separação *dc* que lhe permite pegar o trem por um átimo.

8) Um corpo se move sobre uma reta e duplica sua velocidade, a cada segundo, durante os primeiros 10s. Seja *2 m/s* a velocidade inicial. Qual é a velocidade média nos primeiros *10s*? Responda a essa pergunta elaborando um programa em Java que simula a variação da velocidade e o deslocamento do corpo, com intervalos de tempo variando entre *1 s e 0,1 s*. Avalie os resultados obtidos frente ao resultado analítico (obtido através da integração).

area = tamlado*tamlado*numlados/(4*tan(pi/numlados))