Tecnologia em Análise e Desenvolvimento de Sistemas
Lógica de Programação
Professor Vinícius Fritzen Machado

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

Avaliação de Aprendizagem I

Desenvolva os exercícios abaixo utilizando somente o que foi visto em sala de aula. Novas soluções são encorajadas, no entanto, é necessário que os alunos demonstrem domínio sobre as técnicas apresentadas.

Faça o download da pasta de códigos do Moodle, essa pasta contém a estrutura básica e nome de cada um dos exercícios, **renomeie a pasta com seu nome** e não esqueça de **remover os arquivos .class** antes de enviar a pasta **compactada**.

Os códigos fontes serão avaliados quanto a funcionalidade, legibilidade, estrutura e organização.

Códigos muito similares serão considerados cola e não terá nota atribuída. Façam os exercícios sozinh@s!

O professor resolveu todos exercícios antes, nada além do que foi demonstrado nas aulas é necessário! Aos mais experientes: Sem funções, arrays, switchs =D

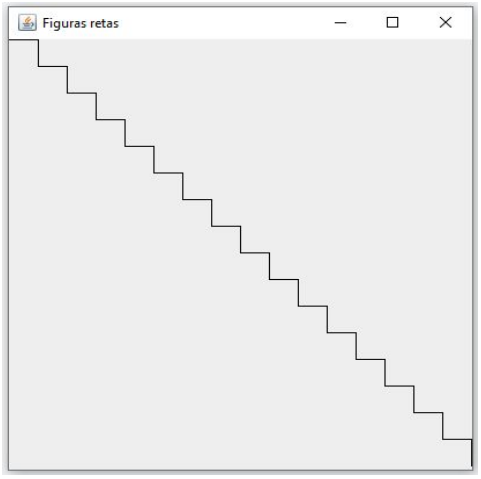
Boa avaliação!!!

1. (Peso 1,0) Comparacoes.java - Faça um programa que leia 100 valores e informe quantos deles são maiores que 0, quantos são iguais a zero e quantos são negativos.

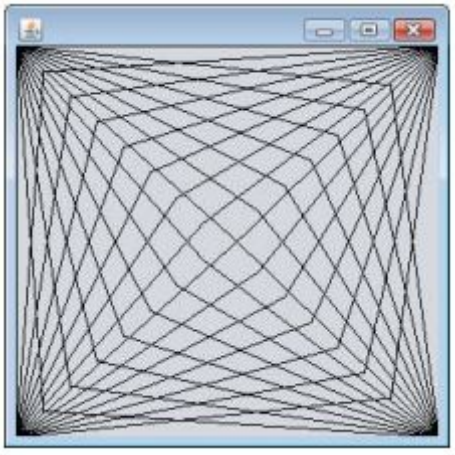
2. (Peso 1,0) Triangulos.java - Escreva um programa que leia o valor de 3 ângulos de um triângulo e escreva se o triângulo é Retângulo, Acutângulo ou Obtusângulo. Considerando:
   - Triângulo Retângulo: possui um ângulo reto. (igual a 90º)
   - Triângulo Obtusângulo: possui um ângulo obtuso. (maior que 90º)
   - Triângulo Acutângulo: possui três ângulos agudos. (menores que 90º)

Atenção: Para os exercícios 3, 4 e 5. A instrução **g.drawLine(x0, y0, x1, y1)** desenha uma linha reta da posição (x0, y0) canto superior esquerdo, até a posição (x1, y1) canto inferior direito. Com essa instrução e as estruturas de repetição, é possível gerar as imagens solicitadas.

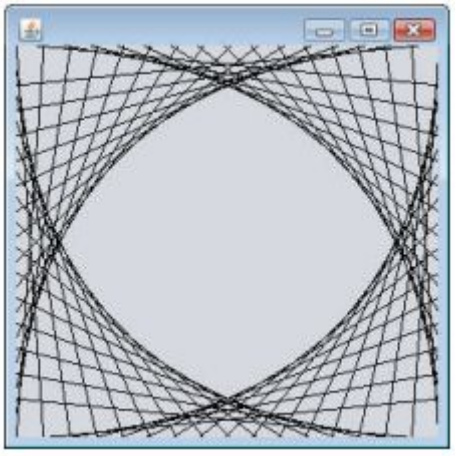
3. (Peso 1,0) Figuras01.java - Modifique o código fonte, na região indicada, utilizando a instrução **g.drawLine(x0, y0, x1, y1)** para gerar a seguinte imagem:



4. (Peso 1,5) Figuras02.java - Modifique o código fonte, na região indicada, utilizando a instrução **g.drawLine(x0, y0, x1, y1)** para gerar a seguinte imagem:



5. (Peso 1,5) Figuras03.java - Modifique o código fonte, na região indicada, utilizando a instrução **g.drawLine(x0, y0, x1, y1)** para gerar a seguinte imagem:



6. (Peso 2,0) AproximacaoDePi - Existem diferentes formas de aproximarmos o valor de PI, https://en.wikipedia.org/wiki/Approximations_of_%CF%80. Utilizando Trigonometria, o matemático Gregory–Leibniz criou a seguinte fórmula:

$$\pi = 4\sum_{n=0}^{\infty}\frac{(-1)^n}{2n+1}$$

Quando aberto o somatório, percebemos que nada mais é do que a soma de várias frações, onde os denominadores são a sequência de números ímpares partindo de 1, e o sinal é intercalado entre positivo e negativo. Após isso a soma é multiplicada por 4 e obtém-se valores bem próximos de PI.

$$4\left(\frac{1}{1}-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+-\cdots\right)$$

Desenvolva um algoritmo que dada a quantidade de interações, calcule a aproximação de PI baseado na fórmula de Gregory Leibniz.

Exemplo de entrada e saída do Algoritmo, note que quanto mais interações são feitas, mais próximo chegamos do valor de PI.

```
C:\dev\tads\logica-prova-i>javac AproximacaoDePi.java

C:\dev\tads\logica-prova-i>java AproximacaoDePi
Quantas interacoes serao feitas?
10
3.0418396189294032

C:\dev\tads\logica-prova-i>java AproximacaoDePi
Quantas interacoes serao feitas?
1000
3.140592653839794

C:\dev\tads\logica-prova-i>java AproximacaoDePi
Quantas interacoes serao feitas?
100000
3.1415826535897198

C:\dev\tads\logica-prova-i>java AproximacaoDePi
Quantas interacoes serao feitas?
10000000
3.1415925535897915

C:\dev\tads\logica-prova-i>java AproximacaoDePi
Quantas interacoes serao feitas?
1000000000
3.1415926525880504

C:\dev\tads\logica-prova-i>
```

7. (Peso 2,0) Fatores Primos - Desenvolva o código necessário para fazer a decomposição em fatores primos. Lembrando que um número é considerado primo quando possui apenas dois divisores, 1 e ele mesmo, por exemplo, o número 17 é primo pois nenhum número entre 2 e 16 tem resto 0 quando tentamos dividir o número 17. O número decomposto em fatores primos nada mais é do que reescrever o número como uma sequência de multiplicações onde todos os valore são primos, por exemplo:

   225 = 3 * 3 * 5 * 5

   1001 = 7 * 11 * 13

Neste exercício, você irá solicitar ao usuário um número positivo e maior que 1 e, apresentará sua decomposição em fatores primos, mostrando um valor primo por linha, valores repetidos aparecerão um ao lado do outro, por exemplo:

```
C:\dev\tads\logica-prova-i>javac FatoresPrimos.java

C:\dev\tads\logica-prova-i>java FatoresPrimos
Informe o numero a ser decomposto em fatores primos:
225
3 3
5 5

C:\dev\tads\logica-prova-i>java FatoresPrimos
Informe o numero a ser decomposto em fatores primos:
864
2 2 2 2 2
3 3 3

C:\dev\tads\logica-prova-i>java FatoresPrimos
Informe o numero a ser decomposto em fatores primos:
1001
7
11
13
```