LISTA DE EXERCÍCIOS – 01

Threads em Java

Exercício 01: Faça um programa que execute dois threads. A primeira imprime todos os números ímpares de 1 a 2000; a segunda todos os números pares de 1 a 2000. Utilizando os dois threads criados, o programa deverá imprimir todos os números de 1 a 2000 em ordem crescente.

Exercício 02: Faça um programa que imprima a quantidade de números primos existes entre dois números também fornecidos pelo usuário (por exemplo: imprimir a quantidade de números primos existentes entre os números 100 e 1000). O usuário também deverá informar a quantidade de threads que deverão ser utilizados para contar esses números. [Dica: divida o intervalo fornecido pelo usuário entre cada thread, de forma que cada thread irá contar quantos números primos existem em cada subintervalo.]

Exercício 03: Faça um programa que imprima os números primos entre 0 e 99999 utilizando um número determinado de threads (o número de threads será informado pelo usuário). Os números deverão ser impressos em ordem crescente.

Exercício 04: Faça um programa multithreading que imprima a soma dos números inteiros em um intervalo fornecido pelo usuário (soma de todos os números inteiros no intervalo de INI até FIM). O usuário escolhe a quantidade de threads a serem usados.

$$soma = \sum_{i=INI}^{FIM} i$$

O usuário deverá indicar a quantidade de threads a serem usados no programa.

Exercício 05: Faça um programa multithreaded que imprima (o usuário escolhe a quantidade de threads a serem usados), em ordem crescente, todos os números perfeitos de 1 a 1.000.000.
Obs.: um número inteiro N é perfeito se a soma dos seus divisores for igual a 2N. Por exemplo: 6, 28, 496, 8128

Exercício 06: Faça um programa multithreaded que imprima todos os números amigáveis entre 1 e 1.000.000. Número amigáveis são pares de números onde um deles é a soma dos divisores do outro (sem considerar o próprio número como divisor). Por exemplo, 220 e 284 são números amigáveis, pois os divisores de 220 são 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 e 110, cuja soma é 284; já os divisores de 284 são 1, 2, 4, 71 e 142 e a soma deles é 220. Os números 17.296 e 18.416 também são amigáveis.

Exercício 07: Pequeno teorema de Fermat: Seja **p** um número primo. Então, para qualquer inteiro **n**, tem-se que **($n^p - n$)** é múltiplo de **p**. Por exemplo: ($2^5 - 2$), ($3^5 - 3$), ($4^5 - 4$), ... são múltiplos de 5. Fazer um programa que mostre que esse teorema é verdadeiro para os 10 primeiros números primos, usando os números inteiros (**n**) 2, 3, 4 e 5. Usar um thread para cada um dos números.

Exercício 08: Fazer um programa multithreaded (número de threads pode ser escolhido pelo usuário) para calcular o valor de π usando a seguinte série infinita:

$$\pi = 4 * \left[1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \cdots\right]$$

Ou seja:

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

Exercício 09: Faça um programa que crie um conjunto com 100.000 números inteiros aleatórios. Após criar o conjunto, o programa deverá permitir que o usuário digite números, onde será informado se o número pertence ou não ao conjunto criado. Deverão ser usados threads para fazer a busca desses números no conjunto.

Exercício 10: Crie uma solução de ordenação de vetor onde cada thread é responsável pela ordenação de uma fração deste vetor. O número de threads e o tamanho do vetor devem ser informados pelo usuário, onde o vetor será iniciado com números aleatórios. A função de ordenação deve ser genérica o suficiente para ser associada a cada thread. Tal função de ordenação deve receber por parâmetro uma identificação, bem como o intervalo de operação sobre o vetor. Ao final da ordenação de todas as frações, o processo pai deverá refazer a ordenação, agora sobre todo o vetor de ordenação. O programa deverá imprimir os valores contidos no vetor:

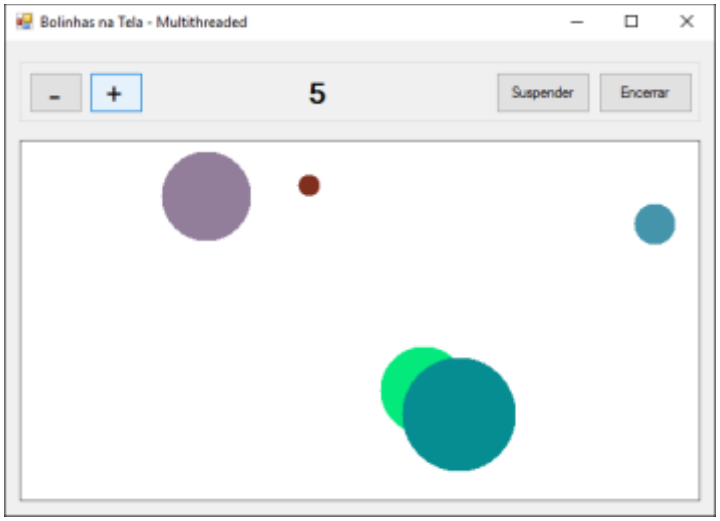
I. Após a inicialização;
II. Após a execução de todos os threads;
III. Após a execução da ordenação pelo processo pai.

Exercício 11: Fazer um programa com os seguintes threads:

1. Gerador de números aleatórios (N números aleatórios);
2. Contador de números pares;
3. Contador de números ímpares

À medida que os números aleatórios são gerados, os dois outros threads deverão contar, respectivamente, os números pares e ímpares gerados. Os threads contadores deverão ser executados simultaneamente com o thread de geração de números aleatórios, de forma independente, ou seja, novos números aleatórios deverão ser gerados mesmo se os anteriores ainda não tiverem sido contados. Ao final, deverão ser impressos a quantidade de números pares e ímpares gerados.

Exercício 12: Faça um programa gráfico multithreaded que cria bolinhas coloridas que deverão ser movimentadas aleatoriamente na tela, onde cada bolinha será criada por um thread, com o seguinte layout:

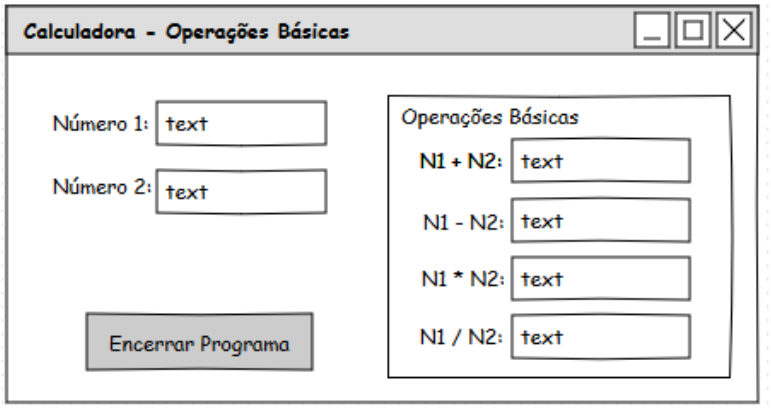


Botões:

- "-": elimina bolinha criada anteriormente
- "+": cria uma nova bolinha
- "Suspender": pausa todos os threads

Exercício 13: Alterar o programa anterior para que seja detectado colisão entre duas bolinhas, de forma que, ao colidir, as bolinhas sejam destruídas.

Exercício 14: Faça um programa multithreaded, usando interface gráfica, conforme o protótipo da tela especificado abaixo:



Funcionamento do Programa:

- Ao iniciar, o programa irá criar 4 threads. Cada thread deverá ser responsável por mostrar o resultado de cada uma das 4 operações realizadas pela calculadora.
- Os threads serão criados chamando a função **MostraResultado (int operação)**, onde operação indica qual das 4 operações o thread irá fazer, a saber:
    - 0: adição
    - 1: subtração
    - 2: multiplicação
    - 3: divisão
- O usuário deverá fornecer nos campos textos de "Número 1" e "Número 2" os dois números a serem utilizados pelos threads. Cada thread deverá fazer o respectivo cálculo da sua operação básica sempre que um dos números for alterado.
- Os threads deverão operar de forma sincronizada com as alterações dos números, ou seja, elas somente poderão fazer os cálculos quando um dos números for alterado, ou seja, elas não poderão ocupar tempo de CPU enquanto os números permanecerem inalterados.
- Ao iniciar o programa, os valores dos campos textos de "Número 1" e "Número 2" deverão ser ZERO.
- Ao clicar no botão "Encerrar Programa", os 4 threads deverão ser encerrados antes do programa ser finalizado.

Exercício 15: Fazer um programa com interface gráfica que movimente horizontalmente um canhão localizado na parte inferior da tela utilizando as setas para a esquerda e direita. Ao pressionar a barra de espaço, o canhão deverá atirar um projétil para cima, na vertical. Na parte superior da tela, deverá existir um alvo móvel movendo-se na horizontal, da esquerda para a direita com velocidade aleatória. Quando o alvo sair da tela, um novo alvo deverá ser gerado. O objetivo do programa é que o usuário acerte o alvo. O programa deverá emitir sons ao atirar e acertar o alvo.