package principal;

/\*\*

 \*

 \* Author .....

\*

 \*/

/\*enunciado\*/

// Existem 240 threads, sendo 120 Escritor e 120 Leitores, e um objeto Buffer de tamanho 1
 com dois métodos:

// class Buffer {

// synchronized Escrever(int i) {...}

// synchronized int Ler() {...}

// }

// (a) As threads Escritor devem ser escalonadas usando ScheduledExecutorService

// (caso 2 em Formas de Escalonamento Java), ou seja, todas essas 120 threads devem ser

// controladas por esse escalonador. Esse escalonador deve liberar uma thread Escritor a c
 cada 1 milissegundo

// (utilize scheduleAtFixedRate) que irá escrever um valor inteiro incremental no Buffer.

// (b) As 120 threads Leitor devem ser controladas por um ExecutorService (caso 1, em

// Formas de Escalonamento Java) usando um pool de tamanho fixo que deve permitir

// até 4 threads Leitor em estado de pronto durante a execução do código. As threads Leitor
 em estado de pronto devem tentar ler o valor escrito no objeto Buffer. Verifique a
 quantidade de threads Leitor que leram o mesmo valor inteiro.

// (c) Na terceira parte do trabalho, a primeira thread Leitor que conseguir

// ler o valor escrito deve configurar a variável do Buffer para zero. Para que as outras threads
 Leitor não leiam zero, você deve utilizar monitor juntamente com wait/notify.

// Isto é, quando uma thread Leitor verificar que o buffer está vazio, este invoca o método
 wait(). E quando uma thread Escritor tiver escrito um dado, este invoca o método notify().
 Verifique o comportamento do programa em relação ao anterior.

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.concurrent.ScheduledExecutorService;

import java.util.concurrent.TimeUnit;

public class Main

{

public static void main( String[] args )

{

//usar o synchronized

 ExecutorService executar = Executors.newFixedThreadPool(4);

 ScheduledExecutorService executar2 = Executors.newScheduledThreadPool(1);

 Buffer sharedLocation = new ImplementacaoBuffer();

 try

 {
 executar.execute(new Leitor( sharedLocation));

 executar2.scheduleAtFixedRate(new Escritor(sharedLocation),0,1,TimeUnit.MILLISECONDS)

 }

 catch ( Exception exception )

 {

 exception.printStackTrace();

 }

 executar.shutdown();

 executar2.shutdown();

 }

}