importjava.util.\*;

/\*\*

\* Исходная задача: Существует два потока(thread) - producer и consumer.

\* Producer - должен дописывать в конец randoms случайное число случайных чисел,

\* Consumer - должен считывать из randomsвсе числа и удалять их оттуда, выводя при этом их на печать

\* Требуется обеспечить потокобезопасность данных с помощью синхронизации.

\* Потоки должны работать строго последовательно, и не должно быть ситуаций, когда один поток отрабатывал бы несколько раз подряд, а другой простаивал

\* В примере представлен кусок "Грязного" кода, в котором допущен ряд ошибок

\* <p>

\* Требуется выполнить рефакторинг кода (переименовать плохо названные объекты/методы/классы,

\* обработать критические секции кода, выполнить оптимизацию и прочее), а также доработать программу таким образом, чтобы исходная задача была выполнена.

\*

\* Желательные условия: дополнительным плюсом будет считаться реализация всех инфраструктурных подсистем, таких как логирование и чтение параметров из файлов конфигурации

\* Решение задачи необходимо отправить ссылкой на опубликованный проект на github.com

\*/

public class ThreadSafeWithSynchronization

{

public List<Integer> Randoms = new ArrayList<>();

public void STARTthreadProcess()

{

PRODUCER.start();

CONSUMER.start();

}

private final Thread PRODUCER = new Thread(() ->

{

Integer i;

while(true)

{

i = 0;

for(; i< new Random().nextInt(100); )

{

Integer n = new Random().nextInt(100);

Randoms.add(n);

synchronized(this)

{

System.out.println(n);

}

i++;

}

System.out.println("wrote");

try

{

Thread.sleep(1000);

}

catch(Exception e)

{

}

});

// Поток для чтения данных. Читает данные из списка Randoms и выводит их в консоль

private final Thread CONSUMER = new Thread(() -> {

Integer i;

while(true)

{

i = 0;

for(;i<Randoms.size(); )

{

System.out.println(Randoms.get(i));

i++;

}

Randoms = null;

System.out.println("read");

try

{

Thread.sleep(1000);

}

catch(Exception e)

{

}

}

});

}