1. Create and start a thread that prints its name for around 5 seconds. One time per second. Run thread in different ways:
    a. By creating separate class that implements Runnable interface
    b. By extending Thread class
    c. By passing lambda function to the constructor of Thread class
    d. By passing method reference to the constructor of Thread class

2. Create nested class Spam that receives an array of time intervals in milliseconds and another connected array of messages. So in case you have five time intervals - you should have five messages. A spam object should print a message and sleep for the corresponding number of milliseconds, connected from another array.

   Spam object should stop doing any work in case the ‘Enter’ key has been pressed. You can imitate ‘Enter’ key pressing by passing ‘\n’ character to the input stream of your program.

   During the demo of this task, imitate enter key pressing after five seconds.

3. Create a class that implements Runnable interface. Declare two separate counters of primitive int type inside this class.

   Create multiple similar threads that will do the same thing: compare the state of the two counters and print the result of comparison to console. After that increments the first counter, then sleep for ten milliseconds and increments the second counter.

   Run this program, interrupt it after 3 seconds of execution and compare the result of execution and console output in case of synchronized and non-synchronized critical sections where increments operations are executed.

4. Create a method that can generate a multidimensional array M x N.

   For the sake of the demo, generate an array with 4 rows and 100 columns.

   Find the maximum integer with the help of multiple threads. Comparison of each int should go with 1 millisecond delay to imitate comparison of big objects. ***This requirement is important in order you could see benefits of multithreading execution. Don’t miss it because it will give you incorrect results.***

   Use the same block of code to find the maximum integer in one thread. Do not forget about 1 millisecond delay. Measure the execution time for multithreading solution and single-thread solution. You should see the advantage of a multithreading solution. If you can’t see that multithreading solution solves this task faster, most likely you wrote your program incorrectly.

   Solve this task in two different ways - with Future objects and with CountDownLatch

synchronizer.

5. Task about readers and writers.

   Declare two different types - Reader & Writer.

   In the demo class, declare a buffer. This may be either String, or StringBuilder or StringBuffer. So it can be any object that may store strings.

   Writer writes messages to the buffer. When the buffer is full, the writer waits until readers read all messages.

   Reader reads messages from the buffer. When there are no messages in the buffer - it waits to be notified by the writer that there are messages added.

   Buffer size is 5 random characters. Readers read the same message from the buffer. Program prints to console what message has been written to the buffer and what messages were read from the buffer.

   Example of console output:

Writer writes:IRUPU
Reader Thread-3:IRUPU
Reader Thread-1:IRUPU
Reader Thread-2:IRUPU
Writer writes:JVJNM
Reader Thread-2:JVJNM
Reader Thread-1:JVJNM
Reader Thread-3:JVJNM
Writer writes:LKFPJ
Reader Thread-2:LKFPJ
Reader Thread-1:LKFPJ
Reader Thread-3:LKFPJ

   Each reader - separate thread.

   Demo program with three readers and one writer.

   Demo program execution for no more than 2 seconds. During this time limit read-write cycles to 3 cycles like in example above.

   Solve this program in different ways:
   - With synchronized blocks and dummy object that you will use to synchronize access of different threads
   - With Reentrant lock