Lab Assignment 05:

Deadline: May 3, 11:59 PM.

Instructions:

The task is similar to the producer–consumer problem discussed in the video. The *SharedMemory* class is used by two threads. The *WriterThread* writes some string to the shared memory at a random interval. The *ReaderThread* reads the memory whenever there are new values and prints those strings.

If the memory is empty, the *ReaderThread* goes to a suspended state. Whenever a new item is inserted, the *WriterThread* gives the suspended *ReaderThread* a wake-up call. You only need to modify the *SharedMemory* class.

Best of luck.

You need to modify the following java code:

```
/**
 *
 * The task is similar to the producer–consumer problem discussed
in the video.
 * The SharedMemory class is used by two threads. The WriterThread
writes some string to the shared
 * memory at a random interval. The ReaderThread reads the memory
whenever there are new values and
 * prints those strings.
 *
 * If the memory is empty, the ReaderThread goes to a suspended
state. Whenever a new item is inserted,
 * the WriterThread gives the suspended ReaderThread a wake-up
call.
 *
 * You only need to modify the SharedMemory class.
 *
 * Best of luck.
 *
 * @author Sifat Ut Taki
 * BRAC University
 */

import java.util.ArrayList;
import java.util.List;

class SharedMemory {

  public int pointer = -1;

  private List<String> registers = new ArrayList<>();

  public String readFromReg() throws InterruptedException {
```

```
    /**
     * TODO:
     * 1. Use monitor for this method [public synchronized String
readFromReg()].
     * 2. If the pointer is at -1, it means there is no item in the
list. So, suspend the readerThread [using wait()].
     * 3. Take the value from the the list.
     * 4. Decrease the pointer by 1.
     * 5. Remove the string from the list,
     * 6. Return the value
     */
    return "";
  }

  public void writeToReg(String value) {
    /**
     * TODO:
     * 1. Use monitor for this method [public synchronized String
writeToReg()].
     * 2. Add the string to the list.
     * 3. Increase the pointer by 1.
     * 4. If the pointer is at 0, it indicates first item in the
list. So, wakeup readerThread [using notifyAll()].
     */
  }
}

// Do not modify this class
class WriterThread extends Thread {

  private String[] values = {
          "Lorem ipsum dolor sit amet, consectetur adipiscing
elit.",
          "Nunc et velit nec eros molestie sagittis.",
          "Aliquam ut ligula ut tortor iaculis dapibus eget et
arcu.",
          "Proin tempor purus ut purus vehicula, eu faucibus ipsum
fringilla.",
          "Praesent id justo ac diam aliquet iaculis.",
          "Suspendisse dignissim turpis malesuada, ultricies turpis
in, molestie lorem.",
          "Nulla auctor elit eget felis congue, sit amet molestie
mi mattis.",
          "Nam nec est nec felis ullamcorper accumsan.",
```

```
        "Maecenas posuere magna a eros semper elementum.",
};

SharedMemory sharedMemory;

public WriterThread(SharedMemory sharedMemory) {
  this.sharedMemory = sharedMemory;
}

@Override
public void run() {
  for (int i = 0 ; i < values.length ; i++) {
    try {
      sharedMemory.writeToReg(values[i]);
      sleep((int)(Math.random() * 1000));
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
  }
}

// Do not modify this class
class ReaderThread extends Thread {

  SharedMemory sharedMemory;

  public ReaderThread(SharedMemory sharedMemory) {
    this.sharedMemory = sharedMemory;
  }

  @Override
  public void run() {
    try {
      while (true) {
        System.out.println(Thread.currentThread().getName() + "
prints: " + sharedMemory.readFromReg());
      }
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
  }
}
```

```
// Do not modify this class
public class LabTask{
  public static void main(String[] args) throws
InterruptedException {

    SharedMemory sharedMemory = new SharedMemory();

    ReaderThread readerThread = new ReaderThread(sharedMemory);
    WriterThread writerThread = new WriterThread(sharedMemory);

    writerThread.start();
    readerThread.start();

    writerThread.join();
      readerThread.stop();
    }
  }
```