An-Najah National University
Faculty of Engineering and IT

جامعة النجاح الوطنية
كلية الهندسة وتكنولوجيا المعلومات

Electrical and Computer Engineering Department
Operating System: (10636451)

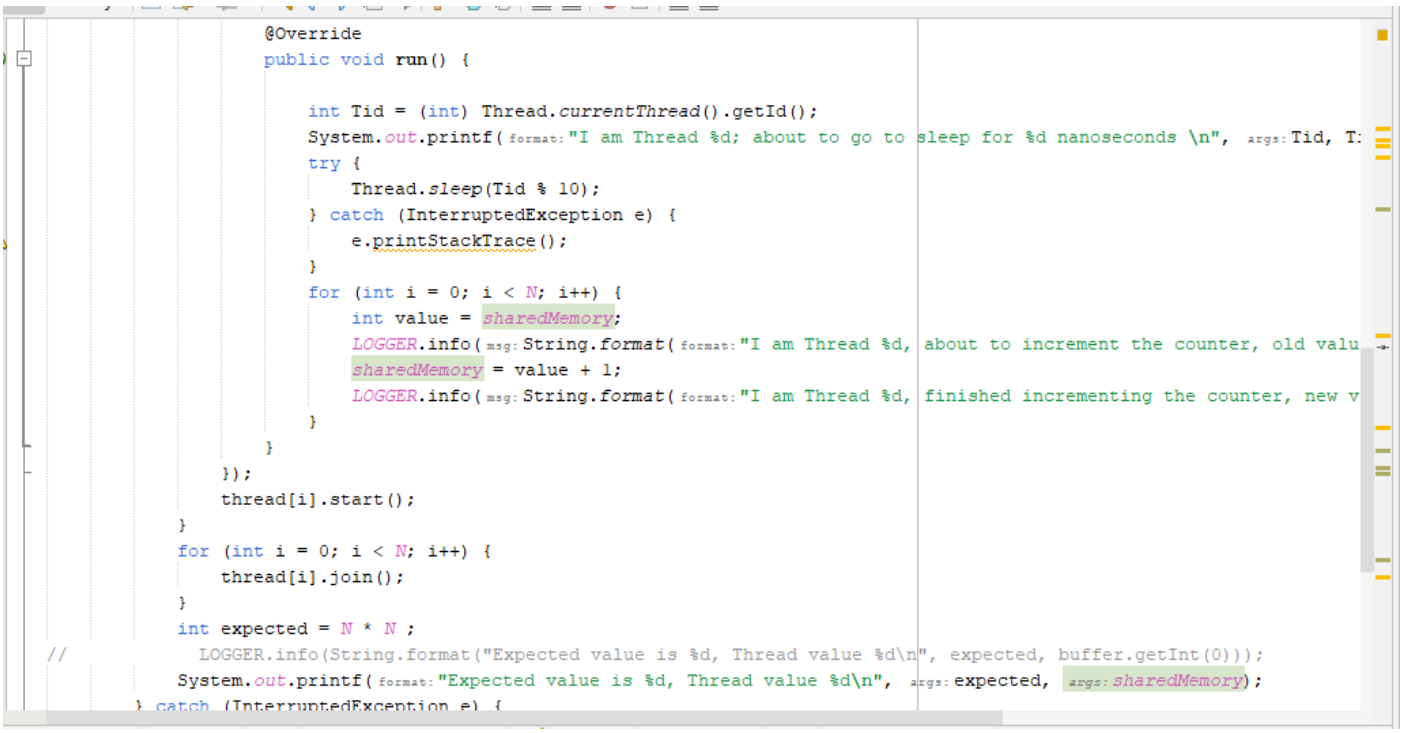
Homework (2) Results

Instructor Name: سليمان ابو خرمة	**Homework #: (2)**
Academic Year: 2023/2024	**Semester: 2nd**
Time: (8 – 9:30) MON./WED.	**Section: 3**
Student Name: احمد شاعر	**Student ID: 11820152**

```
public class NotSynchronized {
    static int sharedMemory = 0;
    static int N = 652;
    public static void main(String[] args) {
        try {
            Thread[] thread = new Thread[N];
            for (int i = 0; i < N; i++) {
                thread[i] = new Thread(new Runnable() {
                    static final Logger LOGGER = Logger.getLogger( name: NotSynchronized.class.getName());
                    static {
                        try {
                            Handler handler = new FileHandler( pattern: "logsUnsynchronized.log"); // log file name
                            LOGGER.addHandler(handler);
                            LOGGER.setUseParentHandlers( useParentHandlers: false);
                            LOGGER.setLevel( newLevel: Level.ALL);
                            handler.setLevel( newLevel: Level.ALL);
                        } catch (IOException e) {
                            e.printStackTrace();
                        }
                    }
                    @Override
                    public void run() {

                        int Tid = (int) Thread.currentThread().getId();
                        System.out.printf( format: "I am Thread %d; about to go to sleep for %d nanoseconds \n", args: Tid, T:
                        try {
```

NotSynchronized > main > try > for (int i = 0; i < N; i++) > Runnable > <static init> > try > catch IOException e

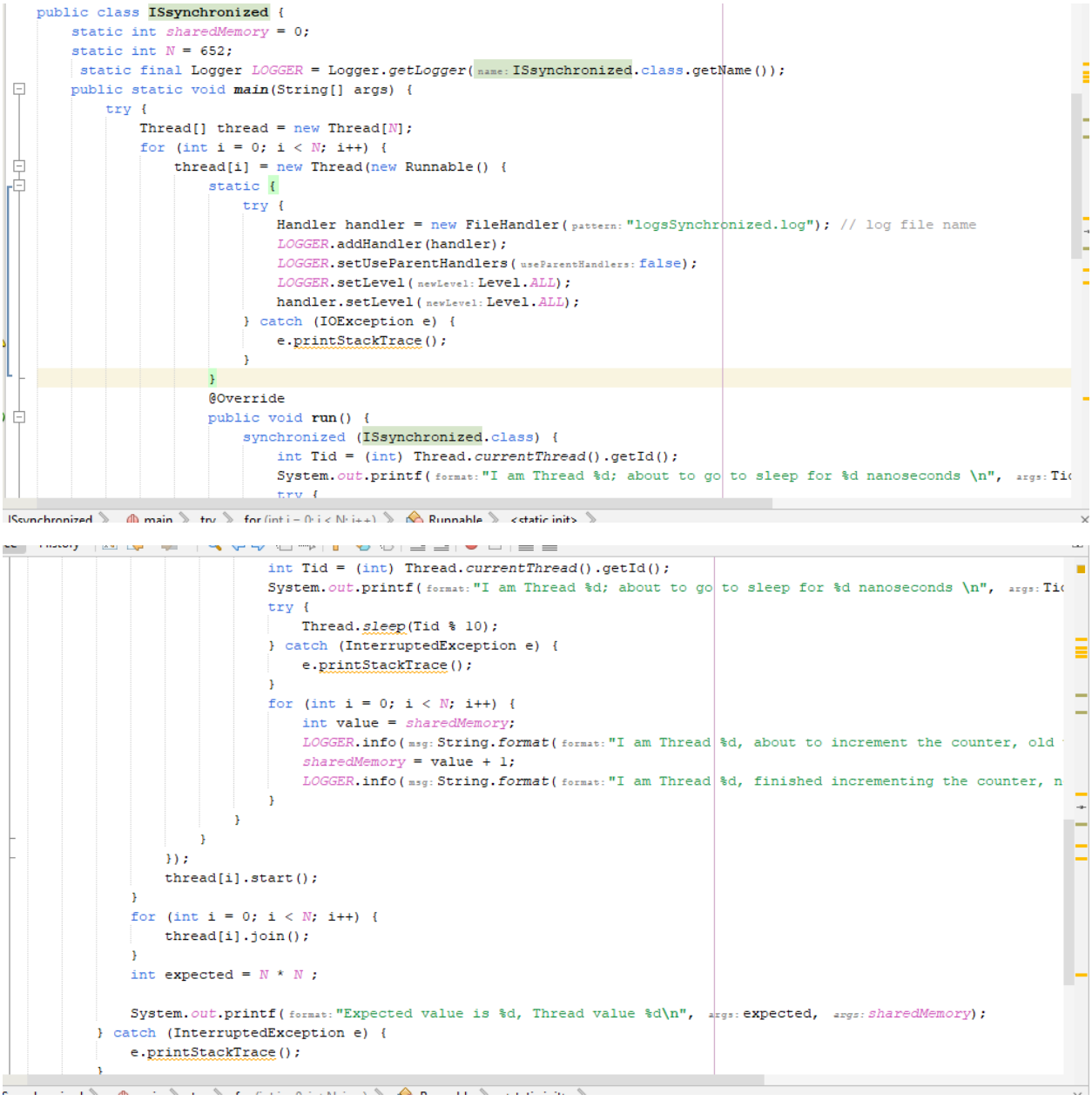


```
                @Override
                public void run() {

                    int Tid = (int) Thread.currentThread().getId();
                    System.out.printf( format: "I am Thread %d; about to go to sleep for %d nanoseconds \n", args: Tid, T:
                    try {
                        Thread.sleep(Tid % 10);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    for (int i = 0; i < N; i++) {
                        int value = sharedMemory;
                        LOGGER.info( msg: String.format( format: "I am Thread %d, about to increment the counter, old valu
                        sharedMemory = value + 1;
                        LOGGER.info( msg: String.format( format: "I am Thread %d, finished incrementing the counter, new v
                    }
                }
            });
            thread[i].start();
        }
        for (int i = 0; i < N; i++) {
            thread[i].join();
        }
        int expected = N * N ;
//          LOGGER.info(String.format("Expected value is %d, Thread value %d\n", expected, buffer.getInt(0)));
        System.out.printf( format: "Expected value is %d, Thread value %d\n", args: expected, args: sharedMemory);
    } catch (InterruptedException e) {
```

In this part (unsynchronized solution)I implemented the main function which first of all, creates the shared memory using a member of class that is global to all functions inside the class and then initializes the integer value to 0 then I created N number of threads which N is 500 + 152 (the last three digits of my ID number), and also created a log file so we can observe the operations done to the variable and in each thread we do N number of additions and then waits for all the threads to complete and then print out the expected value and the value from the thread.

Then for the synchronized solution:



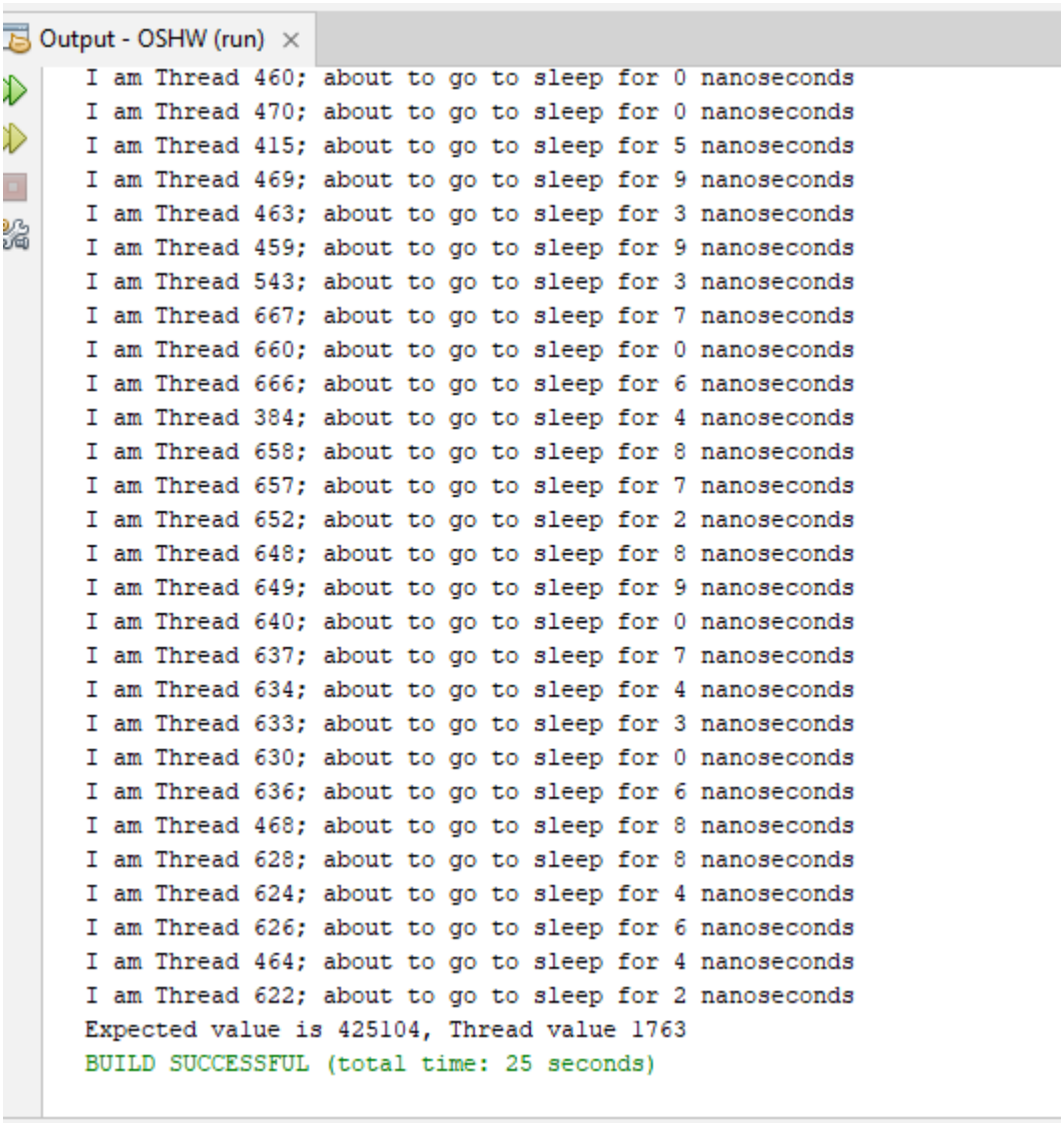
```
public class ISsynchronized {
    static int sharedMemory = 0;
    static int N = 652;
     static final Logger LOGGER = Logger.getLogger( name: ISsynchronized.class.getName());
    public static void main(String[] args) {
        try {
            Thread[] thread = new Thread[N];
            for (int i = 0; i < N; i++) {
                thread[i] = new Thread(new Runnable() {
                    static {
                        try {
                            Handler handler = new FileHandler( pattern: "logsSynchronized.log"); // log file name
                            LOGGER.addHandler(handler);
                            LOGGER.setUseParentHandlers( useParentHandlers: false);
                            LOGGER.setLevel( newLevel: Level.ALL);
                            handler.setLevel( newLevel: Level.ALL);
                        } catch (IOException e) {
                            e.printStackTrace();
                        }
                    }
                    @Override
                    public void run() {
                        synchronized (ISsynchronized.class) {
                            int Tid = (int) Thread.currentThread().getId();
                            System.out.printf( format: "I am Thread %d; about to go to sleep for %d nanoseconds \n",  args: Ti
                            try {
```

```
                            int Tid = (int) Thread.currentThread().getId();
                            System.out.printf( format: "I am Thread %d; about to go to sleep for %d nanoseconds \n",  args: Ti
                            try {
                                Thread.sleep(Tid % 10);
                            } catch (InterruptedException e) {
                                e.printStackTrace();
                            }
                            for (int i = 0; i < N; i++) {
                                int value = sharedMemory;
                                LOGGER.info( msg: String.format( format: "I am Thread %d, about to increment the counter, old
                                sharedMemory = value + 1;
                                LOGGER.info( msg: String.format( format: "I am Thread %d, finished incrementing the counter, n
                            }
                        }
                    }
                });
                thread[i].start();
            }
            for (int i = 0; i < N; i++) {
                thread[i].join();
            }
            int expected = N * N ;

            System.out.printf( format: "Expected value is %d, Thread value %d\n",  args: expected,  args: sharedMemory);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
```

Then for the synchronized solution: I used solution 2 because I solved it by myself and my id number is 11820152 then when we do 11820152 % 3 the reminder will be 2 and now for the explanation: we do the same operations as the unsynchronized solution but we use the synchronized keyword on the code we are running so that only one thread can update the global variable at a time.

Here is the output for the unsynchronized solution:



In this picture we can see the race condition happening when a thread reads a value then the other threads modify on that value already so the output will not be consistent and it will give a wrong output because multiple threads read and modify the same date at the same time.

```
  <date>2023-05-07T16:08:12.473230400Z</date>
  <millis>1683475692473</millis>
  <nanos>230400</nanos>
  <sequence>2795</sequence>
  <logger>NotSynchronized</logger>
  <level>INFO</level>
  <class>NotSynchronized$1</class>
  <method>run</method>
  <thread>221</thread>
  <message>I am Thread 221, finished incrementing the counter, new value is 6
</message>
</record>
<record>
  <date>2023-05-07T16:08:12.473230400Z</date>
  <millis>1683475692473</millis>
  <nanos>230400</nanos>
  <sequence>2794</sequence>
  <logger>NotSynchronized</logger>
  <level>INFO</level>
  <class>NotSynchronized$1</class>
  <method>run</method>
  <thread>630</thread>
  <message>I am Thread 630, about to increment the counter, old value was 5
```

And then when I run the synchronized solution the output look like this:

```
Output - OSHW (run)
   I am Thread 43; about to go to sleep for 3 nanoseconds
   I am Thread 42; about to go to sleep for 2 nanoseconds
   I am Thread 41; about to go to sleep for 1 nanoseconds
   I am Thread 40; about to go to sleep for 0 nanoseconds
   I am Thread 39; about to go to sleep for 9 nanoseconds
   I am Thread 38; about to go to sleep for 8 nanoseconds
   I am Thread 37; about to go to sleep for 7 nanoseconds
   I am Thread 36; about to go to sleep for 6 nanoseconds
   I am Thread 35; about to go to sleep for 5 nanoseconds
   I am Thread 34; about to go to sleep for 4 nanoseconds
   I am Thread 33; about to go to sleep for 3 nanoseconds
   I am Thread 32; about to go to sleep for 2 nanoseconds
   I am Thread 31; about to go to sleep for 1 nanoseconds
   I am Thread 30; about to go to sleep for 0 nanoseconds
   I am Thread 29; about to go to sleep for 9 nanoseconds
   I am Thread 28; about to go to sleep for 8 nanoseconds
   I am Thread 23; about to go to sleep for 3 nanoseconds
   I am Thread 20; about to go to sleep for 0 nanoseconds
   I am Thread 25; about to go to sleep for 5 nanoseconds
   I am Thread 24; about to go to sleep for 4 nanoseconds
   I am Thread 27; about to go to sleep for 7 nanoseconds
   I am Thread 26; about to go to sleep for 6 nanoseconds
   I am Thread 21; about to go to sleep for 1 nanoseconds
   I am Thread 22; about to go to sleep for 2 nanoseconds
   I am Thread 18; about to go to sleep for 8 nanoseconds
   I am Thread 19; about to go to sleep for 9 nanoseconds
   I am Thread 17; about to go to sleep for 7 nanoseconds
   Expected value is 425104, Thread value 425104
   BUILD SUCCESSFUL (total time: 23 seconds)
```

And we can observe in the log file that each thread will read the variable after the previous iteration in the loop and no other thread can access the code until that thread finishes executing the code so there will be no race condition.

```
File   Edit   View
  <level>INFO</level>
  <class>ISsynchronized$1</class>
  <method>run</method>
  <thread>658</thread>
  <message>I am Thread 658, finished incrementing the counter, new value is 7163
</message>
</record>
<record>
  <date>2023-05-07T16:06:31.634545800Z</date>
  <millis>1683475591634</millis>
  <nanos>545800</nanos>
  <sequence>14326</sequence>
  <logger>ISsynchronized</logger>
  <level>INFO</level>
  <class>ISsynchronized$1</class>
  <method>run</method>
  <thread>658</thread>
  <message>I am Thread 658, about to increment the counter, old value was 7163
</message>
</record>
<record>
  <date>2023-05-07T16:06:31.634545800Z</date>
  <millis>1683475591634</millis>
  <nanos>545800</nanos>
  <sequence>14327</sequence>
  <logger>ISsynchronized</logger>
  <level>INFO</level>
  <class>ISsynchronized$1</class>
  <method>run</method>
  <thread>658</thread>
  <message>I am Thread 658, finished incrementing the counter, new value is 7164
</message>
</record>
<record>
  <date>2023-05-07T16:06:31.634545800Z</date>
```

Now for the time measure using Linux(CentOS):

```
[user@localhost OS_HW2]$ javac ISsynchronized.java
[user@localhost OS_HW2]$ time java ISsynchronized
Expected value is 425104, Thread value 425104

real    0m0.247s
user    0m0.123s
sys     0m0.170s
[user@localhost OS_HW2]$ javac NotSynchronized.java
[user@localhost OS_HW2]$ time java NotSynchronized
Expected value is 425104, Thread value 3390

real    0m19.981s
user    0m18.954s
sys     0m6.181s
[user@localhost OS_HW2]$
```

And In Windows:

```
PS C:\Users\ahmad\Downloads\Sem2_2023\OS\HW2\windows> javac .\ISsynchronized_Windows.java
PS C:\Users\ahmad\Downloads\Sem2_2023\OS\HW2\windows> Measure-Command { java ISsynchronized_Windows }


Days              : 0
Hours             : 0
Minutes           : 0
Seconds           : 0
Milliseconds      : 262
Ticks             : 2625294
TotalDays         : 3.03853472222222E-06
TotalHours        : 7.29248333333333E-05
TotalMinutes      : 0.00437549
TotalSeconds      : 0.2625294
TotalMilliseconds : 262.5294
```

```
PS C:\Users\ahmad\Downloads\Sem2_2023\OS\HW2\windows> javac .\NotSynchronized__Windows.java
PS C:\Users\ahmad\Downloads\Sem2_2023\OS\HW2\windows> Measure-Command { java NotSynchronized__Windows }


Days              : 0
Hours             : 0
Minutes           : 0
Seconds           : 20
Milliseconds      : 410
Ticks             : 204107083
TotalDays         : 0.000236235049768519
TotalHours        : 0.00566964119444444
TotalMinutes      : 0.340178471666667
TotalSeconds      : 20.4107083
TotalMilliseconds : 20410.7083


PS C:\Users\ahmad\Downloads\Sem2_2023\OS\HW2\windows>
```

Windows and Linux use different approaches to handle thread scheduling and synchronization. And Linux uses a more lightweight thread model than Windows, and its scheduler is more efficient for multi-threaded workloads so that's why it's a bit faster in Linux.