operating system

The Producer-Consumer Problem

St. ID	St. Name	Department & level
202000163	Amira Wael Amir	CS / L.3
202000240	Joseph Issa Bekhit	CS / L.3
202000232	Thomas Fathy Shehata	CS / L.3
202000711	Mayer Rida	CS / L.3
202000242	John Halim Hanna	CS / L.3
202000699	Marina Ezzat Louis	CS / L.3
201900874	Merola Milad	IS / L.4

1) Solution Pseudocode

1) Consumer

```
void consumer() {
  while(T) {
    wait(F)
    wait(S)
    take()
    signal(S)
    signal(E)
  }
}
```

2) producer

```
void producer() {
    while(T) {
    produce()
    wait(E)
    wait(S)
    append()
    signal(S)
    signal(F)
  }
}
```

2) Examples of Deadlock

The deadlock might happened if there is no consistency between the producer and consumer leading them to access the critical section simultaneously. And this will lead to both of them want to access the buffer to consume and produce but they cannot do this .. this will lead to Deadlock

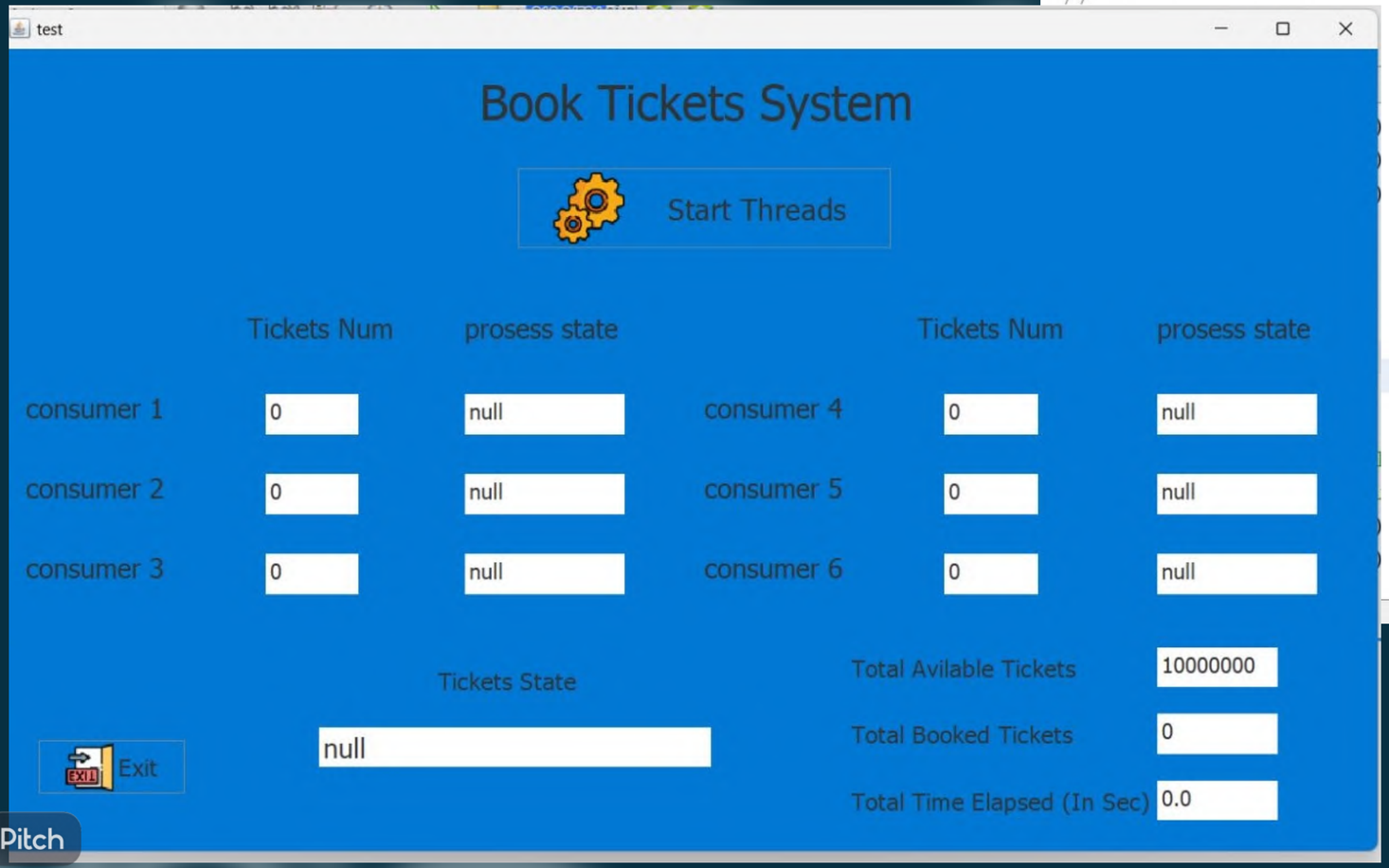
3) How to solve it ??

Deadlock occurs if the consumer succeeds in acquiring semaphore s when the buffer is empty. The starting value for the semaphore n must be 1 and not 0 for deadlock-free operation

we also used synchronized function to solve it

screens for deadlock :

```
        public void producerr() throws InterruptedException {

                while (produced_tik<=buf.Max_tik_num) {
                        synchronized (this) {


                        while (buf.size() == buf.Max ) {

//                        System.out.println("Buffer is full . waiting....");
                        wait();

                                }

                        Wait(buf.empty);
                        Wait(buf.mutex);

                        produced_tik++;
                       buf.produce(produced_tik);

                           Signal(buf.mutex);
                         Signal(buf.full);

                                }

                       }
```



4) Examples of Starvation

The starvation might happen When there is one thread it works, and the rest does not work…

5) How to solve it ??

we solve it by using " notify " method

The notify() method is used to wake up only one thread that’s waiting for an object, and that thread then begins execution.

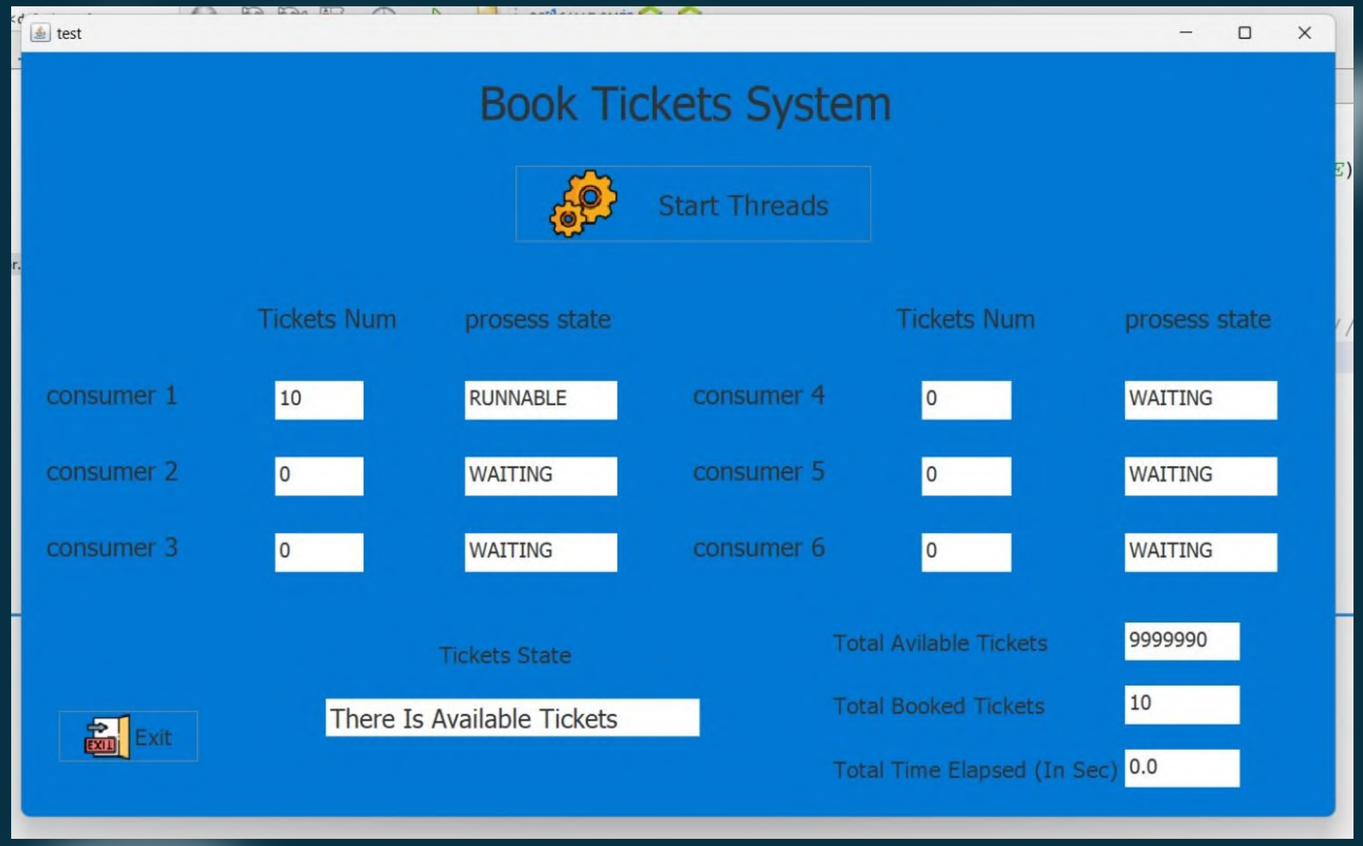
The thread class notify() method is used to wake up a single thread. If multiple threads are waiting for notification, and we use the notify() method, only one thread will receive the notification, and the others will have to wait for more.

If correctly coded producer-consumer problem is always going to be starvation free. Consumer blocking till there is ready buffer to consume. Producer blocking till there is empty buffer to produce. Producer after producing immediately moving buffer to ready buffer queue.

Also , every thread of all threads has the same priority

screens for starvation :

```
        public void Signal(Mysemaphore S) {
//                  notify();
                S.value++;


        }
```



6) Explanation for real world application and how did apply the problem

We have applied this to an airline in the ticketing part where the producer is the airline itself and the consumer is the customers who book tickets from the company

And we have made the multiple consumers in the form of customers who book tickets from the company and we have also made the multiple producers in the body of employees who increase the number of tickets that are booked from the company until it reaches 1000 tickets