Topics

- Thread Synchronization ?
- Why Thread Synchronization ?
- Thread Synchronization Methods?

What is Thread Synchronization?

- When multiple threads share a common data structure then a mechanism is required to ensure that the shared resource will be used by only one thread at a time
- Thread Synchronization → Process which guarantees the shared resource will be used by only one thread at a time
- Key to Synchronization → Monitor [An Object which is used as a mutually exclusive lock]
- Only one thread can Own a Monitor at a Given Point of Time
- In order to Access a Shared Resource, a Thread has to First Acquire the Lock over that Resource. If Lock is granted then a Thread is said have Entered the Monitor. All other Threads Attempting to Acquire the Lock over the Same Shared Resource will Enter into WAITING state.

Why Thread Synchronization?

- Without Proper Synchronization, Threads Can Corrupt Data Structures
- Example

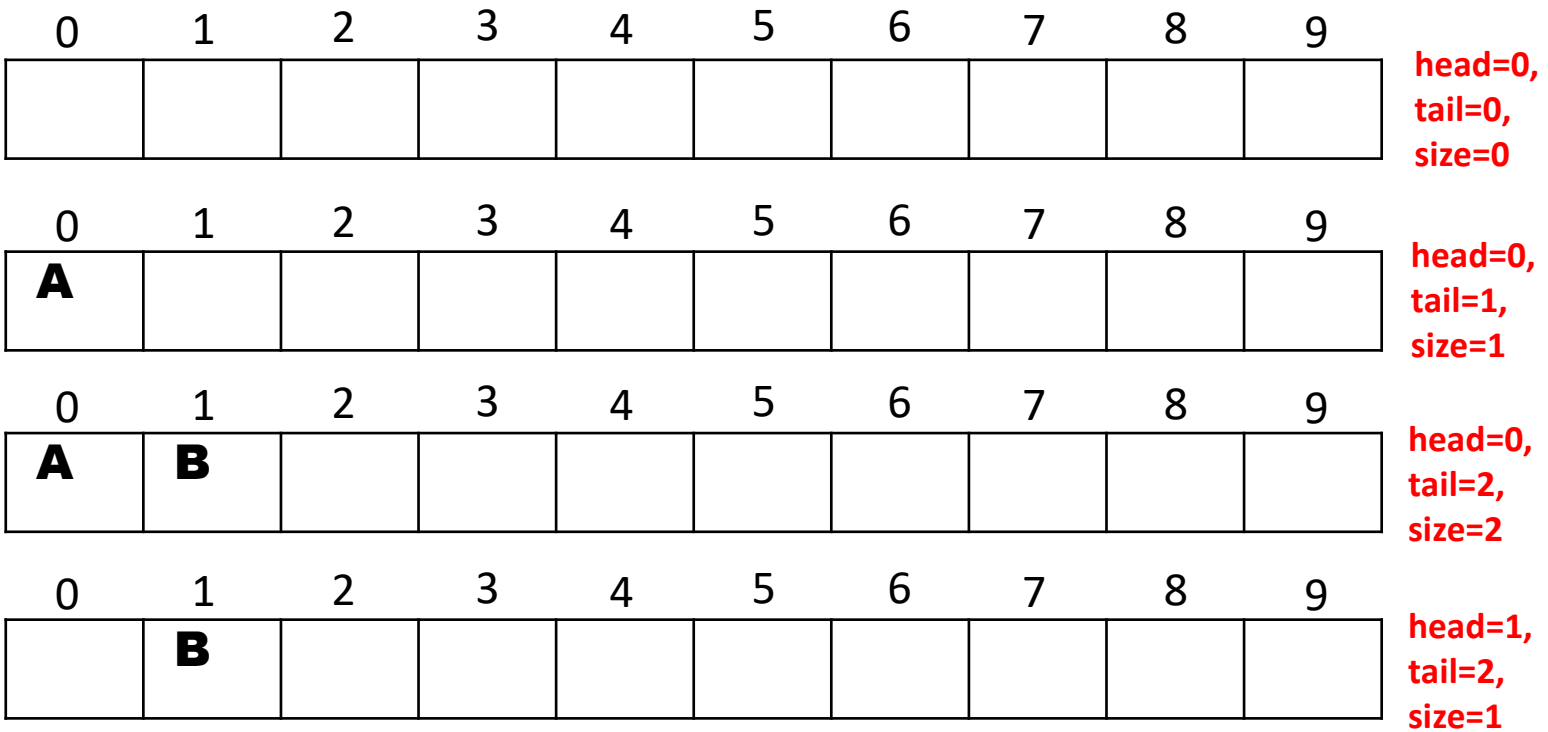
1. Assume we have a Queue of Greetings, with Capacity as 10.
2. Assume there is ProducerThread which keeps on adding a particular string type greeting into the queue for a fixed number of iterations. Again assume that there are used two instances of the ProducerThread say T1 and T2.
3. Assume there is ConsumerThread which keeps on removing greeting from the queue for a fixed number of iterations. Again assume that there is used only one instance of ConsumerThread say T3.

Why Thread Synchronization? …


```
// Queue Class Partial Implementation
public class Queue
{
        private Object[ ]  elements;        // Elements of Queue
        private int head;                   // head location of queue
        private int tail;                   // tail location of queue
        private int size;                   // size of queue
        // Constructor Method
        public Queue(int capacity)          {       }
        // Method to Remove
        public Object removeFirst()         {       } // head++, size--
         // Method to Add
        public void add(Object anotherObject)   {       } // tail++, size++
         // Methods to check if queue is full or empty
        public boolean isFull()             {       }
        public boolean isEmpty()            {       }
} // End of class Queue
```

Why Thread Synchronization? …

- Queue is FIFO (First-in-First-Out) Type Data Structure
- Elements are Added at one End (Tail) and Removed From Other End (Head)
- Initially Head = Tail = 0, Size = 0



Why Thread Synchronization? …

```
class ProducerThread implements Runnable
{
        private Queue queue;
        private int repetitions;
        private static final int DELAY =10;
        public Producer(String greeting, Queue queue , int reps){}
        public void run()
        {
                try
                {
                        int i =1;
                        while(i <= repetitions)
                        {
                                if(!queue.isFull())
                                {
                                        queue.add(i+":"+greeting);
                                        i++
                                }
                                Thread.sleep(DELAY);
                        }// End of while loop
                }// End of try
                catch(InterruptedException e){}
        }// End of Method
}// End of Class
```

Why Thread Synchronization? …


```
class ConsumerThread implements Runnable
{
        private Queue queue;
        private int repetitions;
        private static final int DELAY =10;
        public Consumer(Queue queue , int reps){}
        public void run()
        {
                try
                {
                        int i =1;
                        while(i <= repetitions)
                        {
                                if(!queue.isEmpty())
                                {
                                        Object Obj = queue.removeFirst();
                                        i++
                                }
                                Thread.sleep(DELAY);
                        }// End of while Loop
                }// End of try
                catch(InterruptedException e){}
        }// End of Method
}// End of class
```

Why Thread Synchronization? …

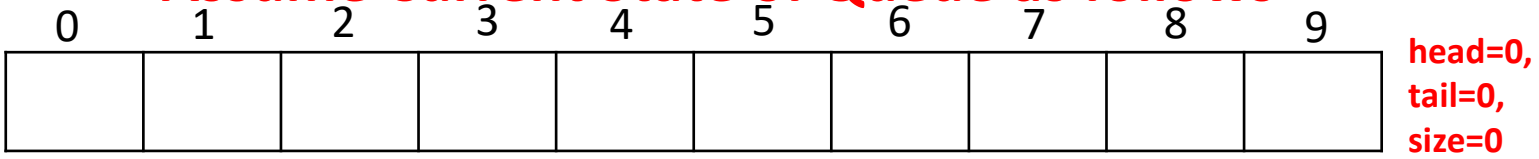

```
class Driver
{
        public static void main(String args[])
        {
                Queue queue = new Queue(10);
                final int repetitions = 100;

                Runnable r1 = new ProducerThread("Hello, World", queue, repetitions);
                Runnable r2 = new ProducerThread("Goodbye, World", queue, repetitions);
                Runnable r3 = new ConsumerThread(queue, 2*repetitions);

                Thread T1 = new Thread(r1);
                Thread T2 = new Thread(r2);
                Thread T3 = new Thread(r3);

                T1.start();
                T2.start();
                T3.start();

        }//End of Method
}//End of class
```

How Queue Can be Corrupted?

Why Thread Synchronization? …

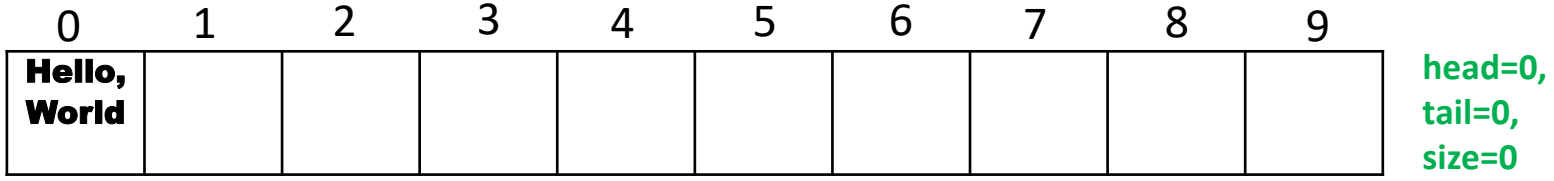
How Queue Can be Corrupted?

Assume Current State of Queue as follows



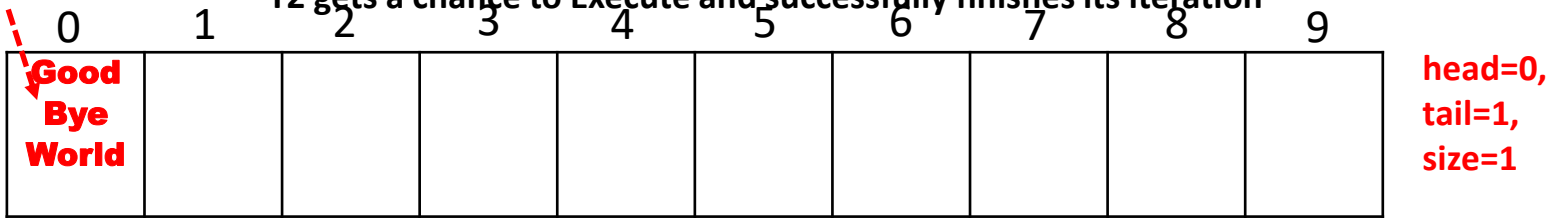
Assume Threads are Scheduled as T1 (Producer), T2 (Producer) and T1 (Producer)

T1 Executing, But just After Adding the Greeting its Allotted Time Slice is Over [T1 is not able to Update the Values of tail and size fields. So T1 will update these values in next cycle]



Overwritten

T2 gets a chance to Execute and successfully finishes its iteration

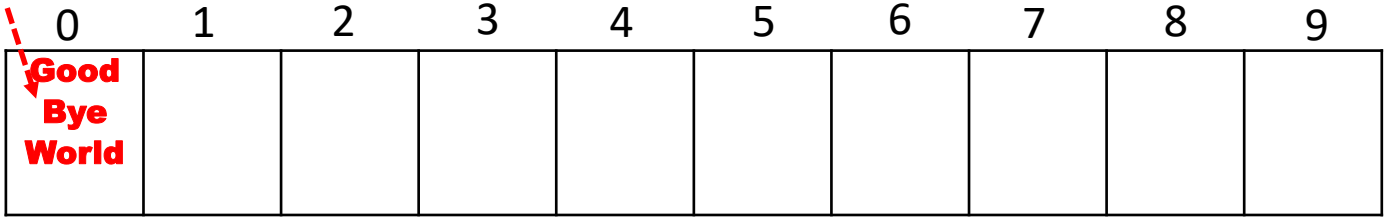


Why Thread Synchronization? …

How Queue Can be Corrupted?

Overwritten

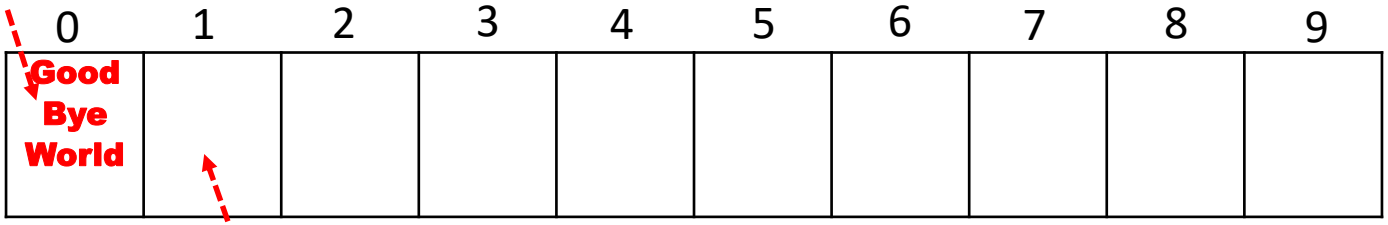
T2 gets a chance to Execute and successfully finishes its iteration



head=0,
tail=1,
size=1

T1 gets a chance to Execute and Tries Finish Its Previous Iteration

Overwritten



head=0,
tail=2,
size=2

Unwritten Location

Synchronization Mechanism

innovate achieve lead

- Using Synchronized Methods
- Using Synchronize Statement

Synchronization Mechanism (Via Synchronized Methods)

- Every Object in Java has an Associated Implicit monitor associated with it
- Mechanism to enter Object's Monitor → Declare the Methods of a class with synchronized keyword
- Example

```
class    Queue
{
         ……….
         public   synchronized    void    add( ….)
         {
                  ……….
         }
         public   synchronized    void    remove( ….)
         {
                  ……….
         }
} // End of class
```

Synchronization Mechanism (Via Synchronized Statement)

- Helps to synchronize the access to an object which does not use synchronized methods
- Syntax

```
synchronized(objRef)
{
        ……..// synchronize block
}
```

- Where ‘objRef’ is a reference to the object being synchronized

Thank You