Multithreading :

Java keyword

Synchronized -> to achieve sequence between threads and used to prevent deadlock

Volatile -> to achieve field visibility

Problems :

Deadlock, starvation, priority inversion

Synchronization in Java

Synchronization in java is the capability *to control the access of multiple threads to any shared resource*.

Java Synchronization is better option where we want **to allow only one thread to access** the shared resource.

Why ?

1. To prevent thread interference.
2. To prevent Inconsistency problem.

There are two types of synchronization

1. Process Synchronization
2. Thread Synchronization

There are two types of thread synchronization mutual exclusive and inter-thread communication.

1. Mutual Exclusive
   1. Synchronized method.
   2. Synchronized block.
   3. static synchronization.
2. Cooperation (Inter-thread communication in java)

### **Mutual Exclusive**

Mutual Exclusive helps keep threads from interfering with one another while sharing data. This can be done by three ways in java:

1. by synchronized method
2. by synchronized block
3. by static synchronization

Multi-threaded programs may often come to a situation where multiple threads try to access the same resources and finally produce erroneous and unforeseen results.

So it needs to be made sure by some synchronization method that only one thread can access the resource at a given point of time.

Java provides a way of creating threads and synchronizing their task by using synchronized blocks. Synchronized blocks in Java are marked with the synchronized keyword.

A synchronized block in Java is synchronized on some object. All synchronized blocks synchronized on the same object can only have one thread executing inside them at a time.

All other threads attempting to enter the synchronized block are blocked until the thread inside the synchronized block exits the block.

Following is the general form of a synchronized block:

// Only one thread can execute at a time.

// sync\_object is a reference to an object

// whose lock associates with the [monitor](http://quiz.geeksforgeeks.org/monitors/).

// The code is said to be synchronized on

// the monitor object

synchronized(sync\_object)

{

// Access shared variables and other

// shared resources

}

This synchronization is implemented in Java with a concept called monitors. Only one thread can own a monitor at a given time. When a thread acquires a lock, it is said to have entered the monitor. All other threads attempting to enter the locked monitor will be suspended until the first thread exits the monitor.

Threads Synchronization

* + Count Down Latch
  + Semaphore
  + Cyclic Barrier
  + Phaser
  + Exchanger