**Multithreading in java** is a process of executing multiple threads simultaneously.

A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking

Life cycle of a Thread (Thread States)

1. [Life cycle of a thread](https://www.javatpoint.com/life-cycle-of-a-thread)
   1. [New](https://www.javatpoint.com/life-cycle-of-a-thread#threadstatenew)
   2. [Runnable](https://www.javatpoint.com/life-cycle-of-a-thread#threadstaterunnable)
   3. [Running](https://www.javatpoint.com/life-cycle-of-a-thread#threadstaterunning)
   4. [Non-Runnable (Blocked)](https://www.javatpoint.com/life-cycle-of-a-thread#threadstateblocked)
   5. [Terminated](https://www.javatpoint.com/life-cycle-of-a-thread#threadstateterminated)

A thread can be in one of the five states. According to sun, there is only 4 states in **thread life cycle in java** new, runnable, non-runnable and terminated. There is no running state.



How to create thread

There are two ways to create a thread:

1. By extending Thread class
2. By implementing Runnable interface.

### Commonly used methods of Thread class:

|  |
| --- |
| 1. **public void run():**is used to perform action for a thread. 2. **public void start():**starts the execution of the thread.JVM calls the run() method on the thread. 3. **public void sleep(long miliseconds):**Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds. 4. **public void join():**waits for a thread to die. 5. **public void join(long miliseconds):**waits for a thread to die for the specified miliseconds. 6. **public int getPriority():**returns the priority of the thread. 7. **public int setPriority(int priority):**changes the priority of the thread. 8. **public String getName():**returns the name of the thread. 9. **public void setName(String name):**changes the name of the thread. 10. **public Thread currentThread():**returns the reference of currently executing thread. 11. **public int getId():**returns the id of the thread. 12. **public Thread.State getState():**returns the state of the thread. 13. **public boolean isAlive():**tests if the thread is alive. 14. **public void yield():**causes the currently executing thread object to temporarily pause and allow other threads to execute. 15. **public void suspend():**is used to suspend the thread(depricated). 16. **public void resume():**is used to resume the suspended thread(depricated). 17. **public void stop():**is used to stop the thread(depricated). 18. **public boolean isDaemon():**tests if the thread is a daemon thread. 19. **public void setDaemon(boolean b):**marks the thread as daemon or user thread. 20. **public void interrupt():**interrupts the thread. 21. **public boolean isInterrupted():**tests if the thread has been interrupted. 22. **public static boolean interrupted():**tests if the current thread has been interrupted. |

### Java Thread Example by extending Thread class

1. **class** Multi **extends** Thread{
2. **public** **void** run(){
3. System.out.println("thread is running...");
4. }
5. **public** **static** **void** main(String args[]){
6. Multi t1=**new** Multi();
7. t1.start();
8. }
9. }

# Thread Scheduler in Java

**Thread scheduler** in java is the part of the JVM that decides which thread should run.There is no guarantee that which runnable thread will be chosen to run by the thread scheduler.Only one thread at a time can run in a single process.

# Sleep method in java

The sleep() method of Thread class is used to sleep a thread for the specified amount of time.

## Example of sleep method in java

1. **class** TestSleepMethod1 **extends** Thread{
2. **public** **void** run(){
3. **for**(**int** i=1;i<5;i++){
4. **try**{Thread.sleep(500);}**catch**(InterruptedException e){System.out.println(e);}
5. System.out.println(i);
6. }
7. }
8. **public** **static** **void** main(String args[]){
9. TestSleepMethod1 t1=**new** TestSleepMethod1();
10. TestSleepMethod1 t2=**new** TestSleepMethod1();
12. t1.start();
13. t2.start();
14. }
15. }

# 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 use Synchronization

The synchronization is mainly used to

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

### Types of Synchronization

There are two types of synchronization

1. Process Synchronization
2. Thread Synchronization

Here, we will discuss only thread synchronization.

### 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)
3. //example of java synchronized method
4. **class** Table{
5. **synchronized** **void** printTable(**int** n){//synchronized method
6. **for**(**int** i=1;i<=5;i++){
7. System.out.println(n\*i);
8. **try**{
9. Thread.sleep(400);
10. }**catch**(Exception e){System.out.println(e);}
11. }
13. }
14. }
16. **class** MyThread1 **extends** Thread{
17. Table t;
18. MyThread1(Table t){
19. **this**.t=t;
20. }
21. **public** **void** run(){
22. t.printTable(5);
23. }
25. }
26. **class** MyThread2 **extends** Thread{
27. Table t;
28. MyThread2(Table t){
29. **this**.t=t;
30. }
31. **public** **void** run(){
32. t.printTable(100);
33. }
34. }
36. **public** **class** TestSynchronization2{
37. **public** **static** **void** main(String args[]){
38. Table obj = **new** Table();//only one object
39. MyThread1 t1=**new** MyThread1(obj);
40. MyThread2 t2=**new** MyThread2(obj);
41. t1.start();
42. t2.start();
43. }
44. }