Multithreading in Java

**Multithreading in**[**Java**](https://www.javatpoint.com/java-tutorial) is a **process of executing multiple threads simultaneously.**

**A thread is a lightweight sub-process**, the smallest unit of processing.

Multiprocessor (many processors i.e. many CPU)

**Multiprocessing (multiple number of CPU or more than one CPU) and multithreading, both are used to achieve multitasking.**

However, we use multithreading than multiprocessing because threads use a shared memory area.

They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process.

Java Multithreading is mostly used in games, animation, etc.

Advantages of Java Multithreading

1) It **doesn't block the user** because threads are independent and you can perform multiple operations at the same time.

2) You **can perform many operations together, so it saves time**.

3) Threads are **independent**, so it doesn't affect other threads if an exception occurs in a single thread.

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. [Blocked](https://www.javatpoint.com/life-cycle-of-a-thread#threadstateblocked)
   5. [Terminated](https://www.javatpoint.com/life-cycle-of-a-thread#threadstateterminated)

NRRBT

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.

But for better understanding the threads, we are explaining it in the 5 states.

The life cycle of the thread in java is controlled by JVM. The java thread states are as follows:



Phases of Thread

1. **New:** In this phase, the thread is created using class "Thread class". It remains in this state till the program **starts** the thread. It is also known as born thread.
2. **Runnable:** In this phase, the instance of the thread is invoked with a start method. The thread control is given to scheduler to finish the execution. It depends on the scheduler, whether to run the thread.
3. **Running:** When the thread starts executing, then the state is changed to "running" state. The scheduler selects one thread from the thread pool, and it starts executing in the application.
4. **Blocked/Waiting:** This is the state when a thread has to wait. As there multiple threads are running in the application, there is a need for synchronization between threads. Hence, one thread has to wait, till the other thread gets executed. Therefore, this state is referred as waiting state.If a thread is in waiting stage and the stage is over, it will move to runnable.
5. **Dead/Terminated:** This is the state when the thread is terminated. The thread is in running state and as soon as it completed processing it is in "dead state".

|  |  |
| --- | --- |
| **Method** | **Description** |
| start() | This method starts the execution of the thread and JVM calls the run() method on the thread. |
| Sleep(int milliseconds) | This method makes the thread sleep hence the thread's execution will pause for milliseconds provided and after that, again the thread starts executing. This help in synchronization of the threads. |
| getName() | It returns the name of the thread. |
| setPriority(int newpriority) | * It changes the priority of the thread. * Minimum=1 * Max=10 * Range=1 to 10 * Default=5 * JVM always prefer maximum priority . |
| yield () | It causes current thread on halt and other threads to execute. |

1)Object.wait():can only be used within the synchronized method

3)object.notify(): to terminate wait state, the current thread will be stopped from execution and the object will be released to be used by others

How to create thread

There are two ways to create a thread:

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

Thread class:

|  |
| --- |
| Thread class provide constructors and methods to create and perform operations on a thread.  Thread class extends Object class and implements Runnable interface. |

Commonly used Constructors of Thread class:

|  |
| --- |
| * Thread() * Thread(String name) * Thread(Runnable r) * Thread(Runnable r, String name) |

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

Runnable interface:

|  |
| --- |
| The Runnable interface should be implemented by any class whose instances are intended to be  executed by a thread. Runnable interface have only one method named run(). |

|  |
| --- |
| 1. **public void run():**is used to perform action for a thread. |

Starting a thread:

|  |
| --- |
| **Start () method** of Thread class is used to start a newly created thread. It performs following tasks:   * A new thread starts. * The thread moves from New state to the Runnable state. * When the thread gets a chance to execute, its target run () method will run. |

1) Java Thread Example by extending Thread class

**class** Multi **extends** Thread

{

**public** **void** run()

{

System.out.println("thread is running...");

}

**public** **static** **void** main(String args[])

{

Multi t1=**new** Multi();

t1.start();  // new,runnable,run()

 }

}

Output:thread is running...

2) Java Thread Example by implementing Runnable interface

1. **class** Multi3 **implements** Runnable{
2. **public** **void** run(){
3. System.out.println("thread is running...");
4. }
6. **public** **static** **void** main(String args[]){
7. Multi3 m1=**new** Multi3();
8. Thread t1 =**new** Thread(m1);
9. t1.start();  //New,runnable,run
10. }
11. }

|  |
| --- |
| If you are not extending the Thread class, your class object would not be treated as a thread object.So you need to explicitely create Thread class object.We are passing the object of your class that implements Runnable so that your class run() method may execute. |

Output:thread is running...

Sleep()

* Thread.Sleep()
* Outside synchronized method and statement
* Interrupt() is used to terminate sleep
* It stops the thread from execution for specified unit of time