**Thread Life Cycle Methods**

We have various methods which can be called on Thread class object. These methods are very useful when writing a multithreaded application. Thread class has following important methods. We will understand various thread states as well later in this tutorial.

|  |  |
| --- | --- |
| Method Signature | Description |
| String getName() | Retrieves the name of running thread in the current context in String format |
| void start() | This method will start a new thread of execution by calling run() method of Thread/runnable object. |
| void run() | This method is the entry point of the thread. Execution of thread starts from this method. |
| void sleep(int sleeptime) | This method suspend the thread for mentioned time duration in argument (sleeptime in ms) |
| void yield() | By invoking this method the current thread pause its execution temporarily and allow other threads to execute. |
| void join() | This method used to queue up a thread in execution. Once called on thread, current thread will wait till calling thread completes its execution |
| boolean isAlive() | This method will check if thread is alive or dead |

**Example**

**public** **class** ThreadDemo {

**public** **static** **void** main(String[] args) **throws** Exception{

// Make object of Runnable

ThreadRunnable anr = **new** ThreadRunnable();

Thread pune = **new** Thread(anr);

pune.setName("Pune");

Thread jaipur = **new** Thread(anr);

jaipur.setName("Jaipur");

System.***out***.println("Thread State before calling start: "+pune.getState());

pune.start();

jaipur.start();

System.***out***.println("Thread State of Pune in Main method before Sleep: " + pune.getState());

System.***out***.println("Thread State of Jaipur in Main method before Sleep: " + jaipur.getState());

Thread.*sleep*(10000);

System.***out***.println("Thread State of Pune in Main method after sleep: " + pune.getState());

System.***out***.println("Thread State of Jaipur in Main method after sleep: " + jaipur.getState());

}

}

**class** ThreadRunnable **implements** Runnable {

@Override

**public** **void** run() {

**for** (**int** x = 1; x < 4; x++) {

System.***out***.println("Run by " + Thread.*currentThread*().getName());

**try** {

Thread.*sleep*(1000);

} **catch** (InterruptedException ex) {

ex.printStackTrace();

}

}

System.***out***.println("Thread State of: "+ Thread.*currentThread*().getName()+ " - "+Thread.*currentThread*().getState());

System.***out***.println("Exit of Thread: "

+ Thread.*currentThread*().getName());

}

}

**Output-**

Thread State before calling start: NEW

Thread State of Pune in Main method before Sleep: RUNNABLE

Thread State of Jaipur in Main method before Sleep: RUNNABLE

Run by Pune

Run by Jaipur

Run by Jaipur

Run by Pune

Run by Pune

Run by Jaipur

Thread State of: Pune - RUNNABLE

Exit of Thread: Pune

Thread State of: Jaipur - RUNNABLE

Exit of Thread: Jaipur

Thread State of Pune in Main method after sleep: TERMINATED

Thread State of Jaipur in Main method after sleep: TERMINATED