Java Concurrency – yield(), sleep() and join() Methods

Last Updated : 16 Jan, 2022

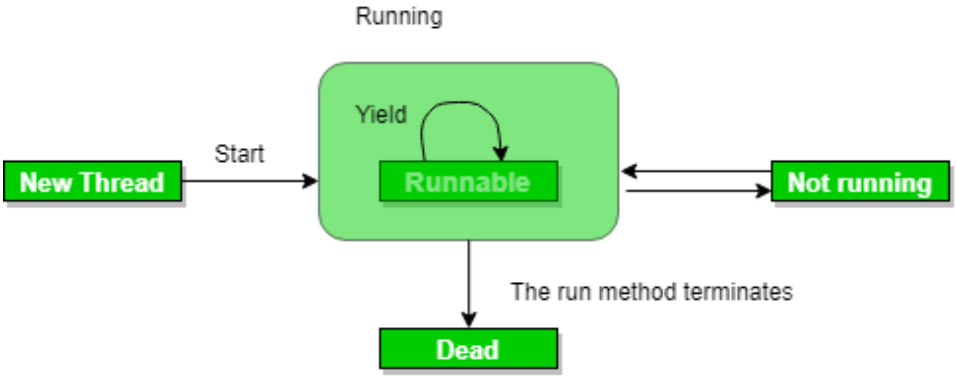
In this article, we will learn what is yield(), join(), and sleep() methods in Java and what is the basic difference between these three. First, we will see the basic introduction of all these three methods, and then we compare these three.

We can prevent the execution of a thread by using one of the following methods of the Thread class. All three methods will be used to prevent the thread from execution.

1. yield() Method

Suppose there are three threads t1, t2, and t3. Thread t1 gets the processor and starts its execution and thread t2 and t3 are in Ready/Runnable state. The completion time for thread t1 is 5 hours and the completion time for t2 is 5 minutes. Since t1 will complete its execution after 5 hours, t2 has to wait for 5 hours to just finish 5 minutes job. In such scenarios where one thread is taking too much time to complete its execution, we need a way to prevent the execution of a thread in between if something important is pending. yield() helps us in doing so.

The **yield()** basically means that the thread is not doing anything particularly important and if any other threads or processes need to be run, they should run. Otherwise, the current thread will continue to run.



Use of yield method:

- Whenever a thread calls **java.lang.Thread.yield** method gives hint to the thread scheduler that it is ready to pause its execution. The thread scheduler is free to ignore this hint.
- If any thread executes the yield method, the thread scheduler checks if there is any thread with the same or high priority as this thread. If the processor finds any thread with higher or same priority then it will move the current thread to Ready/Runnable state and give the processor to another thread and if not – the current thread will keep executing.
- Once a thread has executed the yield method and there are many threads with the same priority is waiting for the processor, then we can't specify which thread will get the execution chance first.
- The thread which executes the yield method will enter in the Runnable state from Running state.
- Once a thread pauses its execution, we can't specify when it will get a chance again it depends on the thread scheduler.
- The underlying platform must provide support for preemptive scheduling if we are using the yield method.

2. sleep() Method

This method causes the currently executing thread to sleep for the specified number of milliseconds, subject to the precision and accuracy of system timers and schedulers.

Syntax:

```
//  sleep for the specified number of milliseconds
public static void sleep(long millis) throws InterruptedException
```

```
//sleep for the specified number of milliseconds plus nano
seconds
public static void sleep(long millis, int nanos)
                         throws InterruptedException
```

Java

```java
// Java program to illustrate
// sleep() method in Java

import java.lang.*;

public class SleepDemo implements Runnable {
    Thread t;
    public void run()
    {
        for (int i = 0; i < 4; i++) {
            System.out.println(
                Thread.currentThread().getName() + "   "
                + i);
            try {
                // thread to sleep for 1000 milliseconds
                Thread.sleep(1000);
            }

            catch (Exception e) {
                System.out.println(e);
            }
        }
    }

    public static void main(String[] args) throws Exception
    {
        Thread t = new Thread(new SleepDemo());

        // call run() function
        t.start();

        Thread t2 = new Thread(new SleepDemo());

        // call run() function
        t2.start();
    }
}
```

Output

```
Thread-1  0
Thread-0  0
Thread-0  1
Thread-1  1
Thread-0  2
Thread-1  2
Thread-1  3
Thread-0  3
```

Note:

- *Based on the requirement we can make a thread to be in a sleeping state for a specified period of time*
- *Sleep() causes the thread to definitely stop executing for a given amount of time; if no other thread or process needs to be run, the CPU will be idle (and probably enter a power-saving mode).*

3. join() Method

The join() method of a Thread instance is used to join the start of a thread's execution to the end of another thread's execution such that a thread does not start running until another thread ends. If join() is called on a Thread instance, the currently running thread will block until the Thread instance has finished executing. The join() method waits at most this many milliseconds for this thread to die. A timeout of 0 means to wait forever

Syntax:

```
// waits for this thread to die.
public final void join() throws InterruptedException

// waits at most this much milliseconds for this thread to die
```

```
public final void join(long millis)
                throws InterruptedException

// waits at most milliseconds plus nanoseconds for this thread to
die.
The java.lang.Thread.join(long millis, int nanos)
```

Java

```java
// Java program to illustrate join() method in Java

import java.lang.*;

public class JoinDemo implements Runnable {
    public void run()
    {
        Thread t = Thread.currentThread();
        System.out.println("Current thread: "
                           + t.getName());

        // checks if current thread is alive
        System.out.println("Is Alive? " + t.isAlive());
    }

    public static void main(String args[]) throws Exception
    {
        Thread t = new Thread(new JoinDemo());
        t.start();

        // Waits for 1000ms this thread to die.
        t.join(1000);

        System.out.println("\nJoining after 1000"
                           + " milliseconds: \n");
        System.out.println("Current thread: "
                           + t.getName());

        // Checks if this thread is alive
        System.out.println("Is alive? " + t.isAlive());
    }
}
```

Output

```
Current thread: Thread-0
Is Alive? true

Joining after 1000 milliseconds:

Current thread: Thread-0
Is alive? false
```

Note:

- *If any executing thread t1 calls join() on t2 i.e; t2.join() immediately t1 will enter into waiting state until t2 completes its execution.*
- *Giving a timeout within join(), will make the join() effect to be nullified after the specific timeout.*

Comparison of yield(), join(), sleep() Methods

Property	yield()	join()	sleep()
Purpose	If a thread wants to pass its execution to give chance to remaining threads of the same priority then we should go for yield()	If a thread wants to wait until completing of some other thread then we should go for join()	If a thread does not want to perform any operation for a particular amount of time, then it goes for sleep()
Is it overloaded?	NO	YES	YES
Is it final?	NO	YES	NO

Property	yield()	join()	sleep()
Is it throws?	NO	YES	YES
Is it native?	YES	NO	sleep(long ms)->native & sleep (long ms, int ns)-> non native
Is it static?	YES	NO	YES

Comment More info

Placement Training Program

Next Article

Inter-thread Communication in Java

Similar Reads

Difference Between sleep and yield Method in Java

In java if a thread doesn't want to perform any operation for a particular amount of time then we should go for the sleep() method, which causes...

4 min read

StringJoiner Class vs String.join() Method to Join String in Java wit...

Prior to Java 8 when we need to concatenate a group of strings we need to write that code manually in addition to this we needed to repeatedly...

6 min read

Difference between wait and sleep in Java