

Thread Priorities

16 February 2026 18:53

Every thread in Java has some priorities. It may be default priority generated by JVM or customized priority provided by the programmer. The valid range of priority is 1 to 10, where one is the minimum priority and 10 is the maximum priority. Thread class defines the following constants to represent some standard priority:

Thread.MAX_PRIORITY : int	x	⌚ ★ sleep(long millis)	void
The maximum priority that a thread can have.		⌚ ★ currentThread()	Thread
Value: 10		⌚ ★ yield()	void
		✉ MAX_PRIORITY	int
		✉ MIN_PRIORITY	int
		✉ NORM_PRIORITY	int

Thread.MIN_PRIORITY : int	x	⌚ ★ sleep(long millis)	void
The minimum priority that a thread can have.		⌚ ★ currentThread()	Thread
Value: 1		⌚ ★ yield()	void
	10	✉ MAX_PRIORITY	int
	11	✉ MIN_PRIORITY	int
		✉ NORM_PRIORITY	int

Thread.NORM_PRIORITY : int	x	⌚ ★ sleep(long millis)	void
The default priority that is assigned to a thread.		⌚ ★ currentThread()	Thread
Value: 5		⌚ ★ yield()	void
		✉ MAX_PRIORITY	int
		✉ MIN_PRIORITY	int
		✉ NORM_PRIORITY	int

Thread scheduler will use priorities while allocating processor time. The thread which is having highest priority will get chance first. If two threads having same priority then we can't expect the exact execution of thread. It depends upon thread scheduler.

```
public final int getPriority()
public final void setPriority(int p)
```

```
t.setPriority(7);
t.setPriority(17); //RE: IllegalArgumentException
```

NORM_PRIORITY(default priority): The NORM_PRIORITY is 5 or the priority of main-thread is 5. But for all remaining threads NORM_PRIORITY is inherited from parent to child. That is, whatever priority parent thread has, the priority will be there for child thread.

main () {

 MyThread t = new MyThread()
 parent thread
 → main-thread priority is 5.
 }
 t - thread priority is 5
 child - thread.

```

package ThreadPriority;
public class FirstThread implements Runnable {
    private String threadName;
    public FirstThread(String threadName) {
        this.threadName = threadName;
    }
    @Override
    public void run() {
        final int max = 10;
        for (int i = 0; i < max; i++) {
            System.out.println(i + ": Child thread = " + threadName);
            try {
                Thread.sleep(1500);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

```

```

package ThreadPriority;
public class ThreadPriorityTest {
    public static void main(String[] args) throws InterruptedException {
        FirstThread ft1 = new FirstThread("ft1");
        FirstThread ft2 = new FirstThread("ft2");
        Thread t2 = new Thread(ft2);
        t2.setPriority(Thread.MAX_PRIORITY);
        Thread t1 = new Thread(ft1);
        t1.setPriority(Thread.MIN_PRIORITY);
        t1.start();
        t2.start();
        Thread.sleep(2000);
        for (int i = 0; i < 10; i++) {
            System.out.println("Main Thread");
        }
    }
}

```

Output:

0: Child thread = ft2 ← *ft2 thread has MAX_PRIORITY*
 0: Child thread = ft1
 1: Child thread = ft1
 1: Child thread = ft2
 Main Thread
 Main Thread
 Main Thread
 Main Thread

Main Thread
Main Thread
Main Thread
Main Thread
Main Thread
Main Thread
2: Child thread = ft1
2: Child thread = ft2
3: Child thread = ft1
3: Child thread = ft2
4: Child thread = ft1
4: Child thread = ft2
5: Child thread = ft2
5: Child thread = ft1
6: Child thread = ft2
6: Child thread = ft1
7: Child thread = ft2
7: Child thread = ft1
8: Child thread = ft1
8: Child thread = ft2
9: Child thread = ft1
9: Child thread = ft2