

## Thread Priorities:

Every Java Thread has a priority that helps the Operating System to determine the order in which threads are scheduled.

An Operating system thread scheduler determines which thread runs next.

Thread scheduling is platform dependent, the behavior of a multithread program could vary across different platforms.

Threads can be assigned priorities. The priorities are used by the OS to find out which runnable thread is to be given the CPU time.

Priorities are given by priority numbers ranging from 1 to 10.

Predefined priority	value
MIN_PRIORITY	1
NORM_PRIORITY	5
MAX_PRIORITY	10

Thread Scheduler will use priorities while allocating processor. The thread which having highest priority will get the chance first.

If two thread having equal priority, which thread having will get chance first?

We can't expect exact execution order, it depends on Thread Scheduler.

Thread class defines the following methods to get and set priority of your thread.

**public final int getPriority();**

**public final void setPriority(int p);**

Range of p:1-10 other than 1-10, it will give IllegalArgumentException.

The default priority only for main thread is 5, but for all remaining threads default priority will be inherited from parent to child i.e. Whatever priority parent thread has the same priority will be there for the child thread.

```
class MyThread extends Thread
{
}
class Test1 {
public static void main(String args[])
{
System.out.println(Thread.currentThread().getPriority());//5
//Thread.currentThread().setPriority(15);//IllegalArgumentException
Thread.currentThread().setPriority(7);//
MyThread t=new MyThread();//main thread create child thread. who execute this line main()
System.out.println(t.getPriority());//7
}}
```

```
F:\Java Code>javac Test1.java

F:\Java Code>java Test1
5
7
```

MyThread t-----→parent class Thread  
 MyThread t-----→parent Thread main Thread();

The priority of thread can be set using the method:

➤ setPriority(int pnum);

To obtain priority of a thread:

➤ int getPriority();

#### Example:

MyThread m = new MyThread();// has priority of 5.

MyThread m1 = new MyThread(8);// has priority of 8.

The thread with highest priority does not claim all the CPU time, The nature of thread behavior will varies greatly from system to system.

#### Program

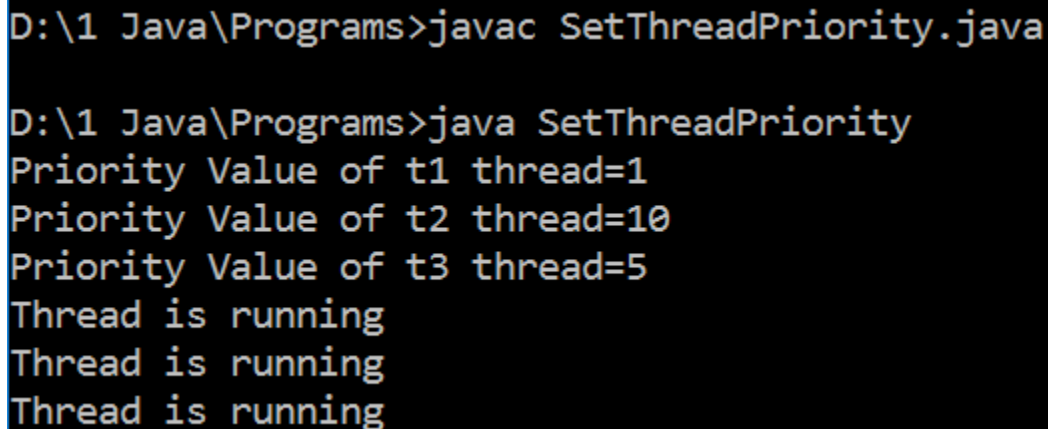
**//This program shows the use of setPriority() and getPriority() method.**

```
class MyThread1 extends Thread {
    public void run() {
        System.out.println("Thread is running");
    }
}
class SetThreadPriority {
    public static void main(String args[]) {
        MyThread1 t1=new MyThread1();//by default priority is 5
        MyThread1 t2=new MyThread1();//by default priority is 5
        MyThread1 t3=new MyThread1();//by default priority is 5
        t1.setPriority(Thread.MIN_PRIORITY);//1
        t2.setPriority(Thread.MAX_PRIORITY);//10
        //t1.setPriority(1);
```

```

//t2.setPriority(10);
System.out.println("Priority Value of t1 thread="+t1.getPriority());
    System.out.println("Priority Value of t2 thread="+t2.getPriority());
    System.out.println("Priority Value of t3 thread="+t3.getPriority());
t1.start();
    t2.start();
    t3.start();
}    }

```



```

D:\1 Java\Programs>javac SetThreadPriority.java

D:\1 Java\Programs>java SetThreadPriority
Priority Value of t1 thread=1
Priority Value of t2 thread=10
Priority Value of t3 thread=5
Thread is running
Thread is running
Thread is running

```

**// This program shows multithreading between two threads with default priority.**

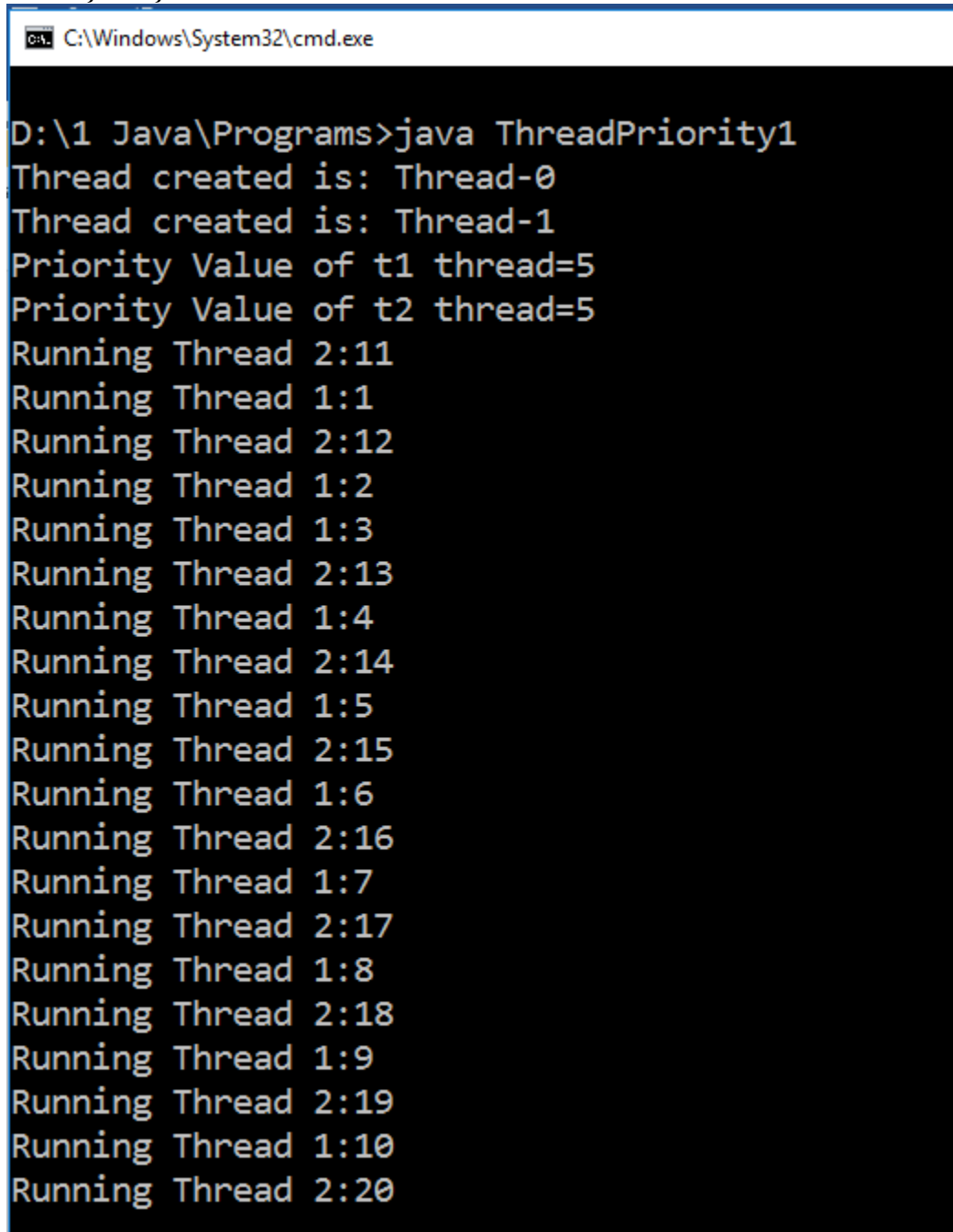
```

class MyThread1 extends Thread {
public void run() {
    for(int i=1;i<=10;i++){
        System.out.println("Running Thread 1:"+i);}
} }
class MyThread2 extends Thread {
public void run() {
    for(int i=1;i<=20;i++){
        System.out.println("Running Thread 2:"+i);}
} }

class ThreadPriority1 {
    public static void main(String args[]) {
        MyThread1 t1=new MyThread1();
        MyThread2 t2=new MyThread2();
        System.out.println("Thread created is: "+t1.getName());
    }
}

```

```
System.out.println("Thread created is: "+t2.getName());
System.out.println("Priority Value of t1 thread="+t1.getPriority());
System.out.println("Priority Value of t2 thread="+t2.getPriority());
t1.start();
t2.start();
} }
```



```
C:\Windows\System32\cmd.exe

D:\1 Java\Programs>java ThreadPriority1
Thread created is: Thread-0
Thread created is: Thread-1
Priority Value of t1 thread=5
Priority Value of t2 thread=5
Running Thread 2:11
Running Thread 1:1
Running Thread 2:12
Running Thread 1:2
Running Thread 1:3
Running Thread 2:13
Running Thread 1:4
Running Thread 2:14
Running Thread 1:5
Running Thread 2:15
Running Thread 1:6
Running Thread 2:16
Running Thread 1:7
Running Thread 2:17
Running Thread 1:8
Running Thread 2:18
Running Thread 1:9
Running Thread 2:19
Running Thread 1:10
Running Thread 2:20
```

**// This program shows multithreading between two threads with different priority.**

```
class MyThread1 extends Thread {
public void run() {
    for(int i=1;i<=10;i++){
        System.out.println("Running Thread 1:"+i);}
} }
class MyThread2 extends Thread {
public void run() {
    for(int i=11;i<=20;i++){
        System.out.println("Running Thread 2:"+i);}
} }

class ThreadPriority2 {
    public static void main(String args[]) {
        MyThread1 t1=new MyThread1();
        MyThread2 t2=new MyThread2();

        t1.setPriority(1); //t1.Thread.MIN_PRIORITY;
        t2.setPriority(10); //t2.setPriority(Thread.MAX_PRIORITY);

        System.out.println("Thread created is: "+t1.getName());
        System.out.println("Thread created is: "+t2.getName());
        System.out.println("Priority Value of t1 thread="+t1.getPriority());
        System.out.println("Priority Value of t2 thread="+t2.getPriority());
        t1.start();
        t2.start();
    } }
```

```
D:\1 Java\Programs>java ThreadPriority2
Thread created is: Thread-0
Thread created is: Thread-1
Priority Value of t1 thread=1
Priority Value of t2 thread=10
Running Thread 2:11
Running Thread 2:12
Running Thread 2:13
Running Thread 2:14
Running Thread 2:15
Running Thread 2:16
Running Thread 2:17
Running Thread 2:18
Running Thread 2:19
Running Thread 2:20
Running Thread 1:1
Running Thread 1:2
Running Thread 1:3
Running Thread 1:4
Running Thread 1:5
Running Thread 1:6
Running Thread 1:7
Running Thread 1:8
Running Thread 1:9
Running Thread 1:10
```

## Daemon Thread in Java

- **Daemon thread in java** is a service provider thread that provides services to user threads for background supporting tasks. It has no role in life, it provides services to the user thread.
- Its life depends on the mercy of user threads i.e. when all the user threads die, JVM terminates this thread automatically.
- It is a low priority thread.
- There are many java daemon threads running automatically e.g. gc, finalizer etc.
- Type `jconsole` on command prompt : It will show information about the loaded classes, memory usage, running threads etc.

### Methods:

1. **public void setDaemon(boolean status)** : is used to mark the current thread as daemon thread or user thread.
2. **public boolean isDaemon()** : is used to check that current is daemon.

```
class MyThread1 extends Thread {
    public void run() {
        if(Thread.currentThread().isDaemon()){//checking for daemon thread
            System.out.println("daemon thread work");
        }
        else{
            System.out.println("user thread1 work");
        }
    }
}

class MyThread2 extends Thread {
    public void run() {
        System.out.println("user thread2 work");
    }
}

class TestDaemonThread {
    public static void main(String args[]) {
        MyThread1 t1=new MyThread1();
        MyThread2 t2=new MyThread2();
        t1.setDaemon(true);//now t1 is daemon thread
        t1.start();
        t2.start();
    }
}
```

```
D:\1 Java\Programs>javac TestDaemonThread.java
```

```
D:\1 Java\Programs>java TestDaemonThread
```

```
user thread2 work
```

```
daemon thread work
```

```
D:\1 Java\Programs>javac TestDaemonThread.java
```

```
D:\1 Java\Programs>java TestDaemonThread
```

```
daemon thread work
```

```
user thread2 work
```