Java Thread Priority in Multithreading

Difficulty Level : Easy Last Updated : 26 Oct, 2021

As we already know java being completely object-oriented works within a multithreading environment in which thread scheduler assigns the processor to a thread based on the priority of thread. Whenever we create a thread in Java, it always has some priority assigned to it. Priority can either be given by JVM while creating the thread or it can be given by the programmer explicitly.

Priorities in threads is a concept where each thread is having a priority which in layman's language one can say every object is having priority here which is represented by numbers ranging from 1 to 10.

- The default priority is set to 5 as excepted.
- Minimum priority is set to 1.
- Maximum priority is set to 10.

Here 3 constants are defined in it namely as follows:

1. public static int NORM_PRIORITY
2. public static int MIN_PRIORITY
3. public static int MAX_PRIORITY

Let us discuss it with an example to get how internally the work is getting executed. Here we will be using the knowledge gathered above as follows:

- We will use *currentThread()* method to get the name of the current thread. User can also use *setName()* method if he/she wants to make names of thread as per choice for understanding purposes.
- *getName()* method will be used to get the name of the thread.

The accepted value of priority for a thread is in the range of 1 to 10.

Let us do discuss how to get and set priority of a thread in java.

1. **public final int getPriority():** java.lang.Thread.getPriority() method returns priority of given thread.

2. **public final void setPriority(int newPriority):** java.lang.Thread.setPriority() method changes the priority of thread to the value newPriority. This method throws IllegalArgumentException if value of parameter newPriority goes beyond minimum(1) and maximum(10) limit.

Example

```
// Java Program to Illustrate Priorities in Multithreading
// via help of getPriority() and setPriority() method

// Importing required classes
import java.lang.*;

// Main class
class ThreadDemo extends Thread {

    // Method 1
    // run() method for the thread that is called
    // as soon as start() is invoked for thread in main()
    public void run()
    {
        // Print statement
        System.out.println("Inside run method");
    }

    // Main driver method
    public static void main(String[] args)
    {
        // Creating random threads
        // with the help of above class
        ThreadDemo t1 = new ThreadDemo();
        ThreadDemo t2 = new ThreadDemo();
        ThreadDemo t3 = new ThreadDemo();

        // Thread 1
        // Display the priority of above thread
        // using getPriority() method
        System.out.println("t1 thread priority : "
                           + t1.getPriority());

        // Thread 1
        // Display the priority of above thread
        System.out.println("t2 thread priority : "
                           + t2.getPriority());

        // Thread 3
        System.out.println("t3 thread priority : "
                           + t3.getPriority());
```

```
        // Setting priorities of above threads by
        // passing integer arguments
        t1.setPriority(2);
        t2.setPriority(5);
        t3.setPriority(8);

        // t3.setPriority(21); will throw
        // IllegalArgumentException

        // 2
        System.out.println("t1 thread priority : "
                           + t1.getPriority());

        // 5
        System.out.println("t2 thread priority : "
                           + t2.getPriority());

        // 8
        System.out.println("t3 thread priority : "
                           + t3.getPriority());

        // Main thread

        // Displays the name of
        // currently executing Thread
        System.out.println(
            "Currently Executing Thread : "
            + Thread.currentThread().getName());

        System.out.println(
            "Main thread priority : "
            + Thread.currentThread().getPriority());

        // Main thread priority is set to 10
        Thread.currentThread().setPriority(10);

        System.out.println(
            "Main thread priority : "
            + Thread.currentThread().getPriority());
    }
}
```

Output

```
t1 thread priority : 5
t2 thread priority : 5
t3 thread priority : 5
t1 thread priority : 2
t2 thread priority : 5
t3 thread priority : 8
```

```
Currently Executing Thread : main
Main thread priority : 5
Main thread priority : 10
```

Output explanation:

- Thread with the highest priority will get an execution chance prior to other threads. Suppose there are 3 threads t1, t2, and t3 with priorities 4, 6, and 1. So, thread t2 will execute first based on maximum priority 6 after that t1 will execute and then t3.
- The default priority for the main thread is always 5, it can be changed later. The default priority for all other threads depends on the priority of the parent thread.

Now geeks you must be wondering out what if we do assign the same priorities to threads than what will happen. All the processing in order to look after threads is carried with help of the thread scheduler. One can refer to the below example of what will happen if the priorities are set to the same and later onwards we will discuss it as an output explanation to have a better understanding conceptually and practically.

Example

```
// Java program to demonstrate that a Child thread
// Getting Same Priority as Parent thread

// Importing all classes from java.lang package
import java.lang.*;

// Main class
// ThreadDemo
// Extending Thread class
class GFG extends Thread {

    // Method 1
    // run() method for the thread that is
    // invoked as threads are started
    public void run()
    {
        // Print statement
        System.out.println("Inside run method");
    }

    // Method 2
    // Main driver method
    public static void main(String[] args)
```

```
    {
        // main thread priority is set to 6 now
        Thread.currentThread().setPriority(6);

        // Current thread is accessed
        // using currentThread() method

        // Print and display main thread priority
        // using getPriority() method of Thread class
        System.out.println(
            "main thread priority : "
            + Thread.currentThread().getPriority());

        // Creating a thread by creating object inside
        // main()
        GFG t1 = new GFG();

        // t1 thread is child of main thread
        // so t1 thread will also have priority 6

        // Print and display priority of current thread
        System.out.println("t1 thread priority : "
                           + t1.getPriority());
    }
}
```

Output

```
main thread priority : 6
t1 thread priority : 6
```

Output explanation:

- If two threads have the same priority then we can't expect which thread will execute first. It depends on the thread scheduler's algorithm(Round-Robin, First Come First Serve, etc)
- If we are using thread priority for thread scheduling then we should always keep in mind that the underlying platform should provide support for scheduling based on thread priority.

All this processing is been carried over with the help of a thread scheduler which can be better visualized withthe help of a video sample been provided below as follows:

This article is contributed by **Dharmesh Singh**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.