Daemon Thread in Java

Difficulty Level : Medium Last Updated : 07 Dec, 2021

Daemon thread in Java is a low-priority thread that runs in the background to perform tasks such as garbage collection. Daemon thread in Java is also a service provider thread that 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.

In simple words, we can say that it provides services to user threads for background supporting tasks. It has no role in life other than to serve user threads.

Example of Daemon Thread in Java: Garbage collection in Java (gc), finalizer, etc.

Properties of Java Daemon Thread

- They can not prevent the JVM from exiting when all the user threads finish their execution.
- JVM terminates itself when all user threads finish their execution.
- If JVM finds a running daemon thread, it terminates the thread and, after that, shutdown it. JVM does not care whether the Daemon thread is running or not.
- It is an utmost low priority thread.

Default Nature of Daemon Thread

By default, the main thread is always non-daemon but for all the remaining threads, daemon nature will be inherited from parent to child. That is, if the parent is Daemon, the child is also a Daemon and if the parent is a non-daemon, then the child is also a non-daemon.

> ***Note:*** *Whenever the last non-daemon thread terminates, all the daemon threads will be ter minated automatically.*

Methods of Daemon Thread

1. void setDaemon(boolean status):
This method marks the current thread as a daemon thread or user thread. For example, if I have a user thread tU then tU.setDaemon(true) would make it a Daemon thread. On the other hand, if I have a Daemon thread tD then calling tD.setDaemon(false) would make it a user thread.

Syntax:

```
public final void setDaemon(boolean on)
```

Parameters:

- **on:** If true, marks this thread as a daemon thread.

Exceptions:

- **IllegalThreadStateException**: if only this thread is active.
- **SecurityException:** if the current thread cannot modify this thread.

2. boolean isDaemon():

This method is used to check that the current thread is a daemon. It returns true if the thread is Daemon. Else, it returns false.

Syntax:

```
public final boolean isDaemon()
```

Returns:

This method returns true if this thread is a daemon thread; false otherwise

---

```
// Java program to demonstrate the usage of
// setDaemon() and isDaemon() method.

public class DaemonThread extends Thread
{
    public DaemonThread(String name){
        super(name);
    }

    public void run()
    {
        // Checking whether the thread is Daemon or not
        if(Thread.currentThread().isDaemon())
        {
            System.out.println(getName() + " is Daemon thread");
        }

        else
        {
```

```
            System.out.println(getName() + " is User thread");
        }
    }

    public static void main(String[] args)
    {

        DaemonThread t1 = new DaemonThread("t1");
        DaemonThread t2 = new DaemonThread("t2");
        DaemonThread t3 = new DaemonThread("t3");

        // Setting user thread t1 to Daemon
        t1.setDaemon(true);

        // starting first 2 threads
        t1.start();
        t2.start();

        // Setting user thread t3 to Daemon
        t3.setDaemon(true);
        t3.start();
    }
}
```

Output:

```
t1 is Daemon thread
t3 is Daemon thread
t2 is User thread
```

Exceptions in a Daemon thread

If you call the setDaemon() method after starting the thread, it would throw **IllegalThreadStateException**.

```
// Java program to demonstrate the usage of
// exception in Daemon() Thread

public class DaemonThread extends Thread
{
    public void run()
```

```
    {
        System.out.println("Thread name: " + Thread.currentThread().getName())
        System.out.println("Check if its DaemonThread: "
                        + Thread.currentThread().isDaemon());
    }

    public static void main(String[] args)
    {
        DaemonThread t1 = new DaemonThread();
        DaemonThread t2 = new DaemonThread();
        t1.start();

        // Exception as the thread is already started
        t1.setDaemon(true);

        t2.start();
    }
}
```

Runtime exception:

```
Exception in thread "main" java.lang.IllegalThreadStateException
    at java.lang.Thread.setDaemon(Thread.java:1352)
    at DaemonThread.main(DaemonThread.java:19)
```

Output:

```
Thread name: Thread-0
Check if its DaemonThread: false
```

This clearly shows that we cannot call the setDaemon() method after starting the thread.

Daemon vs. User Threads

1. **Priority:** When the only remaining threads in a process are daemon threads, the interpreter exits. This makes sense because when only daemon threads remain, there is no other thread for which a daemon thread can provide a service.
2. **Usage:** Daemon thread is to provide services to user thread for background supporting task.

This article is contributed by **Saket Kumar**. If you like GeeksforGeeks and would like to contribute, you can 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.