Object - Oriented Programming 2

Week 4: Multithreading In Java(Sleeping Thread in Java, Naming Thread in Java, Thread Priority in Java and Daemon Thread in Java

By Elubu Joseph - MSc.IS

Lecturer

Department of Information Technology

Kumi University

Email: josebulinda@gmail.com

jose@kumiuniversity.ac.ug

Agenda

1. Sleeping Thread in Java,
2. Naming Thread in Java,
3. Thread Priority in Java and
4. Daemon Thread in Java

Sleeping Thread in Java

Sleeping Thread in Java,

Sleeping a thread, is an act of delaying a thread execution for a specified period of time. To sleep a thread for a specified time, Java provides **sleep()** method which is defined in Thread class. The sleep() method is an overloaded method which versions are given below. It throws **InterruptedException** so make sure to provide proper handler.

It always pause the current thread execution. Any other thread can interrupt the current thread in sleep, in that case InterruptedException is thrown.

Syntax

```
sleep(long millis)throws InterruptedException
sleep(long millis, int nanos)throws InterruptedException
```

Example1 : Sleeping a thread

In this example, we are sleeping threads by using sleep() method. Each thread will sleep for 1000 milliseconds and then resume its execution. See the below example

```
public class SleepingThread extends Thread {
        SleepingThread(String s){ super(s); }
        public void run(){
        System.out.println(Thread.currentThread().getName()+" Started");
        try{
         SleepingThread.sleep(1000);
        }
        catch(InterruptedException ie){
        System.out.println(ie);
        }
        System.out.println(Thread.currentThread().getName()+" Finished");
        }
        public static void main(String[] args){
        SleepingThread ob=new SleepingThread("First Thread");
        SleepingThread ob1=new SleepingThread("Second Thread");
        ob.start();
        ob1.start();
        }
}
```

Example1 : Sleeping a thread-Output

```
1  package Multithreading;
2  public class SleepingThread extends Thread{
3  SleepingThread(String s){ super(s); }
4  public void run(){
5  System.out.println(Thread.currentThread().getName()+"Started")
6  try{
7   SleepingThread.sleep(1000);
8  }
9  catch(InterruptedException ie){
10 System.out.println(ie);}
11 System.out.println(Thread.currentThread().getName()+"Finished"
12          }
13      public static void main(String[] args){
14 SleepingThread ob=new SleepingThread("First Thread");
15 SleepingThread ob1=new SleepingThread("Second Thread");
16 ob.start();
17 ob1.start(); }
18 }
```

```
run:
First ThreadStarted
Second ThreadStarted
First ThreadFinished
Second ThreadFinished
```

Example2 : Sleeping and getting the state of a thread

```
public class SleepingThread1 extends Thread {
        SleepingThread1(String s){ super(s); }
        public void run() {
                System.out.println(Thread.currentThread().getName()+" Started");
        try{
                SleepingThread1.sleep(1500);
                System.out.println(Thread.currentThread().getName()+" Sleeping..");
        }catch(InterruptedException ie){
                System.out.println(ie);
        }
        System.out.println(Thread.currentThread().getName()+" Finished");
        }
        public static void main(String[] args) {
        SleepingThread1 ob1=new SleepingThread1("First thread");
        SleepingThread1 ob2=new SleepingThread1("Second thread");
        System.out.println(ob1.getName()+" State: "+ ob1.getState());
        ob1.start();
        System.out.println(ob1.getName()+" State: "+ ob1.getState());
        System.out.println(ob2.getName()+" State: "+ ob2.getState());
        ob2.start();
        System.out.println(ob2.getName()+" State: "+ ob2.getState());
} }
```

Example2 : Sleeping a thread-Output

```
1  package Multithreading;
2  public class SleepingThread1 extends Thread {
       SleepingThread1(String s){ super(s); }
           public void run() {
5          System.out.println(Thread.currentThread().getName()+" Started");
6          try{
7          SleepingThread1.sleep(1500);
8          System.out.println(Thread.currentThread().getName()+" Sleeping..");
9      }catch(InterruptedException ie){
10         System.out.println(ie); }
11         System.out.println(Thread.currentThread().getName()+" Finished");
12         }
13     public static void main(String[] args) {
14         SleepingThread1 ob1=new SleepingThread1("First Thread");
15         SleepingThread1 ob2=new SleepingThread1("Second Thread");
16  System.out.println(ob1.getName()+" State: "+ ob1.getState()); ob1.start();
17         System.out.println(ob1.getName()+" State: "+ ob1.getState());
18  System.out.println(ob2.getName()+" State: "+ ob2.getState()); ob2.start();
19     System.out.println(ob2.getName()+" State: "+ ob2.getState());
20 }
21 }
```

```
run:
First Thread State: NEW
First Thread State: RUNNABLE
Second Thread State: NEW
Second Thread State: RUNNABLE
First Thread Started
Second Thread Started
First Thread Sleeping..
First Thread Finished
Second Thread Sleeping..
Second Thread Finished
```

Explanation

SleeptingThread1.sleep() method interacts with the thread scheduler to put the current thread in wait state for specified period of time. Once the wait time is over, thread state is changed to runnable state and wait for the CPU for further execution.

So the actual time that current thread sleep depends on the thread scheduler that is part of operating system.

Naming Thread in Java

Each thread in Java has its own name which is set by the JVM by default. Although there are many other attributes associated to the thread like: id, priority etc.

we can get name of a thread by calling getName() method of Thread class.

If we wish to set new name of the thread then **setName()** method can be used. Both methods belongs to Thread class and even we can set name of thread by passing into constructor during creating object. Lets understand by the examples

Naming Thread in Java+.

Syntax of the two methods is given below

```
setName(String name)
getName()
```

Example: Fetch Default Thread Name

lets first fetch the thread name set by the JVM. It is default name so initially we cannot predict it.

```
public class ThreadName extends Thread {
      public void run() {
      System.out.println("Thread running...");
      }
      public static void main(String[] args) {
            ThreadName tn=new ThreadName();
            tn.start();
      System.out.println("Thread name: "+tn.getName());
      }
}
```

Example: Fetch Thread Name-Output

```
1   package Multithreading;
2   public class ThreadName extends Thread {
3           public void run() {
4           System.out.println("Thread running...");
5           }
6           public static void main(String[] args) {
7                   ThreadName tn=new ThreadName();
8                   tn.start();
9           System.out.println("Thread name: "+tn.getName());
0           }
1   }
```

we can also use static currentThread() method of Thread class. it returns thread name, priority etc. it is static method so object is not require to call it. See the below example.

```
run:
Thread running...
Thread name: Thread-0
```

Example: Thread.currentThread()

In this example, we are using currentThread method of Thread class to get name of the thread.

```
public class ThreadName1 extends Thread {
     public void run() {
     System.out.println("Thread running...");
     System.out.println("Thread name:
"+Thread.currentThread());
     }
     public static void main(String[] args) {
          ThreadName1 ob=new ThreadName1();
          ob.start();
     }
}
```

Example: Thread.currentThread()- Output

```
1   package Multithreading;
2   public class ThreadName1 extends Thread {
3           public void run() {
4           System.out.println("Thread running...");
5           System.out.println("Thread name:"
6                   + " "+Thread.currentThread());
7           }
8           public static void main(String[] args) {
9                   ThreadName1 ob=new ThreadName1();
10                  ob.start();
11          }
12  }
```

```
run:
Thread running...
Thread name: Thread[Thread-0,5,main]
```

Example 1: Setting Thread Name Using setName()

In this example, we are using setName() method to set name of the thread and getName() method to get name of the thread.

```
public class setThreadName extends Thread {
      public void run() {
            System.out.println("Thread running...");
      }
      public static void main(String[] args){
      setThreadName stn=new setThreadName();
      stn.start();
      stn.setName("OOPs2");
      System.out.println("Thread Name: "+ stn.getName());
      }
}
```

Example 1: Setting Thread Name-Output

```
1   package Multithreading;
2   public class setThreadName extends Thread {
3       public void run() {
4           System.out.println("Thread running...");
5       }
6       public static void main(String[] args){
7           setThreadName stn=new setThreadName();
8           stn.start();
9           stn.setName("OOPs2");
10          System.out.println("Thread Name: "+ stn.getName());
11      }
12  }
```

```
run:
Thread running...
Thread Name: OOPs2
```

Example 2 : Setting Thread Name using constructor

We can set name of a thread by passing through the constructor. Thread class constructor that allows a string type argument can be used to set name. See the below example.

```
public class setThreadName1 extends Thread {
        setThreadName1(String name){ super(name);
        }
        public void run() {
        System.out.println("Thread running...");
        }
        public static void main(String[] args) {
                setThreadName1 stn=new setThreadName1("KUMU Thread");
                stn.start();
                System.out.println("Thread Name: "+ stn.getName());
        }
}
```

Since we are extending the Thread class so we call its constructor through the super call and passed thread name to the setThreadName1 class.

Example 2 : Setting Name using constructor-Output

```
1   package Multithreading;
2   public class setThreadName1 extends Thread {
3       setThreadName1(String name){
4           super(name);
5       }
6       public void run() {
7           System.out.println("Thread running...");
8       }
9       public static void main(String[] args) {
10          setThreadName1 stn=new setThreadName1("KUMU Thread");
11          stn.start();
12          System.out.println("Thread Name: "+ stn.getName());
13      }
14  }
```

```
run:
Thread running...
Thread Name: KUMU Thread
```

Example 3: Setting Thread Name By Directly passing the name to the Thread class constructor

We can directly pass the thread name to the constructor by creating thread class object.

```
public class setThreadName2 implements Runnable {
      public void run() {
      System.out.println("Thread running...");
      }
      public static void main(String[] args) {
      setThreadName2 ob=new setThreadName2();
      Thread tc = new Thread(ob, "KUMU Thread");
      tc.start();
      System.out.println("Thread Name: "+tc.getName());
}
}
```

Example 3: Setting Thread Name By Directly passing the name to the Thread class constructor-Output

```
1   package Multithreading;
2   public class setThreadName2 implements Runnable {
        public void run() {
4           System.out.println("Thread running...");
5       }
6       public static void main(String[] args) {
7           setThreadName2 ob=new setThreadName2();
8           Thread tc = new Thread(ob, "KUMU Thread");
9           tc.start();
10          System.out.println("Thread Name: "+tc.getName());
11      }
12  }
```

Congratulaions, we learned to fetch thread name using getName(), currentThread() methods, and to set thread name using setName(), passing thread name to the constructor of the thread class method.

```
run:
Thread running...
Thread Name: KUMU Thread
```

Thread Priority in Java

Thread Priority in Java

Priority of a thread describes how early it gets execution and selected by the thread scheduler. In Java, when we create a thread, always a priority is assigned to it.

In a Multithreading environment, the processor assigns a priority to a thread scheduler. The priority is given by the JVM or by the programmer itself explicitly.

The range of the priority is between **1** to **10** and there are **three constant variables** which are **static and used to fetch priority** of a Thread as below:-

Thread Priority in Java + Types of priority

Java provide three kinds of priority options that can be applied to threads, as below;

1. public static **int MIN_PRIORITY**; It holds the minimum priority that can be given to a thread. The value for this is 1.

2. public static int **NORM_PRIORITY**; It is the default priority that is given to a thread if it is not defined, the value for this is 0.

3. public static int **MAX_PRIORITY**; It is the maximum priority that can be given to a thread. The value for this is 10. (Studytonight.com. (n.d.).

Get and Set methods in Thread priority

1. **public final int getPriority();** In Java, getPriority() method is found in java.lang.Thread package. it is used to get the priority of a thread.
2. **public final void setPriority(intnewPriority);** In Java setPriority(intnewPriority) method is in java.lang.Thread package. It is used to set the priority of a thread. The setPriority() method throws IllegalArgumentException if the value of new priority is above minimum and maximum limit.

Example: Fetch Default Thread Priority

If we don’t set priority of a thread then the JVM sets it by default. In this example, we are getting thread’s default priority by using the getPriority() method.

Example: Fetch Thread Priority+

```
class getThreadPrio extends Thread {
	public void run() {
	System.out.println("Thread Running...");
	}
	public static void main(String[]args) {
		getThreadPrio p1 = new getThreadPrio();
		getThreadPrio p2 = new getThreadPrio();
		getThreadPrio p3 = new getThreadPrio();
		p1.start();
		System.out.println("First thread priority : " +
		p1.getPriority());
	System.out.println("Second thread priority : " + p2.getPriority());
	System.out.println("Third thread priority : " + p3.getPriority());
	}
}
```

Example: Fetch Thread Priority++ Output

```
1   package Multithreading;
2   class getThreadPrio extends Thread {
4           public void run() {
5           System.out.println("Thread Running...");
            }
6       public static void main(String[]args) {
7           getThreadPrio p1 = new getThreadPrio();
8           getThreadPrio p2 = new getThreadPrio();
9           getThreadPrio p3 = new getThreadPrio();
10          p1.start();
11          System.out.println("First thread priority : " + p1.getPriority());
12          System.out.println("Second thread priority : " + p2.getPriority());
13          System.out.println("Third thread priority : " + p3.getPriority());
            }
15  }
```

```
run:
Thread Running...
First thread priority : 5
Second thread priority : 5
Third thread priority : 5
```

Example: Fetch Thread priority based on Constants

We can fetch priority of a thread by using some predefined constants provided by the Thread class. these constants returns the max, min and normal priority of a thread.

```
class getThreadPrio1 extends Thread {
      public void run() {
      System.out.println("Thread Running...");
      }
      public static void main(String[]args) {
      getThreadPrio1 p1 = new getThreadPrio1();
      p1.start();
      System.out.println("Max thread priority : " + p1.MAX_PRIORITY);
      System.out.println("Min thread priority : " + p1.MIN_PRIORITY);
      System.out.println("Normal thread priority : " + p1.NORM_PRIORITY);
      }
}
```

Example: Fetch Thread priority based on **Constants-Output**

```
1   package Multithreading;
2   class getThreadPrio1 extends Thread {
3           public void run() {
4           System.out.println("Thread Running...");
5           }
6           public static void main(String[]args) {
7           getThreadPrio1 p1 = new getThreadPrio1();
8           p1.start();
            System.out.println("Max thread priority : " + p1.MAX_PRIORITY);
            System.out.println("Min thread priority : " + p1.MIN_PRIORITY);
            System.out.println("Normal thread priority : " + p1.NORM_PRIORITY);
12          }
13  }
```

```
run:
Thread Running...
Max thread priority : 10
Min thread priority : 1
Normal thread priority : 5
```

Example 1: Set Thread Priority

To set priority of a thread, setPriority() method of thread class is used. This method takes an integer argument that must be between 1 and 10. see the below example.

```
class setThreadPrio extends Thread {
      public void run() {
      System.out.println("Thread Running...");
      }
      public static void main(String[]args) {
       setThreadPrio p1 = new setThreadPrio();
      p1.start();
      p1.setPriority(3);
      int p = p1.getPriority();
      System.out.println("Thread priority : " + p);
      }
}
```

Example 1 : Set Thread Priority-Output

```
1   package Multithreading;
2   class setThreadPrio extends Thread {
3           public void run() {
4           System.out.println("Thread Running...");
5           }
6           public static void main(String[]args) {
7           setThreadPrio p1 = new setThreadPrio();
8           p1.start();
9           p1.setPriority(3); // Seting priority
10          int p = p1.getPriority();
11          System.out.println("Thread priority : " + p);
12          }
13  }
```

```
run:
Thread Running...
Thread priority : 3
```

Example2 : Set Thread Priority

In this example, we are setting priority of two thread and running them to see the effect of thread priority.

Does setting higher priority thread get CPU first. See the below example.

```
class setThreadPrio1 extends Thread {
      public void run() {
      System.out.println("Thread Running...
                        "+Thread.currentThread().getName()); }
      public static void main(String[]args) {
      setThreadPrio1 p1 = new setThreadPrio1();
      setThreadPrio1 p2 = new setThreadPrio1(); // Starting thread
      p1.start();
      p2.start(); // Setting priority
      p1.setPriority(2);
      p2.setPriority(1); // Getting -priority
      int p = p1.getPriority();
      int q = p2.getPriority();
      System.out.println(" First thread priority : " + p);
      System.out.println(" Second thread priority : " + q); } }
```

Example2 : Set Thread Priority-Output

```
1  package Multithreading;
2  class setThreadPrio1 extends Thread {
3      public void run() {
4          System.out.println("Thread Running..."
5                  + ""+Thread.currentThread().getName()); }
6          public static void main(String[]args) {
7          setThreadPrio1 p1 = new setThreadPrio1();
8          setThreadPrio1 p2 = new setThreadPrio1();
9              p1.start();
10             p2.start(); // Setting priority
11             p1.setPriority(2);
12             p2.setPriority(1); // Getting -priority
13             int p = p1.getPriority();
14             int q = p2.getPriority();
15         System.out.println("First thread priority : " + p);
16         System.out.println("Second thread priority : " + q);
17         }
18 }
```

```
run:
Thread Running...Thread-1
Thread Running...Thread-0
First thread priority : 2
Second thread priority : 1
```

Note: Thread priorities cannot guarantee that a higher priority thread will always be executed first than the lower priority thread. The selection of the threads for execution depends upon the thread scheduler which is platform dependent.

Daemon Thread in Java

Daemon threads are a low priority thread that provide supports to user threads. These threads can be user defined and system defined as well. Garbage collection thread is one of the system generated daemon thread that runs in background. They perform tasks such as garbage collection.

Daemon thread does allow JVM from existing until all the threads finish their execution. When a JVM founds daemon threads it terminates the thread and then shutdown itself, (Studytonight.com, n.d.).

Following are the methods in Daemon Thread

1. void setDaemon(boolean check**)**

this method is used to create the current thread as a daemon thread or user thread. If there is a user thread as obj1 then obj1.setDaemon(true) will make it a Daemon thread and if there is a Daemon thread obj2 then calling obj2.setDaemon(false) will make it a user thread.

Syntax

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

Following are the methods in Daemon Thread+

2. boolean isDaemon()

this method is used to check whether the current thread is a daemon or not.

It returns true if the thread is Daemon otherwise it returns false.

Syntax:

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

Example: setting a thread to be Daemon

Lets use this example to create daemon and user threads. To create daemon thread setDaemon() method is used. It takes boolean value either true or false.

```
public class DaemonPro1 extends Thread{
	public DaemonPro1(String n){
		super(n);
	}
	public void run(){
	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) {
	DaemonPro1 D1 = new DaemonPro1("D1");
	DaemonPro1 D2 = new DaemonPro1("D2");
	DaemonPro1 D3 = new DaemonPro1("D3");
	D1.setDaemon(true);
	D1.start();
	D2.start();
	D3.setDaemon(true);
	D3.start();
}}
```

Example: setting a thread to be Daemon - Output

```
1   package Multithreading;
2   public class DaemonPro1 extends Thread{
3       public DaemonPro1(String n){
4           super(n);
5       }
6       public void run(){
7       if(Thread.currentThread().isDaemon()){
8           System.out.println(getName() + " is Daemon thread");
9       }else{
10          System.out.println(getName() + " is User thread");
11      }
12      }
13      public static void main(String[] args) {
14      DaemonPro1 D1 = new DaemonPro1("D1");
15      DaemonPro1 D2 = new DaemonPro1("D2");
16      DaemonPro1 D3 = new DaemonPro1("D3");
17      D1.setDaemon(true);
18      D1.start(); D2.start(); D3.setDaemon(true); D3.start();
19  }}
```

Note thate D2 thread was started without using setDaemon() method.

```
run:
D1 is Daemon thread
D3 is Daemon thread
D2 is User thread
```

Example : Daemon thread Priority

Since Daemon threads are low level threads then we need to check the priority of these threads. The priority we are getting is set by the JVM.

```
public class DaemonPro2 extends Thread {
        public DaemonPro2(String name) {
                super(name);
        }
        public void run(){
        if(Thread.currentThread().isDaemon()) {
        System.out.println(getName() + " is Daemon thread");
        }
        else {
        System.out.println(getName() + " is User thread");
        }
        System.out.println(getName()+" priority
        "+Thread.currentThread().getPriority());
        }
        public static void main(String[] args) {
        DaemonPro2 D1 = new DaemonPro2("D1");
        DaemonPro2 D2 = new DaemonPro2("D2");
        DaemonPro2 D3 = new DaemonPro2("D3");
        D1.setDaemon(true);
        D1.start(); D2.start();
        D3.setDaemon(true);
        D3.start();
} }
```

(Studytonight.com, n.d.).

Example : Daemon thread Priority-Output

```
1  package Multithreading;
2  public class DaemonPro2 extends Thread {
3      public DaemonPro2(String name) {
4          super(name);
5      }
6      public void run(){
7      if(Thread.currentThread().isDaemon()) {
8          System.out.println(getName() + " is Daemon thread");
9      }
10     else {
11         System.out.println(getName() + " is User thread");
12     }
13     System.out.println(getName()+" priority
14     }
15     public static void main(String[] args) {
16         DaemonPro2 D1 = new DaemonPro2("D1");
17         DaemonPro2 D2 = new DaemonPro2("D2");
18         DaemonPro2 D3 = new DaemonPro2("D3");
19         D1.setDaemon(true);
20         D1.start(); D2.start();
21         D3.setDaemon(true);
22         D3.start();
23     }
24 }
```

```
run:
D2 is User thread
D1 is Daemon thread
D3 is Daemon thread
D3 priority                    5
D2 priority                    5
D1 priority                    5
```

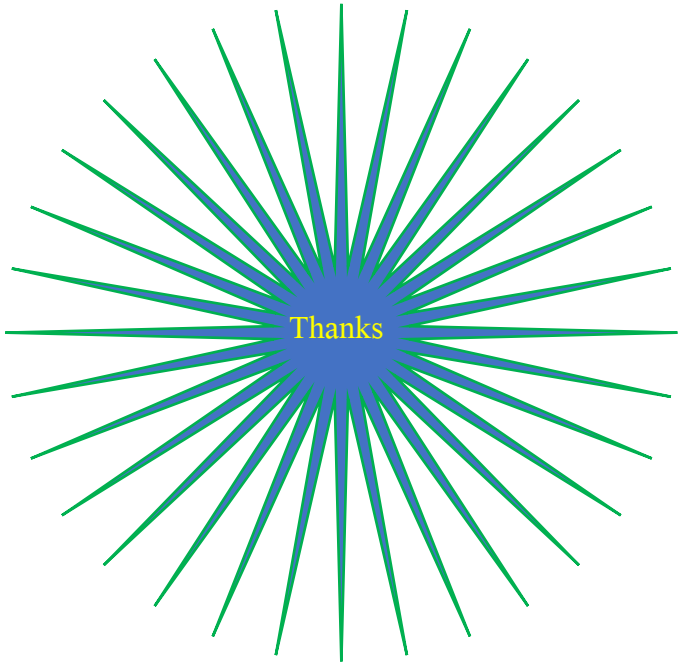
Beware of Error in Program flow.

While creating daemon thread make sure the setDaemon() is called before starting of the thread. Calling it after starting of thread will throw an exception and terminate the program execution.

Summary

1. Sleeping Thread in Java,
2. Naming Thread in Java,
3. Thread Priority in Java and
4. Daemon Thread in Java

Thank you for Listening



References

Java™ Network Programming and Distributed Computing, (David R.,Michael R. 2002), Publisher : Addison Wesley; ISBN: 0201710374

Java thread class. Studytonight.com. (n.d.). Retrieved September 23, 2022, from https://www.studytonight.com/java/thread-class-and-functions.php

Java thread priorities. Studytonight.com. (n.d.). Retrieved September 22, 2022, from https://www.studytonight.com/java/thread-priorities-in-java.php

Java daemon thread. Studytonight.com. (n.d.). Retrieved September 20, 2022, from https://www.studytonight.com/java/daemon-thread-in-java.php

Java naming thread. Studytonight.com. (n.d.). Retrieved September 25, 2022, from https://www.studytonight.com/java/naming-thread-in-java.php