

Naming Thread and Current Thread

Naming Thread

The Thread class provides methods to change and get the name of a thread. By default, each thread has a name, i.e. thread-0, thread-1 and so on. By we can change the name of the thread by using the setName() method. The syntax of setName() and getName() methods are given below:

```
public String getName(): is used to return the name of a thread.
public void setName(String name): is used to change the name of a thread.
```

We can also set the name of a thread directly when we create a new thread using the constructor of the class.

Example of naming a thread : Using setName() Method

FileName: TestMultiNaming1.java

```
class TestMultiNaming1 extends Thread{
    public void run(){
        System.out.println("running...");
    }

    public static void main(String args[]){
        TestMultiNaming1 t1=new TestMultiNaming1();
        TestMultiNaming1 t2=new TestMultiNaming1();
        System.out.println("Name of t1:"+t1.getName());
        System.out.println("Name of t2:"+t2.getName());

        t1.start();
        t2.start();

        t1.setName("Sonoo Jaiswal");
        System.out.println("After changing name of t1:"+t1.getName());
}
```

```
Test it Now
```

Output:

```
Name of t1:Thread-0

Name of t2:Thread-1

After changing name of t1:Sonoo Jaiswal running...

running...
```

Example of naming a thread : Without Using setName() Method

One can also set the name of a thread at the time of the creation of a thread, without using the setName() method. Observe the following code.

FileName: ThreadNamingExample.java

```
// A Java program that shows how one can
// set the name of a thread at the time
// of creation of the thread

// import statement
import java.io.*;

// The ThreadNameClass is the child class of the class Thread
class ThreadName extends Thread
{
```

```
// constructor of the class
ThreadName(String threadName)
{
// invoking the constructor of
// the superclass, which is Thread class.
super(threadName);
}
// overriding the method run()
public void run()
{
System.out.println(" The thread is executing....");
}
}
public class ThreadNamingExample
// main method
public static void main (String argvs[])
{
// creating two threads and settting their name
// using the contructor of the class
ThreadName th1 = new ThreadName("JavaTpoint1");
ThreadName th2 = new ThreadName("JavaTpoint2");
// invoking the getName() method to get the names
// of the thread created above
System.out.println("Thread - 1: " + th1.getName());
System.out.println("Thread - 2: " + th2.getName());
// invoking the start() method on both the threads
th1.start();
th2.start();
}
}
```

Output:

```
Thread - 1: JavaTpoint1

Thread - 2: JavaTpoint2

The thread is executing....

The thread is executing....
```

Current Thread

The currentThread() method returns a reference of the currently executing thread.

```
public static Thread currentThread()
```

Example of currentThread() method

FileName: TestMultiNaming2.java

```
class TestMultiNaming2 extends Thread{
public void run(){
   System.out.println(Thread.currentThread().getName());
}
public static void main(String args[]){
   TestMultiNaming2 t1=new TestMultiNaming2();
   TestMultiNaming2 t2=new TestMultiNaming2();

t1.start();
t2.start();
}
}
```

☑ Test it Now

Output:

Thread-0

Thread-1



 $Next \rightarrow$

Youtube For Videos Join Our Youtube Channel: Join Now

Feedback

• Send your Feedback to feedback@javatpoint.com

Help Others, Please Share









Learn Latest Tutorials









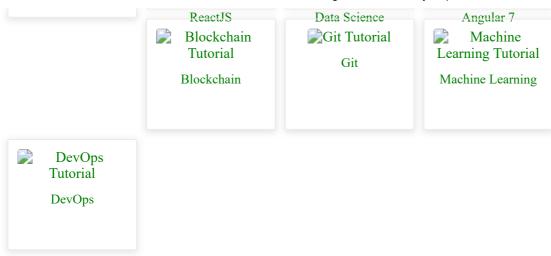


Preparation



Trending Technologies





B.Tech / MCA



Naming a thread in Java- javatpoint

