JAVA.LANG.THREAD CLASS

http://www.tutorialspoint.com/java/lang/java_lang_thread.htm Copyright © tutorialspoint.com

Introduction

The **java.lang.Thread** class is a thread of execution in a program. The Java Virtual Machine allows an application to have multiple threads of execution running concurrently.Following are the important points about Thread:

- Every thread has a priority. Threads with higher priority are executed in preference to threads with lower priority
- Each thread may or may not also be marked as a daemon.
- There are two ways to create a new thread of execution. One is to declare a class to be a subclass of Thread and,
- the other way to create a thread is to declare a class that implements the Runnable interface

Class declaration

Following is the declaration for **java.lang.Thread** class:

```
public class Thread
  extends Object
   implements Runnable
```

Field

Following are the fields for **java.lang.Thread** class:

- **static int MAX_PRIORITY** -- This is the minimum priority that a thread can have.
- **static int NORM_PRIORITY** -- This is the default priority that is assigned to a thread.

Class constructors

S.N.	Constructor & Description
1	**Thread** This allocates a new Thread object.
2	**Thread***Runnabletarget* This allocates a new Thread object.
3	**Thread***Runnabletarget, Stringname* This allocates a new Thread object.
4	**Thread***Stringname* This constructs allocates a new Thread object.
5	

	Thread*ThreadGroupgroup, Runnabletarget* This allocates a new Thread object.
6	**Thread***ThreadGroupgroup, Runnabletarget, Stringname* This allocates a new Thread object so that it has target as its run object, has the specified name as its name, and belongs to the thread group referred to by group.
7	**Thread***ThreadGroupgroup, Runnabletarget, Stringname, longstackSize* This allocates a new Thread object so that it has target as its run object, has the specified name as its name, belongs to the thread group referred to by group, and has the specified stack size.
8	**Thread***ThreadGroupgroup, Stringname* This allocates a new Thread object.

Class methods

S.N.	Method & Description
1	static int activeCount This method returns the number of active threads in the current thread's thread group.
2	void checkAccess This method determines if the currently running thread has permission to modify this thread.
3	protected Object clone This method returns a clone if the class of this object is Cloneable.
4	static Thread currentThread This method returns a reference to the currently executing thread object.
5	static void dumpStack This method prints a stack trace of the current thread to the standard error stream.
6	static int enumerate*Thread[]tarray* This method copies into the specified array every active thread in the current thread's thread group and its subgroups.

7	static Map<Thread,StackTraceElement[]> getAllStackTraces This method returns a map of stack traces for all live threads.
8	ClassLoader getContextClassLoader This method returns the context ClassLoader for this Thread.
9	static Thread.UncaughtExceptionHandler getDefaultUncaughtExceptionHandler This method returns the default handler invoked when a thread abruptly terminates due to an uncaught exception.
10	long getId This method returns the identifier of this Thread.
11	String getName This method returns this thread's name.
12	int getPriority This method Returns this thread's priority.
13	StackTraceElement[] getStackTrace This method returns an array of stack trace elements representing the stack dump of this thread.
14	Thread.State getState This method returns the state of this thread.
15	ThreadGroup getThreadGroup This method returns the thread group to which this thread belongs.
16	Thread.UncaughtExceptionHandler getUncaughtExceptionHandler This method returns the handler invoked when this thread abruptly terminates due to an uncaught exception.
17	static boolean holdsLock*Objectobj* This method returns true if and only if the current thread holds the monitor lock on the specified object.
18	

	void interrupt This method interrupts this thread.
19	static boolean interrupted This method tests whether the current thread has been interrupted.
20	boolean isAlive This method tests if this thread is alive.
21	boolean isDaemon This method tests if this thread is a daemon thread.
22	boolean isInterrupted This method tests whether this thread has been interrupted.
23	void join Waits for this thread to die.
24	void join*longmillis* Waits at most millis milliseconds for this thread to die.
25	void join*longmillis, intnanos* Waits at most millis milliseconds plus nanos nanoseconds for this thread to die.
26	void run If this thread was constructed using a separate Runnable run object, then that Runnable object's run method is called; otherwise, this method does nothing and returns
27	void setContextClassLoader*ClassLoadercl* This method sets the context ClassLoader for this Thread.
28	void setDaemon*booleanon* This method marks this thread as either a daemon thread or a user thread.
29	static void setDefaultUncaughtExceptionHandler*Thread. UncaughtExceptionHandlereh* This method set the default handler invoked when a thread abruptly terminates due to an uncaught exception, and no other handler has been defined for that thread.

30

void setName*Stringname*

This method changes the name of this thread to be equal to the argument name.

31

void setPriority*intnewPriority*

This method changes the priority of this thread.

32

void setUncaughtExceptionHandler*Thread. UncaughtExceptionHandlereh*

This method set the handler invoked when this thread abruptly terminates due to an uncaught exception.

33

static void sleep*longmillis*

This method causes the currently executing thread to sleep *temporarilyceaseexecution* for the specified number of milliseconds, subject to the precision and accuracy of system timers and schedulers.

34

static void sleep*longmillis, intnanos*

This method causes the currently executing thread to sleep *ceaseexecution* for the specified number of milliseconds plus the specified number of nanoseconds, subject to the precision and accuracy of system timers and schedulers.

35

void start

This method causes this thread to begin execution; the Java Virtual Machine calls the run method of this thread.

36

String toString

This method Returns a string representation of this thread, including the thread's name, priority, and thread group.

37

static void yield

This method causes the currently executing thread object to temporarily pause and allow other threads to execute.

Methods inherited

This class inherits methods from the following classes:

- java.lang.Object

Loading [MathJax]/jax/output/HTML-CSS/jax.js