java.lang

Class Thread

```
java.lang.Object
  |
  +--java.lang.Thread
```

All Implemented Interfaces:

Runnable

---

public class **Thread**
extends Object
implements Runnable

...

Since:

JDK1.0

See Also:

`Runnable`, `Runtime.exit(int)`, `run()`, `stop()`

Field Summary

`static int`	**`MAX_PRIORITY`** The maximum priority that a thread can have.
`static int`	**`MIN_PRIORITY`** The minimum priority that a thread can have.
`static int`	**`NORM_PRIORITY`** The default priority that is assigned to a thread.

Constructor Summary

`Thread``()` Allocates a new `Thread` object.
`Thread``(Runnable target)` Allocates a new `Thread` object.
`Thread``(Runnable target, String name)` Allocates a new `Thread` object.
`Thread``(String name)` Allocates a new `Thread` object.
`Thread``(ThreadGroup group, Runnable target)` Allocates a new `Thread` object.
`Thread``(ThreadGroup group, Runnable target, String name)` 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`.
`Thread``(ThreadGroup group, String name)` Allocates a new `Thread` object.

Method Summary

`static int`	**`activeCount`**`()` Returns the current number of active threads in this thread's thread group.
`void`	**`checkAccess`**`()` Determines if the currently running thread has permission to modify this thread.
`int`	**`countStackFrames`**`()` **Deprecated.** *The definition of this call depends on `suspend()`, which is deprecated. Further, the results of this call were never well-defined.*
`static Thread`	**`currentThread`**`()` Returns a reference to the currently executing thread object.
`void`	**`destroy`**`()` Destroys this thread, without any cleanup.
`static void`	**`dumpStack`**`()` Prints a stack trace of the current thread.
`static int`	**`enumerate`**`(Thread[] tarray)` Copies into the specified array every active thread in this thread's thread group and its subgroups.
`ClassLoader`	**`getContextClassLoader`**`()` Returns the context ClassLoader for this Thread.
`String`	**`getName`**`()` Returns this thread's name.

int	**getPriority**() Returns this thread's priority.
ThreadGroup	**getThreadGroup**() Returns the thread group to which this thread belongs.
void	**interrupt**() Interrupts this thread.
static boolean	**interrupted**() Tests whether the current thread has been interrupted.
boolean	**isAlive**() Tests if this thread is alive.
boolean	**isDaemon**() Tests if this thread is a daemon thread.
boolean	**isInterrupted**() Tests whether this thread has been interrupted.
void	**join**() Waits for this thread to die.
void	**join**(long millis) Waits at most millis milliseconds for this thread to die.
void	**join**(long millis, int nanos) Waits at most millis milliseconds plus nanos nanoseconds for this thread to die.
void	**resume**() **Deprecated.** *This method exists solely for use with suspend(), which has been deprecated because it is deadlock-prone. For more information, see Why are Thread.stop, Thread.suspend and Thread.resume Deprecated?.*
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.
void	**setContextClassLoader**(ClassLoader cl) Sets the context ClassLoader for this Thread.
void	**setDaemon**(boolean on) Marks this thread as either a daemon thread or a user thread.
void	**setName**(String name) Changes the name of this thread to be equal to the argument name.
void	**setPriority**(int newPriority) Changes the priority of this thread.
static void	**sleep**(long millis) Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds.

static void	**sleep**(long millis, int nanos) Causes the currently executing thread to sleep (cease execution) for the specified number of milliseconds plus the specified number of nanoseconds.
void	**start**() Causes this thread to begin execution; the Java Virtual Machine calls the run method of this thread.
void	**stop**() **Deprecated.** *This method is inherently unsafe. Stopping a thread with Thread.stop causes it to unlock all of the monitors that it has locked (as a natural consequence of the unchecked ThreadDeath exception propagating up the stack). If any of the objects previously protected by these monitors were in an inconsistent state, the damaged objects become visible to other threads, potentially resulting in arbitrary behavior. Many uses of stop should be replaced by code that simply modifies some variable to indicate that the target thread should stop running. The target thread should check this variable regularly, and return from its run method in an orderly fashion if the variable indicates that it is to stop running. If the target thread waits for long periods (on a condition variable, for example), the interrupt method should be used to interrupt the wait. For more information, see Why are Thread.stop, Thread.suspend and Thread.resume Deprecated?.*
void	**stop**(Throwable obj) **Deprecated.** *This method is inherently unsafe. See stop() (with no arguments) for details. An additional danger of this method is that it may be used to generate exceptions that the target thread is unprepared to handle (including checked exceptions that the thread could not possibly throw, were it not for this method). For more information, see Why are Thread.stop, Thread.suspend and Thread.resume Deprecated?.*
void	**suspend**() **Deprecated.** *This method has been deprecated, as it is inherently deadlock-prone. If the target thread holds a lock on the monitor protecting a critical system resource when it is suspended, no thread can access this resource until the target thread is resumed. If the thread that would resume the target thread attempts to lock this monitor prior to calling resume, deadlock results. Such deadlocks typically manifest themselves as "frozen" processes. For more information, see Why are Thread.stop, Thread.suspend and Thread.resume Deprecated?.*
String	**toString**() Returns a string representation of this thread, including the thread's name, priority, and thread group.
static void	**yield**() Causes the currently executing thread object to temporarily pause and allow other threads to execute.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

---

Submit a bug or feature
For further API reference and developer documentation, see Java 2 SDK SE Developer Documentation. That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.