Java Thread Class

Thread class is the important class on which Java's Multithreading system is based. Thread class, along with its companion interface Runnable will be used to create and run threads for utilizing Multithreading feature of Java.

It provides constructors and methods to support multithreading. It extends object class and implements Runnable interface.

<u>Signature of Thread class</u>

public class Thread extends Object implements Runnable

Thread Class Priority Constants

Field	Description
MAX_PRIORITY	It represents the maximum priority that a thread can have.
MIN_PRIORITY	It represents the minimum priority that a thread can have.
NORM_PRIORITY	It represents the default priority that a thread can have.

Constructors of Thread class

Thread()

Thread(String str)

Thread(Runnable r)

Thread(Runnable r, String str)

Thread(ThreadGroup group, Runnable target)

Thread(ThreadGroup group, Runnable target, String name)

Thread(ThreadGroup group, Runnable target, String name, long stackSize)

Thread(ThreadGroup group, String name)

Thread Class Methods

Thread class also defines many methods for managing threads. Some of them are,

Modifier and Type	Method	Description
Void	start()	It is used to start the execution of the thread.
Void	run()	It is used to do an action for a thread.
static void	sleep()	It sleeps a thread for the specified amount of time.
static Thread	currentThread()	It returns a reference to the currently executing thread object.
Void	join()	It waits for a thread to die.
Int	getPriority()	It returns the priority of the thread.
Void	setPriority()	It changes the priority of the thread.
String	getName()	It returns the name of the thread.
Void	setName()	It changes the name of the thread.
Long	getId()	It returns the id of the thread.
boolean	isAlive()	It tests if the thread is alive.
static void	yield()	It causes the currently executing thread object to pause and allow other threads to execute temporarily.
Void	suspend()	It is used to suspend the thread.
Void	resume()	It is used to resume the suspended thread.
Void	stop()	It is used to stop the thread.

Void	destroy()	It is used to destroy the thread group and all of its subgroups.
boolean	isDaemon()	It tests if the thread is a daemon thread.
Void	setDaemon()	It marks the thread as daemon or user thread.
Void	interrupt()	It interrupts the thread.
boolean	isinterrupted()	It tests whether the thread has been interrupted.
static boolean	interrupted()	It tests whether the current thread has been interrupted.
static int	activeCount()	It returns the number of active threads in the current thread's thread group.
Void	checkAccess()	It determines if the currently running thread has permission to modify the thread.
static boolean	holdLock ()	It returns true if and only if the current thread holds the monitor lock on the specified object.
static void	dumpStack()	It is used to print a stack trace of the current thread to the standard error stream.
StackTraceElement[]	getStackTrace()	It returns an array of stack trace elements representing the stack dump of the thread.
static int	enumerate()	It is used to copy every active thread's thread group and its subgroup into the specified array.

Thread.State	getState()	It is used to return the state of the thread.
ThreadGroup	getThreadGroup()	It is used to return the thread group to which this thread belongs
String	toString()	It is used to return a string representation of this thread, including the thread's name, priority, and thread group.
Void	notify()	It is used to give the notification for only one thread which is waiting for a particular object.
Void	notifyAll()	It is used to give the notification to all waiting threads of a particular object.
Void	setContextClassLoader( )	It sets the context ClassLoader for the Thread.
ClassLoader	getContextClassLoader( )	It returns the context ClassLoader for the thread.
static Thread.UncaughtExceptionHandler	getDefaultUncaughtExceptionHandler()	It returns the default handler invoked when a thread abruptly terminates due to an uncaught exception.
static void	setDefaultUncaughtExceptionHandler()	It sets the default handler invoked when a thread abruptly terminates due to an uncaught exception.