**Constructors of Thread class**

* Thread()
* Thread(String name)
* Thread(object)
* Thread(object, String name)

**Thread()**

Which will be execute to set the predefined name for newly created thread, these names are generally in the form of thread -0, thread -1, ....  
Syntax to call constructor:

Syntax

Thread t=**new** Thread();

**Thread(String name)**

Which can be used to provide user defined name for newly created thread.

Syntax

Thread t=**new** Thread("newthread");

**Thread(object)**

Which can be used to provide default name for newly created user defined thread.

Syntax

UserdefinedThreadclass obj=**new** UserdefinedThreadclass();

Thread t=**new** Thread("obj");

**object, String name**

Which will be used to provide user defined name for the newly created user defined thread.

Syntax

UserdefinedThreadclass obj=**new** UserdefinedThreadclass();

Thread t=**new** Thread(**object**, "secondthread");

**Methods of Thread class**

* getPriority()
* setPriority()
* getName()
* setName()
* isDeamon()
* run()
* start()
* sleep()
* suspend()
* resume()
* stop()
* isAlive()
* currentThread()
* join()
* getState()
* yield()

**getPriority()**

This method is used to get the current priority of thread.

Thread t=**new** Thread();

**int** x=t.getPriority();

System.**out**.println(x);

**setPriority()**

This method is used to set the current priority of thread.

Thread t=**new** Thread();

t.setPriority(any priority number between o to 10)

**or**

t.setPriority(Thread.MAX-PRIORITY)

**getName()**

This method is used to get the current executing thread name.

Thread t=**new** Thread();

String s=t.getName();

System.**out**.println(s);

**setName()**

This method is used to set the userdefined name for the thread.

Thread t=**new** Thread();

t.setName("mythread");

**isDeamon()**

Which returns true if the current thread is background thread otherwise return false.

Thread t=**new** Thread();

**boolean** b=t.isDeamon();

**run()**

Which contains the main business logic that can be executed by multiple threads simultaneously in every user defined thread class run method should be overridden.

**public** Class\_Name **extends** Thread

{

**public** **void** run()

{

.....

.....

}

}

**start()**

Used to convert ready state thread to running state.

Thread t=**new** Thread();

t.start();

**sleep()**

Used to change running state thread to ready state based on time period it is a static method should be called with class reference.

**public** **static** **final** sleep(**long** milisecond)**throws** InterruptedException

{

**try**

{

Thread.sleep(3000);

}

**catch**(InterruptedException ie)

{

........

........

}

}

Once the given time period is completed thread state automatically change from waiting to running state.

**suspend()**

Used to convert running state thread to waiting state, which will never come back to running state automatically.

Thread t=**new** Thread();

t.suspend();

**resume()**

Used to change the suspended thread state(waiting state) to ready state.

Thread t=**new** Thread();

t.resume();

**Note:**Without using suspend() method resume() method can not be use.

**What is the difference between sleep() and suspend()**

Sleep() can be used to convert running state to waiting state and automatically thread convert from waiting state to running state once the given time period is completed. Where as suspend() can be used to convert running state thread to waiting state but it will never return back to running state automatically.

**stop()**

This method is used to convert running state thread to dead state.

Thread t=**new** Thread();

t.stop();

**isAlive()**

Which is return true if the thread is in ready or running or waiting state and return false if the thread is in new or dead state.

Thread t=**new** Thread();

t.isAlive();

**currentThread()**

Used to get the current thread detail like thread name thread group name and priority

Thread t=**new** Thread();

t.currentThread();

**Note:**

* The default thread name is thread-0, (if it is a main thread default name is main)
* The default thread group name is main
* Default thread priority is "5" is normal priority.

**join()**

Which can be used to combined more than one thread into a single group signature is public final void join()throws InterruptedException

**try**

{

t.join();

t2.join();

.....

.....

}

**getState()**

This method is used to get the current state of thread.

Thread t=**new** Thread();

t.getState();

**yield()**

Which will keep the currently executing thread into temporarily pass and allows other threads to execute