# Thead class library:

**Thread class Library mainly contains variables,constructors and methods which are predefined in Thread class library of JAVA:**

## Consrtuctors:

1. **public Thread()**

--> This constructor can be used to create thread class object with the following properties.

**Note: Every thread must have these three properties which are Thread Name, Thread Priority, Thread group Name, and default name of each thread is Thread-0,Thread-1…..and default Thread Priority is 5 and default Thread group Name is main as shown below:**

Thread Name: Thread-0

Thread Priority: 5

Thread group Name : main

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

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

**OP**: Thread[Thread-0, 5, main]

1. **public Thread(String name)**

-->This constructor can be used to create Thread class object with the specified name.

**EX: Thread t=new Thread("Core Java");**

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

**OP**: Thread[Core Java,5,main]

1. **public Thread(Runnable r)**

-->This constructor can be used to create Thread class object with the specified Runnable reference.

**EX: Runnable r=new Thread();**

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

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

**In the above lines of code,TWO Thread class objects are created.**

**OP:** Thread[Thread-1,5,main]

1. **public Thread(Runnable r, String name)**

-->This constructor can be used to create Thread class object with the specified Runnable reference and with the specified name.

**EX:** Runnable r=new Thread();

Thread t=new Thread(r, "Core Java");

System.out.println(t);

**OP**: Thread[Core Java,5,main]

1. **public Thread(ThreadGroup tg, Runnable r)**

--> This constuctor can be used to create Thread class object with the specified ThreadGroup name and with the specified Runnable reference.

**NOTE: To provide ThreadGroup name we have to use a predefined class like java.lang.ThreadGroup, to create ThreadGroup class object we have to use the following Constructor.**

public ThreadGroup(String name)

**EX:** ThreadGroup tg=new ThreadGroup("Java");

Runnable r=new Thread();

Thread t=new Thread(tg, r)

System.out.println(t);

**OP:** Thread[Thread-1,5,Java]

6 **.public Thread(ThreadGroup tg, String name)**

--> This constructor can be used to create Thread class object with the specified ThreadGroup name and with the specified Thread name.

**EX:** ThreadGroup tg=new ThreadGroup("Java");

Thread t=new Thread(tg, "Core Java");

System.out.println(t);

**OP**: Thread[Core Java,5,Java]

1. **public Thread(ThreadGroup tg, Runnable r, String name)**

--> This constuctor constuctor can be used to create Thread class object with the specified ThreadGroup name, with the Runnable reference and with the thread name.

**EX:** ThreadGroup tg=new ThreadGroup("Java");

Runnable r=new Thread();

Thread t=new Thread(tg, r, "Core Java");

**OP**: Thread[Core Java, 5, Java]

**Methods:**

1. **public void setName(String name)**

-->It can be used to set a particular name to the Thread explicitly.

1. **public String getName()**

-->It can be used to get thread name explicitly.

**EX:** class Test

{

public static void main(String[] args)

{

Thread t=new Thread();

System.out.println(t.getName());

t.setName("Core Java");

System.out.println(t.getName());

}

}

**3.public void setPriority(int priority)**

-->It can be used to set a particular priority value to the Thread, but, here the priority value must be provided in the range from 1 to 10, if we provide any other value then JVM will rise an exception like java.lang.IllegalArgumentException.

**To represent Thread priority values, java.lang.Thread class has provided the following constants which are as follows:**

public static final int MIN\_PRIORITY=1;

public static final int NORM\_PRIORITY=5;

public static final int MAX\_PRIORITY=10;

**4.public int getPriority()**

-->It can be used to get priority value of the Thread.

**EX:**

class Test

{

public static void main(String[] args)

{

Thread t=new Thread();

System.out.println(t.getPriority());

t.setPriority(7);

System.out.println(t.getPriority());

t.setPriority(Thread.MAX\_PRIORITY-2);

System.out.println(t.getPriority());

//t.setPriority(Thread.NORM\_PRIORITY+6);-->IllegalArgumentException

}

}

**5.public static int activeCount()**

--> It will return the no. of threads which are in active.[means which are active at present].

**EX:**

class Test

{

public static void main(String[] args)

{

Thread t=new Thread();

t.start();

System.out.println(Thread.activeCount());

}

}

**6.public boolean isAlive()**

--> This method can be used to check whether a thread is in live or not.

**EX:** class Test

{

public static void main(String[] args)

{

Thread t=new Thread();

System.out.println(t.isAlive());

t.start();

System.out.println(t.isAlive());

System.out.println(Thread.activeCount());

}

}

**7.public static Thread currentThread()**

--> It able to return Thread object reference which is executing over the present instruction.

**EX:** class MyThread extends Thread

{

public void run()

{

for(int i=0;i<10;i++)

{

System.out.println(Thread.currentThread().getName());

}

}

}

class Test

{

public static void main(String[] args)

{

MyThread mt1=new MyThread();

MyThread mt2=new MyThread();

MyThread mt3=new MyThread();

mt1.setName("AAA");

mt2.setName("BBB");

mt3.setName("CCC");

mt1.start();

mt2.start();

mt3.start();

}

}

**8.public static void sleep(long time)throws InterruptedException**

--> This method can be used to keep a running thread in sleeping state upto the specified sleep time.

**In general, we will use sleep() method in run() method of user defined Thread class, where to handle InterruptedException we must use try-catch-finally syntax only, we must not use "throws" keyword in run() method prototype, because, we are overriding Thread class or Runnable interface predefined run() method.**

class MyThread extends Thread

{

public void run(){

for (int i=0;i<10; i++)

{

try{

Thread.sleep(1000);

System.out.println(“ USER THREAD :”+i);

}

catch(Exception e){

e.printStackTrace();}

}

}

}

public class Test

{

public static void main(String args[])

{

MyThread mt=new MyThread();

mt.start();

}

}

Once the specified time is completed, then the Thread will out of sleep state and it able to be in active mode/it able to execute its application directly.

**9.public void join()throws InterruptedException**

-->This method will pause a thread to complete a thread on which we accessed join() method , after completion of the respective thread, paused thread will continue its execution part automatically, **means If I want to pause main thread execution until the completion of child thread and once child thread is executed, then I want to continue main thread.**

It will pause a thread to complete another thread.

class MyThread extends Thread

{

public void run()

{

for(int i=0;i<10;i++)

{

System.out.println(" USER THREAD :"+i);

}

}

}

public class Test

{

public static void main(String args[])throws Exception

{

MyThread mt=new MyThread();

mt.start();

mt.join();// BECAUSE OF THIS FIRST user thread will be executed completely and then main thread

for(int i=0;i<10;i++)

{

System.out.println(" MAIN THREAD : "+ i);

}

}

}