**How to create thread:**

There are two ways to create a thread:

1. By extending Thread class
2. By implementing Runnable interface.

**Thread class:**

|  |
| --- |
| Thread class provide **constructors** and **methods** to create and perform operations on a thread. Thread class extends Object class and implements Runnable interface. |

**Commonly used Constructors of Thread class:**

|  |
| --- |
| * Thread() * Thread(String name) * Thread(Runnable r) * Thread(Runnable r,String name) |

**Commonly used methods of Thread class:**

|  |
| --- |
| 1. **public void run():** is used to perform action for a thread. 2. **public void start():** starts the execution of the thread.JVM calls the run() method on the thread. 3. **public static void sleep(long miliseconds):** Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds. 4. **public void join():** waits for a thread to die. 5. **public void join(long miliseconds):** waits for a thread to die for the specified miliseconds. 6. **public int getPriority():** returns the priority of the thread. 7. **public int setPriority(int priority):** changes the priority of the thread. 8. **public String getName():** returns the name of the thread. 9. **public void setName(String name):** changes the name of the thread. 10. **public static Thread currentThread():** returns the reference of currently executing thread. 11. **public int getId():** returns the id of the thread. 12. **public Thread.State getState():** returns the state of the thread. 13. **public boolean isAlive():** tests if the thread is alive. 14. **public void yield():** causes the currently executing thread object to temporarily pause and allow other threads to execute. 15. **public void suspend():** is used to suspend the thread(depricated). 16. **public void resume():** is used to resume the suspended thread(depricated). 17. **public void stop():** is used to stop the thread(depricated). 18. **public boolean isDaemon():** tests if the thread is a daemon thread. 19. **public void setDaemon(boolean b):** marks the thread as daemon or user thread. 20. **public void interrupt():** interrupts the thread. 21. **public boolean isInterrupted():** tests if the thread has been interrupted. 22. **public static boolean interrupted():** tests if the current thread has been interrupted. |

**Runnable interface:**

|  |
| --- |
| The Runnable interface should be implemented by any class whose instances are intended to be executed by a thread. Runnable interface have only one method named run(). |

|  |
| --- |
| 1. **public void run():** is used to perform action for a thread. |

**Starting a thread:**

|  |
| --- |
| **start() method** of Thread class is used to start a newly created thread. It performs following tasks:   * A new thread starts(with new callstack). * The thread moves from New state to the Runnable state. * When the thread gets a chance to execute, its target run() method will run. |

//Ex.1 First Main Thread

class FirstThread

{

public static void main(String args[])

{

Thread t=Thread.currentThread();

System.out.println("First Thread after thread creation "+t);

}

}

//Ex.2 use of getName()

class GetNameMethod

{

public static void main(String args[])

{

Thread t=Thread.currentThread();

System.out.println("First Thread after thread creation : "+t.getName());

}

}

//Ex.3 use of setName()

class Program61\_SetNameMethod

{

public static void main(String args[])

{

Thread t=Thread.currentThread();

System.out.println("First Thread after thread creation : "+t.getName());

t.setName("VowTech");

System.out.println("New Thread Name :"+t.getName());

}

}

//Ex.4 Pausing a Thread

class SleepMethod

{

public static void main(String args[])

{

Thread t=Thread.currentThread();

System.out.println("First Thread after thread creation : "+t.getName());

try

{

System.out.println("Java");

t.sleep(1500);

System.out.println("is the");

t.sleep(3000);

System.out.println("Most Popular language");

}

catch(InterruptedException e)

{

System.out.println("Exception Occerred");

}

}

}

**1)By extending Thread class:**

//Using Thread class

class MyThread extends Thread

{

MyThread(String s)

{

super(s);

start();

}

public void run()

{

for(int i=0;i<2;i++)

{

System.out.println("Thread Name : "+Thread.currentThread().getName());

try

{

System.out.println("Now :"+Thread.currentThread().getName()+" is sleep");

Thread.sleep(5000);

System.out.println("Now :"+Thread.currentThread().getName()+" is wakeup");

}

catch(InterruptedException e)

{

}

}

System.out.println("Finally "+Thread.currentThread().getName()+" is finish");

}

}

class ThreadClass

{

public static void main(String args[])

{

MyThread t1=new MyThread("First Thread");

MyThread t2=new MyThread("Second Thread");

}

}

**Who makes your class object as thread object?**

|  |
| --- |
| **Thread class constructor** allocates a new thread object. When you create object of Multi class, your class constructor is invoked(provided by Compiler) from where Thread class constructor is invoked(by super() as first statement).So your Multi class object is thread object now. |

**2) By implementing the Runnable interface:**

class CreateThread implements Runnable

{

Thread t;

CreateThread(String s)

{

t=new Thread(this,s);

t.start();

}

public void run()

{

for(int i=0;i<2;i++)

{

System.out.println("Thread Name"+Thread.currentThread().getName());

try

{

System.out.println("Now : "+Thread.currentThread().getName()+" is sleep");

Thread.sleep(5000);

System.out.println("Now : "+Thread.currentThread().getName()+" is wake up");

}

catch(InterruptedException e)

{

}

}

System.out.println("Finally: "+Thread.currentThread().getName()+" is finish");

}

}

class RunnableInterface

{

public static void main(String args[])

{

CreateThread t1=new CreateThread("First Thread");

CreateThread t2=new CreateThread("Second Thread");

}

}

|  |
| --- |
| If you are not extending the Thread class, your class object would not be treated as a thread object. So you need to explicitly create Thread class object. We are passing the object of your class that implements Runnable so that your class run() method may execute. |