



# SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)

Approved by AICTE, New Delhi, Affiliated to Anna University, Chennai

Accredited by NAAC-UGC with 'A++' Grade (Cycle III) &

Accredited by NBA (B.E - CSE, EEE, ECE, Mech&B.Tech.IT)

COIMBATORE-641 035, TAMIL NADU



## UNIT V

### MULTITHREADING IN JAVA

#### Java Threads - How to create a thread in Java

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 void sleep(long milliseconds):** 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 milliseconds):** waits for a thread to die for the specified milliseconds.
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 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(deprecated).
16. **public void resume():** is used to resume the suspended thread(deprecated).
17. **public void stop():** is used to stop the thread(deprecated).
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:**

The **start()** method of Thread class is used to start a newly created thread. It performs the 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.

## 1) Java Thread Example by extending Thread class

**FileName:** Multi.java

```
1. class Multi extends Thread{
2. public void run(){
3. System.out.println("thread is running...");
4. }
5. public static void main(String args[]){
6. Multi t1=new Multi();
7. t1.start();
8. }
9. }
```

**Output:**

```
thread is running...
```

## 2) Java Thread Example by implementing Runnable interface

**FileName:** Multi3.java

```
1. class Multi3 implements Runnable{
2. public void run(){
3. System.out.println("thread is running...");
4. }
5.
6. public static void main(String args[]){
7. Multi3 m1=new Multi3();
8. Thread t1 =new Thread(m1); // Using the constructor Thread(Runnable r)
9. t1.start();
10. }
11. }
```

**Output:**

```
thread is running...
```

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 the Thread class object. We are passing the object of your class that implements Runnable so that your class run() method may execute.

### 3) Using the Thread Class: Thread(String Name)

We can directly use the Thread class to spawn new threads using the constructors defined above.

**FileName:** MyThread1.java

```
1. public class MyThread1
2. {
3. // Main method
4. public static void main(String argsv[])
5. {
6. // creating an object of the Thread class using the constructor Thread(String name)
7. Thread t= new Thread("My first thread");
8.
9. // the start() method moves the thread to the active state
10. t.start();
11. // getting the thread name by invoking the getName() method
12. String str = t.getName();
13. System.out.println(str);
14. }
15. }
```

**Output:**

```
My first thread
```

### 4) Using the Thread Class: Thread(Runnable r, String name)

**Observe the following program.**

**FileName:** MyThread2.java

```
1. public class MyThread2 implements Runnable
2. {
3. public void run()
4. {
5. System.out.println("Now the thread is running ...");
6. }
7. }
```

```
8. // main method
9. public static void main(String argsv[])
10. {
11. // creating an object of the class MyThread2
12. Runnable r1 = new MyThread2();
13.
14. // creating an object of the class Thread using Thread(Runnable r, String name)
15. Thread th1 = new Thread(r1, "My new thread");
16.
17. // the start() method moves the thread to the active state
18. th1.start();
19.
20. // getting the thread name by invoking the getName() method
21. String str = th1.getName();
22. System.out.println(str);
23. }
24. }
```

**Output:**

My new thread

Now the thread is running ...