

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:

- ✓ **public void run():** is used to perform action for a thread.
- ✓ **public void start():** starts the execution of the thread.
JVM calls the run() method on the thread.
- ✓ **public void sleep(long milliseconds):** Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds.
- ✓ **public void join():** waits for a thread to die.
- ✓ **public void join(long milliseconds):** waits for a thread to die for the specified milliseconds.
- ✓ **public int getPriority():** returns the priority of the thread.
- ✓ **public int setPriority(int priority):** changes the priority of the thread.
- ✓ **public String getName():** returns the name of the thread.
- ✓ **public void setName(String name):** changes the name of the thread.
- ✓ **public void suspend():** is used to suspend the thread(deprecated).
- ✓ **public void resume():** is used to resume the suspended thread(deprecated).
- ✓ **public void stop():** is used to stop the thread(deprecated).

- ✓ **public int getId():** returns the id of the thread.
- ✓ **public boolean isAlive():** tests if the thread is alive.

Example:

The screenshot shows a Notepad++ editor with a Java file named `TestMultiThread1.java`. The code defines two classes, `ABC` and `XYZ`, both extending `Thread`. `ABC` has a `run()` method that prints "Thread ABC: " followed by its ID. `XYZ` has a `run()` method that prints "Thread XYZ: " followed by its ID. A `TestMultiThread1` class contains a `main` method that creates and starts instances of `ABC` and `XYZ`.

Next to the editor is a Command Prompt window showing the output of the program. It displays the output of `javac TestMultiThread1.java` and `java TestMultiThread1`. The output shows interleaved messages from the two threads, such as "Thread XYZ: 9", "Thread ABC: 7", "Thread ABC: 8", "Thread ABC: 9", etc.

```

1  class ABC extends Thread
2  {
3      public void run()
4      {
5          int i;
6          for(i=0;i<10;i++)
7          {
8              System.out.println("Thread ABC: "+i);
9          }
10     }
11
12     class XYZ extends Thread
13     {
14         public void run()
15         {
16             int i;
17             for(i=0;i<10;i++)
18             {
19                 System.out.println("Thread XYZ: "+i);
20             }
21         }
22     }
23
24     public class TestMultiThread1
25     {
26         public static void main(String args[])
27         {
28             Thread t1=new Thread(new ABC());
29             Thread t2=new Thread(new XYZ());
30             t2.start();
31             t1.start();
32         }
33     }
34
35

```

```

C:\Users\ashish\Desktop\java>javac TestMultiThread1.java
C:\Users\ashish\Desktop\java>java TestMultiThread1
Thread XYZ: 9
Thread ABC: 7
Thread ABC: 8
Thread ABC: 9
Thread XYZ: 0
Thread XYZ: 0
Thread ABC: 1
Thread XYZ: 1
Thread ABC: 2
Thread XYZ: 2
Thread ABC: 3
Thread XYZ: 3
Thread ABC: 4
Thread XYZ: 4
Thread ABC: 5
Thread XYZ: 4
Thread ABC: 6
Thread XYZ: 5
Thread ABC: 7
Thread XYZ: 6
Thread ABC: 8
Thread XYZ: 7
Thread ABC: 9
Thread XYZ: 8
Thread XYZ: 9
C:\Users\ashish\Desktop\java>

```