

Creating a thread in Java

To implement multithreading, Java defines two ways by which a thread can be created.

By implementing the Runnable interface.

By extending the Thread class.

Runnable Interface

It is used to create thread and should be used if you are only planning to override the run() method and no other Thread methods.

Signature

```
public interface Runnable
```

Runnable Interface Method

It provides only single method that must be implemented by the class.

Method	Description
run()	It runs the implemented thread.

Implementing the Runnable Interface

The easiest way to create a thread is to create a class that implements the runnable interface. After implementing runnable interface, the class needs to implement the run() method.

Run Method Syntax:

```
public void run()
```

It introduces a concurrent thread into your program. This thread will end when run() method terminates.

You must specify the code that your thread will execute inside run() method.

run() method can call other methods, can use other classes and declare variables just like any other normal method.

Example

class MyThread implements Runnable

```
{  
    public void run()  
    {  
        System.out.println("concurrent thread started running..");  
    }  
}
```

class MyThreadDemo

```
{  
    public static void main(String args[])  
    {  
        MyThread mt = new MyThread();  
        Thread t = new Thread(mt);  
        t.start();  
    }  
}
```

Output

concurrent thread started running..

To call the run() method, start() method is used. On calling start(), a new stack is provided to the thread and run() method is called to introduce the new thread into the program.

Note: If you are implementing Runnable interface in your class, then you need to explicitly create a Thread class object and need to pass the Runnable interface implemented class object as a parameter in its constructor.

Extending Thread class

This is another way to create a thread by a new class that extends Thread class and create an instance of that class. The extending class must override run() method which is the entry point of new thread.

Example:

```
class MyThread extends Thread
```

```
{
    public void run()
    {
        System.out.println("concurrent thread started running..");
    }
}
```

```
class MyThreadDemo
```

```
{
    public static void main(String args[])
    {
        MyThread mt = new MyThread();
        mt.start();
    }
}
```

Output

concurrent thread started running..

In this case also, we must override the run() and then use the start() method to run the thread. Also, when you create MyThread class object, Thread class constructor will also be invoked, as it is the super class, hence MyThread class object acts as Thread class object.

What if we call run() method directly without using start() method?

In above program if we directly call run() method, without using start() method,

```
public static void main(String args[])  
{  
    MyThread mt = new MyThread();  
    mt.run();  
}
```

Doing so, the thread won't be allocated a new call stack, and it will start running in the current call stack, that is the call stack of the main thread. Hence Multithreading won't be there.

