

Practical 9

Multithreading

Java - Multithreading

Create a Thread by Implementing a Runnable Interface

If your class is intended to be executed as a thread then you can achieve this by implementing a **Runnable** interface. You will need to follow three basic steps –

Step 1

As a first step, you need to implement a `run()` method provided by a **Runnable** interface. This method provides an entry point for the thread and you will put your complete business logic inside this method. Following is a simple syntax of the `run()` method –

```
public void run( )
```

Step 2

As a second step, you will instantiate a **Thread** object using the following constructor –

```
Thread(Runnable threadObj, String threadName);
```

Where, *threadObj* is an instance of a class that implements the **Runnable** interface and **threadName** is the name given to the new thread.

Step 3

Once a Thread object is created, you can start it by calling **start()** method, which executes a call to `run()` method. Following is a simple syntax of `start()` method –

```
void start();
```

Example

```
class RunnableDemo implements Runnable {
    private Thread t;
    private String threadName;

    RunnableDemo( String name) {
        threadName = name;
        System.out.println("Creating " + threadName );
    }

    public void run() {
        System.out.println("Running " + threadName );
        try {
            for(int i = 4; i > 0; i--) {
                System.out.println("Thread: " + threadName + ", " + i);
                // Let the thread sleep for a while.
                Thread.sleep(50);
            }
        }
    }
}
```

```

        } catch (InterruptedException e) {
            System.out.println("Thread " + threadName + " interrupted.");
        }
        System.out.println("Thread " + threadName + " exiting.");
    }

    public void start () {
        System.out.println("Starting " + threadName );
        if (t == null) {
            t = new Thread (this, threadName);
            t.start ();
        }
    }
}

public class TestThread {

    public static void main(String args[]) {
        RunnableDemo R1 = new RunnableDemo( "Thread-1");
        R1.start();

        RunnableDemo R2 = new RunnableDemo( "Thread-2");
        R2.start();
    }
}

```

This will produce the following result –

Output

```

Creating Thread-1
Starting Thread-1
Creating Thread-2
Starting Thread-2
Running Thread-1
Thread: Thread-1, 4
Running Thread-2
Thread: Thread-2, 4
Thread: Thread-1, 3
Thread: Thread-2, 3
Thread: Thread-1, 2
Thread: Thread-2, 2
Thread: Thread-1, 1
Thread: Thread-2, 1
Thread Thread-1 exiting.
Thread Thread-2 exiting.

```

Create a Thread by Extending a Thread Class

The second way to create a thread is to create a new class that extends **Thread** class using the following two simple steps. This approach provides more flexibility in handling multiple threads created using available methods in Thread class.

Step 1

You will need to override **run()** method available in Thread class. This method provides an entry point for the thread and you will put your complete business logic inside this method. Following is a simple syntax of run() method –

```
public void run( )
```

Step 2

Once Thread object is created, you can start it by calling **start()** method, which executes a call to **run()** method. Following is a simple syntax of start() method –

```
void start( );
```

Example

```
class ThreadDemo extends Thread {
    private Thread t;
    private String threadName;

    ThreadDemo( String name) {
        threadName = name;
        System.out.println("Creating " + threadName );
    }

    public void run() {
        System.out.println("Running " + threadName );
        try {
            for(int i = 4; i > 0; i--) {
                System.out.println("Thread: " + threadName + ", " + i);
                // Let the thread sleep for a while.
                Thread.sleep(50);
            }
        } catch (InterruptedException e) {
            System.out.println("Thread " + threadName + " interrupted.");
        }
        System.out.println("Thread " + threadName + " exiting.");
    }

    public void start () {
        System.out.println("Starting " + threadName );
        if (t == null) {
            t = new Thread (this, threadName);
            t.start ();
        }
    }
}

public class TestThread {

    public static void main(String args[]) {
        ThreadDemo T1 = new ThreadDemo( "Thread-1");
        T1.start();

        ThreadDemo T2 = new ThreadDemo( "Thread-2");
        T2.start();
    }
}
```

This will produce the following result –

Output

```
Creating Thread-1
Starting Thread-1
Creating Thread-2
Starting Thread-2
Running Thread-1
```

Thread: Thread-1, 4
Running Thread-2
Thread: Thread-2, 4
Thread: Thread-1, 3
Thread: Thread-2, 3
Thread: Thread-1, 2
Thread: Thread-2, 2
Thread: Thread-1, 1
Thread: Thread-2, 1
Thread Thread-1 exiting.
Thread Thread-2 exiting.

Applet

```
import java.applet.Applet;  
import java.awt.*;  
  
public class GraphicsDemo extends Applet{  
  
    public void paint(Graphics g) {  
        g.setColor(Color.red);  
        g.drawString("Welcome", 50, 50);  
        g.drawLine(20, 30, 20, 300);  
        g.drawRect(70, 100, 30, 30);  
        g.fillRect(170, 100, 30, 30);  
        g.drawOval(70, 200, 30, 30);  
  
        g.setColor(Color.pink);  
        g.fillOval(170, 200, 30, 30);  
        g.drawArc(90, 150, 30, 30, 30, 270);  
        g.fillArc(270, 150, 30, 30, 0, 180);  
  
    }  
}
```