

Multithreading:-

Multithreading in java is a process of executing **multiple threads** simultaneously. Thread is basically a lightweight sub-process, a smallest unit of processing.

The Main Thread

When a Java program starts up, one thread begins running immediately. This is usually called the *main thread* of your program, because it is the one that is executed when your program begins. The main thread is important for two reasons:

- It is the thread from which other “child” threads will be spawned.
- Often, it must be the last thread to finish execution because it performs various shutdown actions.

Although the main thread is created automatically when your program is started, it can be controlled through a **Thread** object. To do so, you must obtain a reference to it by calling the method **currentThread()**, which is a **public static** member of **Thread**. Its general form is shown here:

static Thread currentThread()

This method returns a reference to the thread in which it is called. Once you have a reference to the main thread, you can control it just like any other thread.

Let’s begin by reviewing the following example:

```
// Controlling the main Thread.
class CurrentThreadDemo {
    public static void main(String args[]) {
        Thread t = Thread.currentThread();
        System.out.println("Current thread: " + t);
        // change the name of the thread
        t.setName("My Thread");
        System.out.println("After name change: " + t);
        try {
            for(int n = 5; n > 0; n--) {
                System.out.println(n);
                Thread.sleep(1000);
            }
        } catch (InterruptedException e) {
            System.out.println("Main thread interrupted");
        }
    }
}
```

Here is the output generated by this program:

```
Current thread: Thread[main,5,main]
After name change: Thread[My Thread,5,main]
5
4
3
2
1
```

Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light-weight processes within a process.

Threads can be created by using two mechanisms :

1. Extending the Thread class
2. Implementing the Runnable Interface

Thread creation by implementing the Runnable Interface

We create a new class which implements `java.lang.Runnable` interface and override `run()` method. Then we instantiate a `Thread` object and call `start()` method on this object.

After you create a class that implements **Runnable**, you will instantiate an object of type **Thread** from within that class. **Thread** defines several constructors. The one that we will use is shown here:

```
Thread(Runnable threadOb, String threadName)
```

In this constructor, *threadOb* is an instance of a class that implements the **Runnable** interface. This defines where execution of the thread will begin. The name of the new thread is specified by *threadName*.

After the new thread is created, it will not start running until you call its **start()** method, which is declared within **Thread**. In essence, **start()** executes a call to **run()**. The **start()** method is shown here:

Here is an example that creates a new thread and starts it running:

```
// Create a second thread.
class NewThread implements Runnable {
    Thread t;
    NewThread() {
        // Create a new, second thread
        t = new Thread(this, "Demo Thread");
        System.out.println("Child thread: " + t);
        t.start(); // Start the thread
    }
    // This is the entry point for the second thread.
    public void run() {
        try {
            for(int i = 5; i > 0; i--) {
                System.out.println("Child Thread: " + i);
                Thread.sleep(500);
            }
        } catch (InterruptedException e) {
            System.out.println("Child interrupted.");
        }
        System.out.println("Exiting child thread.");
    }
}

class ThreadDemo {
    public static void main(String args[]) {
        new NewThread(); // create a new thread
        try {
            for(int i = 5; i > 0; i--) {
                System.out.println("Main Thread: " + i);
                Thread.sleep(1000);
            }
        } catch (InterruptedException e) {
            System.out.println("Main thread interrupted.");
        }
        System.out.println("Main thread exiting.");
    }
}
```

```
Child thread: Thread[Demo Thread,5,main]
Main Thread: 5
Child Thread: 5
Child Thread: 4
```

Main Thread: 4
Child Thread: 3
Child Thread: 2
Main Thread: 3
Child Thread: 1
Exiting child thread.
Main Thread: 2
Main Thread: 1
Main thread exiting.

Extending Thread

The second way to create a thread is to create a new class that extends **Thread**, and then to create an instance of that class. The extending class must override the **run()** method, which is the entry point for the new thread. It must also call **start()** to begin execution of the new thread. Here is the preceding program rewritten to extend **Thread**:

```
// Create a second thread by extending Thread
class NewThread extends Thread {
    NewThread() {
        // Create a new, second thread
        super("Demo Thread");
        System.out.println("Child thread: " + this);
        start(); // Start the thread
    }
    // This is the entry point for the second thread.
    public void run() {
        try {
            for(int i = 5; i > 0; i--) {
                System.out.println("Child Thread: " + i);
                Thread.sleep(500);
            }
        } catch (InterruptedException e) {
            System.out.println("Child interrupted.");
        }
        System.out.println("Exiting child thread.");
    }
}

class ExtendThread {
    public static void main(String args[]) {
        new NewThread(); // create a new thread
        try {
            for(int i = 5; i > 0; i--) {
                System.out.println("Main Thread: " + i);
                Thread.sleep(1000);
            }
        } catch (InterruptedException e) {
            System.out.println("Main thread interrupted.");
        }
        System.out.println("Main thread exiting.");
    }
}
```