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:

i. It is the thread from which other "child" threads will be spawned.
ii. It must be the last thread to finish execution because it performs various shutdown actions.

Example:

```
class main_thread
{
	public static void main(String [] args)
	{
		Thread t = Thread.currentThread();
		System.out.println("Current Thread: "+ t);
		t.setName("My Thread");
		System.out.println("After name change: "+ t);
		try
		{
			for(int i=5;i>0;i--)
				System.out.println(i);
			Thread.sleep(3000);
		}
		catch (InterruptedException e)
		{
			System.out.println ("Main Thread Interrupted");
		}
	}
}
```

```
C:\javaprog\thread>javac main_thread.java

C:\javaprog\thread>java main_thread
Current Thread: Thread[main,5,main]
After name change: Thread[My Thread,5,main]
5
4
3
2
1

C:\javaprog\thread>
```

Creating a Thread:

- Implementing runnable interface
- Extend the thread class

Implementing Runnable:

```
class NewThread implements Runnable
{
	Thread t;
	NewThread()
	{
		t = new Thread(this, "Demo Thread");		// Create a new, second thread
		System.out.println("Child thread: " + t);
		t.start(); // Start the 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 ThreadDemoImpl
{
        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.
```

Second example:

```
class FirstThread implements Runnable
{
        public void run()          //This method will be executed when this thread is executed
        {
                for ( int i=1; i<=4; i++)
                {
                        System.out.println( "First Thread: " +i);
                        try{
                                Thread.sleep (1000);
                        }catch (InterruptedException ie)
                        {
                                System.out.println( "First Thread is interrupted." +ie);
                        }
                }
        }
}
```

```
class SecondThread implements Runnable
{
	public void run()
	{
		for ( int i=1; i<=4; i++)
		{
			System.out.println( "Second Thread: " +i);
			try
			{
				Thread.sleep(1000);
			}
			catch (InterruptedException ie)
			{
				System.out.println( "Second Thread is interrupted." +ie);
			}
		}
	}
}
```

```
class ThreadDemoImpl1
{
	public static void main(String args[])
	 {
		 FirstThread   ft = new FirstThread();
		 SecondThread  st = new SecondThread();
		Thread t1 = new Thread(ft);
		t1.start();			//Starting the first thread
		Thread t2 = new Thread(st);
		t2.start();			//Starting the second thread
	}
}
```

```
Second Thread: 1
First Thread: 1
First Thread: 2
Second Thread: 2
Second Thread: 3
First Thread: 3
First Thread: 4
Second Thread: 4
```

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
	}
	public void run()		// This is the entry point for the second thread.
	{
		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.");
	}
}
```

```
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.
```

Second example:

```
class FirstThread extends Thread
{
	public void run()
	{
		for (int i=1; i<=3; i++)
		{
			System.out.println( "First Thread : " +i);
			try{
				Thread.sleep(1000);
			}
			catch (InterruptedException  ie)
			{
				System.out.println("First Thread is interrupted." + ie);
			}
		}
	}
}
class SecondThread extends Thread
{
	public void run()
	{
		for (int i=1; i<=3; i++)
		{
			System.out.println( " Second Thread : " +i);
```

```
                try{
                        Thread.sleep(1000);
                }
                catch (InterruptedException  ie)
                {
                        System.out.println("Second Thread is interrupted." + ie);
                }
        }
    }
}
public class ThreadDemoExtends
{
    public static void main(String args[])
    {
        FirstThread   t1 = new FirstThread();
        SecondThread  t2 = new SecondThread();
        t1.start();     t2.start();
    }
}
```

```
First Thread : 1
Second Thread : 1
First Thread : 2
Second Thread : 2
Second Thread : 3
First Thread : 3
```

Creating multiple threads:

```
class NewThread implements Runnable
{
	String name;
	Thread t;
	NewThread(String threadname)
	{
		name = threadname;
		t = new Thread(this, name);
		System.out.println("New thread: " + t);
		t.start();
	}
	public void run()
	{
		try
		{
			for(int i = 3; i > 0; i--)
			{
				System.out.println(name + ": " + i);
				Thread.sleep(1000);
			}
		}catch(InterruptedException e)
		{
			System.out.println(name + "Interrupted");
		}
		System.out.println(name + " exiting.");
```

```
        }
}
class MultiThreadDemo
{
        public static void main(String args[])
        {
                new NewThread("One");
                new NewThread("Two");
                new NewThread("Three");
                try
                {
                        Thread.sleep(5000);     // wait for other threads to end
                }catch (InterruptedException e)
                {
                        System.out.println("Main thread Interrupted");
                }
                System.out.println("Main thread exiting.");
        }
}
```

```
C:\javaprog\thread>java MultiThreadDemo
New thread: Thread[One,5,main]
New thread: Thread[Two,5,main]
One: 3
New thread: Thread[Three,5,main]
Two: 3
Three: 3
Two: 2
One: 2
Three: 2
Two: 1
One: 1
Three: 1
Two exiting.
One exiting.
Three exiting.
Main thread exiting.

C:\javaprog\thread>
```

isAlive( ) and join( ) :

```
class NewThread implements Runnable
{
	String name;
	Thread t;
	NewThread(String threadname)
	{
		name = threadname;
		t = new Thread(this, name);
		System.out.println("New thread: " + t);
		t.start();
	}
	public void run()
	{
		try {
			for(int i = 3; i > 0; i--)
			{
				System.out.println(name + ": " + i);
				Thread.sleep(1000);
			}
		}catch(InterruptedException e)
		{
			System.out.println(name + " interrupted.");
		}	System.out.println(name + " exiting.");
	}
}
```

```
class DemoJoinIsAlive
{
	public static void main(String args[])
	{
		NewThread ob1 = new NewThread("One");
		NewThread ob2 = new NewThread("Two");
		NewThread ob3 = new NewThread("Three");
		System.out.println("Thread One is alive: "+ ob1.t.isAlive());
		System.out.println("Thread Two is alive: "+ ob2.t.isAlive());
		System.out.println("Thread Three is alive: "+ ob3.t.isAlive());
		try{
			System.out.println("Waiting for threads to finish.");
			ob1.t.join();
			ob2.t.join();
			ob3.t.join();
		}catch(InterruptedException e)
		{
			System.out.println("Main thread Interrupted");
		}
		System.out.println("Thread One is alive: "+ ob1.t.isAlive());
		System.out.println("Thread Two is alive: "+ ob2.t.isAlive());
		System.out.println("Thread Three is alive: "+ ob3.t.isAlive());
		System.out.println("Main thread exiting.");
	}
}
```

Output:

```
C:\javaprog\thread>java DemoJoinIsAlive
New thread: Thread[One,5,main]
New thread: Thread[Two,5,main]
One: 3
Two: 3
New thread: Thread[Three,5,main]
Thread One is alive: true
Thread Two is alive: true
Three: 3
Thread Three is alive: true
Waiting for threads to finish.
Two: 2
One: 2
Three: 2
One: 1
Two: 1
Three: 1
One exiting.
Two exiting.
Three exiting.
Thread One is alive: false
Thread Two is alive: false
Thread Three is alive: false
Main thread exiting.

C:\javaprog\thread>
```

Synchronization:

```
class Callme
{
        synchronized void call(String msg)
        {
                System.out.print("[" + msg);
                try {
                        Thread.sleep(2000);
                }catch(InterruptedException e)
                {
                        System.out.println("Interrupted");
                }
                System.out.println("]");
        }
}
class Caller implements Runnable
{
        String msg;   Callme target;   Thread t;
        public Caller(Callme targ, String s)
        {
                target = targ;     msg = s;
                t = new Thread(this);
                t.start();    }
```

```
        public void run()
        {
                target.call(msg);
        }
}
class Synch
{
        public static void main(String args[])
        {
                Callme target = new Callme();
                Caller ob1 = new Caller(target, "Hello");
                Caller ob2 = new Caller(target, "Synchronized");
                Caller ob3 = new Caller(target, "World");
                try{
                                ob1.t.join();
                                ob2.t.join();
                                ob3.t.join();
                        }catch(InterruptedException e)
                        {
                                System.out.println("Interrupted");
                        }
        }
}
```

Using synchronized statement:

```
class Callme
{
	void call(String msg)
	{
		System.out.print("[" + msg);
		try {
			Thread.sleep(2000);
		}catch(InterruptedException e)
		{
			System.out.println("Interrupted");
		}
		System.out.println("]");
	}
}
class Caller implements Runnable
{
	String msg;
	Callme target;
	Thread t;
	public Caller(Callme targ, String s)
	{
		target = targ;     msg = s;
		t = new Thread(this);
		t.start();
	}
```

```
	public void run()
	{
		synchronized(target)
		{
			target.call(msg);
		}
	}
}
class Synch
{
	public static void main(String args[])
	{
		Callme target = new Callme();
		Caller ob1 = new Caller(target, "Hello");
		Caller ob2 = new Caller(target, "Synchronized");
		Caller ob3 = new Caller(target, "World");
		try{
				ob1.t.join();
				ob2.t.join();
				ob3.t.join();
			}catch(InterruptedException e)
			{
				System.out.println("Interrupted");
			}
	}
}
```

Output:

```
C:\javaprog\thread>javac Synch.java

C:\javaprog\thread>java Synch
[Hello]
[Synchronized]
[World]

C:\javaprog\thread>
```