Sample Example1 – Extending Thread

```
class Thread1 extends Thread
{
   public void run()
   {
     System.out.println("Thread running");
   }
}
public class ExThread1
{
   public static void main(String args[])
   {
      Thread1 t=new Thread1();
      t.start();
   }
}
```

Sample Example2 – Implementing Runnable Interface

```
class Thread2 implements Runnable
{
public void run()
{
System.out.println("thread is running...");
}
}

public class ExThread2
{
public static void main(String args[])
{

   Thread2 t;
   Thread t1 =new Thread(t);
   t1.start();
}
}
```

- **ExThread2 not extending Thread class**
- **Explicitly create Thread class object.**
- **Pass the object of the class that implements Runnable interface**

```
Output:thread is running...
```

Sample Example3 – Making current program as thread

```
public 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");
    }
    }
}
```

```
run:
Current thread: Thread[main,5,main]
After name change: Thread[My Thread,5,main]
5
4
3
2
1
BUILD SUCCESSFUL (total time: 5 seconds)
```

Sample Example4 - Creating Multiple Threads

```
class NewThread implements Runnable
{
    Thread t;
    NewThread()
    {
        t = new Thread(this, "Demo Thread");
        System.out.println("Child thread: " + t);
        t.start(); // Start the thread
    }
    public void run()
    {
      for(int i = 5; i > 0; i--)
      {
        System.out.println("Child Thread: " + i);
        Thread.sleep(500);
      }
      System.out.println("Exiting ch. thread.");
    }
}
```

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

```
run:
Child thread: Thread[Demo Thread,5,main]
Main Thread: 5
Child Thread: 5
Child Thread: 4
Child Thread: 3
Main Thread: 4
Child Thread: 2
Child Thread: 1
Main Thread: 3
Exiting child thread.
Main Thread: 2
Main Thread: 1
Main thread exiting.
BUILD SUCCESSFUL (total time: 5 seconds)
```

Sample Example4 – Multiple threads – Make main thread to wait

```
class Threads implements Runnable
{
String name; // name of thread
Thread t;
Threads(String threadname) {
name = threadname;
t = new Thread(this, name);
System.out.println("New thread: " + t);
t.start(); // Start the thread
}
public void run() {
try {
for(int i = 5; 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 ExThreeThreads {
public static void main(String
    args[])
{
new Threads("One"); // start
    threads
new Threads("Two");
new Threads("Three");
try {
// wait for other threads to end
Thread.sleep(10000);
} catch (InterruptedException e) {
System.out.println("Main thread
    Interrupted");
}
System.out.println("Main thread
    exiting.");
}
}
```

```
run:
New thread: Thread[One,5,main]
New thread: Thread[Two,5,main]
New thread: Thread[Three,5,main]
One: 5
Two: 5
Three: 5
Two: 4
One: 4
Three: 4
One: 3
Three: 3
Two: 3
Two: 2
One: 2
Three: 2
Two: 1
Three: 1
One: 1
Two exiting.
Three exiting.
One exiting.
Main thread exiting.
BUILD SUCCESSFUL (total time: 10
```

isAlive()

isAlive() method determines whether a thread is still running. If it is, the isAlive() method returns a Boolean true value; otherwise, a Boolean false is returned.

Join()

The join() method waits until the child thread terminates and “ joins” the main thread.

In addition, you can use the join() method to specify the amount of time you want to wait for a child thread to terminate.

Example5 – Improved Version

```
class Threads1 implements Runnable {
String name; // name of thread
Thread t;
Threads1(String threadname) {
name = threadname;
t = new Thread(this, name);
System.out.println("New thread: " + t);
t.start(); // Start the thread
}
// This is the entry point for thread.
public void run() {
try {
for(int i = 5; i > 0; i--) {
System.out.println(name + ": " + i);
Thread.sleep(1000);
}
} catch (InterruptedException e) {
System.out.println(name + " interrupted.");
}
System.out.println(name + " exiting.");
}
}
```

```
public class ExThreeThreadsImproved
{
  public static void main(String args[])
{
Threads1 ob1 = new Threads1("One");
Threads1 ob2 = new Threads1("Two");
Threads1 ob3 = new Threads1("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());
System.out.println("Waiting for threads to finish.");
ob1.t.join();
ob2.t.join();
ob3.t.join();
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.");
  }
}
```

Main thread starts the t1 and t2 thread
•Two threads start running in parallel.
•The main thread calls t1.join() to wait
•for the t1 thread to finish.
•The t1 thread completes
•and the t1.join() method returns.
•The main thread calls t2.join() to wait
•for the t2 thread to finish.

```
run:
New thread: Thread[One,5,main]
New thread: Thread[Two,5,main]
One: 5
New thread: Thread[Three,5,main]
Two: 5
Thread One is alive: true
Thread Two is alive: true
Thread Three is alive: true
Waiting for threads to finish.
Three: 5
One: 4
Two: 4
Three: 4
Two: 3
Three: 3
One: 3
One: 2
Two: 2
Three: 2
Two: 1
One: 1
```

Example6 – Thread Priorities

```
class Thread3 implements Runnable
{
String name; // name of thread
Thread t;
Thread3() { }
Thread3(String threadname, int p)
{
        name = threadname;
        t = new Thread(this, name);
        t.setPriority(p);
        System.out.println("New thread: " + t);
        t.start(); // Start the thread
}
public void run()
{
        for(int i = 5; i > 0; i--)
        {
        System.out.println(name + ": " + i);
        Thread.sleep(1000);
        }
        System.out.println(name + " exiting.");
}
}
```

```
public class ThreadPriority  {
public static void main(String args[])
{
Thread3 ob1 = new Thread3("One",8);
Thread3 ob2 = new Thread3("Two",7);
Thread3 ob3 = new Thread3("Three",9);
try {
// wait for other threads to end
Thread.sleep(10000);
} catch (InterruptedException e) {
System.out.println("Main thread
        Interrupted");
}
System.out.println("Main thread
        exiting.");
}
}
```

```
run:
New thread: Thread[One,8,main]
New thread: Thread[Two,7,main]
New thread: Thread[Three,9,main]
One: 5
Two: 5
Three: 5
Two: 4
One: 4
Three: 4
Three: 3
One: 3
Two: 3
Three: 2
One: 2
Two: 2
Three: 1
One: 1
Two: 1
Three exiting.
One exiting.
Two exiting.
Main thread exiting.
BUILD SUCCESSFUL (total time: 10
```

Thread Synchronization

- When two or more threads need access to a shared resource, they need some way to ensure that the resource will be used by only one thread at a time.
- The process by which this synchronization is achieved is called *thread synchronization.*
  - Synchronized method.
  - Synchronized block.

Example1 – Without synchronization

```
class Table
{
    void printTable(int n)//method not synchronized
    {
      for(int i=1;i<=5;i++)
      {
        System.out.println(n*i);
        Thread.sleep(400);
      }
    }
}
class MyThread1 extends Thread
{
    Table t;
    MyThread1(Table t)
    { this.t=t;   }
    public void run()
    {
            t.printTable(5);
    }
}
class MyThread2 extends Thread
{
    Table t;
    MyThread2(Table t)
    {  this.t=t;  }
    public void run()
    {  t.printTable(100);  }
}
public class TestSynchronization
{
public static void main(String args[])
{
    Table obj = new Table();//only one object
    MyThread1 t1=new MyThread1(obj);
    MyThread2 t2=new MyThread2(obj);
    t1.start();
    t2.start();
}
}
```

```
run:
5
100
200
10
15
300
400
20
25
500
BUILD SUCCESSFUL (total time: 2 seconds)
```

Example2 – With synchronization

```
class Table1{
 synchronized void printTable(int n)
//synchronized method
{
  for(int i=1;i<=5;i++){
    System.out.println(n*i);
    try{
     Thread.sleep(400);
    }catch(Exception e){System.out.println(e);}
  }
 }
}

class MyThread11 extends Thread{
Table1 t;
MyThread11(Table1 t){
this.t=t;
}
public void run(){
t.printTable(5);
}
}

class MyThread22 extends Thread{
Table1 t;
MyThread22(Table1 t){
this.t=t;
}
public void run(){
t.printTable(100);
}
}

public class TestSynchronization1{
public static void main(String args[]){
Table1 obj = new Table1();//only one
      object
MyThread11 t1=new MyThread11(obj);
MyThread22 t2=new MyThread22(obj);
t1.start();
t2.start();
}
}
```

```
run:
5
10
15
20
25
100
200
300
400
500
BUILD SUCCESSFUL (total t
```

Interthread Communication

- **wait( ):** This method tells the calling thread to give up the monitor and go to sleep until some other thread enters the same monitor and calls notify( ).
- **notify( ):** This method wakes up the first thread that called wait( ) on the same object.
- **notifyAll( ):** This method wakes up all the threads that called wait( ) on the same object. The highest priority thread will run first.

Producer consumer problem

- Consider the classic queuing problem, where one thread is producing some data and another is consuming it.
- To make the problem more interesting, suppose that the producer has to wait until the consumer is finished before it generates more data.

Example3 – wait and notify method

```
class Q {
int n;
boolean valueSet = false;
synchronized int get() {
if(!valueSet)
try {
wait();
} catch(InterruptedException e) {
System.out.println("InterruptedException caught");
}
System.out.println("Got: " + n);
valueSet = false;
notify();
return n;
}
synchronized void put(int n) {
if(valueSet)
try {
wait();
} catch(InterruptedException e) {
System.out.println("InterruptedException caught");
}
this.n = n;
valueSet = true;
System.out.println("Put: " + n);
notify();
}
}
```

```
class Producer1 implements Runnable {
Q q;
Producer1(Q q) {
this.q = q;
new Thread(this, "Producer").start();
}
public void run() {
int i = 0;
while(true) {
q.put(i++); } } }
class Consumer1 implements Runnable {
Q q;
Consumer1(Q q) {
this.q = q;
new Thread(this, "Consumer").start();
}
public void run() {
while(true) {
q.get();
} } }
public class SynchronizeNofifyWait
{
   public static void main(String args[]) {
Q q = new Q();
new Producer1(q);
new Consumer1(q);
System.out.println("Press Control-C to
stop.");
}   }
```

```
Got: 32090
Put: 32091
Got: 32091
Put: 32092
Got: 32092
Put: 32093
Got: 32093
Put: 32094
Got: 32094
Put: 32095
Got: 32095
Put: 32096
Got: 32096
Put: 32097
Got: 32097
Put: 32098
Got: 32098
Put: 32099
Got: 32099
```