Java Thread Example by extending Thread class

```
class Multi extends Thread{
        public void run(){
                System.out.println("thread is running...");
        }
        public static void main(String args[]){
                Multi t1=new Multi();
                t1.start();
        }
}
```

Java Thread Example by implementing Runnable interface

```
class Multi3 implements Runnable{
        public void run(){
                System.out.println("thread is running...");
        }
        public static void main(String args[]){
                Multi3 m1=new Multi3();
                Thread t1 =new Thread(m1);
                t1.start();
        }
}
```

Syntax of sleep() method in java

The Thread class provides two methods for sleeping a thread:

- public static void sleep(long miliseconds)throws InterruptedException
- public static void sleep(long miliseconds, int nanos)throws InterruptedException

Example of sleep method in java

```
class TestSleepMethod1 extends Thread{
         public void run(){
                for(int i=1;i<5;i++){
                        try{
                                        Thread.sleep(500);
                        }catch(InterruptedException e){System.out.println(e);}
                        System.out.println(i);
                }
        }
        public static void main(String args[]){
                TestSleepMethod1 t1=new TestSleepMethod1();
                TestSleepMethod1 t2=new TestSleepMethod1();
                t1.start();
                t2.start();}}
```

The join() method

The join() method waits for a thread to die. In other words, it causes the currently running threads to stop executing until the thread it joins with completes its task.

Syntax:

```
class TestJoinMethod1 extends Thread{
public void run(){
        for(int i=1;i<=5;i++){
                try{
                        Thread.sleep(500);
                }catch(Exception e){System.out.println(e);}
                 System.out.println(i);
        }
}
public static void main(String args[]){
        TestJoinMethod1 t1=new TestJoinMethod1();
         TestJoinMethod1 t2=new TestJoinMethod1();
        TestJoinMethod1 t3=new TestJoinMethod1();
        t1.start();
        try{
                t1.join();
           }catch(Exception e){System.out.println(e);}
        t2.start();
        t3.start();
        }
}
```

Priority of a Thread (Thread Priority):

Each thread have a priority. Priorities are represented by a number between 1 and 10. In most cases, thread scheduler schedules the threads according to their priority (known as preemptive scheduling). But it is not guaranteed because it depends on JVM specification that which scheduling it chooses.

3 constants defined in Thread class:

1. public static int MIN_PRIORITY
2. public static int NORM_PRIORITY
3. public static int MAX_PRIORITY

Default priority of a thread is 5 (NORM_PRIORITY). The value of MIN_PRIORITY is 1 and the value of MAX_PRIORITY is 10.

Example of priority of a Thread:

```
class TestMultiPriority1 extends Thread{
 public void run(){
   System.out.println("running thread name is:"+Thread.currentThread().getName());
   System.out.println("running thread priority is:"+Thread.currentThread().getPriority());

  }
 public static void main(String args[]){
  TestMultiPriority1 m1=new TestMultiPriority1();
  TestMultiPriority1 m2=new TestMultiPriority1();
  m1.setPriority(Thread.MIN_PRIORITY);
  m2.setPriority(Thread.MAX_PRIORITY);
  m1.start();
  m2.start();
   }
}
```

Daemon Thread in Java

Daemon thread in java is a service provider thread that provides services to the user thread. Its life depend on the mercy of user threads i.e. when all the user threads dies, JVM terminates this thread automatically.

There are many java daemon threads running automatically e.g. gc, finalizer etc.

Points to remember for Daemon Thread in Java

- It provides services to user threads for background supporting tasks. It has no role in life than to serve user threads.
- Its life depends on user threads.
- It is a low priority thread.

Methods for Java Daemon thread by Thread class

The java.lang.Thread class provides two methods for java daemon thread.

No.	Method	Description
1)	public void setDaemon(boolean status)	is used to mark the current thread as daemon thread or user thread.
2)	public boolean isDaemon()	is used to check that current is daemon.

Simple example of Daemon thread in java

```
public class TestDaemonThread1 extends Thread{
   public void run(){
     if(Thread.currentThread().isDaemon())//checking for daemon thread
         System.out.println("daemon thread work");
       Else
         System.out.println("user thread work");
    }
    public static void main(String[] args){
       TestDaemonThread1 t1=new TestDaemonThread1();//creating thread
       TestDaemonThread1 t2=new TestDaemonThread1();
       TestDaemonThread1 t3=new TestDaemonThread1();

        t1.setDaemon(true);//now t1 is daemon thread

        t1.start();//starting threads
        t2.start();
        t3.start();
    }
}
```

The yield() method :

Example of yield() method:

A yield() method is a static method of Thread class and it can stop the currently executing thread and will give a chance to other waiting threads of the same priority. If in case there are no waiting threads or if all the waiting threads have low priority then the same thread will continue its execution.

```
class MyThread extends Thread {
   public void run() {
      for (int i = 0; i < 5; ++i) {
         System.out.println("Child Thread started:");
          Thread.yield();

      }

   }
}
public class Sample1 {

       public static void main(String[] args) {

             MyThread thread = new MyThread();
            thread.start();
            for (int i = 0; i < 5; ++i) {
               System.out.println("Main Thread started:");
            }
       }
```

}

Perform multiple tasks by multiple threads (multitasking in multithreading)

```
class Simple1 extends Thread{
 public void run(){
   System.out.println("task one");
 }
}

class Simple2 extends Thread{
 public void run(){
   System.out.println("task two");
 }
}

 class TestMultitasking3{
 public static void main(String args[]){
  Simple1 t1=new Simple1();
  Simple2 t2=new Simple2();

  t1.start();
  t2.start();
 }
}
```

```
class TestMultitasking4{
 public static void main(String args[]){
  Thread t1=new Thread(){
    public void run(){
      System.out.println("task one");
    }
  };
  Thread t2=new Thread(){
    public void run(){
      System.out.println("task two");
    }
  };


  t1.start();
  t2.start();
 }
}
```