**// Sample Program for the setting the priority of the 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();

 }

}

|  |  |
| --- | --- |
| // This program is not synchronized.  class Callme {  void call(String msg) {  System.out.print("[" + msg);  try {  Thread.sleep(1000);  } 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");  // wait for threads to end  try {  ob1.t.join();  ob2.t.join();  ob3.t.join();  } catch(InterruptedException e) {  System.out.println("Interrupted");  }  }  }  Here is the output produced by this program:  Hello[Synchronized[World]  ]  ] | // This program uses a synchronized block.  class Callme {  void call(String msg) {  System.out.print("[" + msg);  try {  Thread.sleep(1000);  } 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();  }  // synchronize calls to call()  public void run() {  synchronized(target) { // synchronized block  target.call(msg);  }  }  }  class Synch1 {  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");  // wait for threads to end  try {  ob1.t.join();  ob2.t.join();  ob3.t.join();  } catch(InterruptedException e) {  System.out.println("Interrupted");  }  }  } |

**Problem**

You are given two threads. You need to print odd numbers using one thread and even numbers using another thread. You need to print in natural order up to MAX.  
**For example:**  
If MAX is 10, you need to print:

1 2 3 4 5 6 7 8 9 10

So 1 3 5 7 9 will be printed by odd thread  
2 4 6 8 10 will be printed by even thread.

**Solution 1: Using remainder**

You can use concept of remainder here.

* If number%2==1 then Odd will print the number and increment it else will go in the wait state.
* If number%2==0 then Even will print the number and increment it else will go in the wait state.

class OddEvenRunnable implements Runnable{

public int PRINT\_NUMBERS\_UPTO=10;

static int  number=1;

int remainder;

static Object lock=new Object();

OddEvenRunnable(int remainder)

{

this.remainder=remainder;

}

@Override

public void run() {

while (number < PRINT\_NUMBERS\_UPTO) {

synchronized (lock) {

while (number % 2 != remainder) { // wait for numbers other than remainder

try {

lock.wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

System.out.println(Thread.currentThread().getName() + " " + number);

number++;

lock.notifyAll();

}

}

}

}

public class PrintOd**d**EvenMain {

public static void main(String[] args) {

 OddEvenRunnable oddRunnable=new OddEvenRunnable(1);

OddEvenRunnable evenRunnable=new OddEvenRunnable(0);

 Thread t1=new Thread(oddRunnable,"Odd");

Thread t2=new Thread(evenRunnable,"Even");

t1.start();

t2.start();

 } }

Output:

Odd 1  
Even 2  
Odd 3  
Even 4  
Odd 5  
Even 6  
Odd 7  
Even 8  
Odd 9  
Even 10

// Interthread Communication – Sample Program

// Refer ppt also

class Q {  
int n;  
boolean valueSet = false;  
synchronized int get() {  
while(!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) {  
while(valueSet)  
try {  
wait();  
} catch(InterruptedException e) {  
System.out.println("InterruptedException caught");  
}  
this.n = n;  
valueSet = true;  
System.out.println("Put: " + n);  
notify();  
}  
}  
class Producer implements Runnable {  
Q q;  
Producer(Q q) {  
this.q = q;  
new Thread(this, "Producer").start();  
}  
public void run() {  
int i = 0;  
while(true) {  
q.put(i++);  
}  
}  
}  
class Consumer implements Runnable {  
Q q;  
Consumer(Q q) {  
this.q = q;  
new Thread(this, "Consumer").start();  
}  
public void run() {  
while(true) {  
q.get();  
}  
}  
}  
class PCFixed {  
public static void main(String args[]) {  
Q q = new Q();  
new Producer(q);  
new Consumer(q);  
System.out.println("Press Control-C to stop.");  
}  
}

Put: 1  
Got: 1  
Put: 2  
Got: 2  
Put: 3  
Got: 3  
Put: 4  
Got: 4  
Put: 5  
Got: 5