Notify, notifyAll, wait, sleep

class notifyEx extends Thread{

Object lock;

int num;

notifyEx(Object lock,int num){

this.lock = lock;

this.num = num;

}

@Override

public void run(){

try{

synchronized(lock) {lock.wait();}

for(int i =0;i<10;i++){

System.out.print(this.num);

Thread.sleep(1);

}

synchronized(lock) {lock.notify();}

}catch(Exception e){

System.err.print("error");

}

}

}

public class Test {

public static void main(String[] args) throws InterruptedException

{

Object lock = new Object();

notifyEx p1 = new notifyEx(lock,1);

notifyEx p2 = new notifyEx(lock,2);

notifyEx p3 = new notifyEx(lock,3);

p1.start();

p2.start();

p3.start();

try{Thread.sleep(10);}catch(Exception e){}

//notify: only one thread will be revoked

synchronized(lock) {lock.notify();}

//111111111133333333332222222222

//notifyAll: all threads will be revoked

//synchronized(lock) {lock.notifyAll();}

//123123123123123123123213213213

}

}

yield

class yieldEx extends Thread{

int num;

yieldEx(int num){

this.num = num;

}

@Override

public void run(){

try{

for(int i =0;i<40;i++){

System.out.println("("+this.num+" : "+i+")");

if (i % 10 == 0) {

//force to yield the CPU time

Thread.yield();

}

}

}catch(Exception e){

System.err.print("error");

}

}

}

public class Test {

public static void main(String[] args) throws InterruptedException

{

yieldEx p1 = new yieldEx(1);

yieldEx p2 = new yieldEx(2);

// time slot will be very short

// p1.setPriority(Thread.MIN\_PRIORITY);

// p2.setPriority(Thread.MIN\_PRIORITY);

//time slot will be very long

p1.setPriority(Thread.MAX\_PRIORITY);

p2.setPriority(Thread.MAX\_PRIORITY);

p1.start();

p2.start();

}

}

// Thread must change when mod 10 = 0

// (2 : 9)

// (2 : 10)

// (1 : 1)

// (1 : 2)

// .

// .

// .

// (1 : 18)

// (1 : 19)

// (1 : 20)

// (2 : 21)

// (2 : 22)

// .

// .

// .

// (1 : 28)

// (1 : 29)

// (1 : 30)

// (2 : 31)

// (2 : 32)

Join

import java.util.concurrent.\*;

import java.util.\*;

class joinEx extends Thread{

int num,times;

joinEx(int num,int times){

this.num = num;

this.times = times;

}

@Override

public void run(){

try{

for(int i =0;i<times;i++){

System.out.println("("+this.num+" : "+i+")");

Thread.sleep(1);

}

}catch(Exception e){

System.err.print("error");

}

}

}

public class Test {

public static void main(String[] args) throws InterruptedException

{

joinEx p1 = new joinEx(1,10);

joinEx p2 = new joinEx(2,5);

p1.start();

p2.start();

p2.join();

//p2 has join the main, so main will wait until p2 finish.

System.out.println("end of main");

}

}

// (1 : 0)

// (2 : 0)

// (1 : 1)

// (2 : 1)

// (1 : 2)

// (2 : 2)

// (1 : 3)

// (2 : 3)

// (1 : 4)

// (2 : 4)

// (1 : 5)

// end of main

// (1 : 6)

// (1 : 7)

// (1 : 8)

// (1 : 9)