Name Roll:Number

Q6

Code:

class Shared {

synchronized void doWork() {

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

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

System.out.println(e);

}

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

}

}

class MyThreading extends Thread {

Shared sh;

public MyThreading(Shared sh, String name) {

super(name);

this.sh = sh;

start();

}

public void run() {

sh.doWork();

}

}

public class SynchronizeEx {

public static void main(String[] args) {

Shared sh = new Shared();

MyThreading t1 = new MyThreading(sh, "Thread1");

MyThreading t2 = new MyThreading(sh, "Thread2");

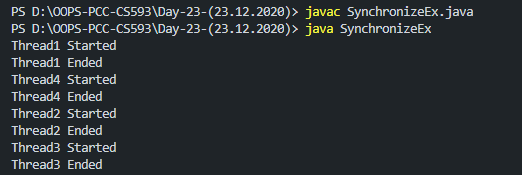
MyThreading t3 = new MyThreading(sh, "Thread3");

MyThreading t4 = new MyThreading(sh, "Thread4");

}

}

Output:



Q7

Code:

public class Ass6Q7 {

public static void main(String[] args){

Sharable sh=new Sharable();

Consumer c=new Consumer("Consumer", sh);

Producer p=new Producer("Producer", sh);

}

}

class Sharable{

int items=0;

synchronized void produce() {

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

try{

Thread.sleep(1000);

}

catch(InterruptedException e){

System.out.println(e);

}

items+=1;

notify();

System.out.println(Thread.currentThread().getName()+" has finished production");

}

synchronized void consume() {

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

try{

wait();

}

catch(InterruptedException e){

System.out.println(e);

}

items-=1;

System.out.println(Thread.currentThread().getName()+" has finished consumption and "+items+" are left");

}

}

class Producer extends Thread{

Sharable sh;

public Producer(String name, Sharable sh){

super(name);

this.sh=sh;

start();

}

public void run(){

sh.produce();

}

}

class Consumer extends Thread{

Sharable sh;

public Consumer(String name, Sharable sh){

super(name);

this.sh=sh;

start();

}

public void run(){

sh.consume();

}

}

Output:

