class Resource{

private int value;

private boolean notEmpty = false;

public synchronized int get() { while(notEmpty == false) {

try {

System.out.println("Consumer waiting for producer to produce"); wait();

}

catch(InterruptedException ie)

{

ie.printStackTrace();

}

}

System.out.println("Consumer consumed "+value); notEmpty = false;

notify(); return value;

}

public synchronized void put(int value) { while(notEmpty == true) {

try {

System.out.println("Producer waiting for consumer to cosume"); wait();

}

catch(InterruptedException ie)

{

ie.printStackTrace();

}

}

System.out.println("Producer produced "+value); this.value = value;

notEmpty = true; notify();

}

}

class Producer extends Thread{ private Resource resource; private int number;

public Producer(Resource resource, int number) { this.resource = resource;

this.number = number; System.out.println("Created Producer Thread");

}

public void run() {

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

resource.put(i); try {

Thread.sleep(1000);

} catch (InterruptedException e) {

// TODO Auto-generated catch block e.printStackTrace();

}

}

}

}

class Consumer extends Thread{ private Resource resource; private int number;

public Consumer(Resource resource, int number) { this.resource = resource;

this.number = number; System.out.println("Created Consumer Thread");

}

public void run() {

int value = 0;

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

value = resource.get();

}

}

}

public class Experiment12 {

public static void main(String[] args) {

// TODO Auto-generated method stub Resource resource = new Resource(); Producer p = new Producer(resource,1); Consumer c1 = new Consumer(resource,1);

//Consumer c2 = new Consumer(resource,2); p.start();

c1.start();

//c2.start();

}

}