















7.9//

package barrier1;

public class CustomBarrier {

    private final int n;

    private int count;

    private final Object lock = new Object();

    public CustomBarrier(int n) {

        this.n = n;

        this.count = 0;

    }

    public void wait1() throws InterruptedException {

        synchronized (lock) {

            count++;

            if (count == n) {

                count = 0;

                lock.notifyAll();

            } else {

                while (count != 0) {

                    lock.wait();

                }

            }

        }

    }

    public static void main(String[] args) {

        int numThreads = 5;

        CustomBarrier barrier = new CustomBarrier(numThreads);

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

            new Thread(() -> {

                try {

                    System.out.println("线程 " + Thread.currentThread().getName() + " 到达屏障点");

                    barrier.wait1();

                    System.out.println("线程 " + Thread.currentThread().getName() + " 继续执行");

                } catch (InterruptedException e) {

                    e.printStackTrace();

                }

            }, "线程" + i).start();

        }

    }

}

package barber*S*hop;

public class barberTest implements Runnable {

    static final private int WAIT\_TIME = 3;

    static public void main(String[] args) {

        new Thread(new barberTest()).start();

    }

    public void run(){

        BarberShop newShop = new BarberShop(15);

        int customerID = 1;

        while(customerID <= 10000){

            new Thread(new Customer( newShop,customerID)).start();

            customerID++;

            SleepUtilities.nap();

        }

    }

}

package barber*S*hop;

public class Customer implements Runnable{

    private BarberShop shop;

    private int Customer;

    private int HairCut\_TIME =5;

    public Customer(BarberShop pShop, int pCustomer) {

        shop = pShop;

        Customer = pCustomer;

    }

    public void run(){

        int sleeptime = (int)(HairCut\_TIME \* Math.random());

        System.out.println("ENTERING SHOP: Customer [" + Customer + "] entering barber shop for haircut.");

        int test = BarberShop.OCCUPIED;

        test = shop.getHairCut(Customer);

        if(test == BarberShop.WAITED){

            System.out.println("Barber's busy: Customer [" + Customer + "] has waited and now wants haircut.");

        }

        else if (test == BarberShop.SLEEPING)

        System.out.println("Barber's asleep: Customer [" + Customer + "] is waking him up and getting haircut.");

        else if (test == BarberShop.FULL){

            System.out.println("Barber Shop full: Customer [" + Customer + "] is leaving shop.");

            return;

        }

        else{

            System.out.println("HAIRCUT: Customer [" + Customer + "] is getting haircut.");

        }

        SleepUtilities.nap();

        System.out.println("LEAVING SHOP: Customer [" + Customer + "] haircut finished: leaving shop.");

        shop.leaveBarberShop(Customer);

    }

}

package barber*S*hop;

public class SleepUtilities {

    private static int NAP\_TIME = 5;

    public static void nap(){

        nap(NAP\_TIME);

    }

    public static void nap(int duration){

        int sleeptime = (int) (NAP\_TIME \* Math.random() );

        try { Thread.sleep(sleeptime\*1000); }

        catch (InterruptedException e) {}

    }

}



