LAB - 10

Page No.
Date 13/02/24

```
class Q {
    int n;
    boolean valueSet = false;
    synchronized int get() {
        while(!valueset) {
            try {
            System.out.println("\n consumer waiting\n");
            wait();
            } catch(InterruptedException e) {
                System.out.println(" caught ");
            }
            System.out.println("Got: " + n);
            valueset = false;
            System.out.println("\n Inform producer\n");
            notify();
            return n;
        }
    }
    Synchronized void put(int n) {
    while (valueset)
    try {
        System.out.println("\n producer waiting\n");
        wait();
    } catch(InterruptedException e) {
            System.out.println(" caught ");
    }
    this.n = n;
    valueset = true;
    System.out.println("Put : " + n);
    System.out.println("\n Inform consumer\n");
    notify();
    }
}
```

```
class consumer implements Runnable {
    Q q;
    Consumer (Q q) {
        this.q = q;
        new Thread (this, "consumer").start();
    }
    public void run() {
        int i = 0;
        while(i < 2) {
            int r = q.get();
            System.out.println("consumed : " + r);
            i++;
        }
    }
}

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 : 0
Intimate consumer
producer waiting
Got : 0
Intimate producer

Put : 1
Intimate Consumer
Consumer : 0
Got : 1
Intimate producer
Consumer : 1
consumer waiting

LAB-10:

// Deadlock.

```
class A {
    synchronized void foo(B b) {
        String name = Thread.currentThread().getName();
        System.out.println(name + " Entered A.foo");
        try {
            Thread.sleep(1000);
        } catch (Exception e) {
            System.out.println("A interrupted");
        }
        System.out.println(name + " trying to call B.foo()");
        b.last();
    }
    void last() {
        System.out.println("Inside A.last");
    }
}

class B {
    synchronized void bar(A a) {
        String name = Thread.currentThread().getName();
        System.out.println(name + "entered B.bar");
        try {
            Thread.sleep(1000);
        } catch (Exception e) {
            System.out.println("B interrupted");
        }
        System.out.println(name + "trying to call A.last()");
        a.last();
    }
```

```
    void last() {
        System.out.println("Inside A.last");
    }
}

class Deadlock implements Runnable {
    A a = new A();
    B b = new B();
    Deadlock() {
        Thread.currentThread().setName("MainThread");
        Thread t = new Thread(this, "RacingThread");
        t.start();
        a.foo(b);
        System.out.println("Back in main thread");
    }
    public void run() {
        b.bar(a);
        System.out.println("Back in other thread");
    }
    public static void main(String[] args) {
        new Deadlock();
    }
}
```

13/2/24

O/P:

```
MainThread entered A.foo
RacingThread entered B.bar
MainThread trying to call B.last()
Inside A.last()
Back in main Thread.
RacingThread trying to call A.last()
Inside A.last()
Back in other thread.
```