Lab-10

IPC

```
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("Interruption caught");}
        System.out.println("Got " + n);
        valueSet = false;
        System.out.println("\n Intimate Producer");
        notify();
        return n; }
    synchronized void put(int n) {
        while(valueSet) {
        try {
            System.out.println("\n Producers waiting\n");
            wait(); } catch (InterruptedException e)
                System.out.println("Interrupted Exception
                    caught"); }
        this.n = n
        valueSet = true;
```

```
        System.out.println("Put: " + n);
        System.out.println("\n Intimate Consumer\n");
        notify(); } }

class Prodesces implements Runnable {
    Q q;
    Producer (Q q) {
        this.q = q;
        new Thread (this, "Producer").start(); }
    public void run(){
        int i = 0;
        while (i<=15){ q.put(i++); }}}

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<15){
            int r = q.get();
            System.out.println ("consumer:" +r);
            i++; } }}
```

```
class Main {
  public static void main(String args[]){
    Q q = new Q();
    new Producer (q);
    new Consumer(q);
    System.out.println("Press control-C to stop");
  }}
```

Output

```
Put: 0
Intimate Consumer
Producer waiting
Got: 0
Intimate Producer
Put: 1
Intimate Consumer
Producer waiting
Consumed: 0
Got: 1
Intimate Producer
Consumed: 1
Put: 2
Intimate Consumer
Producer waiting
Got: 2
```

Run 6/7/2021

13/2/24

10 - Deadlock

```
class A {
    synchronized void foo(B b) {
    String name = Thread.currentThread().getName();
    System.out.println(name + "entered A.foo");
try {
  Thread.sleep(5000); }
catch (Exception e){
    System.out.println("A interrupted"); }
System.out.println("trying to call B.last()", name);
b.last(); }
void last() {
    System.out.println(" Inside A.last "); } }

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

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

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

OUTPUT

```
MainThread entered a.foo
RacingThread entered b.foo
MainThread trying to calling B.last()
```

Inside A.last
Back in main thread
RacingThread trying to call A.last().
Inside A.last
Back in other thread.

13/2/2024