27Jul24 Lab Program-10

Q) Demonstrate Inter process Communication and Deadlock

Output

A) IPC

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

B) Deadlock

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

Seen
[illegible]
27/11/24

Lab Program -10

Source Code

A) IPC

```
class Q{
   int n;
   boolean valueSet = false;
   synchronized int get(){
      while (!valueSet)
      try{
         System.out.println("In Consumer waiting\n");
         wait();
      }
      catch (InterruptedException e){
         System.out.println("InterruptedException caught");
      }
      System.out.println("Got: " +n);
      valueSet = false;
      System.out.println("In Intimate Producer\n");
      notify();
      return;
   }
   Synchronized void put (int n){
      while (valueSet)
      try{
         System.out.println("InProducer waiting\n");
         wait();
      }
      catch (InterruptedException e) {
         System.out.println ("InterruptedException caught");
      }
```

```
        this.n = n;
        valueSet = true;
        System.out.println ("Put: ", n);
        System.out.println ("Interrupted Consumer\n");
        notify();
    }
}

class Producer implement 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("consumed:"+r);
        i++;
      }
    }
}

class PCFixed{
    public static void main(String args[]){
        System.out.println("USN: 1BM23CS003 \nName: Aaron B Ajay");
        Q q = new Q();
        new Producer(q);
        new Consumer(q);
        System.out.println("Press Control-C to stop.");
    }
}
```

B) <u>DeadLock</u>

```
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.last()");
        b.last();
    }
    void last ~~8~~ (){
        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 ("Main Thread");
        Thread t = new Thread(this, "RacingThread");
        t.start ~~[illegible]~~ ();
        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[]){
    System.out.println(" USN: 1BM23CS003 \n Name: Aaron B Ajay");
    new Deadlock();
}
}
```