13-2-24 Lab Program - 10 (10B)

Demonstrate Inter process communication and deadlock

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

     synchronized void put (int n){
        while (valueSet)
        try {
          System.out.println("\n Producer waiting");
          wait();
          }catch (InterruptedException e) {
                System.out.println("InterruptedException caught");
          }
          this.n =n;
          valueSet = true;
          System.out.println ("Put: " +n);
          System.out.println ("\n Intimate Consumer \n");
        notify(); }
     }
```

```
class Producer implements Runnable {
    Q q;
    Producer (Q q) {
      this.q = q;
      new Thread (this, "Producer"). start();
    }
    public void run () {
      int i = 0;
      while (i < 5) {
        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 < 5) {
        int r = q.get();
        System.out.println ("Consumed : " + r);
        i++;
      }
    }
}


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

Output:

```
  Press Control-C to stop
  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
Intimate Producer
consumed: 2
Put: 3
Intimate Consumer

Producer waiting
Got: 3
Intimate Producer
consumed: 3
Put: 4
Intimate Consumer
Got: 4
Intimate Producer
consumed: 4
```

Program (10B)
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.last()");
        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 ("Main Thread");
            Thread t = new Thread (this, "Racing Thread");
            t.start();
            a.foo(b);
            System.out.println ("Back is main thread");
        }
        public void run()
        {
            b.bar (a);
            System.out.println ("Back is other thread");
        }
        public static void main (String args[])
        {
            new Deadlock ();
        }
    }
```

Output:

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

8
13/2/24