13/2/24

LAB - 10

★ Demonstrate Interprocess Communication.
On: Implementation of a producer & consumer.

→
```
import java.util.*;
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("Interrupted Exception
                                caught\n");
          }
        System.out.println("Got: " + n);
        valueset = false;
        System.out.println("Intimate Producer\n");
        notify();
        return n;
    }
 synchronized void put(int n){
      while(valueset){
          try{
            System.out.println("Producer waiting\n");
            wait();
          }
```

```
catch (InterruptedException e){
    System.out.println("Interrupted exception\n");
  }
this.n = n;
valueSet = true;
System.out.println("Put: "+ n);
System.out.println("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[]){
      Q q = new Q();
      new Producer(q);
      new Consumer(q);
      System.out.println("Press Ctrl-C to stop");
  }
}
```

Output:

Put : 0
Intimate Consumer

Producer waiting

Press Ctrl-C to stop
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

13/2/24

* Deadlock

Q) Demonstrate Deadlocks.

→
```
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("MainThread");
        Thread t = new Thread(this, "RacingThread");
        t.start();
        a.foo(b);
        System.out.println("Back in mainthread");
    }
    public void run(){
        b.bar(a);
        System.out.println("Back in otherthread");
    }

public static void main(String args[]){
    new Deadlock();
    System.out.println("NAME: Shivaraj K Pujari");
    System.out.println("USN: 1BM22CS259");
}
}
```

Output:

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

NAME: Shivaraj K Pujari
USN: 1BM22CS259

13/2/2023