

LAB program 10:

Demonstrate inter process communication

class Q {

int n;

boolean valueSet = false

synchronized int get() {

while (!valueSet) {

try {

System.out.println("In consumer
waiting");

wait();

}

catch (InterruptedException e) {

System.out.println("InterruptedException
caught");

}

System.out.println("Got: " + n);

valueSet = true;

System.out.println("In intimate Producer");

notify();

return n;

}

synchronized void put(int n) {

while (valueSet)

try {

System.out.println("In producer
waiting");

wait();

} catch (InterruptedException e) {

System.out.println("InterruptedException
caught");

}

Thread in

```

        this.n = n;
        valueSet = true;
        System.out.println("put: " + n);
        System.out.println("\n Intimate Consumer");
        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 < 3) {
            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 < 3) {

```

```

int r = q.get();
System.out.println("consumed " + r);
i++;
}
}
}

```

```

public class Corrected {
    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 :

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's
Intimate Consumer
producer waiting

Got: 3
Intimate producer
consumed : 3

Name : Shashidhar B M
USN : 18M22CS257

Dead Lock

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 Interrupt");

}

~~System.out.println(name + " trying to
call B.last()")~~

b.last();

}

void last() {

System.out.println("Entered 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.bar");
}

```

```

    a.bar();
}

```

```

}

```

```

void bar() {
}

```

```

    System.out.println("Inside A.bar");
}

```

```

}

```

```

}

```

```

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);
}

```

```

3 System.out.println("Back in main Thread");
3 public void run(){
    b.bar();
    System.out.println("Back in other
    thread");

```

```

3 public static void main (String args[])
    new Deadlock();

```

```

3

```


3.

Output:

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

Name: ~~Shahidhar B M~~

USN: ~~18M22CS257~~


 13/2/2024