**Consumer Producer**

import java.util.concurrent.BlockingQueue;

import java.util.concurrent.LinkedBlockingQueue;

import java.util.logging.Level;

import java.util.logging.Logger;

public class ProducerConsumerPattern {

    public static void main(String args[]){

     //Creating shared object

     BlockingQueue sharedQueue = new LinkedBlockingQueue();

     //Creating Producer and Consumer Thread

     Thread prodThread = new Thread(new Producer(sharedQueue));

     Thread consThread = new Thread(new Consumer(sharedQueue));

     //Starting producer and Consumer thread

     prodThread.start();

     consThread.start();

    }

}

//Producer Class in java

class Producer implements Runnable {

    private final BlockingQueue sharedQueue;

    public Producer(BlockingQueue sharedQueue) {

        this.sharedQueue = sharedQueue;

    }

    @Override

    public void run() {

        for(int i=0; i<10; i++){

            try {

                System.out.println("Produced: " + i);

                sharedQueue.put(i);

            } catch (InterruptedException ex) {

                Logger.getLogger(Producer.class.getName()).log(Level.SEVERE, null, ex);

            }

        }

    }

}

//Consumer Class in Java

class Consumer implements Runnable{

    private final BlockingQueue sharedQueue;

    public Consumer (BlockingQueue sharedQueue) {

        this.sharedQueue = sharedQueue;

    }

    @Override

    public void run() {

        while(true){

            try {

                System.out.println("Consumed: "+ sharedQueue.take());

            } catch (InterruptedException ex) {

                Logger.getLogger(Consumer.class.getName()).log(Level.SEVERE, null, ex);

            }

        }

    }

}

**Find a loop in the Linked List.**