**LinkedList Class:**

public class LinkedList {  
 Node head;  
 private int count;  
 LinkedList(){  
 head=null;  
 count=0;  
 }  
 boolean isListEmpty(){  
 return (count==0);  
 }  
 int listSize(){  
 return count;  
 }  
 void insertLast(Order ord){  
 Node newNode=new Node();  
 newNode.entry=ord;  
 newNode.next=null;  
 if (head==null){  
 head=newNode;  
 }  
 else {  
 Node n=head;  
 while (n.next!=null){  
 n=n.next;  
 }  
 n.next=newNode;  
 }  
 count++;  
 }  
 void insert(Order ord, int p){Node newNode=new Node();  
 newNode.entry=ord;  
 newNode.next=null;  
 if (p<0 || p>listSize()){  
 System.*out*.println("Not in the range.");  
 }  
 else {  
 Node n=head;  
 for (int i=0;i<p-1;i++){  
 n=n.next;  
 }  
 newNode.next=n.next;  
 n.next=newNode;  
 count++;  
 }  
 }  
 void delete(int p){  
 if (isListEmpty()){  
 System.*out*.println("List is empty.");  
 } else if (p<0 || p>listSize()) {  
 System.*out*.println("Not in the range.");  
 } else if (p==0) {  
 head=head.next;  
 }  
 else {  
 Node n=head;  
 Node n1=null;  
 for (int i=0;i<p-1;i++){  
 n=n.next;  
 }  
 n1=n.next;  
 n.next=n1.next;  
 n1=null;  
 count--;  
 }  
 }  
 void traverseList(){  
 Node n=head;  
 while (n.next!=null){  
 System.*out*.println(n.entry);  
 n=n.next;  
 }  
 System.*out*.println(n.entry);  
 }  
}

**LinkedQueue class:**

public class LinkedQueue {  
 private Node front;  
 private Node rear;  
 private int count;  
  
 LinkedQueue() {  
 front = null;  
 rear = null;  
 count = 0;  
 }  
  
 boolean isQueueEmpty() {  
 return (count == 0);  
 }  
  
 Order serve() {  
 if (isQueueEmpty()) {  
 System.*out*.println("Queue is empty.");  
 return null;  
 } else {  
 Order element = front.entry;  
 front = front.next;  
 count--;  
 return element;  
 }  
 }  
  
 void append(Order ord) {  
 Node oldRear = rear;  
 rear = new Node();  
 rear.entry = ord;  
 rear.next = null;  
 if (isQueueEmpty()) {  
 front = rear;  
 } else {  
 oldRear.next = rear;  
 }  
 count++;  
 }  
}

**Node Class:**

public class Node {  
 Order entry;  
 Node next;  
}

**Order Class:**

public class Order {  
 private int orderName;  
 private String customerName;  
 private String orderDetails;  
 private String status;  
 public Order(int orderName, String customerName, String orderDetails,  
 String status) {  
 this.orderName = orderName;  
 this.customerName = customerName;  
 this.orderDetails = orderDetails;  
 this.status = status;  
 }  
 public int getOrderName() {  
 return orderName;  
 }  
 public String getCustomerName() {  
 return customerName;  
 }  
 public String getOrderDetails() {  
 return orderDetails;  
 }  
 public String getStatus() {  
 return status;  
 }  
 public void setStatus(String status) {  
 this.status = status;  
 }  
}

**Main Class:**

import java.util.Scanner;  
  
public class Main {  
 public static void processNextOrder(LinkedList list,LinkedQueue queue){  
 if (!queue.isQueueEmpty()){  
 Order ord=queue.serve();  
 int ordNumber=ord.getOrderName();  
 Node currentNode= list.head;  
 while (currentNode!=null){  
 if (currentNode.entry.getOrderName()==ordNumber){  
 currentNode.entry.setStatus("Processed");  
 break;  
 }  
 currentNode=currentNode.next;  
 }  
 System.*out*.println("Order number "+ordNumber+" has been processed.");  
 }  
 else {  
 System.*out*.println("No orders to process.");  
 }  
 }  
 public static void printOrderStatus(LinkedList list,int orderNumber){  
 Node currentNode= list.head;  
 while (currentNode!=null){  
 if (currentNode.entry.getOrderName()==orderNumber){  
 System.*out*.println("Status: "+currentNode.entry.getStatus());  
 break;  
 }  
 currentNode=currentNode.next;  
 }  
 }  
 public static void cancelOrder(LinkedList list,int orderNumber){  
 Node currentNode= list.head;  
 while (currentNode!=null){  
 if (currentNode.entry.getOrderName()==orderNumber){  
 currentNode.entry.setStatus("Canceled");  
 break;  
 }  
 currentNode=currentNode.next;  
 }  
 }  
 public static void printOrderStatusAfterCancellation(LinkedList list,int orderNumber){  
 Node currentNode= list.head;  
 while (currentNode!=null){  
 if (currentNode.entry.getOrderName()==orderNumber){  
 System.*out*.println(currentNode.entry.getStatus());  
 break;  
 }  
 currentNode=currentNode.next;  
 }  
 }  
 public static void main(String[] args) {  
 LinkedList list=new LinkedList();  
 LinkedQueue queue=new LinkedQueue();  
 list.insertLast(new Order(101,"Nimal","Product A","Processing"));  
 queue.append(new Order(101,"Nimal","Product A","Processing"));  
 list.insertLast(new Order(102,"Kamala","Product B","Pending"));  
 queue.append(new Order(102,"Kamala","Product B","Pending"));  
 list.insertLast(new Order(103,"Sunil","Product C","Processing"));  
 queue.append(new Order(103,"Sunil","Product C","Processing"));  
 list.insertLast(new Order(104,"Amal","Product D","Pending"));  
 queue.append(new Order(104,"Amal","Product D","Pending"));  
 list.insertLast(new Order(105,"Nayana","Product D","Processing"));  
 queue.append(new Order(105,"Nayana","Product D","Processing"));  
 System.*out*.print("The person who placed first order in the system: ");  
 Order firstOrder=list.head.entry;  
 System.*out*.println(firstOrder.getCustomerName());  
 int processesCount=0;  
 Node currentNode= list.head;  
 while (currentNode!=null){  
 if (currentNode.entry.getStatus().equals("Processing")){  
 processesCount++;  
 }  
 currentNode=currentNode.next;  
 }  
 System.*out*.print("Processers which are currently being existed: ");  
 System.*out*.println(processesCount);  
 *processNextOrder*(list,queue);  
 Scanner input=new Scanner(System.*in*);  
 System.*out*.println("Enter order number: ");  
 int num= input.nextInt();  
 *printOrderStatus*(list,num);  
 System.*out*.println("Enter order number that wants to cancel: ");  
 int ordNum= input.nextInt();  
 *cancelOrder*(list,ordNum);  
 *printOrderStatusAfterCancellation*(list,ordNum);  
 }  
  
}