```
import java.util.*;
class Node {     //class node
protected int data;
protected Node link;

public Node()    {    link = null;       data = 0;    }    //  Constructor
public Node(int d,Node n)     {   data = d;     link = n;  }   // Constructor
public void setLink(Node n) {    link = n;      }    //Function to set link to
next Node
public void setData(int d)  {    data = d;    }           //Function to set data
to current Node
public Node getLink()  { return link;  }            //Function to get link to
next node
public int getData()  {   return data;  }           // Function to get data from
current Node
}
 class linkedQueue  {                                          //Class
linkedQueue
    protected Node front, rear;
    public int size;

    public linkedQueue()   {   front = null;    rear = null;     size =
0;    }    //Constructor
    public boolean isEmpty()  {    return front == null;    }     /*  Function to
check if queue is empty */
    public int getSize()   {    return size;     }    /*  Function to get the
size of the queue */

    /*  Function to insert an element to the queue */
    public void insert(int data) {
        Node nptr = new Node(data, null);
        if (rear == null)   {   front = nptr;   rear = nptr;   }
        else   {
            rear.setLink(nptr);
            rear = rear.getLink();
        }
        size++ ;
}

  /*  Function to remove front element from the queue */
      public int remove()   {
        if (isEmpty() )  throw new NoSuchElementException("Underflow Exception");
        Node ptr = front;
        front = ptr.getLink();
        if (front == null)
```

```
            rear = null;
        size-- ;
        return ptr.getData();
    }
    /*  Function to check the front element of the queue */
    public int peek() {
        if (isEmpty() )    throw new NoSuchElementException("Underflow
Exception");
        return front.getData();
    }
    /*  Function to display the status of the queue */
    public void display() {
        System.out.print("\nQueue = ");
        if (size == 0) { System.out.print("Empty\n");    return ;   }
        Node ptr = front;
        while (ptr != rear.getLink() ) {
           System.out.print(ptr.getData()+" ");
            ptr = ptr.getLink();
        }
        System.out.println();
    }
}

/*  Class LinkedQueueImplement  */
public class LinkedQueueImplement {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        /* Creating object of class linkedQueue */
        linkedQueue lq = new linkedQueue();
        /* Perform Queue Operations */
        System.out.println("Linked Queue Test\n");
        char ch;
        do  {
            System.out.println("\nQueue Operations");
            System.out.println("1. insert");
            System.out.println("2. remove");
            System.out.println("3. peek");
            System.out.println("4. check empty");
            System.out.println("5. size");
            int choice = scan.nextInt();
            switch (choice)  {
            case 1 :
                System.out.println("Enter integer element to insert");
                lq.insert( scan.nextInt() );
                break;
```

```
            case 2 :
                try  {  System.out.println("Removed Element = "+
lq.remove());     }
                catch (Exception e)     {    System.out.println("Error : " +
e.getMessage());   }
                break;
            case 3 :
                try  {   System.out.println("Peek Element = "+ lq.peek());    }
                catch (Exception e)  {   System.out.println("Error : " +
e.getMessage());    }
                break;


            case 4 :  System.out.println("Empty status = "+
lq.isEmpty());  break;
            case 5 :  System.out.println("Size = "+ lq.getSize());    break;
            default :  System.out.println("Wrong Entry \n ");   break;
            }
            /* display queue */
            lq.display();
            System.out.println("\nDo you want to continue (Type y or n) \n");
            ch = scan.next().charAt(0);
        } while (ch == 'Y'|| ch ==
'y');
    } q
}
```

Output:

```
Linked Queue Test


Queue Operations
1. insert
2. remove
3. peek
4. check empty
5. size
1
Enter integer element to insert
2

Queue = 2

Do you want to continue (Type y or n)

y

Queue Operations
1. insert
2. remove
3. peek
4. check empty
5. size
1
Enter integer element to insert
20

Queue = 2 20

Do you want to continue (Type y or n)

y

Queue Operations
1. insert
2. remove
3. peek
4. check empty
5. size
3
Peek Element = 2

Queue = 2 20

Do you want to continue (Type y or n)

y

Queue Operations
1. insert
2. remove
3. peek
4. check empty
5. size
4
Empty status = false

Queue = 2 20
```