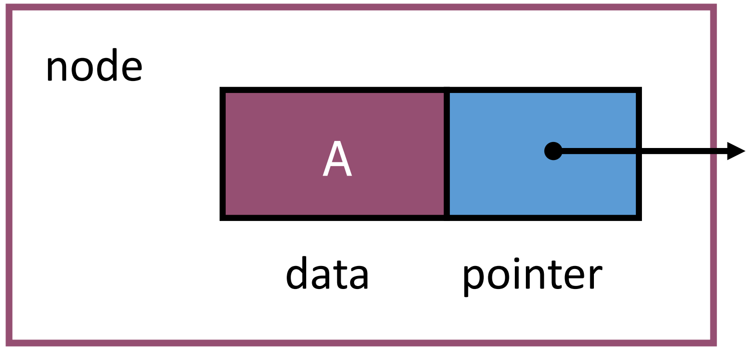
Algoritma dan Struktur Data

L12. Singly Linked Structure

Amil Ahmad Ilham

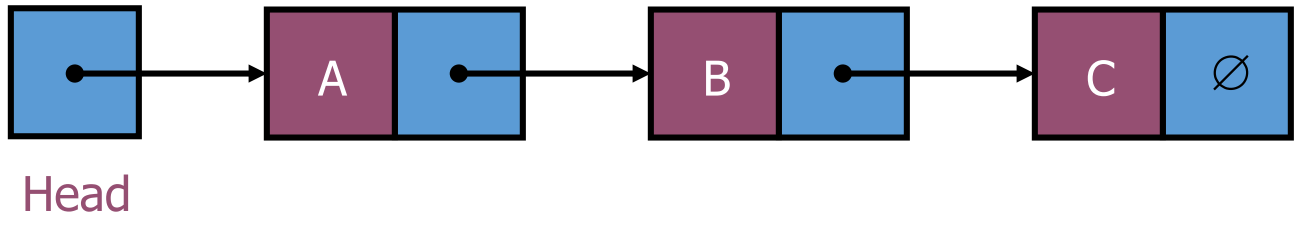
Singly Linked Structure

- A *linked list* is a series of connected *nodes.*
- Each node contains at least
  - A piece of data (any type)
  - Pointer to the next node in the list



Singly Linked Structure

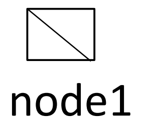
- ***Head***: pointer to the first node
- The last node points to `NULL`



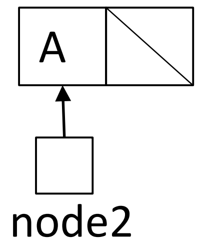
Singly Linked Node Object

```
// A Node class for a singly linked list
public class Node {
   public Object element;
   public Node next;

   //Constructor
   public Node(){
      element = null;
      next = null;
   }

   public Node(Object e, Node n){
      element = e;
      next = n;
   }
}
```

Node node1 = null;



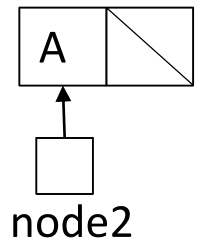
Node node2 = new Node(“A”, null);



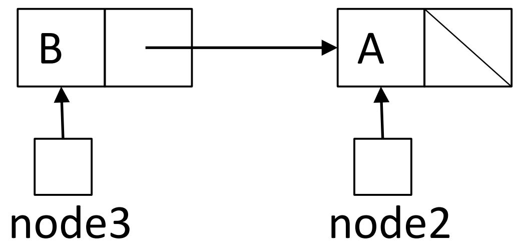
Singly Linked Node Object

```
// A Node class for a singly linked list
public class Node {
   public Object element;
   public Node next;

   //Constructor
   public Node(){
      element = null;
      next = null;
   }

   public Node(Object e, Node n){
      element = e;
      next = n;
   }
}
```

Node node2 = new Node("A", null);



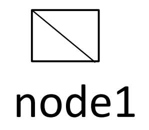
Node node3 = new Node("B", node2);



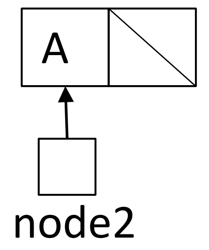
Singly Linked Node Object

```
// A Node class for a singly linked list
public class Node {
   public Object element;
   public Node next;

   //Constructor
   public Node(){
      element = null;
      next = null;
   }

   public Node(Object e, Node n){
      element = e;
      next = n;
   }
}
```

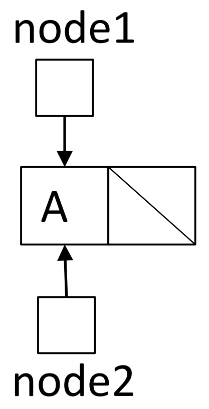
Node node1 = null;



Node node2 = new Node(“A”, null);



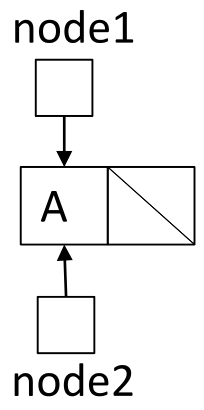
node1 = node2;



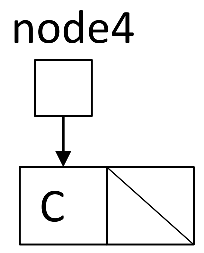
Singly Linked Node Object

```
// A Node class for a singly linked list
public class Node {
   public Object element;
   public Node next;

   //Constructor
   public Node(){
      element = null;
      next = null;
   }

   public Node(Object e, Node n){
      element = e;
      next = n;
   }
}
```

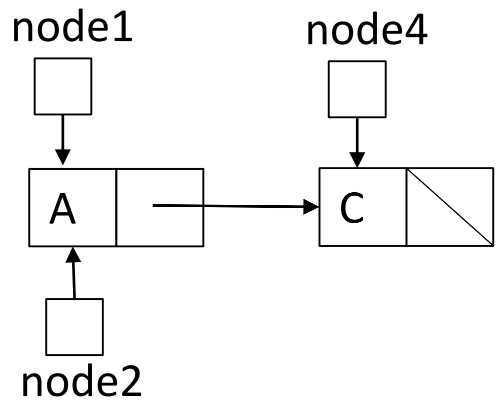
node1 = node2;



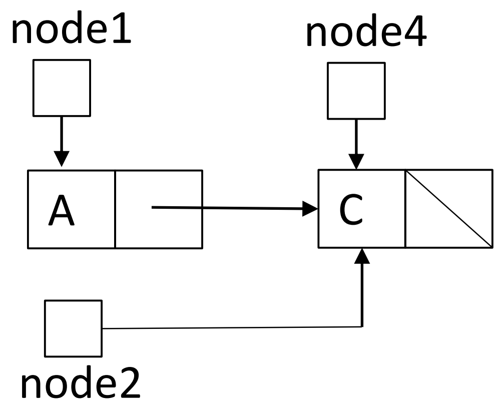
Node node4 = new Node(“C”, null);



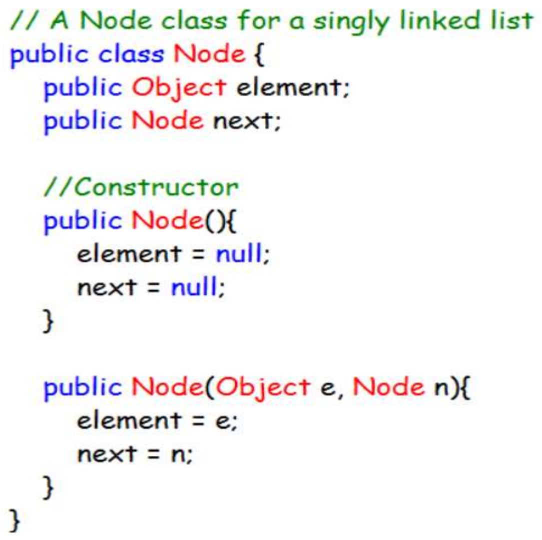
node2.next = node4;



node2 = node4;

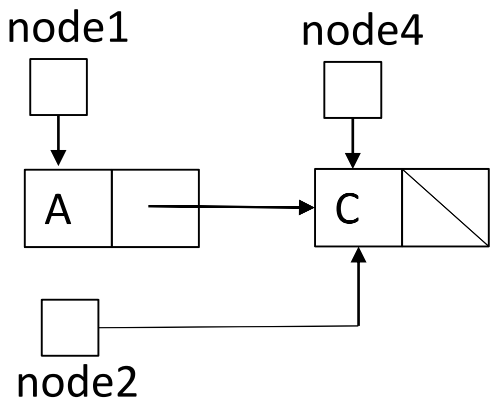


Singly Linked Node Object

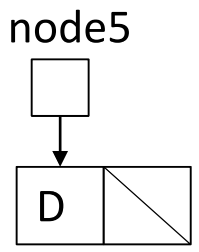


```
// A Node class for a singly linked list
public class Node {
    public Object element;
    public Node next;

    //Constructor
    public Node(){
        element = null;
        next = null;
    }

    public Node(Object e, Node n){
        element = e;
        next = n;
    }
}
```

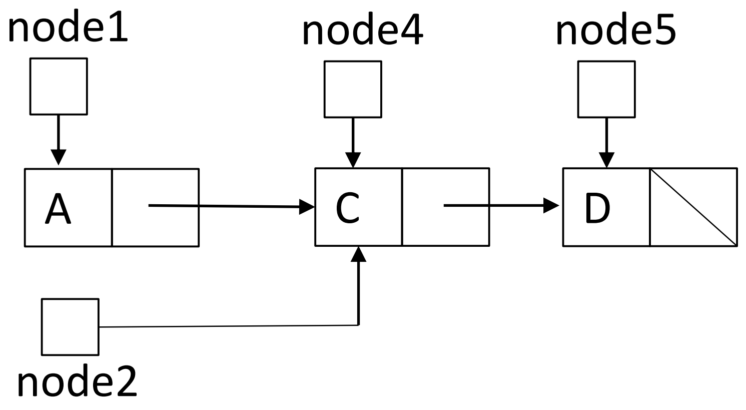
node2 = node4;



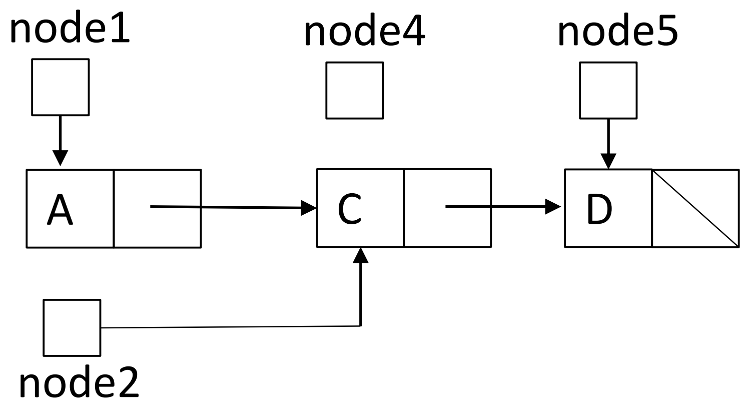
Node node5 = new Node(“D”, null);



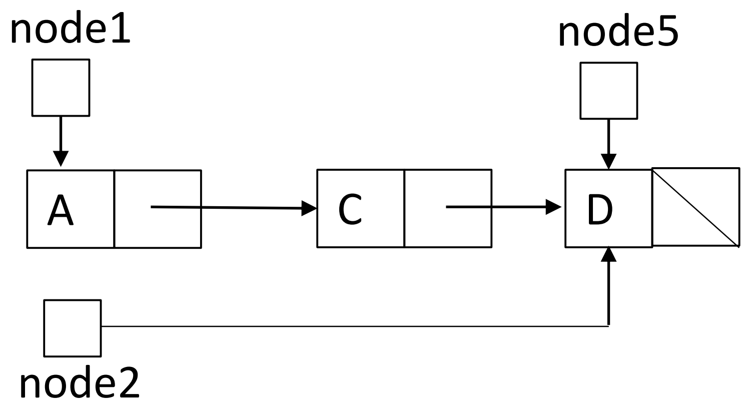
node2.next = node5;



node4 = null;

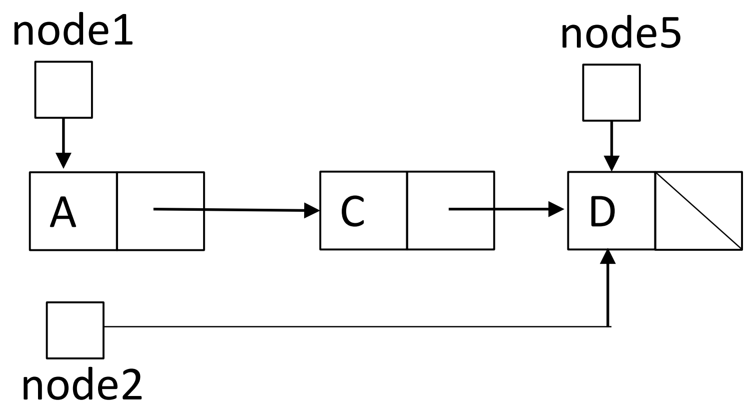


node2 = node5;

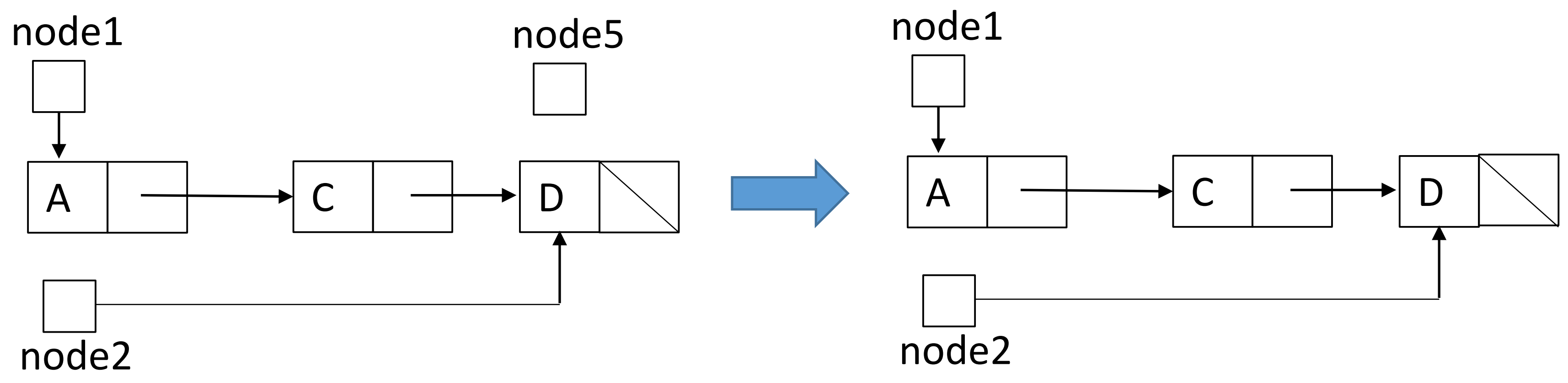


Singly Linked Node Object

node2 = node5;

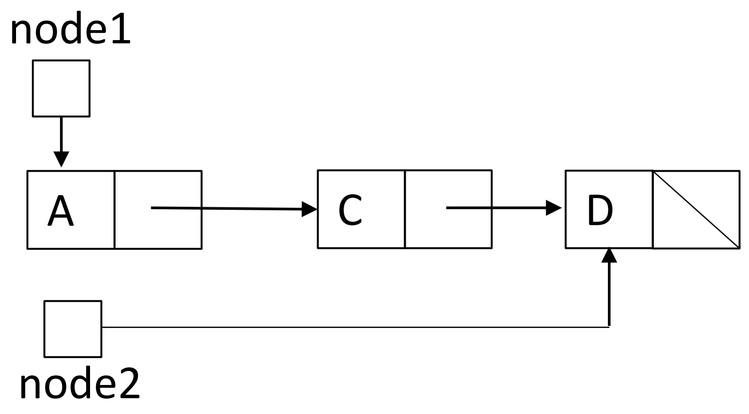


node5 = null;



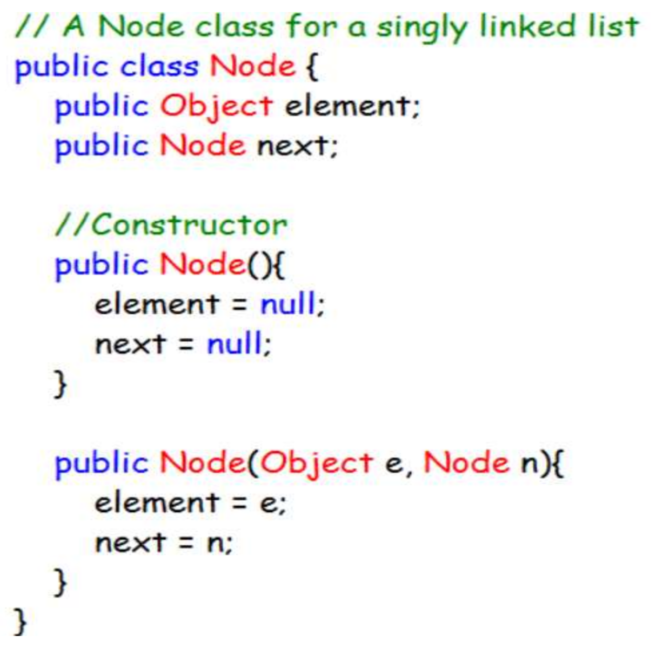
```
// A Node class for a singly linked list
public class Node {
   public Object element;
   public Node next;

   //Constructor
   public Node(){
      element = null;
      next = null;
   }

   public Node(Object e, Node n){
      element = e;
      next = n;
   }
}
```

Singly Linked Node Object



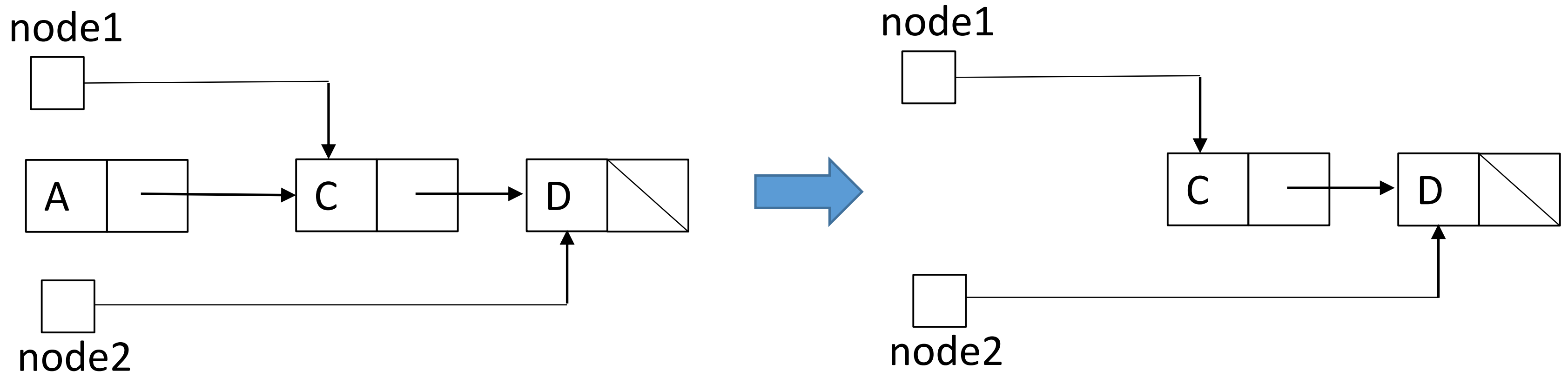
//return A

return node1.element;



```
// A Node class for a singly linked list
public class Node {
   public Object element;
   public Node next;

   //Constructor
   public Node(){
      element = null;
      next = null;
   }

   public Node(Object e, Node n){
      element = e;
      next = n;
   }
}
```

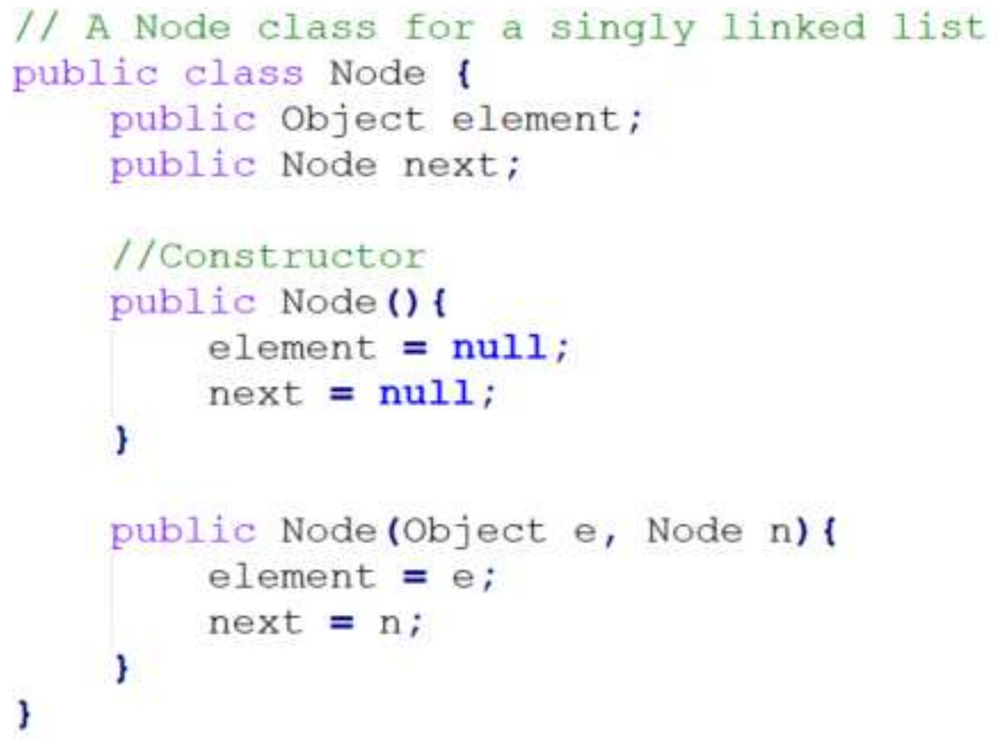
node1 = node1.next



Implementasi Queue dengan Singly Linked List

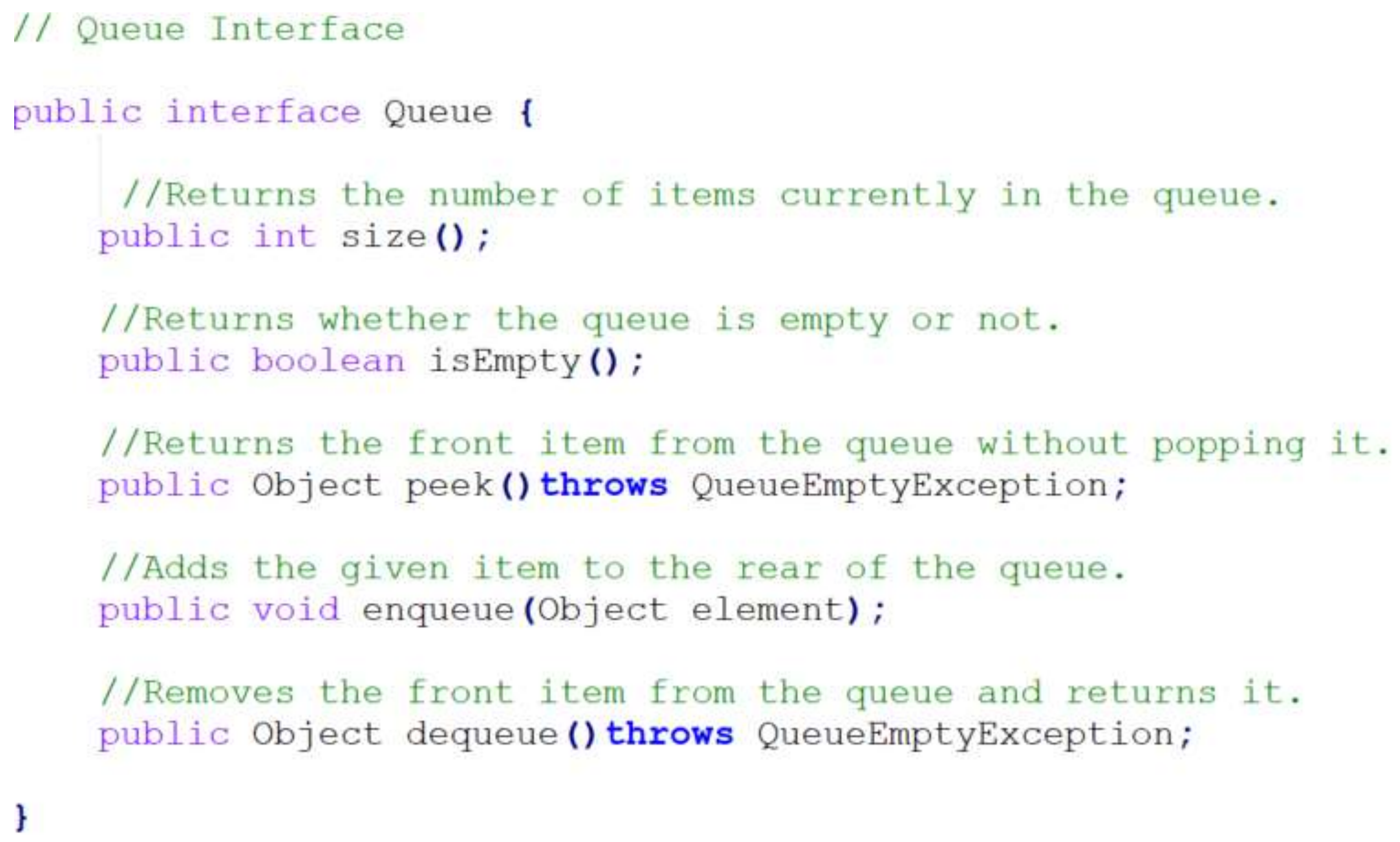
- Node.java
- Queue.java
- LinkedQueue.java
- QueueEmptyException.java
- QueueTest.java

Node.java



```java
// A Node class for a singly linked list
public class Node {
    public Object element;
    public Node next;

    //Constructor
    public Node(){
        element = null;
        next = null;
    }

    public Node(Object e, Node n){
        element = e;
        next = n;
    }
}
```

Queue.java

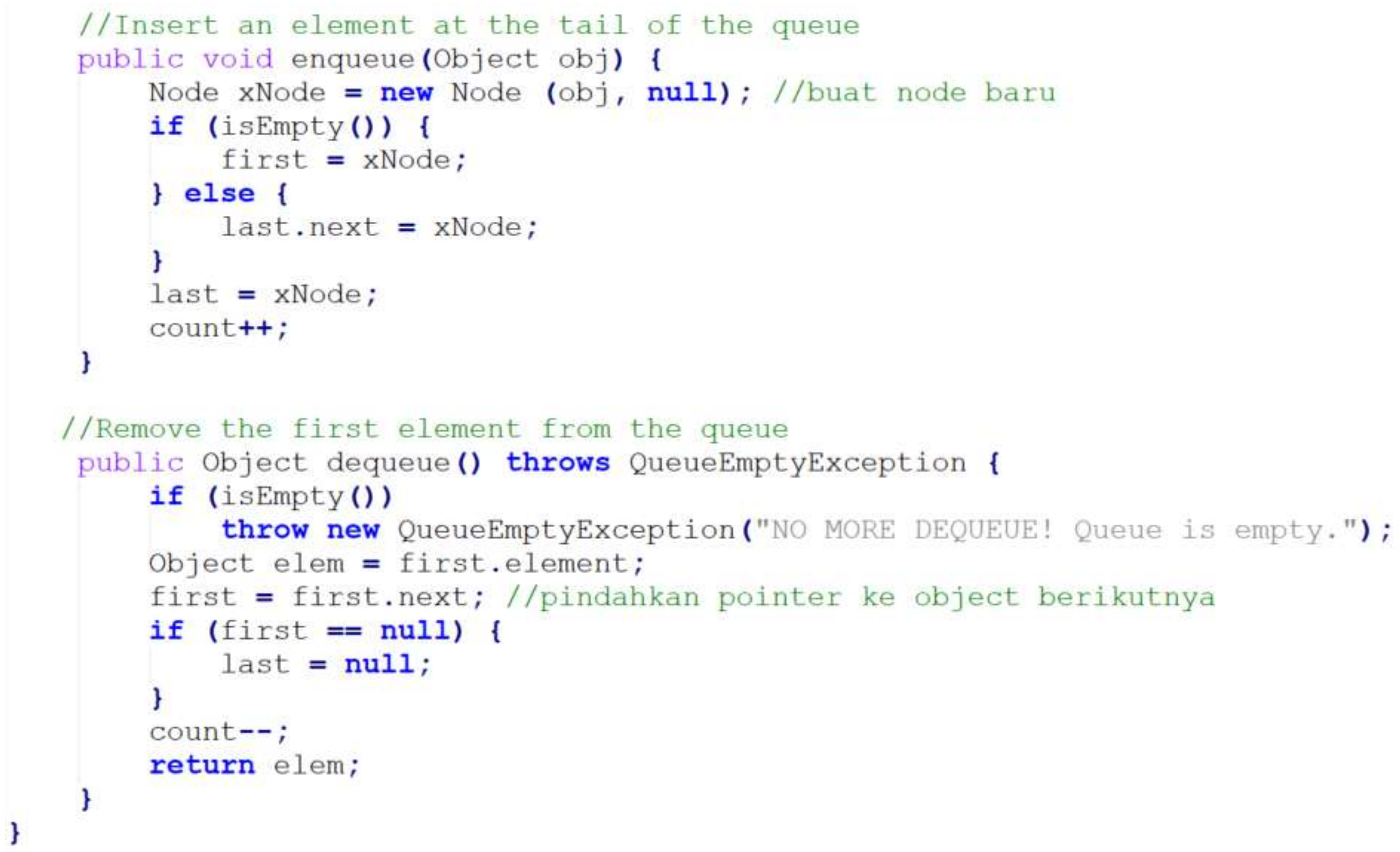


```
// Queue Interface

public interface Queue {

     //Returns the number of items currently in the queue.
    public int size();

    //Returns whether the queue is empty or not.
    public boolean isEmpty();

    //Returns the front item from the queue without popping it.
    public Object peek()throws QueueEmptyException;

    //Adds the given item to the rear of the queue.
    public void enqueue(Object element);

    //Removes the front item from the queue and returns it.
    public Object dequeue()throws QueueEmptyException;

}
```

LinkedQueue.java

```java
//Queue implementation using Linked List

public class LinkedQueue implements Queue {
    private Node first; //node paling depan
    private Node last;  //node paling belakang
    private int count;  //jumlah node

   //constructor
   public LinkedQueue(){
       first = null;
       last = null;
       count = 0;
   }

    //current Queue size
    public int size() {
        return count;
    }

    //Return whether the queue is empty or not
    public boolean isEmpty() {
        return (size() == 0);
    }

    //Return the first element in the queue
    public Object peek() throws QueueEmptyException {
        if (isEmpty())
            throw new QueueEmptyException("Queue is empty.");
        return first.element;
    }
```

LinkedQueue.java

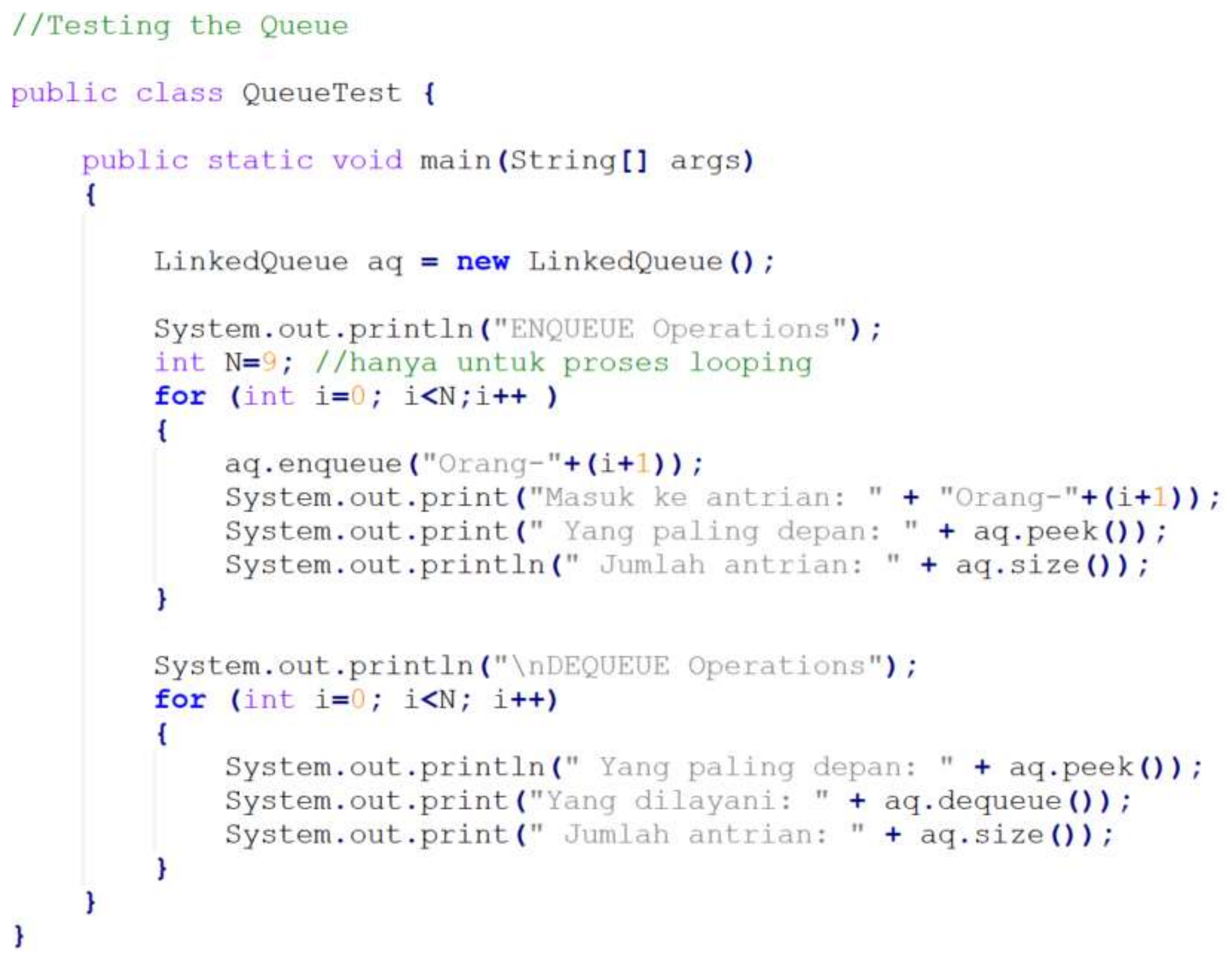


```java
    //Insert an element at the tail of the queue
    public void enqueue(Object obj) {
        Node xNode = new Node (obj, null); //buat node baru
        if (isEmpty()) {
            first = xNode;
        } else {
            last.next = xNode;
        }
        last = xNode;
        count++;
    }

   //Remove the first element from the queue
    public Object dequeue() throws QueueEmptyException {
        if (isEmpty())
            throw new QueueEmptyException("NO MORE DEQUEUE! Queue is empty.");
        Object elem = first.element;
        first = first.next; //pindahkan pointer ke object berikutnya
        if (first == null) {
            last = null;
        }
        count--;
        return elem;
    }
}
```

QueueEmptyException.java

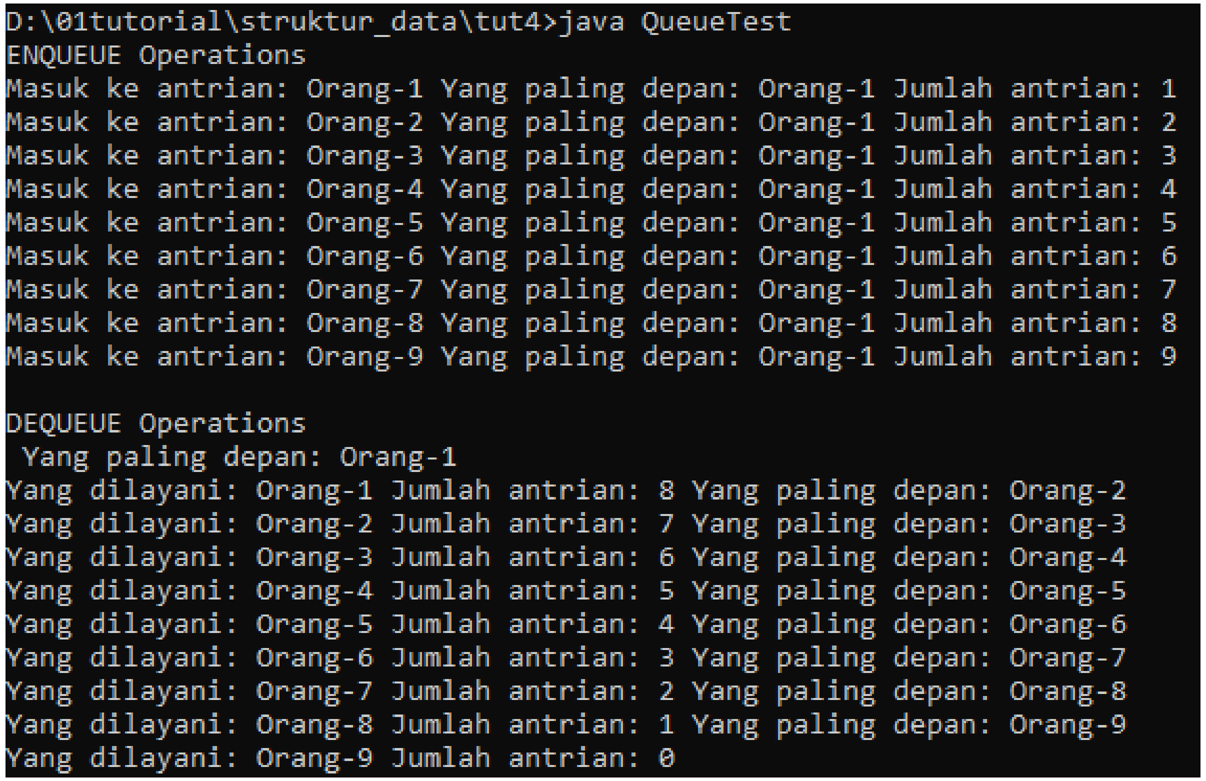
```java
// Queue Empty Exception

public class QueueEmptyException extends RuntimeException {
    public QueueEmptyException(String err){
        super(err);
    }
}
```

QueueTest.java



```java
//Testing the Queue

public class QueueTest {

    public static void main(String[] args)
    {

        LinkedQueue aq = new LinkedQueue();

        System.out.println("ENQUEUE Operations");
        int N=9; //hanya untuk proses looping
        for (int i=0; i<N;i++ )
        {
            aq.enqueue("Orang-"+(i+1));
            System.out.print("Masuk ke antrian: " + "Orang-"+(i+1));
            System.out.print(" Yang paling depan: " + aq.peek());
            System.out.println(" Jumlah antrian: " + aq.size());
        }

        System.out.println("\nDEQUEUE Operations");
        for (int i=0; i<N; i++)
        {
            System.out.println(" Yang paling depan: " + aq.peek());
            System.out.print("Yang dilayani: " + aq.dequeue());
            System.out.print(" Jumlah antrian: " + aq.size());
        }
    }
}
```

Output



```
D:\01tutorial\struktur_data\tut4>java QueueTest
ENQUEUE Operations
Masuk ke antrian: Orang-1 Yang paling depan: Orang-1 Jumlah antrian: 1
Masuk ke antrian: Orang-2 Yang paling depan: Orang-1 Jumlah antrian: 2
Masuk ke antrian: Orang-3 Yang paling depan: Orang-1 Jumlah antrian: 3
Masuk ke antrian: Orang-4 Yang paling depan: Orang-1 Jumlah antrian: 4
Masuk ke antrian: Orang-5 Yang paling depan: Orang-1 Jumlah antrian: 5
Masuk ke antrian: Orang-6 Yang paling depan: Orang-1 Jumlah antrian: 6
Masuk ke antrian: Orang-7 Yang paling depan: Orang-1 Jumlah antrian: 7
Masuk ke antrian: Orang-8 Yang paling depan: Orang-1 Jumlah antrian: 8
Masuk ke antrian: Orang-9 Yang paling depan: Orang-1 Jumlah antrian: 9

DEQUEUE Operations
 Yang paling depan: Orang-1
Yang dilayani: Orang-1 Jumlah antrian: 8 Yang paling depan: Orang-2
Yang dilayani: Orang-2 Jumlah antrian: 7 Yang paling depan: Orang-3
Yang dilayani: Orang-3 Jumlah antrian: 6 Yang paling depan: Orang-4
Yang dilayani: Orang-4 Jumlah antrian: 5 Yang paling depan: Orang-5
Yang dilayani: Orang-5 Jumlah antrian: 4 Yang paling depan: Orang-6
Yang dilayani: Orang-6 Jumlah antrian: 3 Yang paling depan: Orang-7
Yang dilayani: Orang-7 Jumlah antrian: 2 Yang paling depan: Orang-8
Yang dilayani: Orang-8 Jumlah antrian: 1 Yang paling depan: Orang-9
Yang dilayani: Orang-9 Jumlah antrian: 0
```

Tugas

- Pelajari class LinkedQueue, run QueueTest dengan benar sehingga outputnya sama dengan output di slide 18. Tidak perlu diupload ke Sikola.