

Η Διασύνδεση της Απλά Συνδεδεμένης Λίστας

```
public interface List {  
    public boolean isEmpty();  
    // Tests if list is empty (returns true)  
    public int size();  
    // Returns the current number of elements of the list  
    public void insertFirst(Object data);  
    // inserts a new node containing data to the end of the list  
    public void insertLast(Object data);  
    // inserts a new node containing data in front of the list  
    public Object removeFirst() throws ListEmptyException;  
    // removes the first element of the list and returns its content  
    public Object removeLast() throws ListEmptyException;  
    // removes the last element of the list and returns its content  
}
```

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

Γ' ΕΞΑΜΗΝΟ

Η κλάση Node

```
public class Node {  
    private Object item;  
    private Node next;  
  
    public Node() {  
        this(null, null);  
    }  
    public Node(Object it, Node n) {  
        item = it;  
        next = n;  
    }  
  
    public void setItem(Object newItem) {  
        item = newItem; }  
  
    public void setNext(Node newNext) {  
        next = newNext; }  
  
    public Object getItem() {  
        return item; }  
  
    public Node getNext() {  
        return next; }  
  
    public String toString(){  
        return item.toString(); }  
}
```

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

Γ' ΕΞΑΜΗΝΟ

Η κλάση LinkedList

```
public class LinkedList implements List {  
  
    private Node first;  
    private Node last;  
  
    public LinkedList(){  
        first = last = null;  
    }  
  
    public boolean isEmpty() {  
        return first == null;  
    }  
  
    public Node getFirst(){  
        return first;  
    }  
  
    public Node getLast(){  
        return last;  
    }  
}
```

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

Γ' ΕΞΑΜΗΝΟ

Η κλάση LinkedList

```
public void insertFirst(Object data) {  
    if(isEmpty())  
        first = last = new Node(data, null);  
    else  
        first = new Node(data, first);  
}  
  
public void insertLast(Object data) {  
    if(isEmpty())  
        first = last = new Node(data, null);  
    else {  
        Node temp = new Node(data, null);  
        last.setNext(temp);  
        last = temp;  
    }  
}
```

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

Γ' ΕΞΑΜΗΝΟ

Η κλάση LinkedList

```

public Object removeFirst() throws ListEmptyException {
    if(isEmpty())
        throw new ListEmptyException("List is Empty.");
    Object removedItem = first.getItem();
    if(first == last)
        first = last = null;
    else
        first = first.getNext();
    return removedItem;
}

public Object removeLast() throws ListEmptyException {
    if(isEmpty())
        throw new ListEmptyException("List is Empty.");
    Object removedItem = last.getItem();
    if(first == last)
        first = last = null;
    else{
        Node position;
        for(position = first; position.getNext() != last;
            position = position.getNext());
        last = position;
        position.setNext(null);
    }
    return removedItem;
}

```

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

Γ' ΕΞΑΜΗΝΟ

Η κλάση LinkedList

```

public int size() {
    int size = 0;
    Node position = first;
    while (position !=null) {
        position = position.getNext();
        size++;
    }
    return size;
}

```

ή εναλλακτικά

```

public int size() {
    int size = 0;
    for(Node position = first; position != null;
        position = position.getNext())
        size++;
    return size;
}

```

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

Γ' ΕΞΑΜΗΝΟ

Η κλάση LinkedList

```

public void printList() throws ListEmptyException{
    if(isEmpty())
        throw new ListEmptyException("List is Empty.");
    for(Node position = first; position != null;
        position = position.getNext())
        System.out.println(position.getItem());
}

public Object maxOfList() {
    if(isEmpty())
        throw new ListEmptyException("List is Empty.");
    Object max = first.getItem();
    Node position = first.getNext();
    while (position !=null) {
        Comparable CoMax=(Comparable)max;
        Comparable CoItem=(Comparable)position.getItem();
        if ((CoMax.compareTo(CoItem)<0))
            max=position.getItem();
        position=position.getNext();
    }
    return max;
}

//if (((Comparable)max).compareTo((Comparable)position.getItem()) <0)

```

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

Γ' ΕΞΑΜΗΝΟ

Ενδεικτική χρήση της κλάσης LinkedList

```

public static void main(String[] args) {

    LinkedList L = new LinkedList();
    LinkedList L2 = new LinkedList();

    L.insertLast("Stamatis");
    L.insertLast("Adamakis");
    L.insertLast("Sferiou");
    L.insertLast("Santouris");
    L.insertLast("Iliopoulos");

    try {
        System.out.println("LIST SIZE: "+ L.size());
        System.out.println("LIST MAX: "+ L.maxOfList());
        System.out.println("LIST:"); L.printList();
    }
    catch (ListEmptyException Error) {
        System.out.println("IS EMPTY");
    }
}

```

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

Γ' ΕΞΑΜΗΝΟ

Σύγκριση δύο αντικειμένων

```
// Σε κάθε κλάση που δημιουργούμε πρέπει να
// συμπεριλάβουμε μέθοδο compareTo

public class Student implements Comparable {

    private String name;
    private int am;
    // κ.λπ.

    @Override
    public int compareTo(Object ob) {
        return getAM() - ((Student)ob).getAM();
    }

    @Override
    public String toString() {
        String s = name + ", " + am + "\n";
        return s;
    }
}
```