ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΑΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩ
ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ & ΑΝΑΛΥΣΗ ΑΛΓΟΡΙΘΜΩΝ

ΑΣΚΗΣΗ 3

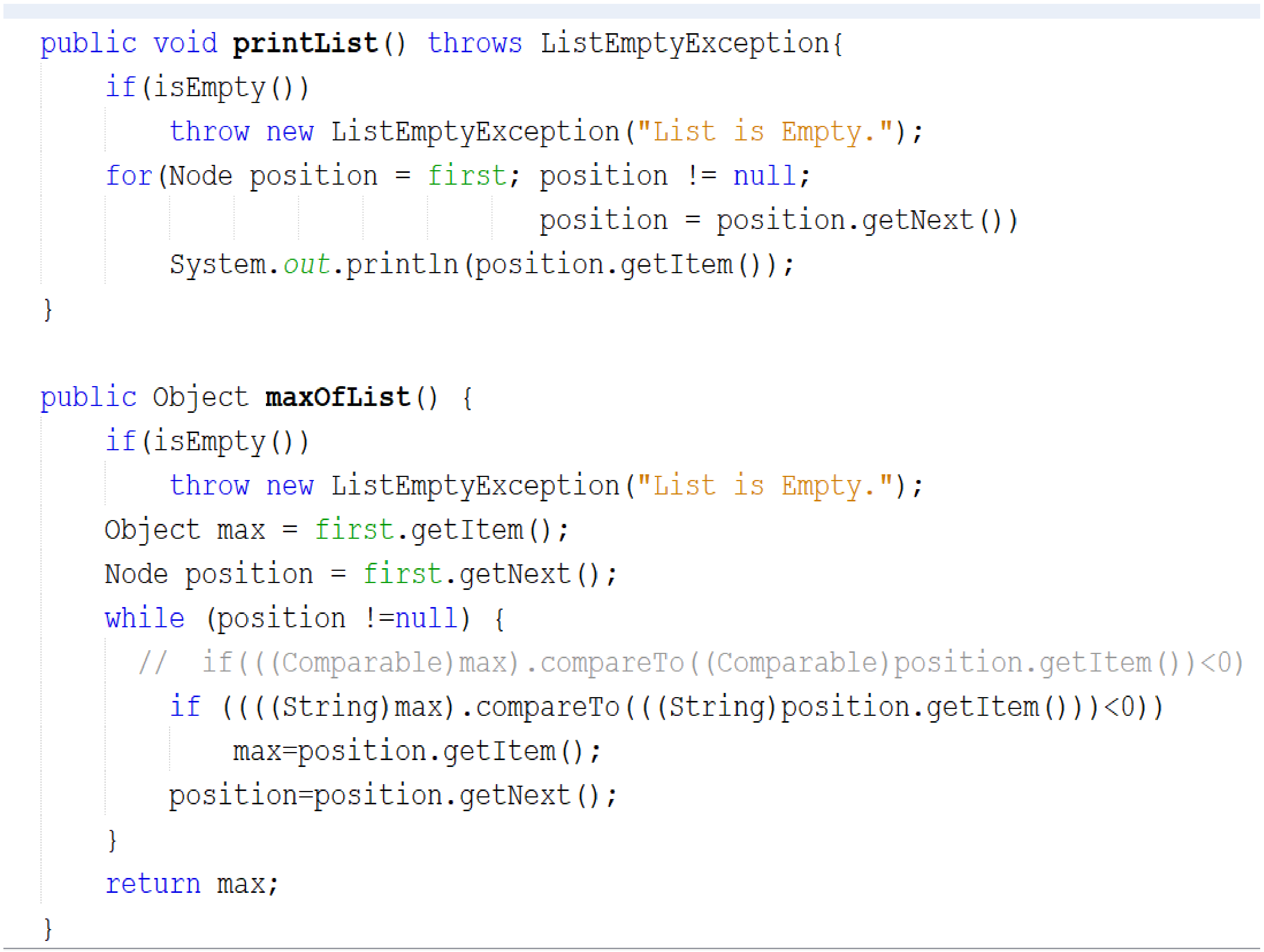
ΑΠΛΑ ΣΥΝΔΕΔΕΜΕΝΕΣ ΛΙΣΤΕΣ
(simple linked lists)

(Βλέπε http://www.iee.ihu.gr/~demos/teaching_GR.html)

Ασδρέ Κατερίνα
asdre@ihu.gr

```
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;
    }
```

```java
public void setFirst(Node first) {
    this.first = first;
}

public void setLast(Node last) {
    this.last = last;
}


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

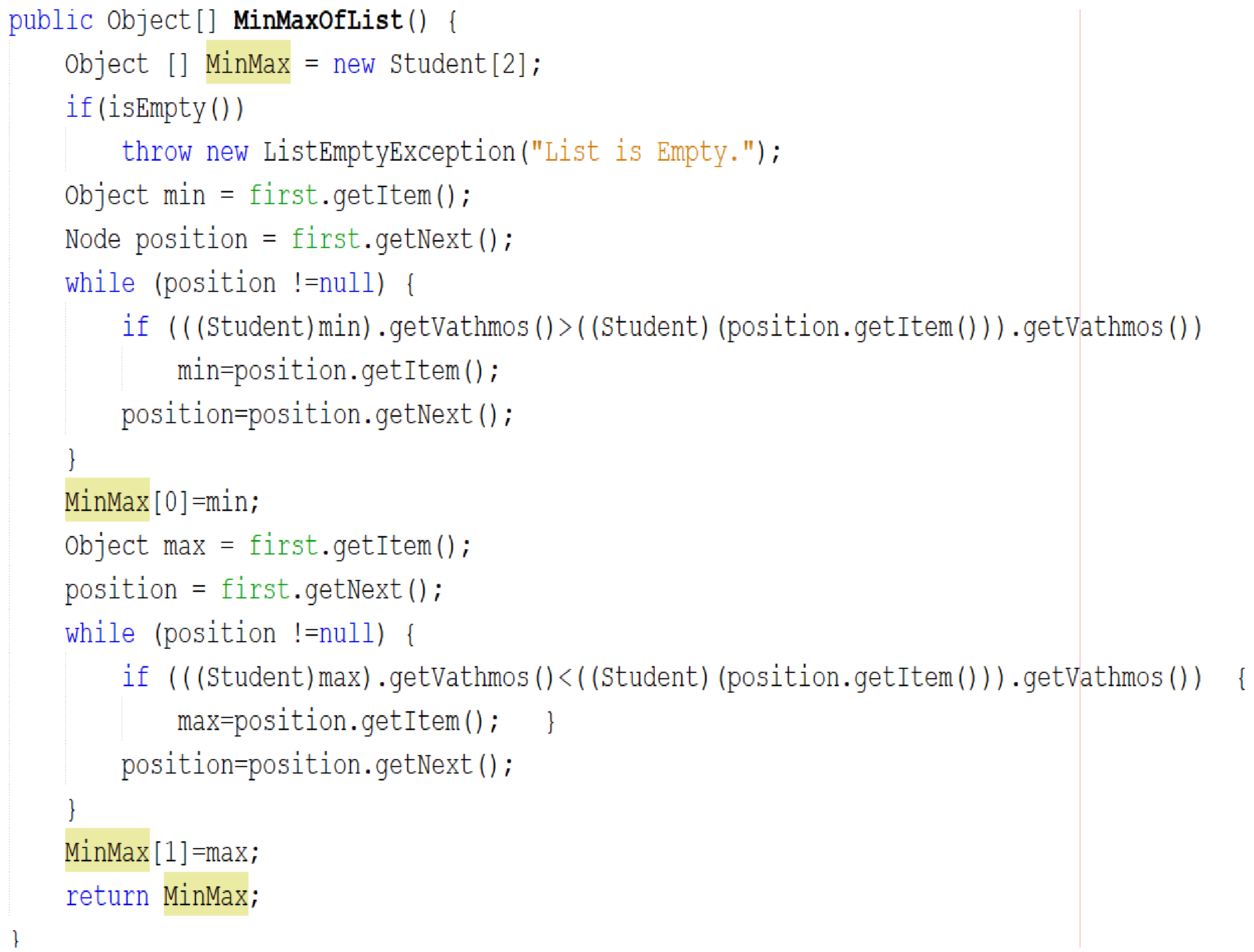


```java
public void printList() throws ListEmptyException{
    if(isEmpty())
        throw new ListEmptyException("List is Empty.");
    for(Node position = first; position != null;
                                position = position.getNext())
        System.out.println(position.getItem());
}

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

```
public boolean exists(Object data) {
    if(isEmpty())
        throw new ListEmptyException("List is Empty.");
    Node position = first;
    while (position !=null) {
        if (position.getItem().equals(data))return true;
        position=position.getNext();
    }
    return false;
}
```

```java
public LinkedList sort() {
    Node trace, current, min;
    trace = getFirst();
    while (trace!=null)
      { current = trace;
        min = trace;
        while (current!=null)
          { if (((String)(current.getItem())).compareTo((String)(min.getItem()))<0)
                min=current;
            current = current.getNext();
          } //endwhile current
        String temp = (String)trace.getItem();
        trace.setItem(min.getItem());
        min.setItem(temp);
        trace = trace.getNext();
      }//endwhile trace
    return this;
 }
```

```
public LinkedList BubbleSort() {
    Node current = getFirst();
    while (current != null) {
         Node second = current.getNext();
         while (second != null) {
            if (((String)(current.getItem())).compareTo((String)(second.getItem()))>0) {
                String temp =(String) current.getItem();
                current.setItem(second.getItem());
                 second.setItem(temp);
             }
            second = second.getNext();
         }
    current = current.getNext();
    }
    return this;
 }
```



```java
public Object[] MinMaxOfList() {
    Object [] MinMax = new Student[2];
    if(isEmpty())
        throw new ListEmptyException("List is Empty.");
    Object min = first.getItem();
    Node position = first.getNext();
    while (position !=null) {
        if (((Student)min).getVathmos()>((Student)(position.getItem())).getVathmos())
            min=position.getItem();
        position=position.getNext();
    }
    MinMax[0]=min;
    Object max = first.getItem();
    position = first.getNext();
    while (position !=null) {
        if (((Student)max).getVathmos()<((Student)(position.getItem())).getVathmos())   {
            max=position.getItem();    }
        position=position.getNext();
    }
    MinMax[1]=max;
    return MinMax;
}
```

```
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;
    }
}
```

```java
    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;
    }
}
```