CS 635 Advanced Object-Oriented Design & Programming
Spring Semester, 2012
Doc 15 Assignment 2 Comments
Mar 8, 2012
Issues

Info Hiding - don't let users of LinkedList class have access to nodes

Use Java's naming conventions

Full names
structureOfName
Use existing names for method
What name do you use for method to add elements to a collection

Format consistently

Use Java interfaces
Don't make new ones to replace existing ones
This includes the parent class of LinkedList class

Half a class - just data
What Type of Cohesion is this?

```java
public class Node<E> {

    public boolean contains(E dataValue) {
        return this.next.checkNode(dataValue);
    }
}
```
public class SortedLinkedList<T> extends AbstractSequentialList {

    public boolean add(int index, T data) {
        What should happen here?
    }

    
}
Principles of OO Design, or Everything I Know About Programming, I Learned from Dilbert

Alan Knight

http://alanknightsblog.blogspot.com/2011/10/principles-of-oo-design-or-everything-i.html
1. Never do any work that you can get someone else to do for you

Example 1 Total of bills that have been paid this quarter for a factory

```plaintext
total := 0
aFactory billings do: [:each |
  (each status == #paid and: [each date > startDate])
  ifTrue: [total := total + each amount]].
```

versus

```plaintext
total := aPlant totalBillingsPaidSince: startDate.
```

http://alanknightsblog.blogspot.com/2011/10/principles-of-oo-design-or-everything-i.html
2. Avoid Responsibility

If you must accept a responsibility, keep it as vague as possible.

For any responsibility you accept, try to pass the real work off to somebody else.
In iterator class

public boolean hasNext() {
    if (currentNode.next.isEnd() == true )
        return false;
    return true;
}

public boolean hasNext() {
    return !currentNode.next.isEnd();
}
public boolean hasNext() {
    return currentNode.hasNext();
}
Using the Dilbert Rules

```java
public boolean hasNext() {
    return currentNode.hasNext();
}
```

Verses

```java
public boolean hasNext() {
    return currentNode.next().getValue() != null;
}
```

```java
public boolean hasNext() {
    return currentNode.hasNext();
}
```
public boolean add(E item) {
    if (isFirstElement())
        return addFirst(item);
    }
    if (isLastElement(item)) {
        return addLast(item)
    }
    ListNode<E> next = head.getNextNode();
    ListNode<E> current;
    for (int ii == 0; ii < length; ii++) {
        current = next;
        if (itemBelongsBeforeCurrent(item, current)) {
            return appendItemBeforeCurrent(item, current);
        }
    }
    return false;
}
public boolean add(E element) {
    return head.add(element,comparer);
}

public boolean add(E element, Comparator comparer) {
    return next.add(element,comparer);
}

public boolean add(E element, Comparator comparer) {
    if (comparer(element, data) == -1) {
        return prepend(element);
    }
    return next.add(element,comparer);
}

public boolean add(E element, Comparator comparer) {
    return prepend(element);
}
public boolean add(E element) {
    NodeIterator nodes = new NodeIterator(head);
    while (nodes.hasNext() ) {
        Node current = nodes.next();
        if (current.elementBelongsBefore(element, comparer))
            return current.prepend(element);
    }
    return false;
}
Visiter pattern

void visitMiddleNode(Node aNode) {
    System.out.println(aNode.getValue());
}

Thursday, March 8, 12
Is this the Visiter pattern

```java
public String accept(Visiter aVisiter) {
    String answer = "( ";
    Node current = head.getNext();
    while (!current.isTail()) {
        answer = answer + current.getValue() + " - ";
        current = current.getNext();
    }
    return answer;
}
```

LinkedList Class
Is this the Visiter pattern

```java
public String accept(Visiter aVisiter) {
    String answer = "(";
    Node current = head.getNext();
    while (!current.isTail()) {
        answer = answer + aVisiter.visitMiddleNode() + " - ";
        current = current.getNext();
    }
    return answer;
}
```

```java
String visitMiddleNode(Node aNode) {
    return aNode.getValue();
}
```

Linkedlist Class

Visiter class
Is this the Visiter pattern

```java
public String accept(Visiter aVisiter) {
    return aVisitor.visitMiddleNode(head.getNext());
}
```

```java
String visitMiddleNode(Node aNode) {
    String answer = "(";
    Node current = aNode;
    while (!current.isTail()) {
        answer = answer + aVisiter.visitMiddleNode() + " - ";
        current = current.getNext();
    }
    return answer;
}
```

Visitor class

LinkedList Class