CS 535 Object-Oriented Programming & Design
Fall Semester, 2003
Doc 13 Linked List Example

References

CS535 lecture notes, Fall 1994

Ordered Double Linked List – Iterative Solution
Solution 1 – Dumb Node

```
Smalltalk defineClass: #DumbNode
   superclass: #{Core.Object}
   indexedType: #none
   private: false
   instanceVariableNames: 'value previous next '
   classInstanceVariableNames: ''
   imports: ''
   category: 'CS535'
```

DumbNode methodsFor: 'accessing'

```
next
   ^next

next: aNode
   next := aNode

previous
   ^previous

previous: aNode
   previous := aNode

value
   ^value

value: anObject
   value := anObject
```

```
Smalltalk defineClass: #LinkedListDumbNode
   superclass: #{Core.SequenceableCollection}
   indexedType: #none
   private: false
   instanceVariableNames: 'head tail size '
   classInstanceVariableNames: ''
   imports: ''
   category: 'CS535'

LinkedListDumbNode class methodsFor: 'instance creation'

new
   ^super new initialize
```

Instance methods

```
initialize
   size := 0.

at: anInteger
   | current |
   (anInteger < 1) | (anInteger > size)
      ifTrue:[SubscriptOutOfBoundsError raiseSignal].
   current := head.
   (anInteger -1) timesRepeat: [current :=current next].
   ^current value

size
   ^size
```

```
add: aMagnitude
   | current previous newNode |
   size := size + 1.
   head ifNil:
         [tail := head := DumbNode new.
         head value: aMagnitude.
         ^aMagnitude].
   newNode := DumbNode new.
   newNode value: aMagnitude.
   tail value < aMagnitude
      ifTrue:
         [tail next: newNode.
         newNode previous: tail.
         tail := newNode.
         ^aMagnitude].
   aMagnitude < head value
      ifTrue:
         [head previous: newNode.
         newNode next: head.
         head := newNode.
         ^aMagnitude].
   current := head.
   previous := nil.
   [current value < aMagnitude] whileTrue:
         [previous := current.
         current := current next].
   previous next: newNode.
   newNode previous: previous.
   newNode next: current.
   current previous: newNode.
   ^aMagnitude
```

Solution 2 – Good Node

```
Smalltalk defineClass: #GoodNode
   superclass: #{Core.Object}
   indexedType: #none
   private: false
   instanceVariableNames: 'value previous next '
   classInstanceVariableNames: ''
   imports: ''
   category: 'CS535'
```

GoodNode class methodsFor: 'instance creation'

```
value: aMagnitude
   ^self new value: aMagnitude
```

Instance Methods

```
append: aNode
   aNode next: next.
   next ifNotNil: [next previous: aNode].
   aNode previous: self.
   next := aNode

next
   ^next

next: anObject
   next := anObject

previous
   ^previous
```

```
previous: anObject
   previous := anObject

value
   ^value

value: anObject
   value := anObject
```

```
Smalltalk defineClass: #LinkedListGoodNode
   superclass: #{Core.SequenceableCollection}
   indexedType: #none
   private: false
   instanceVariableNames: 'head tail size '
   classInstanceVariableNames: ''
   imports: ''
   category: 'CS535'
```

LinkedListGoodNode class methodsFor: 'instance creation'

```
new
   ^super new initialize
```

Instance Methods

```
initialize
   size := 0.

size
   ^size

at: anInteger
   | current |
   (anInteger < 1) | (anInteger > size)
      ifTrue:[SubscriptOutOfBoundsError raiseSignal].
   current := head.
   (anInteger -1) timesRepeat: [current :=current next].
   ^current value
```

```
add: aMagnitude
   | current previous newNode |
   size := size + 1.
   head ifNil:
         [tail := head := GoodNode value: aMagnitude.
         ^aMagnitude].
   newNode := GoodNode value: aMagnitude.
   tail value < aMagnitude
      ifTrue:
         [tail append: newNode.
         tail := newNode.
         ^aMagnitude].
   aMagnitude < head value
      ifTrue:
         [newNode append: head.
         head := newNode.
         ^aMagnitude].
   current := head.
   previous := nil.
   [current value < aMagnitude] whileTrue:
         [previous := current.
         current := current next].
   previous append: newNode.
   ^aMagnitude
```

Solution 3 SmartNode

```
Smalltalk defineClass: #SmartNode
   superclass: #{Core.Object}
   indexedType: #none
   private: false
   instanceVariableNames: 'value previous next '
   classInstanceVariableNames: ''
   imports: ''
   category: 'CS535'
```

SmartNode class methodsFor: 'instance creation'

```
value: aMagnitude
   ^self new value: aMagnitude
```

Instance Methods

```
< aNode
   ^value < aNode value

next
   ^next

next: anObject
   next := anObject

prepend: aNode
   aNode next: self.
   aNode previous: previous.
   previous ifNotNil: [previous next: aNode].
   previous := aNode
```

```
previous
   ^previous

previous: anObject
   previous := anObject

value
   ^value

value: anObject
   value := anObject
```

```
Smalltalk defineClass: #HeadNode
   superclass: #{Smalltalk.SmartNode}
   indexedType: #none
   private: false
   instanceVariableNames: ''
   classInstanceVariableNames: ''
   imports: ''
   category: 'CS535'
```

Instance Methods

```
< aNode
   "Act like value is negative infinity"
   ^true
```

```
Smalltalk defineClass: #TailNode
   superclass: #{Smalltalk.SmartNode}
   indexedType: #none
   private: false
   instanceVariableNames: ''
   classInstanceVariableNames: ''
   imports: ''
   category: 'CS535'
```

Instance Methods

```
< aNode
   "Act like value is infinity"
   ^false
```

```
Smalltalk defineClass: #LinkedListSmartNode
   superclass: #{Core.SequenceableCollection}
   indexedType: #none
   private: false
   instanceVariableNames: 'head tail size '
   classInstanceVariableNames: ''
   imports: ''
   category: 'CS535'
```

LinkedListSmartNode class methodsFor: 'instance creation'

```
new
   ^super new initialize
```

Instance Methods

```
at: anInteger
   | current |
   (anInteger < 1) | (anInteger > size)
      ifTrue:[SubscriptOutOfBoundsError raiseSignal].
   current := head.
   anInteger timesRepeat: [current :=current next].
   ^current value

initialize
   size := 0.
   head := HeadNode new.
   tail := TailNode new.
   tail prepend: head

size
   ^size
```

```
add: aMagnitude
   | newNode current |
   size := size + 1.
   newNode := SmartNode value: aMagnitude.
   current := head.
   [current < newNode] whileTrue: [current := current next].
   current prepend: newNode.
   ^aMagnitude
```

OOPS verses Procedural Solution of Ordered Single Linked List Problem CS 535 Fall 1994

Operations on List:

- Add elements
- Remove element
- Print list
- Test of inclusion
- Access

	Procedural	C++ Data Structure Text	C++ OOPS
Lines of Code (LOC)	100	56	25
Functions defined	5	21	18
LOC/function	20	2.7	1.4
Add functions	1	3	3
LOC for add	23	11	4