

Chapter 17: Linked List Lab

Node Constructors:

```
9      public Node()
10     {
11         this.x = null;
12         this.next = null;
13     }
14
15     /**
16      * Create a node with the value of _x
17      * @param _x
18      */
19     public Node(AnyType _x)
20     {
21         this.x = _x;
22         this.next = null;
23     }
24
25     /**
26      * Create a node with the value of _x and a pointer of _n
27      * @param _x
28      * @param _n
29      */
30     public Node(AnyType _x, Node<AnyType> _n)
31     {
32         this.x = _x;
33         this.next = _n;
34     }
```

Node Methods:

```
36     /**
37      * Get the value of this node
38      * @return The node's value
39      */
40     public AnyType getItem()
41     {
42         return this.x;
43     }
44
45     /**
46      * Set the value of this node to _x
47      * @param _x
48      */
49     public void setItem(AnyType _x)
50     {
51         this.x = _x;
52     }
53
54     /**
55      * Set the pointer of this node to _n
56      * @param _n
57      */
58     public void setNextNode(Node<AnyType> _n)
59     {
60         this.next = _n;
61     }
62
63     /**
64      * Get the pointer of this node
65      * @return The node's pointer
66      */
67     public Node<AnyType> getNextNode()
68     {
69         return this.next;
70     }
71
72     /**
73      * Create a string from all nodes
74      */
75     @Override
76     public String toString()
77     {
78         String value = "";
79         Node<AnyType> current = this;
80
81         while (current != null)
82         {
83             value += String.format("format:%s ", current.getItem().toString());
84             current = current.getNextNode();
85         }
86
87         if (value.length() > 0)
88         {
89             return value.substring(beginIndex:0, value.length() - 1);
90         }
91
92         return null;
93     }
```

## Chapter 17: Linked List Lab

### Linked List Constructor and Methods:

```
5 public LinkedList()
6 {
7     list = new Node<AnyType>();
8 }
9
10 /**
11  * Test if the list is logically empty.
12  * @return true if empty, false otherwise.
13  */
14 public boolean isEmpty() {
15     return (this.list.getNextNode() == null);
16 }
17
18 /**
19  * Make the list logically empty.
20  */
21 public void makeEmpty()
22 {
23     this.list.setNextNode(_n:null);
24 }
25
26
27 /**
28  * Insert at the front
29  * @param x the item to insert.
30  */
31 public void insertFront(AnyType _x)
32 {
33     this.list.setNextNode
34     (
35         new Node<AnyType>(_x, list.getNextNode())
36     );
37 }
38
39 /**
40  * Return Node corresponding to the first node containing an item.
41  * @param x the item to search for.
42  * @return a Node; node is not valid if item is not found.
43  */
44 public Node<AnyType> find(AnyType _x)
45 {
46     Node<AnyType> found = null;
47     Node<AnyType> current = this.list.getNextNode();
48
49     while (current != null)
50     {
51         if (current.getItem() == _x)
52         {
53             found = current;
54             break;
55         }
56         current = current.getNextNode();
57     }
58
59     return found;
60 }
61
62 /**
63  * Remove the first occurrence of an item.
64  * @param x the item to remove.
65  */
66 public void remove(AnyType _x)
67 {
68     Node<AnyType> previous = this.list;
69     Node<AnyType> current = this.list.getNextNode();
70
71     while (current != null)
72     {
73         if (current.getItem() == _x)
74         {
75             previous.setNextNode(current.getNextNode());
76             break;
77         }
78         previous = current;
79         current = current.getNextNode();
80     }
81 }
```

## Chapter 17: Linked List Lab

---

```
86  @Override
87  public String toString()
88  {
89      String value = "";
90      Node<AnyType> current = this.list.getNextNode();
91
92      while (current != null)
93      {
94          value += String.format(format:"%s ", current.getItem().toString());
95          current = current.getNextNode();
96      }
97
98      if (value.length() > 0)
99      {
100         return value.substring(beginIndex:0, value.length() - 1);
101     }
102
103     return null;
104 }
105
106 /**
107  * Return the size of the list
108  * @return
109  */
110 public static <AnyType> int listSize(LinkedList<AnyType> _theList)
111 {
112     Node<AnyType> current = _theList.list.getNextNode();
113     int length = 0;
114
115     while (current != null)
116     {
117         length++;
118         current = current.getNextNode();
119     }
120
121     return length;
122 }
```

### Testing Output:

```
c:\Users\wes\github-repos\cs2420_summer2023\Chapter17\LinkedListLab - VS Code Console
Testing Node Methods
Finished Testing Node Methods
Testing LinkedList methods
Finished LinkedList testing
Press any key to continue . . .
```