

**INF1010**

CURRICULUM

**V2012**

Java

Joakim  
Myrvoll  
Johansen

## Container with linked list (iterates)

```
1 import java.util.Iterator;
2 import inf1010.lib.two.IfICollection;
3
4
5 public class OrderedSet<E extends Comparable<? super E>> implements IfICollection<E> {
6     int size;
7     E[] arr;
8     private ListNode first, last, tmp;
9     ListIterator it;
10
11     public OrderedSet(E p){
12         first = new ListNode(null);
13         last = new ListNode(null);
14         first.next = last;
15         last.pre = first;
16         it = new ListIterator();
17         size = 0;
18
19         this.add(p);
20     }
21     public OrderedSet(){
22         first = new ListNode(null);
23         last = new ListNode(null);
24         first.next = last;
25         last.pre = first;
26         it = new ListIterator();
27         size = 0;
28     }
29
30     /**
31      * Adds object to collection
32      *
33      * @param e
34      */
35     public boolean add(E e) {
36         it = new ListIterator();
37         tmp = first.next;
38         ListNode add = new ListNode(e);
39
40         // already exists?
41         if(this.contains(e)){
42             return false;
43         }
44         while(it.hasNext()){
45             it.next();
46             if(add.current.compareTo(tmp.current) < 0 && tmp.current != null){
47                 add.next = tmp;
48                 add.pre = tmp.pre;
49                 tmp.pre.next = add;
50                 tmp.pre = add;
51                 size++;
52                 return true;
53             }
54             tmp = tmp.next;
55         }
56         // added last in list
57         add.next = last;
58         add.pre = last.pre;
59         last.pre.next = add;
60         last.pre = add;
61         size++;
62
63         return true;
64     }
65
66     /**
67      * Checks if collection contains object
68      *
69      * @param e
70      */
71     public boolean contains(E e) {
72         if(e == null) throw new java.lang.NullPointerException();
73         for(E o : this){
74             if(o.compareTo(e) == 0){
75                 System.out.println("This " + e.getClass().getSimpleName() + " already exist.");
76                 return true;
77             }
78         }
79         return false;
80     }
81
82     /**
83      * Removes object from collection
84      *
85      * @param e
86      */
87     public boolean remove(E e) {
88         it = new ListIterator();
89         if(e == null) throw new java.lang.NullPointerException();
90
91         while(it.hasNext()){
92             E o = it.next();
93             if(o.compareTo(e) == 0){
94                 it.remove();
95                 return true;
96             }
97         }
98         throw new UnsupportedOperationException();
99     }
100 }
```

```

101     public int size() {
102         return size;
103     }
104
105     /**
106     * Checks if collection is empty
107     */
108     public boolean isEmpty() {
109         return size == 0;
110     }
111
112     /**
113     * Clears collection
114     */
115     public void clear() {
116         first.next = last;
117         last.pre = first;
118         size = 0;
119     }
120
121     /**
122     * Get object from collection
123     *
124     * @param index
125     * @return o
126     */
127     public E get(int index) {
128         if(index < 0 || index >= size) throw new java.lang.IndexOutOfBoundsException();
129
130         it = new ListIterator();
131         int i = 0;
132         while(it.hasNext()){
133             E o = it.next();
134             if(i++ == index && o != null){
135                 return o;
136             }
137         }
138         throw new java.lang.IndexOutOfBoundsException();
139     }
140
141     /**
142     * Colelct collection in an array
143     *
144     * @param a    empty array
145     * @return a    full array
146     */
147     public E[] toArray(E[] a) {
148         int t = 0;
149
150         if(a == null) throw new java.lang.NullPointerException();
151
152         for(E o : this){
153             a[t++] = o;
154         }
155         for(int i = t; i < a.length; i++){
156             a[i] = null;
157         }
158
159         return a;
160     }
161
162     /**
163     * {@inheritDoc}
164     */
165     public Iterator<E> iterator() {
166         return new ListIterator();
167     }
168
169     /**
170     * Iterator
171     */
172     public class ListIterator implements Iterator<E>{
173         ListNode t = first;
174         ListNode p, n = null;
175         boolean used = false;;
176
177         public boolean hasNext() {
178             return t.next != last;
179         }
180
181         public E next() {
182             if(t.next == last) throw new java.util.NoSuchElementException();
183             t = t.next;
184             used = false;
185             return t.current;
186         }
187
188         public void remove() {
189             if(t == first || used == true) throw new java.lang.IllegalStateException();
190
191             p = t.pre;
192             n = t.next;
193
194             p.next = n;
195             n.pre = p;
196             size--;
197             used = true;
198         }
199     }
200
201     /**
202     * List
203     *
204     * @author joakimaj
205     */
206     private class ListNode{
207         private E current;
208         private ListNode next;
209         private ListNode pre;
210
211         private ListNode(E e){
212             current = e;
213         }
214     }
215 }
216 }

```

## IO (read from/write to file)

```
1  import java.io.File;
2  import java.io.FileNotFoundException;
3  import java.io.FileWriter;
4  import java.util.Scanner;
5
6  /**
7   * reader/writer
8   */
9  public class SortIO {
10     private String[] words;
11     private int x = 0, wordCnt;
12
13     /**
14      * Reads from file
15      *
16      * @param filename
17      */
18     public String[] readFromFile(String filename) {
19         try {
20             Scanner s = new Scanner (new File(filename));
21
22             wordCnt = s.nextInt();
23             words = new String[wordCnt];
24
25             while (s.hasNext()) {
26                 words[x++] = s.next();
27             }
28
29             if(words[wordCnt - 1] == null){
30                 System.err.println("Error: wordCnt is too high");
31                 System.exit(1);
32             }
33         } catch (FileNotFoundException fnfe) {
34             System.err.println(fnfe.getMessage());
35             System.exit(0);
36         }
37
38         return words;
39     }
40
41     /**
42      * Writes to file
43      *
44      * @param inArr, filename
45      */
46     public void writeToFile(String[] inArr, String filename) {
47         // checks if inArr is correct length and if last object is null
48         // (this was specified as an requirement in the task description)
49         if(inArr.length == wordCnt && inArr[wordCnt - 1] != null){
50
51             try{
52                 // make file
53                 FileWriter out = new FileWriter(filename);
54                 out.write(inArr.length + "");
55
56                 for(int i = 0; i < inArr.length; i++){
57                     out.write("\r\n" + inArr[i]);
58                 }
59
60                 out.close();
61             }catch (Exception e){
62                 System.err.println("Error: " + e.getMessage());
63             }
64         }
65     }
66 }
67 }
```