INF1010

CURRICULUM

V2012

Java

Joakim
Myrvoll
Johansen

Container with linked list (iterates)

```java
import java.util.Iterator;
import inf1010.lib.two.IfiCollection;


public class OrderedSet<E extends Comparable<? super E>> implements IfiCollection<E> {
    int size;
    E[] arr;
    private ListNode first, last, tmp;
    ListIterator it;

    public OrderedSet(E p){
        first = new ListNode(null);
        last = new ListNode(null);
        first.next = last;
        last.pre = first;
        it = new ListIterator();
        size = 0;

        this.add(p);
    }
    public OrderedSet(){
        first = new ListNode(null);
        last = new ListNode(null);
        first.next = last;
        last.pre = first;
        it = new ListIterator();
        size = 0;
    }

    /**
     * Adds object to collection
     *
     * @param e
     */
    public boolean add(E e) {
        it = new ListIterator();
        tmp = first.next;
        ListNode add = new ListNode(e);

        // already exists?
        if(this.contains(e)){
            return false;
        }
        while(it.hasNext()){
            it.next();
            if(add.current.compareTo(tmp.current) < 0 && tmp.current != null){
                add.next = tmp;
                add.pre = tmp.pre;
                tmp.pre.next = add;
                tmp.pre = add;
                size++;
                return true;
            }
            tmp = tmp.next;
        }
        // added last in list
        add.next = last;
        add.pre = last.pre;
        last.pre.next = add;
        last.pre = add;
        size++;

        return true;
    }

    /**
     * Checks if collection cointains object
     *
     * @param e
     */
    public boolean contains(E e) {
        if(e == null) throw new java.lang.NullPointerException();
        for(E o : this){
            if(o.compareTo(e) == 0){
                System.out.println("This " + e.getClass().getSimpleName() + " already exist.");
                return true;
            }
        }
        return false;
    }

    /**
     * Removes object from collection
     *
     * @param e
     */
    public boolean remove(E e) {
        it = new ListIterator();
        if(e == null) throw new java.lang.NullPointerException();

        while(it.hasNext()){
            E o = it.next();
            if(o.compareTo(e) == 0){
                it.remove();
                return true;
            }
        }
        throw new UnsupportedOperationException();
    }

```

```
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 collecion
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     /**
203      * List
204      *
205      * @author joakimmj
206      */
207     private class ListNode{
208         private E current;
209         private ListNode next;
210         private ListNode pre;
211
212         private ListNode(E e){
213             current = e;
214         }
215     }
216 }
```

IO (read from/write to file)

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.util.Scanner;

/**
 * reader/writer
 */
public class SortIO {
    private String[] words;
    private int x = 0, wordCnt;

    /**
     * Reads from file
     *
     * @param filename
     */
    public String[] readFromFile(String filename) {
        try {
            Scanner s = new Scanner (new File(filename));

            wordCnt = s.nextInt();
            words = new String[wordCnt];

            while (s.hasNext()) {
                words[x++] = s.next();
            }

            if(words[wordCnt - 1] == null){
                System.err.println("Error: wordCnt is too high");
                System.exit(1);
            }
        } catch (FileNotFoundException fnfe) {
            System.err.println(fnfe.getMessage());
            System.exit(0);
        }

        return words;
    }

    /**
     * Writes to file
     *
     * @param inArr, filename
     */
    public void writeToFile(String[] inArr, String filename) {
        // checks if inArr is correct length and if last object is null
        // (this was specified as an requirement in the task description)
        if(inArr.length == wordCnt && inArr[wordCnt -1] != null){

            try{
                // make file
                FileWriter out = new FileWriter(filename);
                out.write(inArr.length + "");

                for(int i = 0; i < inArr.length; i++){
                    out.write("\r\n" + inArr[i]);
                }

                out.close();
            }catch (Exception e){
                System.err.println("Error: " + e.getMessage());
            }

        }
    }
}
```