☐ Evgen959 / Advanced_Backend (Public)

Advanced_Backend / Lesen 037 / code / d06_3_2 / src / MyLikedList.java 📮

078446a · 7 minutes ago

185 lines (159 loc) · 4.07 KB

Evgen959 newLes37

```
<>
Code
         Blame
    1
           import java.util.Iterator;
    2
    3
           public class MyLikedList<E> implements MyList<E>{
    4
               private Node<E> head = null;
    5
               private Node<E> tail = null;
    6
    7
               private int size = 0;
    9
               @Override
               public boolean add(E element) {
   10
   11
                   Node<E> node = new Node<>(tail, null, element);
   12
                   size++;
                   if (tail!=null){
   13
                       tail.setNext(node);
   14
   15
                   if (head==null){
   16
                       head = node;
   17
   18
   19
                   tail = node;
   20
                   return true;
   21
               }
   22
   23
               public boolean isEmpty(){
   24
                   return head==null;
   25
               }
   26
               // 0....size-1
   27
   28
               @Override
               public boolean add(int index, E element) {
   29
                   if (index>=size){
   30
                        return add(element);
   31
   32
                   }
   33
                   Node<E> node = new Node<>(null,null,element);
   34
                   Node<E> next = getNode(index);
   35
                   if(next == null||index<=0){ //добавляем ноду в 0 индекс
   36
                       next=head;
   37
                       head=node;
```

```
38
39
                Node<E> prev = next.getPrev();
40
                next.setPrev(node);
41
                node.setNext(next);
42
                node.setPrev(prev);
43
                if (prev!=null){
                    prev.setNext(node);
44
45
46
                size++;
                return false;
47
48
            }
49
            @Override
50
            public E get(int index) {
51
52
                Node<E> node = getNode(index);
53
                return (node!=null)?node.getValue():null;
54
            }
55
56
            private Node<E> getNode(int index){
57
                if (index>=size || index<0 || head==null){</pre>
58
                    return null;
59
                }
60
                int counter = 0;
                Node<E> aktiveNode = head;
61
62
                while (aktiveNode!=null && counter<index){</pre>
                    aktiveNode = aktiveNode.getNext();
63
                    counter++;
64
                }
65
66
                return aktiveNode;
            }
67
68
69
            @Override
            public int size() {
70
                return size;
71
72
            }
73
            private E remove(Node<E> node){// удоляем ноду
74
75
                if (node==null){
76
                    return null;
77
                }
78
                Node<E> prev = node.getPrev();
79
                Node<E> next = node.getNext();
80
81
                if (prev!=null){
                    prev.setNext(next);
82
83
                } else {
84
                    head = next;
85
86
                if (next!=null){
87
                    next.setPrev(prev);
88
                } else {
89
                    tail = prev;
```

```
91
                 size--;
92
                 node.setPrev(null);
93
                 node.setNext(null);
94
                 E removedValue = node.getValue();
95
                 return removedValue;
96
             }
97
            @Override
98
99
             public E remove(int index) {
100
                 Node<E> node = getNode(index); // ищит ноду
101
                 return remove(node);
102
             }
103
104
            @Override
             public E remove() {
105
                 return remove(tail);
106
107
             }
108
            @Override
109
110
             public E set(int index, E element) {
                 return null;
111
112
             }
113
            @Override
114
115 🗸
             public String toString() {
                 if (head==null){
116
117
                     return "[]";
118
                 }
119
                 StringBuilder sb = new StringBuilder();
120
                 Node<E> currenNode = head;
121
                 while (currenNode!=null){
                     sb.append(currenNode.getValue()).append(";");
122
                     currenNode=currenNode.getNext();
123
124
                 sb.setLength(sb.length()-1);
125
                 return "[" + sb.toString() + ']';
126
127
            }
128
            @Override
129
130
             public Iterator<E> iterator(){
131
                 return this.new MyListIteraror();
132
             }
133
134 🗸
             private class MyListIteraror implements Iterator<E>{
135
                 private Node<E> current = head;
136
137
                 @Override
138
                 public boolean hasNext() {
                     return current!=null;
139
140
                 }
141
142
                 @Override
143
                 public E next() {
```

```
144
                     E value = current.value;
145
                     current=current.next;
                     return value;
146
147
                 }
148
            }
149
            public class Node<E> {
150 🗸
                 Node<E> prev;
151
152
                 Node<E> next;
153
                 E value;
154
155 🗸
                 public Node(Node<E> prev, Node next, E value) {
156
                     this.prev = prev;
157
                     this.next = next;
                     this.value = value;
158
159
                 }
160
                 public Node<E> getPrev() {
161
162
                     return prev;
163
                 }
164
165
                 public void setPrev(Node<E> prev) {
166
                     this.prev = prev;
                 }
167
168
                 public Node<E> getNext() {
169
                     return next;
170
171
172
                 public void setNext(Node<E> next) {
173
174
                     this.next = next;
175
                 }
176
                 public E getValue() {
177
                     return value;
178
179
                 }
180
                 public void setValue(E value) {
181
182
                     this.value = value;
183
                 }
184
             }
185
        }
```