Advanced_Backend / Lesen 035 / code / d05_30_3 / src / MyLikedList.java

Evgen959 newLes35 b517e12 · 34 minutes ago

162 lines (141 loc) · 3.63 KB

Code Blame Raw

```java
public class MyLikedList<E> implements MyList<E>{

    private Node<E> head = null;
    private Node<E> tail = null;
    private int size = 0;

    @Override
    public boolean add(E element) {
        Node<E> node = new Node<>(tail, null, element);
        size++;
        if (tail!=null){
            tail.setNext(node);
        }
        if (head==null){
            head = node;
        }
        tail = node;
        return true;
    }

    public boolean isEmpty(){
        return  head==null;
    }

    //  0....size-1
    @Override
    public boolean add(int index, E element) {
        if (index>=size){
            return add(element);
        }
        Node<E> node = new Node<>(null,null,element);
        Node<E> next = getNode(index);
        if(next == null||index<=0){ //добавляем ноду в 0 индекс
            next=head;
            head=node;
        }
        Node<E> prev = next.getPrev();
```

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

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

```
144             }
145
146             public Node<E> getNext() {
147                 return next;
148             }
149
150             public void setNext(Node<E> next) {
151                 this.next = next;
152             }
153
154             public E getValue() {
155                 return value;
156             }
157
158             public void setValue(E value) {
159                 this.value = value;
160             }
161         }
162     }
```