Advanced_Backend / Lesen 035 / code / d05_30_3 / test / MyLikedListTest.java

Evgen959 newLes35 b517e12 · 36 minutes ago

227 lines (193 loc) · 6.44 KB

Code Blame Raw

```java
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;

class MyLikedListTest {

    @Test
    void add() {
        MyList<String> list = new MyLikedList<>();
        list.add("Jack");
        list.add("John");
        list.add("Nick");

        Assertions.assertEquals(3, list.size());
        Assertions.assertEquals("Jack", list.get(0));
        Assertions.assertEquals("John", list.get(1));
        Assertions.assertEquals("Nick", list.get(2));
    }

    @Test
    @DisplayName("add(index, element): add element at the middle of list")
    void addAtTheMiddleOfList() {
        MyList<String> list = new MyLikedList<>();
        list.add("Jack");
        list.add("John");
        list.add("Nick");
        list.add(2,"Poul");

        Assertions.assertEquals(4, list.size());
        Assertions.assertEquals("Jack", list.get(0));
        Assertions.assertEquals("John", list.get(1));
        Assertions.assertEquals("Poul", list.get(2));
        Assertions.assertEquals("Nick", list.get(3));
    }

    @Test
    @DisplayName("add(index, element): element at the 0 position")
    void addAtTheMiddleOfList1() {
```

```
39              MyList<String> list = new MyLikedList<>();
40              list.add("Jack");
41              list.add("John");
42              list.add("Nick");
43              list.add(0,"Poul");
44
45              Assertions.assertEquals(4, list.size());
46              Assertions.assertEquals("Poul", list.get(0));
47              Assertions.assertEquals("Jack", list.get(1));
48              Assertions.assertEquals("John", list.get(2));
49              Assertions.assertEquals("Nick", list.get(3));
50          }
51
52          @Test
53          @DisplayName("add(index, element): element at the 0 position")
54          void addAtTheMiddleOfList2() {
55              MyList<String> list = new MyLikedList<>();
56              list.add("Jack");
57              list.add("John");
58              list.add("Nick");
59              list.add(3,"Poul");
60
61              Assertions.assertEquals(4, list.size());
62              Assertions.assertEquals("Jack", list.get(0));
63              Assertions.assertEquals("John", list.get(1));
64              Assertions.assertEquals("Nick", list.get(2));
65              Assertions.assertEquals("Poul", list.get(3));
66          }
67
68
69
70          @Test
71          @DisplayName("Add element by index at the 0 position")
72          void addAtTheMiddleOfList3() {
73              MyList<String> list = new MyLikedList<>();
74              list.add(3,"Poul");
75
76              Assertions.assertEquals(1, list.size());
77              Assertions.assertEquals("Poul", list.get(0));
78          }
79
80          @Test
81          @DisplayName("add(index, element): several elements")
82          void addAtTheMiddleOfList4() {
83              MyList<String> list = new MyLikedList<>();
84              list.add(0,"Jack");
85              list.add(1, "John");
86              list.add(0,"Nick");
87              list.add(1,"Poul");
88
89              Assertions.assertEquals(4, list.size());
90              Assertions.assertEquals("Nick", list.get(0));
91              Assertions.assertEquals("Poul", list.get(1));
92              Assertions.assertEquals("Jack", list.get(2));
```

```
93              Assertions.assertEquals("John", list.get(3));
94          }
95
96          @Test
97          @DisplayName("add(element): integers list")
98          void addInteger() {
99              MyList<Integer> list = new MyLikedList<>();
100             list.add(1);
101             list.add(3);
102             list.add(5);
103
104             Assertions.assertEquals(3, list.size());
105             Assertions.assertEquals(1, list.get(0));
106             Assertions.assertEquals(3, list.get(1));
107             Assertions.assertEquals(5, list.get(2));
108         }
109
110         @Test
111         @DisplayName("add(element): after removing")
112         void add1() {
113             MyList<String> list = new MyLikedList<>();;
114             list.add("Jack");
115             list.add("John");
116             list.add("Nick");
117             list.remove(2);
118             list.add("Ann");
119
120             Assertions.assertEquals(3, list.size());
121             Assertions.assertEquals("Jack", list.get(0));
122             Assertions.assertEquals("John", list.get(1));
123             Assertions.assertEquals("Ann", list.get(2));
124         }
125
126         @Test
127         void get() {
128             MyList<String> list = new MyLikedList<>();
129             list.add("Jack");
130             list.add("John");
131             list.add("Nick");
132
133             Assertions.assertEquals("John", list.get(1));
134         }
135
136         @Test
137         void size() {
138             MyList<String> list = new MyLikedList<>();
139             list.add("Jack");
140             list.add("John");
141             list.add("Nick");
142             Assertions.assertEquals(3, list.size());
143         }
144
145         @Test
146         @DisplayName("size should be 0 if list is empty")
```

```
147         void size1() {
148             MyList<String> list = new MyLikedList<>();
149             Assertions.assertEquals(0, list.size());
150         }
151 
152         @Test
153         @DisplayName("regular remove")
154         void remove() {
155             MyList<String> list = new MyLikedList<>();
156             list.add("Jack");
157             list.add("John");
158             list.add("Nick");
159             list.remove(1);
160 
161             Assertions.assertEquals(2, list.size());
162             Assertions.assertEquals("Jack", list.get(0));
163             Assertions.assertEquals("Nick", list.get(1));
164         }
165 
166         @Test
167         @DisplayName("regular tail remove")
168         void remove1() {
169             MyList<String> list = new MyLikedList<>();;
170             list.add("Jack");
171             list.add("John");
172             list.add("Nick");
173             list.remove(2);
174 
175             Assertions.assertEquals(2, list.size());
176             Assertions.assertEquals("Jack", list.get(0));
177             Assertions.assertEquals("John", list.get(1));
178         }
179 
180         @Test
181         @DisplayName("remove head remove")
182         void remove2() {
183             MyList<String> list = new MyLikedList<>();
184             list.add("Jack");
185             list.add("John");
186             list.add("Nick");
187             list.remove(0);
188 
189             Assertions.assertEquals(2, list.size());
190             Assertions.assertEquals("John", list.get(0));
191             Assertions.assertEquals("Nick", list.get(1));
192         }
193 
194         @Test
195         @DisplayName("remove() last remove")
196         void remove3() {
197             MyList<String> list = new MyLikedList<>();
198             list.add("Jack");
199             list.add("John");
200             list.add("Nick");
```

```
201             list.remove();
202             boolean isSizeCorrect = list.size()==2;
203             list.remove();
204             isSizeCorrect = isSizeCorrect? list.size()==1: false;
205             list.remove();
206             isSizeCorrect = isSizeCorrect? list.size()==0: false;
207
208             Assertions.assertTrue(isSizeCorrect);
209             Assertions.assertNull(list.get(0));
210         }
211
212         @Test
213         @DisplayName("remove() last remove from single element list")
214         void remove4() {
215             MyList<String> list = new MyLikedList<>();
216             list.add("Jack");
217             String removedString = list.remove();
218
219             Assertions.assertEquals(0, list.size());
220             Assertions.assertEquals("Jack", removedString);
221             Assertions.assertNull(list.get(0));
222         }
223
224         @Test
225         void set() {
226         }
227     }
```