Evgen959 / Advanced_Backend Public

Code Issues Pull requests Actions Projects Security Insights

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() {
        MyList<String> list = new MyLikedList<>();
        list.add("Jack");
        list.add("John");
        list.add("Nick");
        list.add(0,"Poul");

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

    @Test
    @DisplayName("add(index, element): element at the 0 position")
    void addAtTheMiddleOfList2() {
        MyList<String> list = new MyLikedList<>();
        list.add("Jack");
        list.add("John");
        list.add("Nick");
        list.add(3,"Poul");

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



    @Test
    @DisplayName("Add element by index at the 0 position")
    void addAtTheMiddleOfList3() {
        MyList<String> list = new MyLikedList<>();
        list.add(3,"Poul");

        Assertions.assertEquals(1, list.size());
        Assertions.assertEquals("Poul", list.get(0));
    }

    @Test
    @DisplayName("add(index, element): several elements")
    void addAtTheMiddleOfList4() {
        MyList<String> list = new MyLikedList<>();
        list.add(0,"Jack");
        list.add(1, "John");
        list.add(0,"Nick");
        list.add(1,"Poul");

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

```
            Assertions.assertEquals("John", list.get(3));
        }

        @Test
        @DisplayName("add(element): integers list")
        void addInteger() {
            MyList<Integer> list = new MyLikedList<>();
            list.add(1);
            list.add(3);
            list.add(5);

            Assertions.assertEquals(3, list.size());
            Assertions.assertEquals(1, list.get(0));
            Assertions.assertEquals(3, list.get(1));
            Assertions.assertEquals(5, list.get(2));
        }

        @Test
        @DisplayName("add(element): after removing")
        void add1() {
            MyList<String> list = new MyLikedList<>();;
            list.add("Jack");
            list.add("John");
            list.add("Nick");
            list.remove(2);
            list.add("Ann");

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

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

            Assertions.assertEquals("John", list.get(1));
        }

        @Test
        void size() {
            MyList<String> list = new MyLikedList<>();
            list.add("Jack");
            list.add("John");
            list.add("Nick");
            Assertions.assertEquals(3, list.size());
        }

        @Test
        @DisplayName("size should be 0 if list is empty")
```

```java
    void size1() {
        MyList<String> list = new MyLikedList<>();
        Assertions.assertEquals(0, list.size());
    }

    @Test
    @DisplayName("regular remove")
    void remove() {
        MyList<String> list = new MyLikedList<>();
        list.add("Jack");
        list.add("John");
        list.add("Nick");
        list.remove(1);

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

    @Test
    @DisplayName("regular tail remove")
    void remove1() {
        MyList<String> list = new MyLikedList<>();;
        list.add("Jack");
        list.add("John");
        list.add("Nick");
        list.remove(2);

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

    @Test
    @DisplayName("remove head remove")
    void remove2() {
        MyList<String> list = new MyLikedList<>();
        list.add("Jack");
        list.add("John");
        list.add("Nick");
        list.remove(0);

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

    @Test
    @DisplayName("remove() last remove")
    void remove3() {
        MyList<String> list = new MyLikedList<>();
        list.add("Jack");
        list.add("John");
        list.add("Nick");
```

```java
            list.remove();
            boolean isSizeCorrect = list.size()==2;
            list.remove();
            isSizeCorrect = isSizeCorrect? list.size()==1: false;
            list.remove();
            isSizeCorrect = isSizeCorrect? list.size()==0: false;

            Assertions.assertTrue(isSizeCorrect);
            Assertions.assertNull(list.get(0));
        }

        @Test
        @DisplayName("remove() last remove from single element list")
        void remove4() {
            MyList<String> list = new MyLikedList<>();
            list.add("Jack");
            String removedString = list.remove();

            Assertions.assertEquals(0, list.size());
            Assertions.assertEquals("Jack", removedString);
            Assertions.assertNull(list.get(0));
        }

        @Test
        void set() {
        }
    }
```