ait-tr / cohort44E Public

generated from ait-tr/cohort-xx.x

Code Issues Pull requests Actions Projects Security Insights

cohort44E / basic_programming / lesson_34 / code / demo20240529_GenericLinkedList / test / MyLinkedListTest.java

Andy179176 20240529 461fabc · 9 hours ago

158 lines (135 loc) · 4.38 KB

Code Blame Raw

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

import static org.junit.jupiter.api.Assertions.*;

class MyLinkedListTest {

    @Test
    void add() {
        MyList<String> list = new MyLinkedList<>();
        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
    void addInteger() {
        MyList<Integer> list = new MyLinkedList<>();
        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 after removing")
    void add1() {
        MyList<String> list = new MyLinkedList<>();
        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 MyLinkedList<>();
        list.add("Jack");
        list.add("John");
        list.add("Nick");

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

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

    @Test
    @DisplayName("size should be 0 if list is empty")
    void size1() {
        MyList<String> list = new MyLinkedList<>();
        Assertions.assertEquals(0, list.size());
    }


    @Test
    @DisplayName("regular remove")
    void remove() {
        MyList<String> list = new MyLinkedList<>();
        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("remove tail element")
    void remove1() {
        MyList<String> list = new MyLinkedList<>();
        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 element")
    void remove2() {
        MyList<String> list = new MyLinkedList<>();
        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 element")
    void remove3() {
        MyList<String> list = new MyLinkedList<>();
        list.add("Jack");
        list.add("John");
        list.add("Nick");
        String removed = list.remove();
        boolean isSizeCorrect = list.size()==2;
        boolean isValueCorrect = removed.equals("Nick");
        removed = list.remove();
        isSizeCorrect = isSizeCorrect? list.size()==1:false;
        isValueCorrect = isValueCorrect? removed.equals("John"):false;
        removed = list.remove();
        isSizeCorrect = isSizeCorrect?list.size()==0:false;
        isValueCorrect = isValueCorrect? removed.equals("Jack"):false;

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

    @Test
    @DisplayName("remove() last element from single element list")
    void remove4() {
        MyList<String> list = new MyLinkedList<>();
```

```java
            list.add("Jack");
            String removedString = list.remove();

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

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