Evgen959 / Advanced_Backend Public

Code Issues Pull requests Actions Projects Security Insights

Advanced_Backend / Lesen 032 / code / hw_na_les031_1 / test / MainTest.java

Evgen959 newLes32 74dd8bb · 10 hours ago

119 lines (93 loc) · 4.36 KB

Code Blame Raw

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

import java.util.ArrayList;
import java.util.List;

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

class MainTest {

    @Test
    @DisplayName("selectEmployeeForBonus: regular case")
    void selectEmployeeForBonus() {
        List<Employee> list = new ArrayList<>();
        list.add(new Employee("John", 2020));
        list.add(new Employee("Anna", 2016));
        list.add(new Employee("Jack", 1999));
        list.add(new Employee("Mark", 1995));
        list.add(new Employee("Mark", 1997));
        list.add(new Employee("Oleg", 2000));
        list.add(new Employee("Valerii", 2005));

        List<Employee> expectedList = new ArrayList<>();
        expectedList.add(new Employee("Mark", 1997));
        expectedList.add(new Employee("Mark", 1995));
        expectedList.add(new Employee("Jack", 1999));

        List<Employee> result = Main.selectEmployeeForBonus(list, 3);
        Assertions.assertTrue(isEqualsIgnoreOrder(expectedList,result),
                String.format("%nExpected: %s%nActual:   %s%n", expectedList, result));

        //Assertions.assertEquals(expectedList, result);
    }

    @Test
    @DisplayName("selectEmployeeForBonus: regular case case| more than N employees")
    void selectEmployeeForBonus2() {
        List<Employee> list = new ArrayList<>();
        list.add(new Employee("John", 2020));
        list.add(new Employee("Anna", 2016));
        list.add(new Employee("Jack", 1999));
        list.add(new Employee("Tom", 1999));
        list.add(new Employee("Mark", 1995));
        list.add(new Employee("Mark", 1997));
        list.add(new Employee("Igor", 1999));
        list.add(new Employee("Oleg", 2000));
        list.add(new Employee("Valerii", 2005));

        List<Employee> expectedList = new ArrayList<>();
        expectedList.add(new Employee("Mark", 1997));
        expectedList.add(new Employee("Mark", 1995));
        expectedList.add(new Employee("Jack", 1999));
        expectedList.add(new Employee("Tom", 1999));
        expectedList.add(new Employee("Igor", 1999));


        List<Employee> result = Main.selectEmployeeForBonus(list, 3);
        Assertions.assertTrue(isEqualsIgnoreOrder(expectedList,result),
                String.format("%nExpected: %s%nActual:   %s%n", expectedList, result));

        //Assertions.assertEquals(expectedList, result);
    }

    @Test
    @DisplayName("selectEmployeeForBonus: all employee sould be selected")
    void selectAllEmployeeForBonus() {
        List<Employee> list = new ArrayList<>();
        list.add(new Employee("John", 2020));
        list.add(new Employee("Anna", 2016));
        list.add(new Employee("Jack", 1999));
        list.add(new Employee("Mark", 1995));

        List<Employee> expectedList = new ArrayList<>();
        expectedList.add(new Employee("John", 2020));
        expectedList.add(new Employee("Anna", 2016));
        expectedList.add(new Employee("Jack", 1999));
        expectedList.add(new Employee("Mark", 1995));

        List<Employee> result = Main.selectEmployeeForBonus(list, 7);
        Assertions.assertTrue(isEqualsIgnoreOrder(expectedList,result),
                String.format("%nExpected: %s%nActual:   %s%n", expectedList, result));

        //Assertions.assertEquals(expectedList, result);
    }


    @Test
    @DisplayName("createSortedCopy: by year")
    void createSortedCopy() {
        List<Employee> list = new ArrayList<>();
        list.add(new Employee("John", 2020));
        list.add(new Employee("Anna", 2016));
        list.add(new Employee("Jack", 1999));
        list.add(new Employee("Mark", 1995));

        List<Employee> expected = new ArrayList<>();
        expected.add(new Employee("Mark", 1995));
        expected.add(new Employee("Jack", 1999));
        expected.add(new Employee("Anna", 2016));
        expected.add(new Employee("John", 2020));


        List<Employee> result = Main.createSortedCopy(list, new ComparatorEmployeeByYear());

        Assertions.assertEquals(expected, result);
        Assertions.assertFalse(result==list); // проверяем, что листы разные


    }

    private boolean isEqualsIgnoreOrder(List<Employee> list1,List<Employee> list2){
        return      list1!=null
                &&  list2!=null
                &&  list1.size() == list2.size()
                && list1.containsAll(list2);
    }

}
```