Лабораторная работа №3
Компонентное или модульное тестирование (юнит-тестирование)

Цель работы: научиться технике модульного тестирования с помощью фреймворка JUnit.

Требования: тестируемая программа на языке Java, набор модульных тестов к ней

Написать модульные тесты с использованием JUnit 4 для каждого из классов программы.
Соблюдать Java Conventions.
Обеспечить максимальное покрытие тестами, модульное тестирование которого проводится.
При составлении тест-кейсов использовать техники тест-дизайна (эквивалентное разделение, анализ граничных значений, причина / следствие, предугадывание ошибки).
Обработка исключительных ситуаций должна быть реализована только через механизм Java Exceptions.

Вариант: 17

Задание: Реализовать класс “Смартфон” со следующими полями: производитель, название модели, диагональ дисплея, операционная система, толщина корпуса. Обеспечить возможность установки и считывания всех полей. Добавить возможность “связывания смартфонов”, которое возможно только в случае, если у двух смартфонов один и тот же производитель, либо одинаковая операционная система.

1. Реализация класса Smartphone

Файл Smartphone.java:

```java
package ru.miet.CourceTesting.Lr3;

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

/**
 *
 */
public class Smartphone {

    /**
     * Производитель
     */
    private String vendor;
    /**
     * Название модели
     */
    private String modelName;
    /**
     * Диагональ дисплея
     */
    private double displaySize;
    /**
     * Операционная система
     */
    private String os;
    /**
     * Толщина устройства
     */
    private double thickness;
    /**
     *
     */
    private List<Smartphone> linkedDevices = new ArrayList<>();

    /**
     *
     * @param vendor
     * @param modelName
     * @param displaySize
     * @param os
```

```
42.       * @param thickness
43.       */
44.      public Smartphone(String vendor, String modelName, double displaySize, String os, double thickness)
{
45.          setVendor(vendor);
46.          setModelName(modelName);
47.          setDisplaySize(displaySize);
48.          setOs(os);
49.          setThickness(thickness);
50.      }
51.
52.      /**
53.       *
54.       * @return
55.       */
56.      public String getVendor() {
57.          return vendor;
58.      }
59.
60.      /**
61.       *
62.       * @return
63.       */
64.      public String getModelName() {
65.          return modelName;
66.      }
67.
68.      /**
69.       *
70.       * @return
71.       */
72.      public double getDisplaySize() {
73.          return displaySize;
74.      }
75.
76.      /**
77.       *
78.       * @return
79.       */
80.      public String getOs() {
81.          return os;
82.      }
83.
84.      /**
85.       *
86.       * @return
87.       */
88.      public double getThickness() {
89.          return thickness;
90.      }
91.
92.      /**
93.       *
94.       * @return
95.       */
96.      public List<Smartphone> getLinkedDevices() {
97.          return new ArrayList<>(linkedDevices);
98.      }
99.
100.     /**
101.      *
102.      * @param vendor
103.      */
104.     public void setVendor(String vendor) {
105.         if (vendor == null || vendor.trim().isEmpty()) {
106.             throw new IllegalArgumentException("Производитель не может быть пустым");
107.         }
108.         this.vendor = vendor.trim();
109.     }
110.
111.     /**
112.      *
113.      * @param modelName
114.      */
115.     public void setModelName(String modelName) {
116.         if (modelName == null || modelName.trim().isEmpty()) {
117.             throw new IllegalArgumentException("Название модели не может быть пустым");
118.         }
```

```
119.          this.modelName = modelName.trim();
120.      }
121.
122.      /**
123.       *
124.       * @param displaySize
125.       */
126.      public void setDisplaySize(double displaySize) {
127.          if (displaySize == Double.NaN || displaySize <= 0) {
128.              throw new IllegalArgumentException("Диагональ дисплея должна быть положительной");
129.          }
130.          this.displaySize = displaySize;
131.      }
132.
133.      /**
134.       *
135.       * @param os
136.       */
137.      public void setOs(String os) {
138.          if (os == null || os.trim().isEmpty()) {
139.              throw new IllegalArgumentException("ОС не может быть пустой");
140.          }
141.          this.os = os.trim();
142.      }
143.
144.      /**
145.       *
146.       * @param thickness
147.       */
148.      public void setThickness(double thickness) {
149.          if (thickness == Double.NaN || thickness <= 0) {
150.              throw new IllegalArgumentException("Толщина корпуса должна быть положительной");
151.          }
152.          this.thickness = thickness;
153.      }
154.
155.      /**
156.       * Выполняет связываение с другой моделью телефона
157.       *
158.       * @param other
159.       * @throws InvalidLinkingException - если телефоны уже связаны или
160.       *                                   производитель/системы различны
161.       */
162.      public void linkTo(Smartphone other) throws InvalidLinkingException {
163.          if (this == other) {
164.              throw new InvalidLinkingException("Невозможно связать смартфон с самим собой");
165.          }
166.
167.          if (linkedDevices.contains(other)) {
168.              throw new InvalidLinkingException("Смартфоны уже связаны");
169.          }
170.
171.          boolean isSameVendor = this.vendor.equalsIgnoreCase(other.vendor);
172.          boolean isSameOS = this.os.equalsIgnoreCase(other.os);
173.
174.          if (!isSameVendor && !isSameOS) {
175.              throw new InvalidLinkingException("Связь возможна только при совпадении производителя или
ОС");
176.          }
177.
178.          this.linkedDevices.add(other);
179.          other.linkedDevices.add(this);
180.      }
181. }
182.
```

Файл InvalidLinkingException.java:

```
1. package ru.miet.CourceTesting.Lr3;
2.
3. class InvalidLinkingException extends Exception {
4.      public InvalidLinkingException(String message) {
5.   super(message);
6.      }
7. }
8.
```

2. Реализация класса SmartphoneTest

```java
package ru.miet.CourceTesting.Lr3;

import java.util.List;

import org.junit.Before;
import org.junit.Test;

public class SmartphoneTest {

    private Smartphone samsungAndroid;
    private Smartphone xiaomiAndroid;
    private Smartphone samsungCustomOS;
    private Smartphone appleIOS;

    @Before
    public void setUp() {
        samsungAndroid = new Smartphone("Samsung", "S24", 6.2, "Android", 7.9);
        xiaomiAndroid = new Smartphone("Xiaomi", "Note 13", 6.6, "Android", 8.1);
        samsungCustomOS = new Smartphone("Samsung", "Fold", 7.6, "One UI", 6.9);
        appleIOS = new Smartphone("Apple", "iPhone 16", 6.1, "iOS", 7.4);
    }

    // Тестирование конструктора и базовой функциональности
    @Test
    public void testConstructorAndGetters() {
        assertEquals("Samsung", samsungAndroid.getVendor());
        assertEquals("S24", samsungAndroid.getModelName());
        assertEquals(6.2, samsungAndroid.getDisplaySize(), 0.001);
        assertEquals(7.9, samsungAndroid.getThickness(), 0.001);
    }

    // Тестирование сеттеров с валидацией (анализ граничных значений)
    @Test
    public void setVendor_EmptyValue_ThrowsException() {
        assertThrows(IllegalArgumentException.class, () -> {
            samsungAndroid.setVendor("");
        });
        assertThrows(IllegalArgumentException.class, () -> {
            samsungAndroid.setVendor(null);
        });
    }

    // Тестирование сеттеров с валидацией (анализ граничных значений)
    @Test
    public void setModelName_EmptyValue_ThrowsException() {
        assertThrows(IllegalArgumentException.class, () -> {
            samsungAndroid.setModelName(null);
        });
        assertThrows(IllegalArgumentException.class, () -> {
            samsungAndroid.setModelName("");
        });
    }

    // Тестирование сеттеров с валидацией (анализ граничных значений)
    @Test
    public void setOS_EmptyValue_ThrowsException() {
        assertThrows(IllegalArgumentException.class, () -> {
            samsungAndroid.setOs("");
        });
        assertThrows(IllegalArgumentException.class, () -> {
            samsungAndroid.setOs(null);
        });
    }

    // Тестирование сеттеров с валидацией (анализ граничных значений)
    @Test
    public void setThinckness_EmptyValue_ThrowsException() {
        assertThrows(IllegalArgumentException.class, () -> {
            samsungAndroid.setThickness(0);
        });
        assertThrows(IllegalArgumentException.class, () -> {
            samsungAndroid.setThickness(-1);
        });
        assertThrows(IllegalArgumentException.class, () -> {
            samsungAndroid.setThickness(Double.NaN);
```

```
        });
    }

    // Тестирование сеттеров с валидацией (анализ граничных значений)
    @Test
    public void setDisplaySize_ZeroValue_ThrowsException() {
        assertThrows(IllegalArgumentException.class, () -> {
            samsungAndroid.setDisplaySize(0);
        });
        assertThrows(IllegalArgumentException.class, () -> {
            samsungAndroid.setDisplaySize(-1);
        });
        assertThrows(IllegalArgumentException.class, () -> {
            samsungAndroid.setDisplaySize(Double.NaN);
        });
    }

    @Test
    public void setOs_ValidValue_UpdatesCorrectly() {
        samsungAndroid.setOs("Android 14");
        assertEquals("Android 14", samsungAndroid.getOs());
    }

    // Тестирование связывания (техника причина/следствие)
    @Test
    public void linkTo_SameOS_Success() throws Exception {
        samsungAndroid.linkTo(xiaomiAndroid);
        assertTrue(devicesAreLinked(samsungAndroid, xiaomiAndroid));
    }

    @Test
    public void linkTo_SameVendor_Success() throws Exception {
        samsungAndroid.linkTo(samsungCustomOS);
        assertTrue(devicesAreLinked(samsungAndroid, samsungCustomOS));
    }

    @Test(expected = InvalidLinkingException.class)
    public void linkTo_DifferentVendorAndOS_ThrowsException() throws Exception {
        samsungAndroid.linkTo(appleIOS);
    }

    @Test(expected = InvalidLinkingException.class)
    public void linkTo_SelfLinking_ThrowsException() throws Exception {
        samsungAndroid.linkTo(samsungAndroid);
    }

    @Test(expected = InvalidLinkingException.class)
    public void linkTo_DuplicateLinking_ThrowsException() throws Exception {
        samsungAndroid.linkTo(xiaomiAndroid);
        samsungAndroid.linkTo(xiaomiAndroid);
    }

    // Тестирование исключительных ситуаций (предугадывание ошибок)
    @Test
    public void linkTo_NullDevice_ThrowsNPE() {
        assertThrows(NullPointerException.class, () -> {
            samsungAndroid.linkTo(null);
        });
    }

    // Вспомогательные методы
    private boolean devicesAreLinked(Smartphone a, Smartphone b) {
        return a.getLinkedDevices().contains(b) && b.getLinkedDevices().contains(a);
    }

    // Тестирование возвращаемой копии списка
    @Test
    public void getLinkedDevices_ReturnsCopy() throws Exception {
        samsungAndroid.linkTo(xiaomiAndroid);
        List<Smartphone> links = samsungAndroid.getLinkedDevices();
        links.clear();
        assertFalse(samsungAndroid.getLinkedDevices().isEmpty());
    }
}
```

3. Результаты покрытия и тестирования

После прогона всех тест-кейсов были получены следующие результаты:

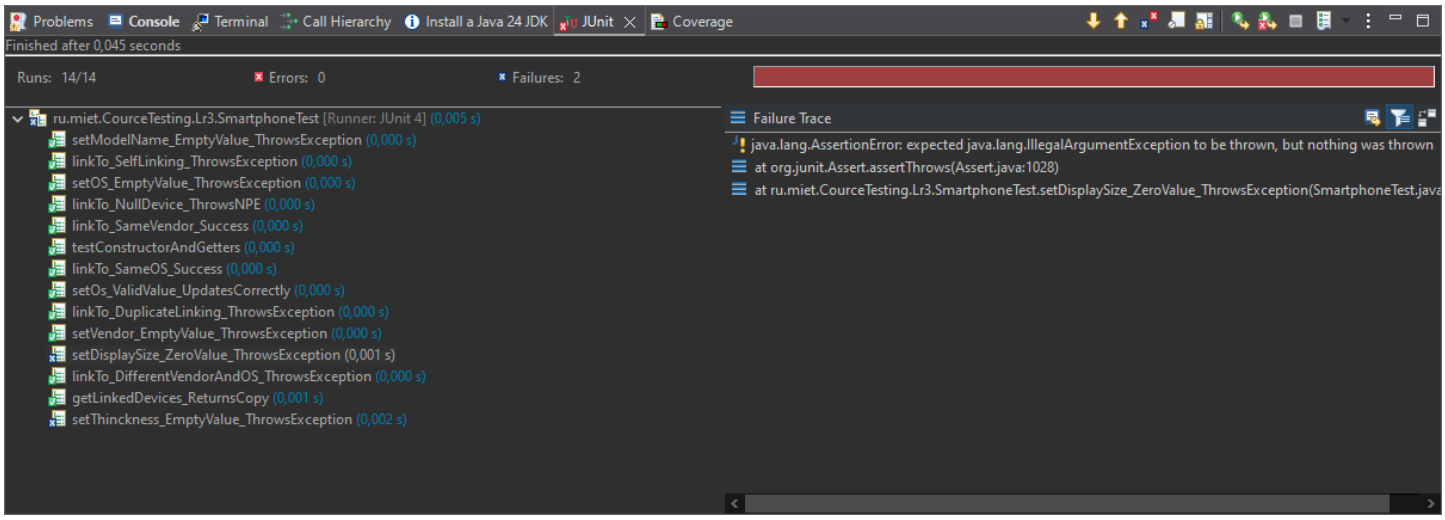

Рис.1: Результаты прогона теста.

Можно заметить, что два тест-кейса провалились, поскольку:

«Тестировщик» в тест кейсе заложил проверку double значения на NaN:

```
84      // Тестирование сеттеров с валидацией (анализ граничных значений)
85      @Test
86      public void setDisplaySize_ZeroValue_ThrowsException() {
87          assertThrows(IllegalArgumentException.class, () -> {
88              samsungAndroid.setDisplaySize(0);
89          });
90          assertThrows(IllegalArgumentException.class, () -> {
91              samsungAndroid.setDisplaySize(-1);
92          });
93          assertThrows(IllegalArgumentException.class, () -> {
94              samsungAndroid.setDisplaySize(Double.NaN);
95          });
96      }
```

А «Разработчик» выполняет проверку неправильно:

```
126     public void setDisplaySize(double displaySize) {
127         if (displaySize == Double.NaN || displaySize <= 0) {
128             throw new IllegalArgumentException("Диагональ дисплея должна быть положительной");
129         }
130         this.displaySize = displaySize;
131     }
```

Результаты покрытия исходного кода:

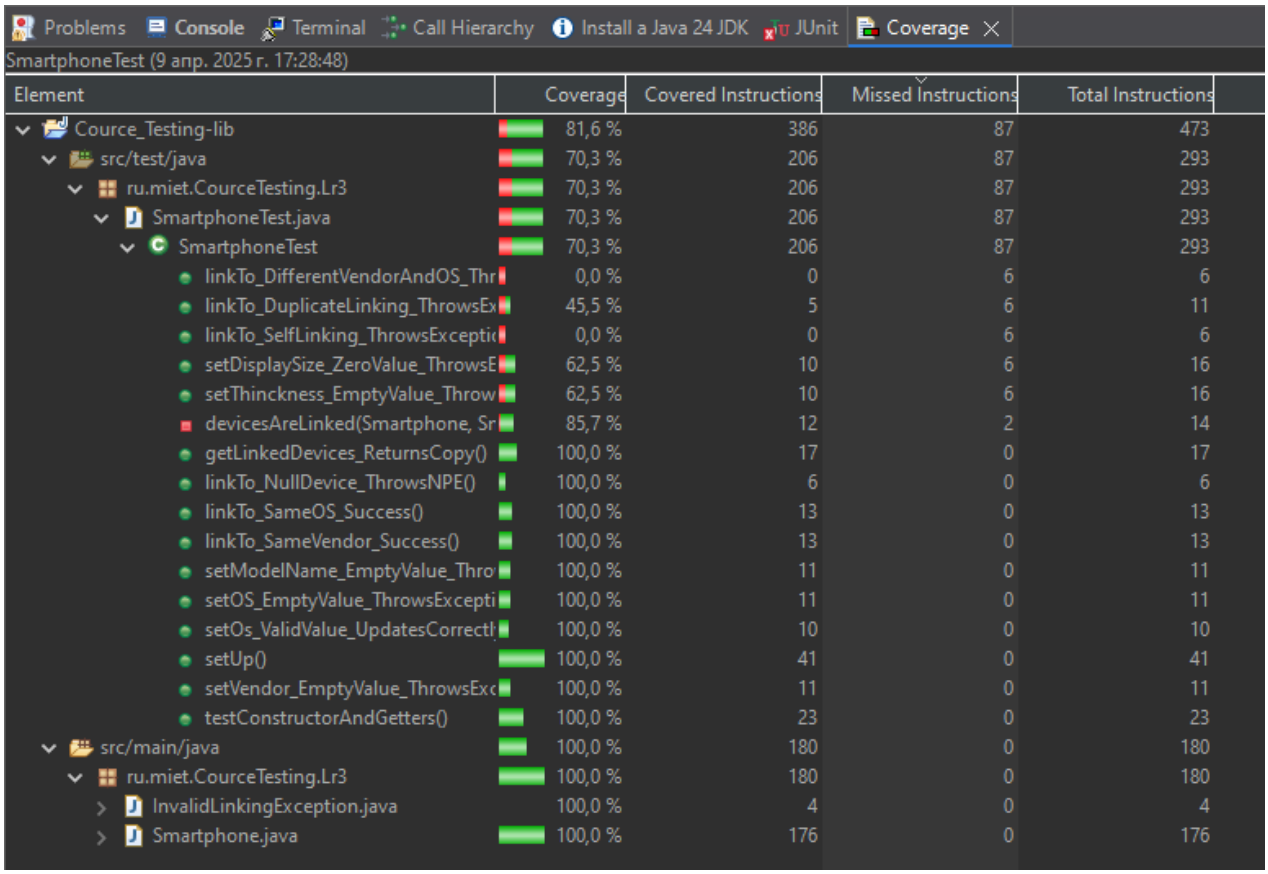

Рис.2: Результаты покрытия исходного кода