

# Лабораторная работа №4

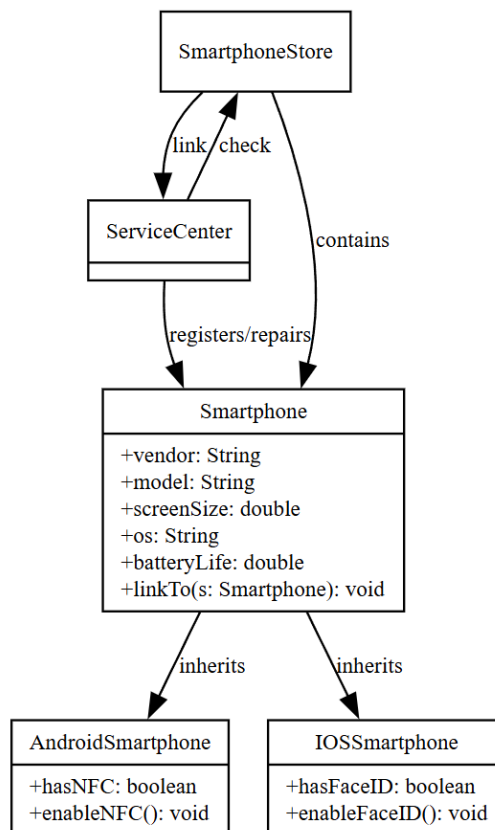
## Интеграционное тестирование

**Цель работы:** научиться выполнять интеграционное тестирование ПО.

**Отчет по лабораторной работе:** тестируемая программа на языке Java, набор интеграционных и модульных тестов к ней на основе JUnit 4.

**Задание:** Сформировать иерархию классов вокруг класса, протестированного в лабораторной работе №3. К примеру, для класса «Животное» это могут быть наследники «Кошка», «Собака» и агрегирующие классы «Зоопарк» и «Лес». Провести модульное тестирование написанных классов. Провести инкрементальное интеграционное тестирование «сверху вниз».

### 1. Разработка иерархии классов



**Рис.1:** Диаграмма иерархии классов

## 2. Реализация классов

### Класс AndroidSmartphone

```
1. package ru.miet.CourseTesting.Lr4;
2.
3. import ru.miet.CourseTesting.Lr3.Smartphone;
4.
5. /**
6.  *
7.  */
8. public class AndroidSmartphone extends Smartphone {
9.     private String androidVersion;
10.    private boolean hasNFC;
11.    private boolean isNFCEnable;
12.
13.    public AndroidSmartphone(String vendor, String modelName, double displaySize, double thickness,
14.        String androidVersion, boolean hasNFC) {
15.        super(vendor, modelName, displaySize, "Android", thickness);
16.        setAndroidVersion(androidVersion);
17.        setHasNFC(hasNFC);
18.        updateStateNFC(false);
19.    }
20.
21.    public String getAndroidVersion() {
22.        return androidVersion;
23.    }
24.
25.    public void setAndroidVersion(String androidVersion) {
26.        this.androidVersion = androidVersion;
27.    }
28.
29.    public boolean isHasNFC() {
30.        return hasNFC;
31.    }
32.
33.    public void setHasNFC(boolean hasNFC) {
34.        this.hasNFC = hasNFC;
35.    }
36.
37.    private void updateStateNFC(boolean state) {
38.        isNFCEnable = state;
39.    }
40.
41.    public boolean getStateNFC() {
42.        return isNFCEnable;
43.    }
44.
45.    public void enableNFC() {
46.        if (hasNFC) {
47.            System.out.println("NFC включен на " + getModelName());
48.            updateStateNFC(!isNFCEnable);
49.        } else {
50.            System.out.println("Устройство не поддерживает NFC.");
51.        }
52.    }
53. }
```

## Класс IOSSmartphone

```
1. package ru.miet.CourseTesting.Lr4;
2.
3. import ru.miet.CourseTesting.Lr3.Smartphone;
4.
5. /**
6.  *
7.  */
8. public class IOSSmartphone extends Smartphone {
9.     private String iosVersion;
10.    private boolean hasFaceID;
11.    private boolean isFaceIDEnable;
12.
13.    public IOSSmartphone(String vendor, String modelName, double displaySize, double thickness, String
iosVersion,
14.        boolean hasFaceID) {
15.        super(vendor, modelName, displaySize, "iOS", thickness);
16.        setIosVersion(iosVersion);
17.        setHasFaceID(hasFaceID);
18.        updateStateFaceID(false);
19.    }
20.
21.    public String getIosVersion() {
22.        return iosVersion;
23.    }
24.
25.    public void setIosVersion(String iosVersion) {
26.        this.iosVersion = iosVersion;
27.    }
28.
29.    public boolean isHasFaceID() {
30.        return hasFaceID;
31.    }
32.
33.    public void setHasFaceID(boolean hasFaceID) {
34.        this.hasFaceID = hasFaceID;
35.    }
36.
37.    private void updateStateFaceID(boolean state) {
38.        isFaceIDEnable = state;
39.    }
40.
41.    public boolean getStateFaceID() {
42.        return isFaceIDEnable;
43.    }
44.
45.    public void enableFaceID() {
46.        if (hasFaceID) {
47.            System.out.println("Face ID включен на " + getModelName());
48.            updateStateFaceID(!isFaceIDEnable);
49.        } else {
50.            System.out.println("Устройство не поддерживает Face ID.");
51.        }
52.    }
53. }
```

## Класс PhoneStore

```
1. package ru.miet.CourseTesting.Lr4;
2.
3. import java.util.ArrayList;
4. import java.util.List;
5.
6. import ru.miet.CourseTesting.Lr3.Smartphone;
7.
8. /**
9.  *
10. */
11. public class PhoneStore {
12.     protected String storeName;
13.     protected List<Smartphone> smartphones;
14.     protected List<Smartphone> soldSmartphones;
15.     protected List<ServiceCenter> certifiedServiceCenter;
16.
17.     public PhoneStore(String storeName) {
18.         this.storeName = storeName;
19.         smartphones = new ArrayList<>();
20.         soldSmartphones = new ArrayList<>();
21.         certifiedServiceCenter = new ArrayList<>();
22.     }
23.
24.     public String getStoreName() {
25.         return storeName;
26.     }
27.
28.     public void setStoreName(String storeName) {
29.         if (storeName == null || storeName.trim().isEmpty()) {
30.             throw new IllegalArgumentException("Название продавца не может быть пустым");
31.         }
32.         this.storeName = storeName;
33.     }
34.
35.     public List<Smartphone> getSmartphones() {
36.         return smartphones;
37.     }
38.
39.     public List<Smartphone> getSelledSmartphones() {
40.         return soldSmartphones;
41.     }
42.
43.     public void pushServiceCenter(ServiceCenter sc) {
44.         if (sc == null) {
45.             throw new IllegalArgumentException("Сервисный центр не может быть null");
46.         }
47.         certifiedServiceCenter.add(sc);
48.     }
49.
50.     public List<ServiceCenter> getCertifiedServiceCenter() {
51.         return certifiedServiceCenter;
52.     }
53.
54.     public void addSmartphone(Smartphone smartphone) {
55.         if (smartphone == null) {
56.             throw new IllegalArgumentException("Смартфон не может быть null");
57.         }
58.         smartphones.add(smartphone);
59.     }
60.
61.     public void sellSmartphone(Smartphone smartphone) {
62.         if (smartphone == null) {
63.             throw new IllegalArgumentException("Смартфон не может быть null");
64.         }
65.         if (!smartphones.contains(smartphone)) {
66.             throw new IllegalArgumentException("Смартфон не принадлежит магазину");
67.         }
68.         smartphones.remove(smartphone);
69.         soldSmartphones.add(smartphone);
70.     }
71. }
```

## Класс ServiceCenter

```
1. package ru.miet.CourseTesting.Lr4;
2.
3. import java.util.ArrayList;
4. import java.util.List;
5.
6. import ru.miet.CourseTesting.Lr3.Smartphone;
7.
8. /**
9.  *
10. */
11. public class ServiceCenter {
12.     private String centerName;
13.     private List<Smartphone> repairedSmartphones;
14.     private List<PhoneStore> officialStores;
15.
16.     public ServiceCenter(String centerName) {
17.         this.centerName = centerName;
18.         repairedSmartphones = new ArrayList<>();
19.         officialStores = new ArrayList<>();
20.     }
21.
22.     public String getCenterName() {
23.         return centerName;
24.     }
25.
26.     public void setCenterName(String centerName) {
27.         this.centerName = centerName;
28.     }
29.
30.     public List<Smartphone> getRepairedSmartphones() {
31.         return repairedSmartphones;
32.     }
33.
34.     public void pushOfficialStore(PhoneStore store) {
35.         if (store == null) {
36.             throw new IllegalArgumentException("Магазин не может быть null");
37.         }
38.
39.         officialStores.add(store);
40.     }
41.
42.     /**
43.      * Метод для регистрации ремонта
44.      *
45.      * @param smartphone
46.      */
47.     public boolean registerRepair(Smartphone smartphone) {
48.         if (officialStores.stream().anyMatch(x -> x.getSelledSmartphones().contains(smartphone))) {
49.             repairedSmartphones.add(smartphone);
50.             System.out
51.                 .println("Смартфон " + smartphone.getModelName() + " зарегистрирован для ремонта
52. в " + centerName);
53.             return true;
54.         }
55.         return false;
56.     }
57.
58.     /**
59.      * Метод для завершения ремонта
60.      *
61.      * @param smartphone
62.      */
63.     public boolean completeRepair(Smartphone smartphone) throws IllegalArgumentException {
64.         if (!repairedSmartphones.contains(smartphone)) {
65.             throw new IllegalArgumentException("Устройство не принадлежит данному СЦ");
66.         }
67.
68.         repairedSmartphones.remove(smartphone);
69.         System.out.println("Смартфон " + smartphone.getModelName() + " был успешно отремонтирован.");
70.         return true;
71.     }
72. }
```

### 3. Разработка тестов

#### а. Модульные тесты

##### Файл AndroidSmartphoneTest

```
1. package ru.miet.CourceTesting.Lr4;
2.
3. import static org.junit.jupiter.api.Assertions.assertEquals;
4. import static org.junit.jupiter.api.Assertions.assertTrue;
5.
6. import org.junit.jupiter.api.BeforeEach;
7. import org.junit.jupiter.api.Test;
8.
9. /**
10.  * Модульный тест андроид-устройств
11.  */
12. class AndroidSmartphoneTest {
13.     private AndroidSmartphone androidSmartphone;
14.
15.     @BeforeEach
16.     public void setUp() {
17.         androidSmartphone = new AndroidSmartphone("Samsung", "Galaxy S21", 6.2, 0.7, "11", true);
18.     }
19.
20.     @Test
21.     public void testCreateAndroidSmartphone() {
22.         assertEquals("Samsung", androidSmartphone.getVendor());
23.         assertEquals("Galaxy S21", androidSmartphone.getModelName());
24.         assertEquals(6.2, androidSmartphone.getDisplaySize());
25.         assertEquals("Android", androidSmartphone.getOs());
26.         assertEquals(0.7, androidSmartphone.getThickness());
27.         assertEquals("11", androidSmartphone.getAndroidVersion());
28.         assertTrue(androidSmartphone.isHasNFC());
29.     }
30.
31.     @Test
32.     public void testEnableNFC() {
33.         androidSmartphone.enableNFC();
34.         assertTrue(androidSmartphone.getStateNFC());
35.     }
36. }
```

##### Файл IOSSmartphoneTest

```
1. package ru.miet.CourceTesting.Lr4;
2.
3. import static org.junit.jupiter.api.Assertions.assertEquals;
4. import static org.junit.jupiter.api.Assertions.assertTrue;
5.
6. import org.junit.jupiter.api.BeforeEach;
7. import org.junit.jupiter.api.Test;
8.
9. /**
10.  * Модульный тест IOS-устройств
11.  */
12. class IOSSmartphoneTest {
13.     private IOSSmartphone iosSmartphone;
14.
15.     @BeforeEach
16.     public void setUp() {
17.         iosSmartphone = new IOSSmartphone("Apple", "iPhone 13", 6.1, 0.7, "15", true);
18.     }
19.
20.     @Test
21.     public void testCreateIOSSmartphone() {
22.         assertEquals("Apple", iosSmartphone.getVendor());
23.         assertEquals("iPhone 13", iosSmartphone.getModelName());
24.         assertEquals(6.1, iosSmartphone.getDisplaySize());
25.         assertEquals("iOS", iosSmartphone.getOs());
26.         assertEquals(0.7, iosSmartphone.getThickness());
27.         assertEquals("15", iosSmartphone.getIosVersion());
28.         assertTrue(iosSmartphone.isHasFaceID());
29.     }
30.
31.     @Test
32.     public void testEnableFaceID() {
33.         iosSmartphone.enableFaceID();
34.         assertTrue(iosSmartphone.getStateFaceID());
35.     }
36. }
```

## Файл PhoneStoreTest

```
1. package ru.miet.CourceTesting.Lr4;
2.
3. import static org.junit.Assert.assertThrows;
4. import static org.junit.jupiter.api.Assertions.assertEquals;
5.
6. import org.junit.jupiter.api.BeforeEach;
7. import org.junit.jupiter.api.Test;
8.
9. /**
10.  * Модульный тест магазина
11.  */
12. class PhoneStoreTest {
13.     private PhoneStore store;
14.
15.     @BeforeEach
16.     public void setUp() {
17.         store = new PhoneStore("Тестовый Магазин");
18.     }
19.
20.     @Test
21.     public void testAddSmartphone() {
22.         var smartphone = new AndroidSmartphone("Samsung", "Galaxy S21", 6.2, 0.7, "11", true);
23.         store.addSmartphone(smartphone);
24.         assertEquals(1, store.getSmartphones().size());
25.         assertEquals("Galaxy S21", store.getSmartphones().get(0).getModelName());
26.     }
27.
28.     @Test
29.     public void testAddNullSmartphone() {
30.         assertThrows(IllegalArgumentException.class, () -> {
31.             store.addSmartphone(null);
32.         });
33.         assertEquals(0, store.getSmartphones().size());
34.     }
35.
36.     @Test
37.     public void testInitStoreName() {
38.         assertEquals("Тестовый Магазин", store.getStoreName());
39.     }
40.
41.     @Test
42.     public void testRenameStoreName() {
43.         store.setStoreName("New name");
44.         assertEquals("New name", store.getStoreName());
45.     }
46.
47.     @Test
48.     public void testSellSmartphone() {
49.         var smartphone = new AndroidSmartphone("Samsung", "Galaxy S21", 6.2, 0.7, "11", true);
50.         store.addSmartphone(smartphone);
51.         store.sellSmartphone(smartphone);
52.         assertEquals(0, store.getSmartphones().size());
53.         assertEquals(1, store.getSelledSmartphones().size());
54.     }
55.
56.     @Test
57.     public void testSellFakeSmartphone() {
58.         var smartphone = new AndroidSmartphone("Samsung", "Galaxy S21", 6.2, 0.7, "11", true);
59.         var smartphone1 = new AndroidSmartphone("Samsung", "Galaxy S433", 10.2, 0.1, "110", true);
60.         store.addSmartphone(smartphone);
61.         assertThrows(IllegalArgumentException.class, () -> {
62.             store.sellSmartphone(smartphone1);
63.         });
64.         assertEquals(1, store.getSmartphones().size());
65.         assertEquals(0, store.getSelledSmartphones().size());
66.     }
67. }
```

## Файл ServiceCenterTest

```
1. package ru.miet.CourceTesting.Lr4;
2.
3. import static org.junit.jupiter.api.Assertions.assertEquals;
4.
5. import org.junit.jupiter.api.BeforeEach;
6. import org.junit.jupiter.api.Test;
7.
8. /**
9.  * Модульный тест сервисного центра
10. */
11. class ServiceCenterTest {
12.     private ServiceCenter serviceCenter;
13.
14.     @BeforeEach
15.     public void setUp() {
16.         serviceCenter = new ServiceCenter("Сервисный Центр");
17.     }
18.
19.     @Test
20.     public void testRegisterAndCompleteRepair() {
21.         var smartphone = new IOSSmartphone("Apple", "iPhone 13", 6.1, 0.7, "15", true);
22.         serviceCenter.registerRepair(smartphone);
23.         assertEquals(1, serviceCenter.getRepairedSmartphones().size());
24.         serviceCenter.completeRepair(smartphone);
25.         assertEquals(0, serviceCenter.getRepairedSmartphones().size());
26.     }
27. }
```



## б. Интеграционные тесты «сверху вниз»

### Файл FullIntergationTest

```
1. package ru.miet.CourseTesting.Lr4;
2.
3. import static org.junit.jupiter.api.Assertions.assertFalse;
4. import static org.junit.jupiter.api.Assertions.assertThrows;
5. import static org.junit.jupiter.api.Assertions.assertTrue;
6.
7. import org.junit.jupiter.api.BeforeEach;
8. import org.junit.jupiter.api.Test;
9.
10. /**
11.  * Интеграционный тест магазина и сервисного центра
12.  */
13. class FullIntergationTest {
14.
15.     private PhoneStore store;
16.     private ServiceCenter serviceCenter;
17.
18.     @BeforeEach
19.     public void setUp() {
20.         serviceCenter = new ServiceCenter("Сервисный Центр");
21.         store = new PhoneStore("Тестовый Магазин");
22.         store.pushServiceCenter(serviceCenter);
23.         serviceCenter.pushOfficialStore(store);
24.     }
25.
26.     @Test
27.     public void defaultTestSellAndRestoreSmartphones() {
28.         var smartphone1 = new AndroidSmartphone("Samsung", "Galaxy S21", 6.2, 0.7, "11", true);
29.         var smartphone2 = new IOSSmartphone("Apple", "iPhone 13", 6.1, 0.7, "15", true);
30.
31.         // Отправка устройств на прилавок магазина
32.         store.addSmartphone(smartphone1);
33.         store.addSmartphone(smartphone2);
34.
35.         // Продажа одного из устройств
36.         store.sellSmartphone(smartphone1);
37.
38.         var services = store.getCertifiedServiceCenter();
39.         // Отправка в сервисный центр устройства по гарантии
40.         assertTrue(services.stream().anyMatch(x -> x.registerRepair(smartphone1)));
41.
42.         // Ремонт устройства
43.         assertTrue(serviceCenter.completeRepair(smartphone1));
44.     }
45.
46.     @Test
47.     public void TestSellAndRestoreFakeSmartphones() {
48.         var smartphone1 = new AndroidSmartphone("Samsung", "Galaxy S21", 6.2, 0.7, "11", true);
49.         var smartphone2 = new IOSSmartphone("Apple", "iPhone 13", 6.1, 0.7, "15", true);
50.         var fakeStore = new PhoneStore("Фейковый магазин");
51.
52.         // Магазин пытается скопировать сертификаты на ремонт в официальных ЦС
53.         store.pushServiceCenter(serviceCenter);
54.
55.         // Отправка устройств на прилавок магазина
56.         fakeStore.addSmartphone(smartphone1);
57.         fakeStore.addSmartphone(smartphone2);
58.
59.         // Продажа одного из устройств
60.         fakeStore.sellSmartphone(smartphone1);
61.
62.         var services = store.getCertifiedServiceCenter();
63.         // Отправка в сервисный центр устройства по гарантии
64.         assertFalse(services.stream().anyMatch(x -> x.registerRepair(smartphone1)));
65.
66.         // Ремонт устройства
67.         assertThrows(IllegalArgumentException.class, () -> {
68.             serviceCenter.completeRepair(smartphone1);
69.         });
70.     }
71. }
```

## Файл IntergationTestPhoneStore

```
1. package ru.miet.CourseTesting.Lr4;
2.
3. import static org.junit.jupiter.api.Assertions.assertEquals;
4.
5. import org.junit.jupiter.api.BeforeEach;
6. import org.junit.jupiter.api.Test;
7.
8. import ru.miet.CourseTesting.Lr3.Smartphone;
9.
10. /**
11.  * Интеграционный тест магазина с Mock сервисным центром и реальными телефонами
12.  */
13. class IntergationTestPhoneStore {
14.     private PhoneStore store;
15.
16.     @BeforeEach
17.     public void setUp() {
18.         store = new PhoneStore("Тестовый Магазин");
19.     }
20.
21.     @Test
22.     public void testAddAndRetrieveSmartphones() {
23.         Smartphone smartphone1 = new AndroidSmartphone("Samsung", "Galaxy S21", 6.2, 0.7, "11", true);
24.         Smartphone smartphone2 = new IOSSmartphone("Apple", "iPhone 13", 6.1, 0.7, "15", true);
25.
26.         store.addSmartphone(smartphone1);
27.         store.addSmartphone(smartphone2);
28.
29.         assertEquals(2, store.getSmartphones().size());
30.         assertEquals("Samsung", store.getSmartphones().get(0).getVendor());
31.         assertEquals("Apple", store.getSmartphones().get(1).getVendor());
32.     }
33.
34.     @Test
35.     public void testServiceCenterIntegration() {
36.         store.pushServiceCenter(new ServiceCenterMock("Сервисный Центр")); // Создаем заглушку
37.         assertEquals(1, store.getCertifiedServiceCenter().size());
38.     }
39. }
40.
41. class ServiceCenterMock extends ServiceCenter {
42.     public ServiceCenterMock(String centerName) {
43.         super(centerName);
44.         // TODO Auto-generated constructor stub
45.     }
46. }
```

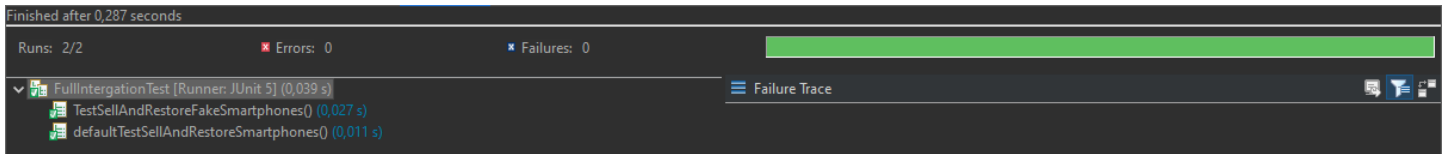
## Файл IntergationTestServiceCenter

```
1. package ru.miet.CourceTesting.Lr4;
2.
3. import static org.junit.jupiter.api.Assertions.assertEquals;
4. import static org.junit.jupiter.api.Assertions.assertTrue;
5.
6. import org.junit.jupiter.api.BeforeEach;
7. import org.junit.jupiter.api.Test;
8.
9. import ru.miet.CourceTesting.Lr3.Smartphone;
10.
11. /**
12.  * Интеграционный тест сервисного центра с Mock магазинами и реальными
13.  * телефонами
14.  */
15. class IntergationTestServiceCenter {
16.     private ServiceCenter serviceCenter;
17.     private Smartphone smartphone;
18.
19.     @BeforeEach
20.     public void setUp() {
21.         serviceCenter = new ServiceCenter("Сервисный Центр");
22.         smartphone = new AndroidSmartphone("Samsung", "Galaxy S21", 6.2, 0.7, "11", true);
23.
24.         var mockStore = new PhoneStoreMock("Mock-store", smartphone);
25.         serviceCenter.pushOfficialStore(mockStore);
26.     }
27.
28.     @Test
29.     public void testRegisterAndCompleteRepair() {
30.         assertTrue(serviceCenter.registerRepair(smartphone));
31.         assertEquals(1, serviceCenter.getRepairedSmartphones().size());
32.
33.         assertTrue(serviceCenter.completeRepair(smartphone));
34.         assertEquals(0, serviceCenter.getRepairedSmartphones().size());
35.     }
36.
37. }
38.
39. class PhoneStoreMock extends PhoneStore {
40.     public PhoneStoreMock(String storeName, Smartphone solded) {
41.         super(storeName);
42.         soldedSmartphones.add(selled);
43.     }
44. }
```

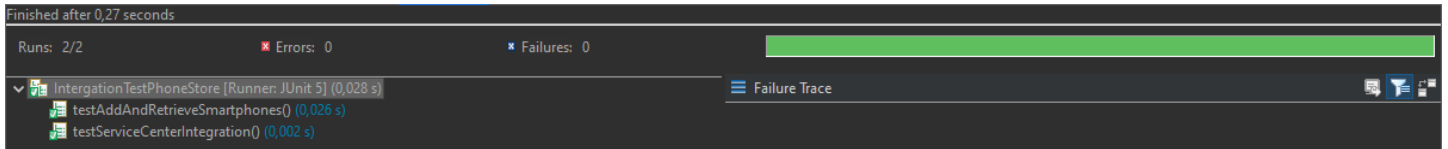
#### 4. Результаты прогона

Для выполнения инкрементального интеграционного тестирования «сверху вниз» выполним разработанные тесты в следующей последовательности:

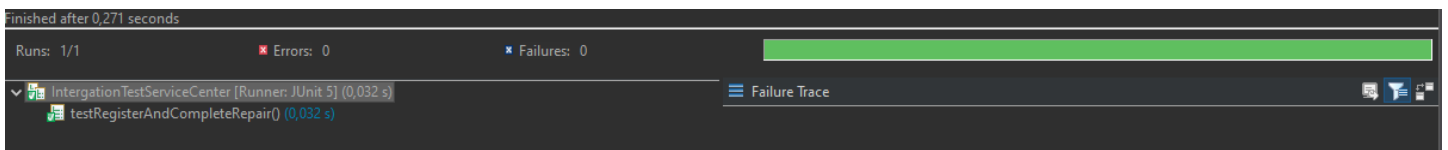
**FullIntegrationTest -> IntergrationTestPhoneStore -> IntegrationTestServiceCenter**



**Рис.2:** 1 Этап - прогон тестов FullIntegrationTest

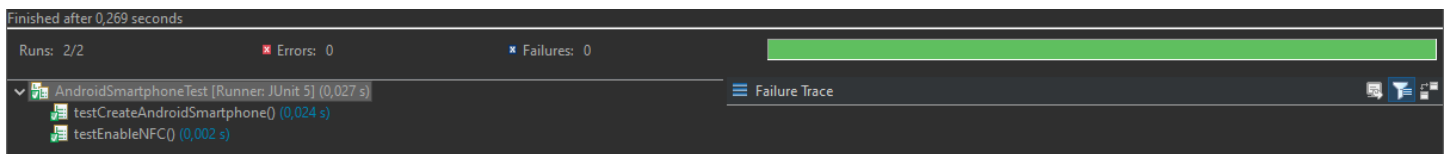


**Рис.3:** 2 Этап - прогон тестов IntergrationTestPhoneStore

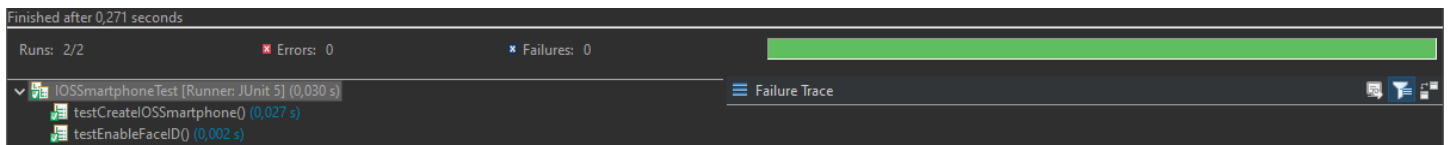


**Рис.4:** 3 Этап - прогон тестов IntegrationTestServiceCenter

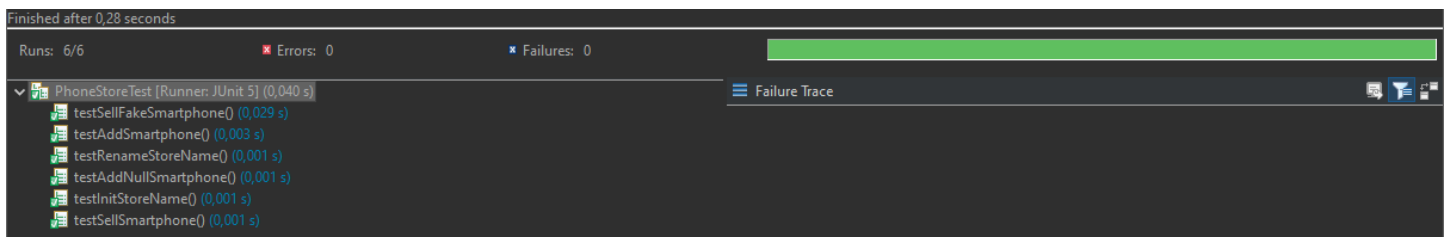
Результаты прогона модульных тестов:



**Рис.5:** прогон тестов AndroidSmartphoneTest



**Рис.6:** прогон тестов IOSSmartphoneTest



**Рис.7:** прогон тестов PhoneStoreTest