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**

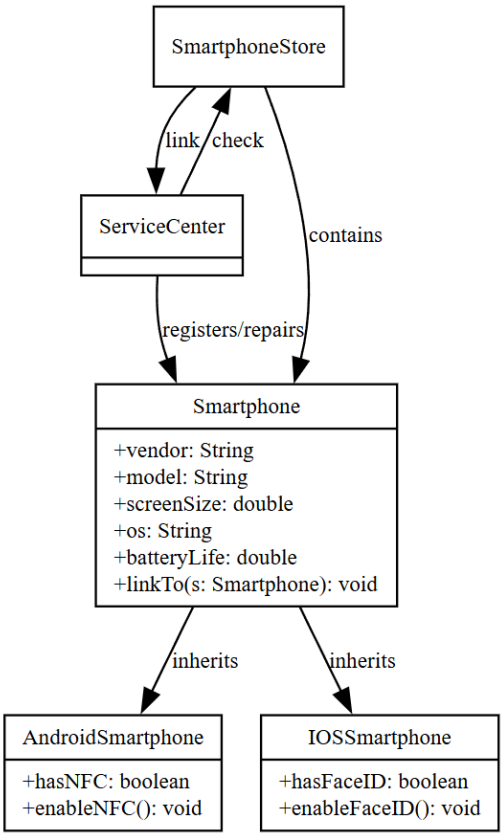

Рис.1: Диаграмма иерархии классов

2. Реализация классов

Класс AndroidSmartphone

```java
package ru.miet.CourceTesting.Lr4;

import ru.miet.CourceTesting.Lr3.Smartphone;

/**
 *
 */
public class AndroidSmartphone extends Smartphone {
    private String androidVersion;
    private boolean hasNFC;
    private boolean isNFCEnable;

    public AndroidSmartphone(String vendor, String modelName, double displaySize, double thickness,
            String androidVersion, boolean hasNFC) {
        super(vendor, modelName, displaySize, "Android", thickness);
        setAndroidVersion(androidVersion);
        setHasNFC(hasNFC);
        updateStateNFC(false);
    }

    public String getAndroidVersion() {
        return androidVersion;
    }

    public void setAndroidVersion(String androidVersion) {
        this.androidVersion = androidVersion;
    }

    public boolean isHasNFC() {
        return hasNFC;
    }

    public void setHasNFC(boolean hasNFC) {
        this.hasNFC = hasNFC;
    }

    private void updateStateNFC(boolean state) {
        isNFCEnable = state;
    }

    public boolean getStateNFC() {
        return isNFCEnable;
    }

    public void enableNFC() {
        if (hasNFC) {
            System.out.println("NFC включен на " + getModelName());
            updateStateNFC(!isNFCEnable);
        } else {
            System.out.println("Устройство не поддерживает NFC.");
        }
    }
}
```

Класс IOSSmartphone

```java
package ru.miet.CourceTesting.Lr4;

import ru.miet.CourceTesting.Lr3.Smartphone;

/**
 *
 */
public class IOSSmartphone extends Smartphone {
    private String iosVersion;
    private boolean hasFaceID;
    private boolean isFaceIDEnable;

    public IOSSmartphone(String vendor, String modelName, double displaySize, double thickness, String
iosVersion,
            boolean hasFaceID) {
        super(vendor, modelName, displaySize, "iOS", thickness);
        setIosVersion(iosVersion);
        setHasFaceID(hasFaceID);
        updateStateFaceID(false);
    }

    public String getIosVersion() {
        return iosVersion;
    }

    public void setIosVersion(String iosVersion) {
        this.iosVersion = iosVersion;
    }

    public boolean isHasFaceID() {
        return hasFaceID;
    }

    public void setHasFaceID(boolean hasFaceID) {
        this.hasFaceID = hasFaceID;
    }

    private void updateStateFaceID(boolean state) {
        isFaceIDEnable = state;
    }

    public boolean getStateFaceID() {
        return isFaceIDEnable;
    }

    public void enableFaceID() {
        if (hasFaceID) {
            System.out.println("Face ID включен на " + getModelName());
            updateStateFaceID(!isFaceIDEnable);
        } else {
            System.out.println("Устройство не поддерживает Face ID.");
        }
    }
}
```

Класс PhoneStore

```java
package ru.miet.CourceTesting.Lr4;

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

import ru.miet.CourceTesting.Lr3.Smartphone;

/**
 *
 */
public class PhoneStore {
    protected String storeName;
    protected List<Smartphone> smartphones;
    protected List<Smartphone> selledSmartphones;
    protected List<ServiceCenter> certifiedServiceCenter;

    public PhoneStore(String storeName) {
        this.storeName = storeName;
        smartphones = new ArrayList<>();
        selledSmartphones = new ArrayList<>();
        certifiedServiceCenter = new ArrayList<>();
    }

    public String getStoreName() {
        return storeName;
    }

    public void setStoreName(String storeName) {
        if (storeName == null || storeName.trim().isEmpty()) {
            throw new IllegalArgumentException("Название продавца не может быть пустым");
        }
        this.storeName = storeName;
    }

    public List<Smartphone> getSmartphones() {
        return smartphones;
    }

    public List<Smartphone> getSelledSmartphones() {
        return selledSmartphones;
    }

    public void pushServiceCenter(ServiceCenter sc) {
        if (sc == null) {
            throw new IllegalArgumentException("Сервисный центр не может быть null");
        }
        certifiedServiceCenter.add(sc);
    }

    public List<ServiceCenter> getCertifiedServiceCenter() {
        return certifiedServiceCenter;
    }

    public void addSmartphone(Smartphone smartphone) {
        if (smartphone == null) {
            throw new IllegalArgumentException("Смартфон не может быть null");
        }
        smartphones.add(smartphone);
    }

    public void sellSmartphone(Smartphone smartphone) {
        if (smartphone == null) {
            throw new IllegalArgumentException("Смартфон не может быть null");
        }
        if (!smartphones.contains(smartphone)) {
            throw new IllegalArgumentException("Смартфон не принадлежит магазину");
        }
        smartphones.remove(smartphone);
        selledSmartphones.add(smartphone);
    }
}
```

Класс ServiceCenter

```java
package ru.miet.CourceTesting.Lr4;

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

import ru.miet.CourceTesting.Lr3.Smartphone;

/**
 *
 */
public class ServiceCenter {
    private String centerName;
    private List<Smartphone> repairedSmartphones;
    private List<PhoneStore> officialStores;

    public ServiceCenter(String centerName) {
        this.centerName = centerName;
        repairedSmartphones = new ArrayList<>();
        officialStores = new ArrayList<>();
    }

    public String getCenterName() {
        return centerName;
    }

    public void setCenterName(String centerName) {
        this.centerName = centerName;
    }

    public List<Smartphone> getRepairedSmartphones() {
        return repairedSmartphones;
    }

    public void pushOfficialStore(PhoneStore store) {
        if (store == null) {
            throw new IllegalArgumentException("Магазин не может быть null");
        }

        officialStores.add(store);
    }

    /**
     * Метод для регистрации ремонта
     *
     * @param smartphone
     */
    public boolean registerRepair(Smartphone smartphone) {
        if (officialStores.stream().anyMatch(x -> x.getSelledSmartphones().contains(smartphone))) {
            repairedSmartphones.add(smartphone);
            System.out
                    .println("Смартфон " + smartphone.getModelName() + " зарегистрирован для ремонта
в " + centerName);
            return true;
        }
        return false;
    }

    /**
     * Метод для завершения ремонта
     *
     * @param smartphone
     */
    public boolean completeRepair(Smartphone smartphone) throws IllegalArgumentException {
        if (!repairedSmartphones.contains(smartphone)) {
            throw new IllegalArgumentException("Устройство не принадлежит данному СЦ");
        }

        repairedSmartphones.remove(smartphone);
        System.out.println("Смартфон " + smartphone.getModelName() + " был успешно отремонтирован.");
        return true;
    }
}
```

3. Разработка тестов

a. Модульные тесты

Файл AndroidSmartphoneTest

```java
package ru.miet.CourceTesting.Lr4;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertTrue;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

/**
 * Модульный тест андройд-устройств
 */
class AndroidSmartphoneTest {
    private AndroidSmartphone androidSmartphone;

    @BeforeEach
    public void setUp() {
    androidSmartphone = new AndroidSmartphone("Samsung", "Galaxy S21", 6.2, 0.7, "11", true);
    }
    @Test
    public void testCreateAndroidSmartphone() {
    assertEquals("Samsung", androidSmartphone.getVendor());
    assertEquals("Galaxy S21", androidSmartphone.getModelName());
    assertEquals(6.2, androidSmartphone.getDisplaySize());
    assertEquals("Android", androidSmartphone.getOs());
    assertEquals(0.7, androidSmartphone.getThickness());
    assertEquals("11", androidSmartphone.getAndroidVersion());
    assertTrue(androidSmartphone.isHasNFC());
    }
    @Test
    public void testEnableNFC() {
    androidSmartphone.enableNFC();
    assertTrue(androidSmartphone.getStateNFC());
    }
}
```

Файл IOSSmartphoneTest

```java
package ru.miet.CourceTesting.Lr4;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertTrue;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

/**
 * Модульный тест IOS-устройств
 */
class IOSSmartphoneTest {
    private IOSSmartphone iosSmartphone;

    @BeforeEach
    public void setUp() {
    iosSmartphone = new IOSSmartphone("Apple", "iPhone 13", 6.1, 0.7, "15", true);
    }
    @Test
    public void testCreateIOSSmartphone() {
    assertEquals("Apple", iosSmartphone.getVendor());
    assertEquals("iPhone 13", iosSmartphone.getModelName());
    assertEquals(6.1, iosSmartphone.getDisplaySize());
    assertEquals("iOS", iosSmartphone.getOs());
    assertEquals(0.7, iosSmartphone.getThickness());
    assertEquals("15", iosSmartphone.getIosVersion());
    assertTrue(iosSmartphone.isHasFaceID());
    }
    @Test
    public void testEnableFaceID() {
    iosSmartphone.enableFaceID();
    assertTrue(iosSmartphone.getStateFaceID());
    }
}
```

Файл PhoneStoreTest

```java
package ru.miet.CourceTesting.Lr4;

import static org.junit.Assert.assertThrows;
import static org.junit.jupiter.api.Assertions.assertEquals;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

/**
 * Модульный тест магазина
 */
class PhoneStoreTest {
    private PhoneStore store;

    @BeforeEach
    public void setUp() {
    store = new PhoneStore("Тестовый Магазин");
    }

    @Test
    public void testAddSmartphone() {
    var smartphone = new AndroidSmartphone("Samsung", "Galaxy S21", 6.2, 0.7, "11", true);
    store.addSmartphone(smartphone);
    assertEquals(1, store.getSmartphones().size());
    assertEquals("Galaxy S21", store.getSmartphones().get(0).getModelName());
    }

    @Test
    public void testAddNullSmartphone() {
    assertThrows(IllegalArgumentException.class, () -> {
        store.addSmartphone(null);
    });
    assertEquals(0, store.getSmartphones().size());
    }

    @Test
    public void testInitStoreName() {
    assertEquals("Тестовый Магазин", store.getStoreName());
    }

    @Test
    public void testRenameStoreName() {
    store.setStoreName("New name");
    assertEquals("New name", store.getStoreName());
    }

    @Test
    public void testSellSmartphone() {
    var smartphone = new AndroidSmartphone("Samsung", "Galaxy S21", 6.2, 0.7, "11", true);
    store.addSmartphone(smartphone);
    store.sellSmartphone(smartphone);
    assertEquals(0, store.getSmartphones().size());
    assertEquals(1, store.getSelledSmartphones().size());
    }

    @Test
    public void testSellFakeSmartphone() {
    var smartphone = new AndroidSmartphone("Samsung", "Galaxy S21", 6.2, 0.7, "11", true);
    var smartphone1 = new AndroidSmartphone("Samsung", "Galaxy S433", 10.2, 0.1, "110", true);
    store.addSmartphone(smartphone);
    assertThrows(IllegalArgumentException.class, () -> {
        store.sellSmartphone(smartphone1);
    });
    assertEquals(1, store.getSmartphones().size());
    assertEquals(0, store.getSelledSmartphones().size());
    }
}
```

Файл ServiceCenterTest

```java
package ru.miet.CourceTesting.Lr4;

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

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

/**
 * Модульный тест сервисного центра
 */
class ServiceCenterTest {
    private ServiceCenter serviceCenter;

    @BeforeEach
    public void setUp() {
    serviceCenter = new ServiceCenter("Сервисный Центр");
    }

    @Test
    public void testRegisterAndCompleteRepair() {
    var smartphone = new IOSSmartphone("Apple", "iPhone 13", 6.1, 0.7, "15", true);
    serviceCenter.registerRepair(smartphone);
    assertEquals(1, serviceCenter.getRepairedSmartphones().size());
    serviceCenter.completeRepair(smartphone);
    assertEquals(0, serviceCenter.getRepairedSmartphones().size());
    }
}
```

b. Интеграционные тесты «сверху вниз»

Файл FullIntergationTest

```java
package ru.miet.CourceTesting.Lr4;

import static org.junit.jupiter.api.Assertions.assertFalse;
import static org.junit.jupiter.api.Assertions.assertThrows;
import static org.junit.jupiter.api.Assertions.assertTrue;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

/**
 * Интеграционный тест магазина и сервисного центра
 */
class FullIntergationTest {

    private PhoneStore store;
    private ServiceCenter serviceCenter;

    @BeforeEach
    public void setUp() {
    serviceCenter = new ServiceCenter("Сервисный Центр");
    store = new PhoneStore("Тестовый Магазин");
    store.pushServiceCenter(serviceCenter);
    serviceCenter.pushOfficialStore(store);
    }

    @Test
    public void defaultTestSellAndRestoreSmartphones() {
    var smartphone1 = new AndroidSmartphone("Samsung", "Galaxy S21", 6.2, 0.7, "11", true);
    var smartphone2 = new IOSSmartphone("Apple", "iPhone 13", 6.1, 0.7, "15", true);

    // Отправка устройств на прилавок магазина
    store.addSmartphone(smartphone1);
    store.addSmartphone(smartphone2);

    // Продажа одного из устройств
    store.sellSmartphone(smartphone1);

    var services = store.getCertifiedServiceCenter();
    // Отправка в сервисный центр устройства по гарантии
    assertTrue(services.stream().anyMatch(x -> x.registerRepair(smartphone1)));

    // Ремонт устройства
    assertTrue(serviceCenter.completeRepair(smartphone1));
    }

    @Test
    public void TestSellAndRestoreFakeSmartphones() {
    var smartphone1 = new AndroidSmartphone("Samsung", "Galaxy S21", 6.2, 0.7, "11", true);
    var smartphone2 = new IOSSmartphone("Apple", "iPhone 13", 6.1, 0.7, "15", true);
    var fakeStore = new PhoneStore("Фейковый магазин");

    // Магазин пытается скопировать сертификаты на ремонт в официанльных СЦ
    store.pushServiceCenter(serviceCenter);

    // Отправка устройств на прилавок магазина
    fakeStore.addSmartphone(smartphone1);
    fakeStore.addSmartphone(smartphone2);

    // Продажа одного из устройств
    fakeStore.sellSmartphone(smartphone1);

    var services = store.getCertifiedServiceCenter();
    // Отправка в сервисный центр устройства по гарантии
    assertFalse(services.stream().anyMatch(x -> x.registerRepair(smartphone1)));

    // Ремонт устройства
    assertThrows(IllegalArgumentException.class, () -> {
        serviceCenter.completeRepair(smartphone1);
    });
    }
}
```

Файл IntergationTestPhoneStore

```java
package ru.miet.CourceTesting.Lr4;

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

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import ru.miet.CourceTesting.Lr3.Smartphone;

/**
 * Интеграционный тест магазина с Mock сервисным центром и реальными телефонами
 */
class IntergationTestPhoneStore {
    private PhoneStore store;

    @BeforeEach
    public void setUp() {
    store = new PhoneStore("Тестовый Магазин");
    }

    @Test
    public void testAddAndRetrieveSmartphones() {
    Smartphone smartphone1 = new AndroidSmartphone("Samsung", "Galaxy S21", 6.2, 0.7, "11", true);
    Smartphone smartphone2 = new IOSSmartphone("Apple", "iPhone 13", 6.1, 0.7, "15", true);

    store.addSmartphone(smartphone1);
    store.addSmartphone(smartphone2);

    assertEquals(2, store.getSmartphones().size());
    assertEquals("Samsung", store.getSmartphones().get(0).getVendor());
    assertEquals("Apple", store.getSmartphones().get(1).getVendor());
    }

    @Test
    public void testServiceCenterIntegration() {
    store.pushServiceCenter(new ServiceCenterMock("Сервисный Центр")); // Создаем заглушку
    assertEquals(1, store.getCertifiedServiceCenter().size());
    }
}

class ServiceCenterMock extends ServiceCenter {
    public ServiceCenterMock(String centerName) {
    super(centerName);
    // TODO Auto-generated constructor stub
    }
}
```

Файл IntergationTestServiceCenter

```java
package ru.miet.CourceTesting.Lr4;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertTrue;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import ru.miet.CourceTesting.Lr3.Smartphone;

/**
 * Интеграционный тест сервисного центра с Mock магазинами и реальными
 * телефонами
 */
class IntergationTestServiceCenter {
    private ServiceCenter serviceCenter;
    private Smartphone smartphone;

    @BeforeEach
    public void setUp() {
    serviceCenter = new ServiceCenter("Сервисный Центр");
    smartphone = new AndroidSmartphone("Samsung", "Galaxy S21", 6.2, 0.7, "11", true);

    var mockStore = new PhoneStoreMock("Mock-store", smartphone);
    serviceCenter.pushOfficialStore(mockStore);
    }

    @Test
    public void testRegisterAndCompleteRepair() {
    assertTrue(serviceCenter.registerRepair(smartphone));
    assertEquals(1, serviceCenter.getRepairedSmartphones().size());

    assertTrue(serviceCenter.completeRepair(smartphone));
    assertEquals(0, serviceCenter.getRepairedSmartphones().size());
    }

}

class PhoneStoreMock extends PhoneStore {
    public PhoneStoreMock(String storeName, Smartphone selled) {
    super(storeName);
    selledSmartphones.add(selled);
    }
}
```

4. Результаты прогона

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

FullIntegrationTest -> **IntergationTestPhoneStore** -> **IntegrationTestServiceCenter**

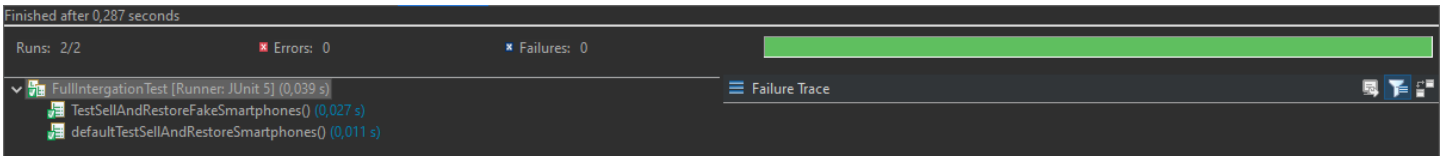

Рис.2: 1 Этап - прогон тестов FullIntegrationTest

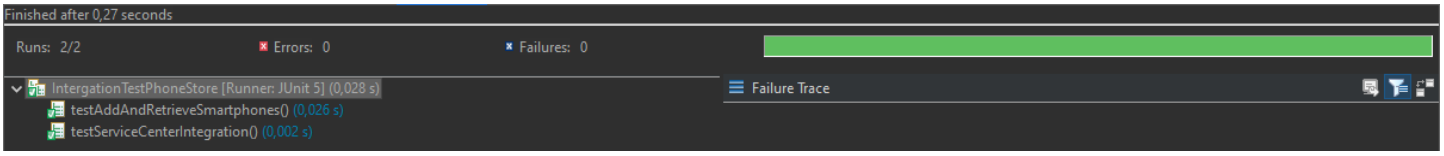

Рис.3: 2 Этап - прогон тестов IntergationTestPhoneStore

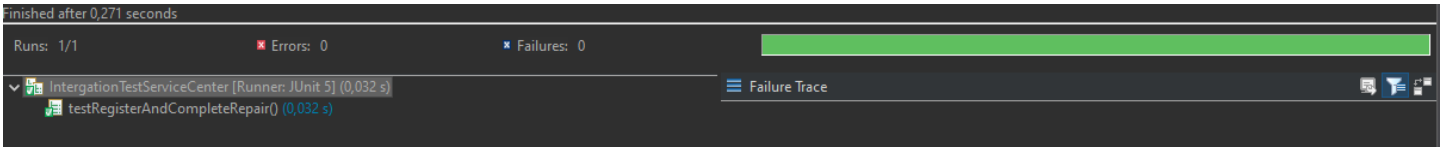

Рис.4: 3 Этап - прогон тестов IntegrationTestServiceCenter

Результаты прогона модульных тестов:

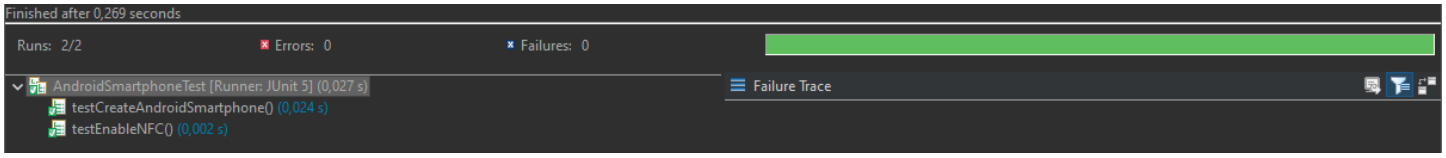

Рис.5: прогон тестов AndroidSmartphoneTest

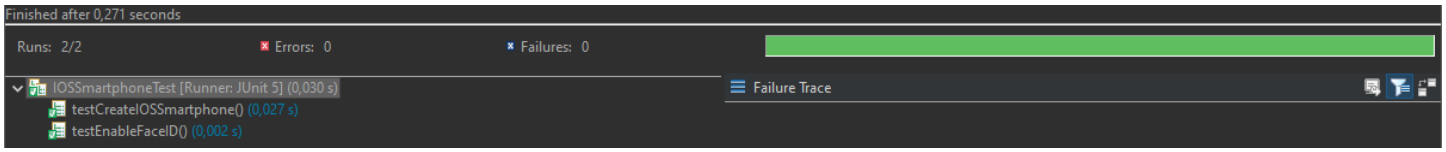

Рис.6: прогон тестов IOSSmartphoneTest

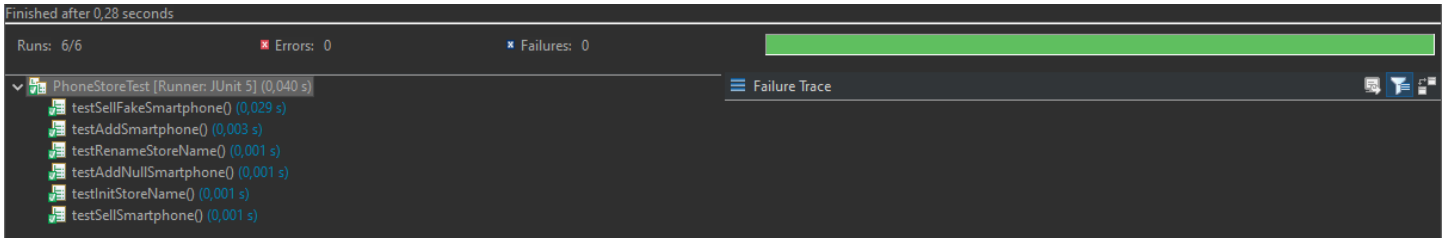

Рис.7: прогон тестов PhoneStoreTest