

Министерство образования и науки Российской Федерации  
ФГБОУ ВО Рыбинский государственный авиационный технический  
университет имени П.А. Соловьева

Факультет радиоэлектроники и информатики  
Кафедра математического и программного обеспечения  
электронных вычислительных средств

## **ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

по дисциплине  
**Тестирование и отладка  
программного обеспечения**

по теме  
**Интеграционное тестирование**

Студенты группы ИПБ-13  
Преподаватель к.т.н., ст. преп.

Болотин Д. И.  
Ивашин А.В.  
Воробьев К. А.

Рыбинск 2016

# Содержание

<b>1. Общее описание тестируемой системы</b>	<b>3</b>
<b>2. Общее описание тестируемых классов</b>	<b>5</b>
2.1. Ограничения для шифруемого/расшифруемого текста . . . . .	6
<b>3. Общее описание тестирующих классов</b>	<b>7</b>
<b>4. Тестирование взаимодействия класса Login</b>	<b>8</b>
4.1. Закрытие программы при неверной авторизации . . . . .	8
4.2. Реакция программы при неверном пароле . . . . .	8
4.3. Реакция программы при не существующем логине . . . . .	9
4.4. Реакция программы при верной авторизации . . . . .	9
4.5. Тестирование перехода к окну регистрации . . . . .	9
<b>5. Тестирование взаимодействия классов     MonoAlphabetCipher и BitReverseCipher</b>	<b>11</b>
<b>6. Тестирование взаимодействия класса MainWindow</b>	<b>13</b>
<b>7. Выводы</b>	<b>15</b>
<b>Приложение 1. Исходные коды тестируемых классов</b>	<b>16</b>
<b>Приложение 2. Исходные коды тестирующих классов</b>	<b>31</b>

# 1. Общее описание тестируемой системы

Прект предназначен для шифрования/дешифрования текстов методами Моноалфавитной замены, Побитовой перестановки. На Рисунке 1 представлена диаграмма классов проекта.

После запуска программы пользователь видит окно авторизации (Рис. 2), с помощью которого он может либо авторизоваться: введя логин (аглийский алфавит) и пароль; зарегистрироваться (Рис. 3): введя информацию о себе и о предпочтительных методах шифрования (обязательными являются поля: логин, пароль и метод шифрования); либо же завершить работы с программой с помощью кнопки "Выход".

После прохождения авторизации пользователю открывается главное окно приложения, в котором ему, из списка методов шифрования, будут доступны лишь те методы, которые были выбраны им на этапе регистрации (Рис. 4).

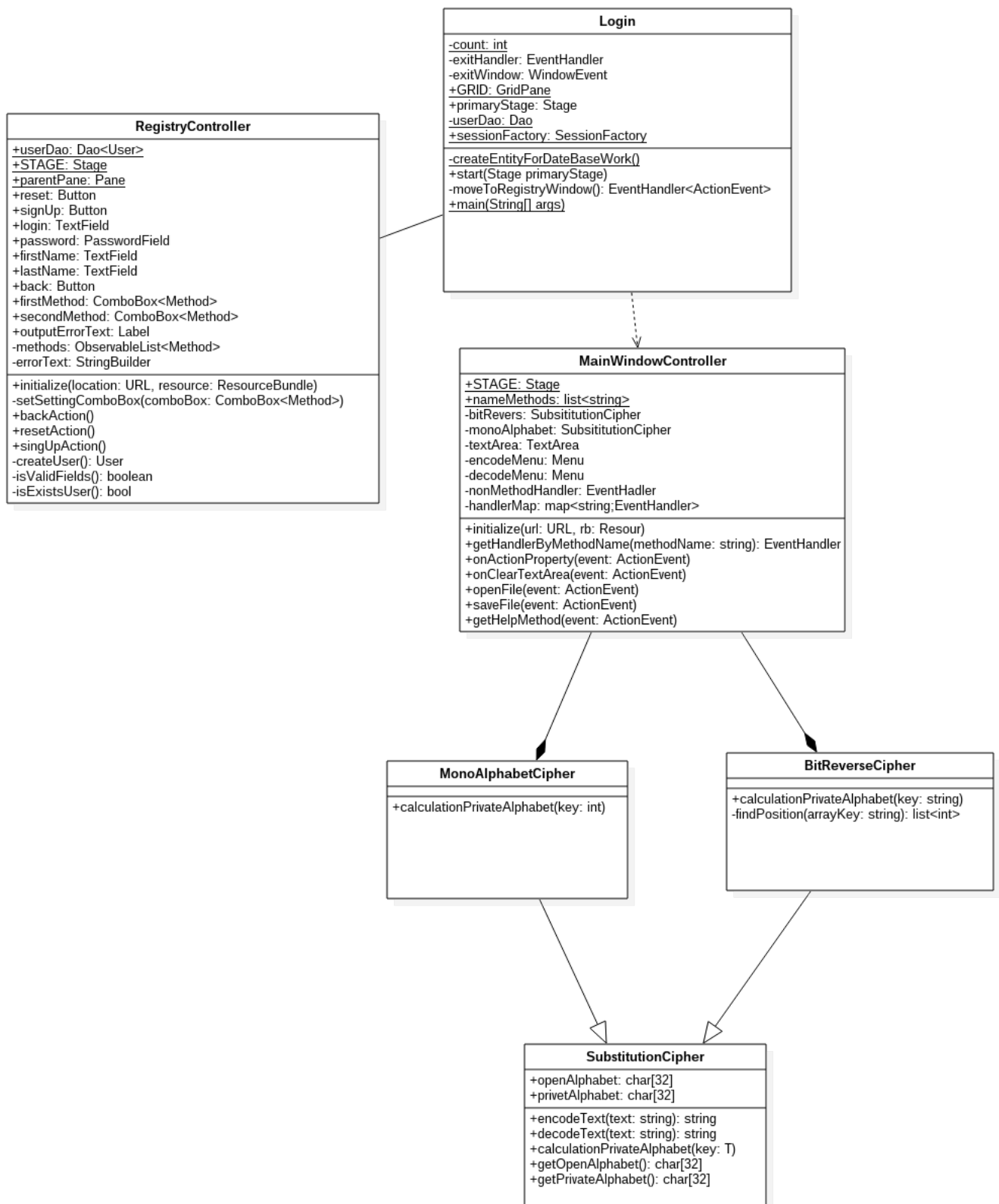


Рис. 1 – Диаграмма классов проекта

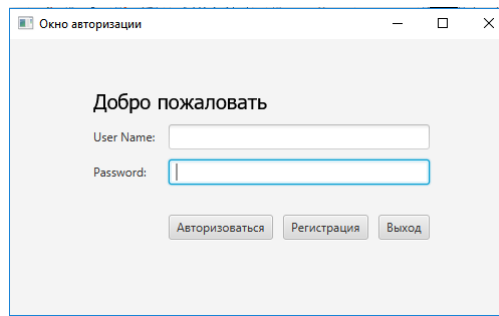


Рис. 2 – Окно авторизации

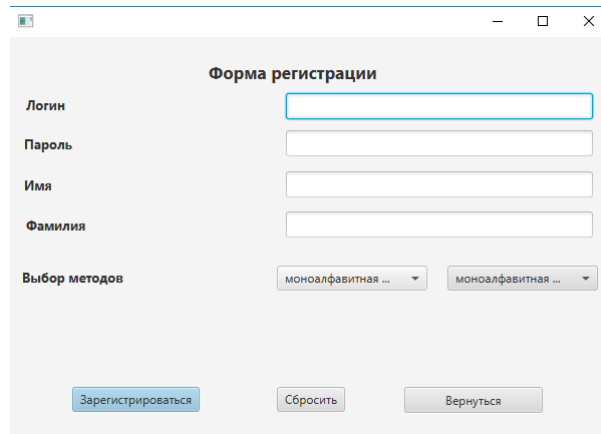


Рис. 3 – Окно регистрации

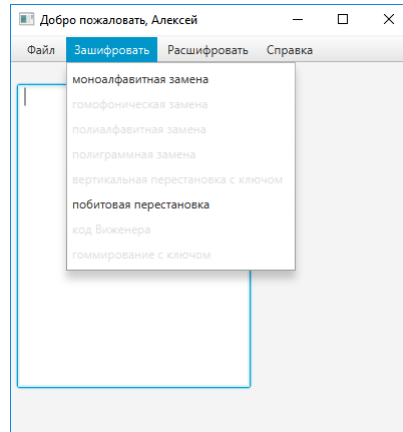


Рис. 4 – Главное окно программы

## 2. Общее описание тестируемых классов

В ходе данного тестирования проверяется только взаимодействие классов программы между собой, взаимодействие с другими системами не проверяется.

Тестированию подвержены классы, взаимодействующие друг с другом непосредственно, а также взаимодействие MonoAlphabetCipher и BitReversCipher посредством комбинации различных способов шифрования/дешифрования. Тестирование добавление пользователя в БД в данной работе не производится.

Для проведения тестирования использована библиотека JUnit, а также TestFX, которая нужна для моделирования взаимодействия пользователя с интерфейсом.

## 2.1. Ограничения для шифруемого/расшифруемого текста

Текст принимаемый методами шифрования состоит из строчных букв русского алфавита, из которого исключены буквы 'ё' и 'й', а так же пробела. Оставшиеся символы не входят в состав открытого алфавита, по этому при их использовании их в тексте пользователь увидит следующее сообщение: "Недопустимый символ. Расшифровка/Шифрование не возможно" (Последнее предложение зависит от выбранного пользователем метода. Пример на рисунке (5).

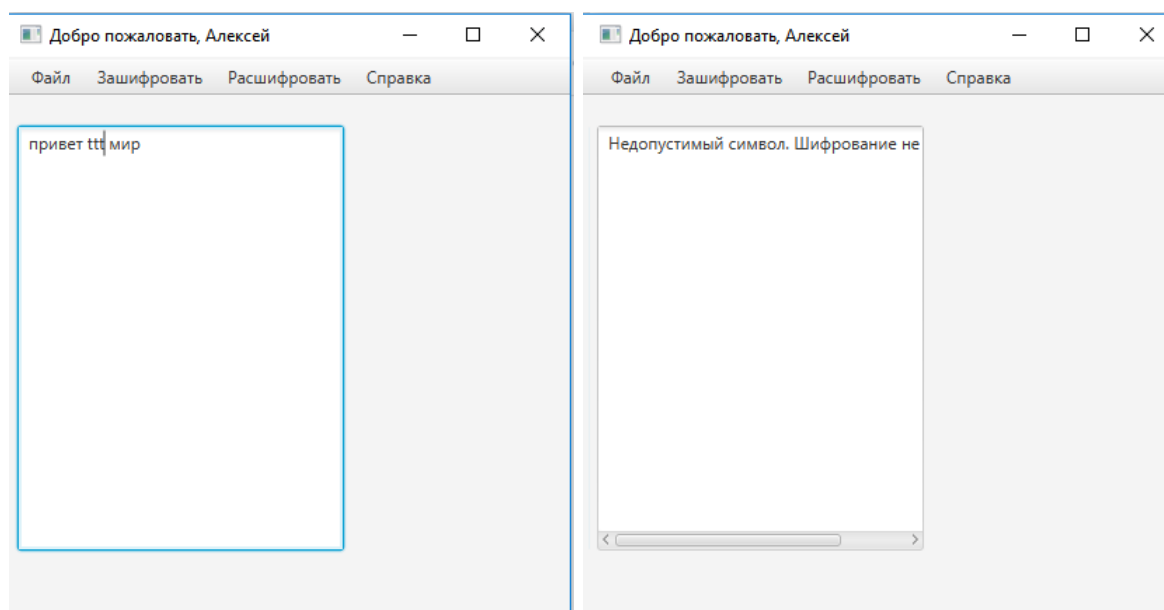


Рис. 5 – Не верный символ в тексте

### 3. Общее описание тестирующих классов

Для проведения интеграционного тестирования была разработана система классов для тестирования пользовательского интерфейса(т.к. почти все классы программы взаимодействуют только через него), а также класс для тестирования взаимодействия методов шифрования (под названием IntegrationCipherTest). Диаграмма классов для тестирования пользовательского Gui представлена на рисунке 6, на ней упущены имена методов и атрибутов.

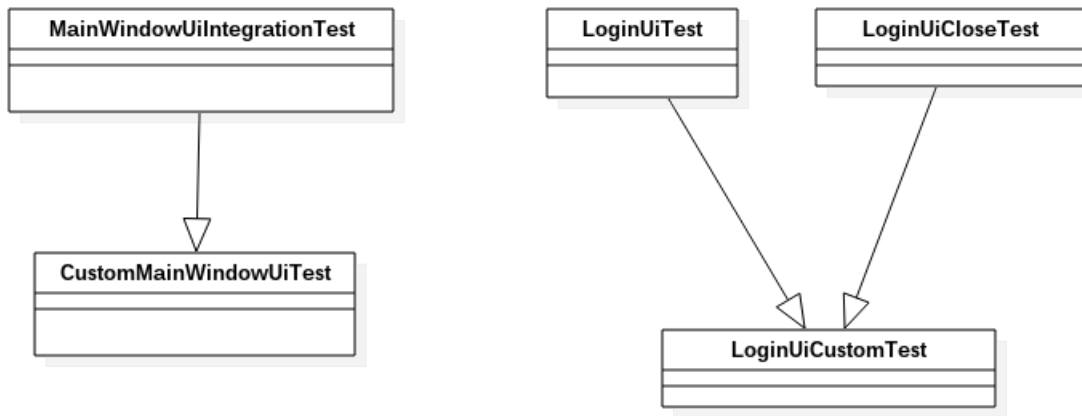


Рис. 6 – Диаграмма тестирующих классов

## 4. Тестирование взаимодействия класса Login

Login - это главный класс приложения, который запускает его и одновременно является контроллером для окна авторизации. Пользователь должен пройти авторизацию или зарегистрироваться. Если после третьей попытки не удалось авторизоваться, то программа закрывается.

### 4.1. Закрытие программы при неверной авторизации

С помощью TestFx моделируется попытка неудачной авторизации: три раза вводим не верны пароль. На каждом шаге проверяем появилось ли в специальной строке сообщение об ошибке: "Не верный пароль выделенное красным цветом. (Листинг 1).

Листинг 1 – Тестирование неверной авторизации

```
1 public class LoginUiCloseTest extends LoginUiCustomTest {
2     @Test
3     public void threeWrongAuthorizationTest () {
4         for (int i = 0; i < 3; i++) {
5             clickOn(userName).write("Aleksey");
6             clickOn(password).write("tttt");
7             clickOn(authorization).sleep(100);
8             verifyThat(resultAuthorization, hasText("Не верный пароль"));
9             assertEquals(resultAuthorization.getFill(), Color.FIREBRICK);
10            userName.clear();
11            password.clear();
12        }
13        assertEquals(stage.isShowing(), false);
14        assertEquals(sessionFactory.isClosed(), true);
15    }
16 }
```

### 4.2. Реакция программы при неверном пароле

При вводе корректного (существующего) логина пользователя, но не корректного пароля для него в специальной строке формы авторизации должна появиться надпись "Неверный пароль выделенное красным цветом. Этот и последующие тесты класса Login описаны в классе LoginUiTest (Листинг ??).

Листинг 2 – Тестирование правильной авторизации

```
1 protected void openRegistryWindowAndTest () {
2     clickOn(registry);
3     assertEquals(gridPane.isDisable(), true);
4     waitUntil("#registryPane", visible());
5 }
6
7 protected void closeRegistryWindowAndTest () {
8     clickOn("#back");
9     assertEquals(gridPane.isDisable(), false);
10 }
11
12 @Test
```



```

13 public void wrongAuthorizationTest () {
14     assertEquals(sessionFactory.isClosed(), false);
15     clickOn(userName).write("Aleksey");
16     clickOn(password).write("retdfgdfh");
17     clickOn(authorization);
18     verifyThat(resultAuthorization, hasText("Не верный пароль"));
19     assertEquals(resultAuthorization.getFill(), Color.FIREBRICK);
20 }

```

### 4.3. Реакция программы при не существующем логине

После нажатия на кнопку авторизации в БД ищется пользователь с указанным логином, если не возвращается пользователь с таким логином, то должно появиться сообщение: "Такого пользователя не существует" (Листинг ??).

Листинг 3 – Тестирование ввода неверного пароля

```

1 @Test
2 public void thisUserDoesntExistTest () {
3     assertEquals(sessionFactory.isClosed(), false);
4     clickOn(userName).write("ТестНик");
5     clickOn(password).write("tttt");
6     clickOn(authorization);
7     verifyThat(resultAuthorization, hasText("Такого пользователя не существует"));
8     assertEquals(resultAuthorization.getFill(), Color.FIREBRICK);
9 }

```

### 4.4. Реакция программы при верной авторизации

При прохождении авторизации окно авторизации должно закрыться, а вместо него должно появиться основное окно программы (Листинг 4).

Листинг 4 – Тестирование прохождения авторизации

```

1 @Test
2 public void openMainWindowTest () {
3     assertEquals(sessionFactory.isClosed(), false);
4     clickOn(userName).write("Aleksey");
5     clickOn(password).write("yui");
6     clickOn(authorization);
7     verifyThat(resultAuthorization, hasText("Пароль верный"));
8     assertEquals(resultAuthorization.getFill(), Color.GREEN);
9     waitUntil("#AnchorPane", visible());
10 }

```

### 4.5. Тестирование перехода к окну регистрации

При нажатии на кнопку "Регистрация", должно появиться соответствующее окно, а окно для авторизации исчезнуть (Листинг 5).

Листинг 5 – Тестирование перехода к окну регистрации

```
1  @Test
2  public void openRegistryWindowTest () {
3      openRegistryWindowAndTest ();
4      closeRegistryWindowAndTest ();
5  }
```

## 5. Тестирование взаимодействия классов MonoAlphabetCipher и BitReverseCipher

Тестирование осуществляется посредством многократного шифрования и расшифрования текста различными методами. После последовательного шифрования и расшифрования разными методами с неизменными ключами текст должен совпасть с исходным. Если провести такой же тест, но при расшифровании изменить ключ одного из методов, то результат должен не совпасть с исходным. Далее проведём тест, при котором зашифруем несколько раз обоими методами с разными ключами и расшифруем в обратном порядке с соответствующими ключами, результат должен совпасть с исходным. Все эти три теста описаны в классе IntegrationCipherTest, который представлен в листинге 6.

Листинг 6 – класс IntegrationCipherTest

```
1 public class IntegrationCipherTest {
2     private static SubstitutionCipher<Integer> monoAlphabet;
3     private static SubstitutionCipher<String> bitRevers;
4     private String actualText;
5
6     @BeforeClass
7     public static void test() {
8         monoAlphabet = new MonoAlphabetCipher();
9         bitRevers = new BitReversCipher();
10    }
11
12    @Before
13    public void createCipherClass() {
14        actualText = "специальный текст для тестов";
15    }
16
17    @Test
18    public void integrationCipherEqualsTest() {
19        monoAlphabet.calculationPrivateAlphabet(4);
20        bitRevers.calculationPrivateAlphabet("32451");
21        String expectedText = monoAlphabet.encodeText(actualText);
22        expectedText = bitRevers.decodeText(expectedText);
23        expectedText = bitRevers.encodeText(expectedText);
24        expectedText = monoAlphabet.decodeText(expectedText);
25        Assert.assertEquals(expectedText, actualText);
26    }
27
28    @Test
29    public void integrationCipherNotEqualsTest() {
30        monoAlphabet.calculationPrivateAlphabet(4);
31        bitRevers.calculationPrivateAlphabet("32451");
32        String testText = monoAlphabet.encodeText(actualText);
33        testText = bitRevers.decodeText(testText);
34        bitRevers.calculationPrivateAlphabet("52341");
35        String expectedText = bitRevers.encodeText(testText);
36        Assert.assertNotEquals(expectedText, testText);
37    }
38
39    @Test
```

```

40 public void integrationDeepCipherEqualsTest () {
41     monoAlphabet.calculationPrivateAlphabet (2);
42     String expectedText = monoAlphabet.encodeText (actualText);
43
44     bitRevers.calculationPrivateAlphabet ("53241");
45     expectedText = bitRevers.encodeText (expectedText);
46     Assert.assertNotEquals (expectedText , actualText);
47
48     monoAlphabet.calculationPrivateAlphabet (3);
49     expectedText = monoAlphabet.decodeText (expectedText);
50
51     bitRevers.calculationPrivateAlphabet ("24531");
52     expectedText = bitRevers.decodeText (expectedText);
53     Assert.assertNotEquals (expectedText , actualText);
54
55     expectedText = bitRevers.encodeText (expectedText);
56     expectedText = monoAlphabet.encodeText (expectedText);
57     Assert.assertNotEquals (expectedText , actualText);
58
59     bitRevers.calculationPrivateAlphabet ("53241");
60     expectedText = bitRevers.decodeText (expectedText);
61
62     monoAlphabet.calculationPrivateAlphabet (2);
63     expectedText = monoAlphabet.decodeText (expectedText);
64
65     Assert.assertEquals (expectedText , actualText);
66 }
67 }

```

## 6. Тестирование взаимодействия класса MainWindow

MainWindow - класс отвечающий за главное окно программы. Он взаимодействует с классами, реализующими методы шифрования. Проведём тестирование, аналогичное тестированию классов MonoAlphabetCipher и BitReverseCipher, но теперь классы для шифрования будут взаимодействовать не напрямую, а через MainWindow (как и происходит на самом деле), посредством моделирования работы пользователя с gui. Оно представлено в листинге 7. Предыдущее тестирование направлено на проверку корректности классов шифрования и их взаимодействия, а текущее преимущественно на корректность работы MainWindow с данными классами.

Листинг 7 – класс MainWindowUiIntegrationTest

```
1 public class MainWindowUiIntegrationTest extends CustomMainWindowUiTest {
2     private TextArea textArea;
3     String actualText;
4
5     @BeforeClass
6     public static void mainWindowSettings() {
7         MainWindowController.nameMethods = Arrays.asList(
8             methodNames.get(0), methodNames.get(5)
9         );
10    }
11
12    @Before
13    public void findTextArea() {
14        textArea = find("#textArea");
15        actualText = "специальный текст для тестов";
16        robot.clickOn(textArea).write(actualText);
17    }
18
19    @After
20    public void clearTextArea() {
21        textArea.clear();
22    }
23
24    @Test
25    public void integrationCipherEqualsTest() {
26        robot.clickOn("#encodeMenu").clickOn("#encodeMonoAlphabet")
27            .clickOn("#txtFieldDialog").write('4').clickOn("#okDialog");
28        robot.clickOn("#decodeMenu").clickOn("#decodeBitRevers")
29            .clickOn("#txtFieldDialog").write("32451").clickOn("#okDialog");
30        robot.clickOn("#encodeMenu").clickOn("#encodeBitRevers")
31            .clickOn("#txtFieldDialog").write("32451").clickOn("#okDialog");
32        robot.clickOn("#decodeMenu").clickOn("#decodeMonoAlphabet")
33            .clickOn("#txtFieldDialog").write('4').clickOn("#okDialog");
34        Assert.assertEquals(textArea.getText(), actualText);
35    }
36
37    @Test
38    public void integrationCipherNotEqualsTest() {
39        robot.clickOn("#encodeMenu").clickOn("#encodeMonoAlphabet")
40            .clickOn("#txtFieldDialog").write('4').clickOn("#okDialog");
41        robot.clickOn("#decodeMenu").clickOn("#decodeBitRevers")
42            .clickOn("#txtFieldDialog").write("32451").clickOn("#okDialog");
```

```

43     String testText = textArea.getText();
44     robot.clickOn("#decodeMenu").clickOn("#decodeBitRevers")
45         .clickOn("#txtFieldDialog").write("52341").clickOn("#okDialog");
46     Assert.assertNotEquals(textArea.getText(), testText);
47 }
48
49 @Test
50 public void integrationDeepCipherEqualsTest() {
51     robot.clickOn("#encodeMenu").clickOn("#encodeMonoAlphabet")
52         .clickOn("#txtFieldDialog").write("2").clickOn("#okDialog");
53     robot.clickOn("#encodeMenu").clickOn("#encodeBitRevers")
54         .clickOn("#txtFieldDialog").write("53241").clickOn("#okDialog");
55     Assert.assertNotEquals(textArea.getText(), actualText);
56     robot.clickOn("#decodeMenu").clickOn("#decodeMonoAlphabet")
57         .clickOn("#txtFieldDialog").write('3').clickOn("#okDialog");
58     robot.clickOn("#decodeMenu").clickOn("#decodeBitRevers")
59         .clickOn("#txtFieldDialog").write("24531").clickOn("#okDialog");
60     Assert.assertNotEquals(textArea.getText(), actualText);
61     robot.clickOn("#encodeMenu").clickOn("#encodeBitRevers")
62         .clickOn("#txtFieldDialog").write("24531").clickOn("#okDialog");
63     robot.clickOn("#encodeMenu").clickOn("#encodeMonoAlphabet")
64         .clickOn("#txtFieldDialog").write('3').clickOn("#okDialog");
65     Assert.assertNotEquals(textArea.getText(), actualText);
66     robot.clickOn("#decodeMenu").clickOn("#decodeBitRevers")
67         .clickOn("#txtFieldDialog").write("53241").clickOn("#okDialog");
68     robot.clickOn("#decodeMenu").clickOn("#decodeMonoAlphabet")
69         .clickOn("#txtFieldDialog").write('2').clickOn("#okDialog");
70     Assert.assertEquals(textArea.getText(), actualText);
71 }
72 }

```

## 7. Выводы

В ходе выполнения лабораторной работы была разработана система интеграционного тестирования проекта. Для написания тестов была использована библиотека JUnit, а действия пользователя моделировались с помощью библиотеки TestFX. Тесты составлялись из расчёта на то, что всякое непосредственное взаимодействие классов программы должно быть покрыто. В результате тестирования была обнаружена ошибка, не замеченная при модульном тестировании: в базовом классе SubstitutionCipher поле `privateAlphabet` (закрытый алфавит - правила сопоставления символов при шифровании) было объявлено как `static`, т.е. все наследники этого класса ссылались на одно и то же поле закрытого алфавита, из-за чего в данном поле хранился закрытый алфавит того класса, который последним выполнил метод `calculationPrivateAlphavit`. Такое поведение классов шифрования недопустимо, но не было замечено на предыдущем этапе, т.к. оба наследника создавались одновременно и вначале выполняли шифрование алфавита, а после уже шифровали текст. Других ошибок в работе приложения обнаружено не было. Так же была обнаружена, плавающая ошибка (Гейзенбаг) для окна регистрации, при моделировании работы с ним через класс `robotFx`, предоставляемый библиотекой TestFx, было замечено, что иногда робот промахивается и не попадает в то поле, которое должен был кликнуть для ввода текста, из-за чего тесты проваливались. После внимательного разбора кода приложения и fxml файла описания интерфейса был сделан вывод, что это ошибка именно самой библиотеки. Вызванное, скорее всего, разной работой с fxml описанием интерфейса и описанием с помощью классов.

# Приложение 1. Исходные коды тестируемых классов

Листинг 8 – код модуля SubstitutionCipher.java

```
1 package encryptionMethods.base;
2
3 /**
4  * Created by Алексей on 16.12.2016.
5  */
6 public abstract class SubstitutionCipher<T> {
7     // Открытый алфавит
8     protected static final char[] openAlphabet = new char[32];
9     // Закрытый алфавит
10    protected static final char[] privateAlphabet = new char[32];
11
12    public SubstitutionCipher() {
13        createOpenAlphabet();
14    }
15
16    private void createOpenAlphabet() {
17        openAlphabet[0] = '\u0020';
18        for (int i = 0; i < 9; i++) {
19            openAlphabet[i + 1] = (char) ('a' + i);
20        }
21        for (int i = 10; i < 32; ++i) {
22            openAlphabet[i] = (char) ('a' + i);
23        }
24    }
25
26    public abstract void calculationPrivateAlphabet(T key);
27
28    /**
29     * Закодировать текст
30     *
31     * @param originalText текст
32     * @return закодированный текст
33     */
34    public String encodeText(String originalText) {
35        originalText = originalText.replaceAll("\n", " ").toLowerCase();
36        char[] text = originalText.toCharArray();
37        char[] result = new char[text.length];
38        int index;
39        for (int i = 0; i < text.length; i++) {
40            index = contains(openAlphabet, text[i]);
41            if (index < 0) return "Недопустимый символ. Шифрование не возможно";
42            result[i] = privateAlphabet[index];
43        }
44        return String.valueOf(result);
45    }
46
47    /**
48     * Раскодировать текст
49     *
50     * @param originalText текст
51     * @return раскодированный текст
```



```

52  */
53  public String decodeText(String originalText) {
54      char[] text = originalText.toCharArray();
55      char[] result = new char[text.length];
56      int index;
57      for (int i = 0; i < text.length; i++) {
58          index = contains(privateAlphabet, text[i]);
59          if (index < 0) return "Недопустимый символ. Расшифровка не возможно";
60          result[i] = openAlphabet[index];
61      }
62      return String.valueOf(result);
63  }
64
65  private int contains(char[] chars, char symbol) {
66      for (int i = 0; i < chars.length; i++) {
67          if (chars[i] == symbol) {
68              return i;
69          }
70      }
71      return -1;
72  }
73
74  public char[] getOpenAlphabet() {
75      return openAlphabet;
76  }
77
78  public char[] getPrivateAlphabet() {
79      return privateAlphabet;
80  }
81  }

```

Листинг 9 – код модуля MonoAlphabetCipher.java

```
1 package encryptionMethods.monoAlphabet;
2
3 import encryptionMethods.base.SubstitutionCipher;
4
5 /**
6  * Класс для кодирования текста моноалфавитным методом Created by Алексей on
7  * 25.03.2016.
8  */
9 public class MonoAlphabetCipher extends SubstitutionCipher<Integer> {
10
11     public MonoAlphabetCipher() {
12         super();
13     }
14
15     /**
16     * Создать закрытый алфавит
17     *
18     * @param key ключ смещения
19     */
20     @Override
21     public void calculationPrivateAlphabet(Integer key) {
22         for (int i = 0; i < 32; i++) {
23             privateAlphabet[i] = openAlphabet[Math.floorMod(i + key, 32)];
24         }
25     }
26 }
```

Листинг 10 – код модуля BitReversCipher.java

```

1 package encryptionMethods.bitrevers;
2
3 import encryptionMethods.base.SubstitutionCipher;
4
5 /**
6  * Класс для кодирования текста методом побитовой перестановки Created by
7  * Алексей on 25.03.2016.
8  */
9 public class BitReversCipher extends SubstitutionCipher<String> {
10     public BitReversCipher() {
11         super();
12     }
13
14     /**
15      * Вычислить закрытый алфавит по заданому ключу
16      *
17      * @param key ключ
18      */
19     @Override
20     public void calculationPrivateAlphabet (String key) {
21         int[] arrayKey = findPosition(key);
22         for (int i = 0; i < 32; i++) {
23             StringBuilder stI = new StringBuilder(Integer.toBinaryString(i));
24             for (int j = stI.length(); j < 5; j++) {
25                 stI.insert(0, "0");
26             }
27             StringBuilder charAlph = new StringBuilder();
28             char[] mass = stI.toString().toCharArray();
29             for (int anArrayKey : arrayKey) {
30                 charAlph.append(mass[anArrayKey]);
31             }
32             int exitI = Integer.parseInt(charAlph.toString(), 2);
33             privateAlphabet[i] = openAlphabet[exitI];
34         }
35     }
36
37     /**
38      * Найти в каких позициях произошла перестановка
39      *
40      * @param arrayKey
41      * @return
42      */
43     private int[] findPosition(String arrayKey) {
44         int[] result = new int[arrayKey.length()];
45         result[0] = arrayKey.indexOf("1");
46         result[1] = arrayKey.indexOf("2");
47         result[2] = arrayKey.indexOf("3");
48         result[3] = arrayKey.indexOf("4");
49         result[4] = arrayKey.indexOf("5");
50         return result;
51     }
52 }

```

# Листинг 11 – код модуля MainWindowController.java

```

1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package controller;
7
8  import encryptionMethods.base.SubstitutionCipher;
9  import encryptionMethods.bitrevers.BitReversCipher;
10 import encryptionMethods.monoAlphabet.MonoAlphabetCipher;
11 import javafx.event.ActionEvent;
12 import javafx.event.EventHandler;
13 import javafx.fxml.FXML;
14 import javafx.fxml.Initializable;
15 import javafx.geometry.Pos;
16 import javafx.scene.Scene;
17 import javafx.scene.control.*;
18 import javafx.scene.layout.HBox;
19 import javafx.scene.layout.VBox;
20 import javafx.stage.FileChooser;
21 import javafx.stage.Stage;
22 import javafx.stage.StageStyle;
23 import javafx.stage.WindowEvent;
24 import org.hibernate.SessionFactory;
25 import utils.FileWorker;
26 import utils.alert.ErrorAlert;
27 import utils.alert.InformAlert;
28
29 import java.io.File;
30 import java.io.IOException;
31 import java.net.URL;
32 import java.util.HashMap;
33 import java.util.List;
34 import java.util.Map;
35 import java.util.ResourceBundle;
36 import java.util.function.Consumer;
37 import java.util.function.Predicate;
38
39 import static utils.UtilFunctions.isNullString;
40
41 /**
42  * FXML Controller class
43  *
44  * @author Алексей
45  */
46 public class MainWindowController implements Initializable {
47
48     public static SessionFactory sessionFactory;
49     public static Stage stage;
50     public static List<String> nameMethods;
51     public MenuItem decodeMonoAlphabet;
52
53     @FXML
54     private TextArea textArea;

```

```

55 @FXML
56 private Menu encodeMenu;
57 @FXML
58 private Menu decodeMenu;
59
60 private SubstitutionCipher<Integer> monoAlphabet;
61 private SubstitutionCipher<String> bitRevers;
62 private EventHandler<ActionEvent> nonMethodHandler;
63 private Map<String, EventHandler<ActionEvent>> handlerMap;
64 private Predicate<MenuItem> filterMethods;
65 private Consumer<MenuItem> getDesiredMethods;
66
67 public MainWindowController() {
68
69 }
70
71 /**
72  * Initializes the controller class.
73  */
74 @Override
75 public void initialize(URL url, ResourceBundle rb) {
76     initializationPrivateField();
77     encodeMenu.getItems().stream().filter(filterMethods).forEach(getDesiredMethods);
78     decodeMenu.getItems().stream().filter(filterMethods).forEach(getDesiredMethods);
79     initializeStage();
80 }
81
82 private void initializationPrivateField() {
83     monoAlphabet = new MonoAlphabetCipher();
84     bitRevers = new BitReversCipher();
85     handlerMap = new HashMap<>();
86     //<editor-fold desc="Блок определения лямб для фильтрации списков меню шифрования"
87     filterMethods = menuItem -> {
88         String id = menuItem.getId();
89         return (nameMethods != null) && nameMethods.contains(id.substring(6));
90     };
91     getDesiredMethods = menuItem -> {
92         String id = menuItem.getId();
93         menuItem.setOnAction(getHandlerByMethodName(id));
94         menuItem.setDisable(false);
95     };
96     //</editor-fold>
97     //<editor-fold desc="Создание обработчика для ситуации когда метод не реализован"
98     nonMethodHandler = event -> {
99         Stage dialog = new Stage();
100         dialog.initStyle(StageStyle.UTILITY);
101         dialog.setTitle("Ввод ключа");
102         VBox box = new VBox();
103         box.setAlignment(Pos.CENTER);
104         HBox buttons = new HBox();
105         buttons.setAlignment(Pos.CENTER);
106         Button buttonOk = new Button("Ok");
107         buttonOk.setId("okDialog");
108         buttonOk.setOnAction((ActionEvent evt) -> {
109             dialog.close();

```

```

110     });
111
112     buttons.getChildren().addAll(buttonOk);
113     Label label = new Label("Метод не реализован");
114     label.setId("labelDialog");
115     box.getChildren().addAll(label, buttons);
116     Scene scene = new Scene(box, 300, 100);
117     dialog.setScene(scene);
118     dialog.show();
119 };
120 //</editor-fold>
121 //<editor-fold desc="Заполнения словаря обработчиков шифрования/расшифрования дл
122 handlerMap.put("encodeMonoAlphabet", event -> {
123     Stage dialog = new Stage();
124     dialog.initStyle(StageStyle.UTILITY);
125     dialog.setTitle("Окно ввода ключа");
126     VBox box = new VBox();
127     box.setAlignment(Pos.CENTER);
128     HBox buttons = new HBox();
129     TextField txtField = new TextField();
130     txtField.setId("txtFieldDialog");
131     buttons.setAlignment(Pos.CENTER);
132     Button buttonOk = new Button("Ok");
133     buttonOk.setId("okDialog");
134     buttonOk.setOnAction((ActionEvent evt) -> {
135         Integer key = Integer.valueOf(txtField.getText());
136         monoAlphabet.calculationPrivateAlphabet(key);
137         dialog.close();
138         String codeText = monoAlphabet
139             .encodeText(textArea.getText().replaceAll("\n", " "));
140         textArea.clear();
141         textArea.setText(codeText);
142     });
143     Button buttonEx = new Button("Cancel");
144     buttonEx.setId("cancelDialog");
145     buttonEx.setOnAction(evt -> {
146         dialog.close();
147     });
148     buttons.getChildren().addAll(buttonOk, buttonEx);
149     Label label = new Label("Введите ключ");
150     label.setId("labelDialog");
151     box.getChildren().addAll(label, txtField, buttons);
152     Scene scene = new Scene(box, 300, 100);
153     dialog.setScene(scene);
154     dialog.show();
155 });
156
157 handlerMap.put("encodeBitRevers", event -> {
158     Stage dialog = new Stage();
159     dialog.initStyle(StageStyle.UTILITY);
160     dialog.setTitle("Окно ввода ключа");
161     VBox box = new VBox();
162     box.setAlignment(Pos.CENTER);
163     HBox buttons = new HBox();
164     TextField txtField = new TextField();

```

```

165     txtField.setId("txtFieldDialog");
166     buttons.setAlignment(Pos.CENTER);
167     Button buttonOk = new Button("Ok");
168     buttonOk.setId("okDialog");
169     buttonOk.setOnAction((ActionEvent evt) -> {
170         bitRevers.calculationPrivateAlphabet(txtField.getText());
171         dialog.close();
172         String codeText = bitRevers
173             .encodeText(textArea.getText().replaceAll("\n", " "));
174         textArea.clear();
175         textArea.setText(codeText);
176     });
177     Button buttonEx = new Button("Cancel");
178     buttonEx.setId("cancelDialog");
179     buttonEx.setOnAction(evt -> {
180         dialog.close();
181     });
182     buttons.getChildren().addAll(buttonOk, buttonEx);
183     Label label = new Label("Введите ключ");
184     label.setId("labelDialog");
185     box.getChildren().addAll(label, txtField, buttons);
186     Scene scene = new Scene(box, 300, 100);
187     dialog.setScene(scene);
188     dialog.show();
189 });
190
191 handlerMap.put("decodeMonoAlphabet", event -> {
192     Stage dialog = new Stage();
193     dialog.initStyle(StageStyle.UTILITY);
194     dialog.setTitle("Окно ввода ключа");
195     VBox box = new VBox();
196     box.setAlignment(Pos.CENTER);
197     HBox buttons = new HBox();
198     TextField txtField = new TextField();
199     txtField.setId("txtFieldDialog");
200     buttons.setAlignment(Pos.CENTER);
201     Button buttonOk = new Button("Ok");
202     buttonOk.setId("okDialog");
203     buttonOk.setOnAction((ActionEvent evt) -> {
204         int key = Integer.valueOf(txtField.getText());
205         monoAlphabet.calculationPrivateAlphabet(key);
206         dialog.close();
207         String decodeText = monoAlphabet
208             .decodeText(textArea.getText().replaceAll("\n", " "));
209         textArea.clear();
210         textArea.setText(decodeText);
211     });
212     Button buttonEx = new Button("Cancel");
213     buttonEx.setId("cancelDialog");
214     buttonEx.setOnAction(evt -> {
215         dialog.close();
216     });
217     buttons.getChildren().addAll(buttonOk, buttonEx);
218     Label label = new Label("Введите ключ");
219     label.setId("labelDialog");

```

```

220         box.getChildren().addAll(label, txtField, buttons);
221         Scene scene = new Scene(box, 300, 100);
222         dialog.setScene(scene);
223         dialog.show();
224     });
225
226     handlerMap.put("decodeBitRevers", event -> {
227         Stage dialog = new Stage();
228         dialog.initStyle(StageStyle.UTILITY);
229         dialog.setTitle("Окно ввода ключа");
230         VBox box = new VBox();
231         box.setAlignment(Pos.CENTER);
232         HBox buttons = new HBox();
233         TextField txtField = new TextField();
234         txtField.setId("txtFieldDialog");
235         buttons.setAlignment(Pos.CENTER);
236         Button buttonOk = new Button("Ok");
237         buttonOk.setId("okDialog");
238         buttonOk.setOnAction((ActionEvent evt) -> {
239             bitRevers.calculationPrivateAlphabet(txtField.getText());
240             dialog.close();
241             String decodeText = bitRevers
242                 .decodeText(textArea.getText().replaceAll("\n", " "));
243             textArea.clear();
244             textArea.setText(decodeText);
245         });
246         Button buttonEx = new Button("Cancel");
247         buttonEx.setId("cancelDialog");
248         buttonEx.setOnAction(evt -> {
249             dialog.close();
250         });
251         buttons.getChildren().addAll(buttonOk, buttonEx);
252         Label label = new Label("Введите ключ");
253         label.setId("labelDialog");
254         box.getChildren().addAll(label, txtField, buttons);
255         Scene scene = new Scene(box, 300, 100);
256         dialog.setScene(scene);
257         dialog.show();
258     });
259     //</editor-fold>
260 }
261
262 private void initializeStage() {
263     if (stage == null) {
264         stage = new Stage();
265         stage.setTitle("Добро пожаловать, Неизвестный");
266     }
267     ;
268     stage.setOnCloseRequest(we -> sessionFactory.close());
269 }
270
271
272 private EventHandler<ActionEvent> getHandlerByMethodName(String methodName) {
273     if (handlerMap.containsKey(methodName)) {
274         return handlerMap.get(methodName);

```



```

275     }
276     return nonMethodHandler;
277 }
278
279 @FXML
280 private void onActionProperty(ActionEvent event) {
281     stage.fireEvent(new WindowEvent(
282         stage,
283         WindowEvent.WINDOW_CLOSE_REQUEST
284     ));
285 }
286
287 @FXML
288 private void onClearTextArea(ActionEvent event) {
289     textArea.clear();
290 }
291
292 @FXML
293 private void openFile(ActionEvent event) throws IOException {
294     FileChooser fileChooser = new FileChooser(); //Класс работы с диалогом выборки и
295     fileChooser.setTitle("Open Document"); //Заголовок диалога
296     FileChooser.ExtensionFilter extFilter = new FileChooser.ExtensionFilter("txt fil
297     fileChooser.getExtensionFilters().add(extFilter);
298
299     File file = fileChooser.showOpenDialog(stage);
300     String str = FileWorker.readFile(file).toString();
301     Alert alert = (isNullString(str) || str.equals("Файл не найден")) ?
302         new ErrorAlert("Ошибка чтения файла: " + (isNullString(str) ? "Смотрите
303         : new InformAlert("Файл прочитан.");
304     textArea.setText(str);
305     alert.showAndWait();
306 }
307
308 @FXML
309 private void saveFile(ActionEvent event) {
310     FileChooser fileChooser = new FileChooser(); //Класс работы с диалогом выборки и
311     fileChooser.setTitle("Save Document"); //Заголовок диалога
312     FileChooser.ExtensionFilter extFilter = new FileChooser.ExtensionFilter("txt fil
313     fileChooser.getExtensionFilters().add(extFilter);
314     File file = fileChooser.showSaveDialog(stage); //Указываем текущую сцену
315     boolean result = FileWorker.writeFile(file, textArea.getText());
316     Alert alert = result ? new ErrorAlert("Ошибка записи в файл: Смотрите лог")
317         : new InformAlert("Текст записан в файл");
318     alert.showAndWait();
319 }
320
321 @FXML
322 private void getHelpMethod(ActionEvent event) {
323     Stage dialog = new Stage();
324     dialog.initStyle(StageStyle.UTILITY);
325     dialog.setTitle("Справка");
326     VBox box = new VBox();
327     box.setAlignment(Pos.CENTER);
328     TextArea txtAreaHelp = new TextArea();
329     txtAreaHelp.appendText("Моноалфавитная замена \n");

```

```

330     txtAreaHelp.appendText(
331         "При Моноалфавитной замене каждой букве открытого текста ставится соотве
332         + "закрытого текста из этого же алфавита.  $y_i=(K_1 X_i+K_2)$  по
333         + " $K_1$  и  $K_2$ — это константы.  $K_1=1, K_2$ — это смещение \n"
334         + "символов закрытого алфавита относительно открытого алфавита, \n"
335         + "если  $K_1=1, K_2=3$ , то это так называемый код Цезаря.  $X_i$ — это
336         + " $i$  символа открытого алфавита  $Y_i$ — это код  $i$  символа закрытого
337     );
338     txtAreaHelp.appendText("Побитовая перестановка\n");
339     txtAreaHelp.appendText(
340         "Несколько более сложной является побитовая перестановка, при которой в
341         + "соответствии с вектором перестановки изменяются позиции разря
342         + "символов открытого текста, обычно берутся ASCII коды. \n"
343     );
344
345     HBox buttons = new HBox();
346     buttons.setAlignment(Pos.CENTER);
347     Button buttonOk = new Button("Ok");
348     buttonOk.setOnAction((ActionEvent evt) -> {
349         dialog.close();
350     });
351
352     buttons.getChildren().addAll(buttonOk);
353     box.getChildren().addAll(txtAreaHelp, buttons);
354     Scene scene = new Scene(box, 600, 200);
355     dialog.setScene(scene);
356     dialog.show();
357 }
358
359 }

```

Листинг 12 – код модуля Login.java

```

1 package controller;
2
3 import database.dao.Dao;
4 import database.entity.User;
5 import database.service.DaoFactory;
6 import database.service.DataBaseService;
7 import javafx.application.Application;
8 import javafx.event.ActionEvent;
9 import javafx.event.EventHandler;
10 import javafx.fxml.FXMLLoader;
11 import javafx.geometry.Insets;
12 import javafx.geometry.Pos;
13 import javafx.scene.Parent;
14 import javafx.scene.Scene;
15 import javafx.scene.control.Button;
16 import javafx.scene.control.Label;
17 import javafx.scene.control.PasswordField;
18 import javafx.scene.control.TextField;
19 import javafx.scene.layout.GridPane;
20 import javafx.scene.layout.HBox;
21 import javafx.scene.paint.Color;
22 import javafx.scene.text.Font;
23 import javafx.scene.text.FontWeight;

```

```

24 import javafx.scene.text.Text;
25 import javafx.stage.Stage;
26 import javafx.stage.WindowEvent;
27 import org.hibernate.SessionFactory;
28 import utils.UtilFunctions;
29
30 import java.io.IOException;
31 import java.util.logging.Level;
32 import java.util.logging.Logger;
33
34 import static utils.UtilFunctions.isNullString;
35
36 /**
37  * @author Алексей
38  */
39 public class Login extends Application {
40
41     private static int count = 0;
42     private EventHandler<ActionEvent> exitHandler;
43     private WindowEvent exitWindow;
44     public static GridPane GRID;
45     public Stage primaryStage;
46     private static Dao<User> userDao;
47     public static SessionFactory sessionFactory;
48
49     static {
50         createEntityForDateBaseWork();
51     }
52
53     {
54         exitWindow = new WindowEvent(
55             primaryStage,
56             WindowEvent.WINDOW_CLOSE_REQUEST
57         );
58         exitHandler = event -> primaryStage.fireEvent(exitWindow);
59     }
60
61     private static void createEntityForDateBaseWork() {
62         DataBaseService dataBaseService = DataBaseService.instanceDataBaseService();
63         sessionFactory = dataBaseService.getSessionFactory();
64         userDao = DaoFactory.getInstance(sessionFactory).getUserDao();
65     }
66
67     @Override
68     public void start(Stage primaryStage) {
69         primaryStage.setOnCloseRequest(we -> sessionFactory.close());
70         this.primaryStage = primaryStage;
71         primaryStage.setTitle("Окно авторизации");
72
73         GRID = new GridPane();
74         GRID.setAlignment(Pos.CENTER);
75         GRID.setVgap(10);
76         GRID.setHgap(10);
77         GRID.setPadding(new Insets(25, 25, 25, 25));
78

```

```

79     Text sceneTitle = new Text("Добро пожаловать");
80     sceneTitle.setFont(Font.font("Tahoma", FontWeight.NORMAL, 20));
81     GRID.add(sceneTitle, 0, 0, 2, 1);
82
83     Label userName = new Label("User Name: ");
84     GRID.add(userName, 0, 1);
85
86     TextField userTextField = new TextField();
87     userTextField.setId("login");
88     GRID.add(userTextField, 1, 1);
89
90     Label password = new Label("Password: ");
91     GRID.add(password, 0, 2);
92
93     PasswordField pwBox = new PasswordField();
94     pwBox.setId("password");
95     GRID.add(pwBox, 1, 2);
96
97     Button sign = new Button("Авторизоваться");
98     sign.setId("authorization");
99     Button exit = new Button("Выход");
100    exit.setId("exit");
101    Button registration = new Button("Регистрация");
102    registration.setId("checkIn");
103    registration.setOnAction(moveToRegistryWindow());
104
105    HBox hbSign = new HBox(10);
106    hbSign.setAlignment(Pos.BOTTOM_LEFT);
107    hbSign.getChildren().add(sign);
108    hbSign.getChildren().add(registration);
109    hbSign.getChildren().add(exit);
110    GRID.add(hbSign, 1, 5);
111
112    final Text actionTarget = new Text();
113    actionTarget.setId("resultAuthorization");
114    GRID.add(actionTarget, 1, 6);
115
116    exit.setOnAction(exitHandler);
117
118    sign.setOnAction(event -> {
119        actionTarget.setFill(Color.FIREBRICK);
120        String name = userTextField.getText();
121        User user = userDao.getEntityByStringProperty("login", name);
122        if (user == null) {
123            actionTarget.setText("Такого пользователя не существует");
124            return;
125        }
126        String realPassword = user.getPassword();
127        String pass = UtilFunctions.md5Custom(pwBox.getText());
128        if (!realPassword.equals(pass)) {
129            ++count;
130            if (count == 3) {
131                primaryStage.fireEvent(exitWindow);
132            }
133            actionTarget.setText("Не верный пароль");

```

```

134         return;
135     }
136     actionTarget.setFill(Color.GREEN);
137     actionTarget.setText("Пароль верный");
138     Parent root = null;
139     Stage mainWindowStage = new Stage();
140     try {
141         MainWindowController.stage = mainWindowStage;
142         MainWindowController.sessionFactory = sessionFactory;
143         MainWindowController.nameMethods = user.getMethods();
144         String firstName = isNullString(user.getFirstName()) ? "Неизвестный" : user.getFirstName();
145         mainWindowStage.setTitle("Добро пожаловать, " + firstName);
146
147         root = FXMLLoader.load(this.getClass().getResource("/fxml/MainWindow.fxml"));
148     };
149
150     primaryStage.close(); // закрытие формы авторизации
151     Scene scene = new Scene(root, 400, 400);
152
153     mainWindowStage.setScene(scene);
154     mainWindowStage.show();
155 } catch (IOException ex) {
156     Logger.getLogger(Login.class.getName()).log(Level.SEVERE, null, ex);
157 }
158
159 });
160
161 Scene scene = new Scene(GRID, 500, 275);
162 primaryStage.setScene(scene);
163
164 primaryStage.show();
165 }
166
167 private EventHandler<ActionEvent> moveToRegistryWindow() {
168     return event -> {
169         Parent root = null;
170         try {
171             GRID.setDisable(true);
172             root = FXMLLoader.load(this.getClass().getResource("/fxml/Registry.fxml"));
173             Scene scene = new Scene(root, 600.0D, 400.0D);
174             Stage registryStage = new Stage();
175             registryStage.setScene(scene);
176             RegistryController.STAGE = registryStage;
177             RegistryController.parentPane = GRID;
178             RegistryController.userDao = userDao;
179             registryStage.show();
180         } catch (IOException e) {
181             e.printStackTrace();
182         }
183     };
184 }
185
186 /**
187  * @param args the command line arguments

```

```
189      */
190      public static void main(String [] args) {
191          launch(args);
192      }
193
194 }
```

## Приложение 2. Исходные коды тестирующих классов

Листинг 13 – код модуля CustomMainWindowUiTest.java

```
1 package ui.custom;
2
3 import javafx.fxml.FXMLLoader;
4 import javafx.scene.Parent;
5 import org.loadui.testfx.GuiTest;
6 import org.testfx.api.FxRobot;
7
8 import java.io.IOException;
9 import java.util.Arrays;
10 import java.util.List;
11
12 /**
13  * Created by Алексей on 25.12.2016.
14  */
15 public class CustomMainWindowUiTest extends GuiTest {
16     protected static FxRobot robot;
17     protected static List<String> methodNames;
18
19     static {
20         robot = new FxRobot();
21
22         methodNames = Arrays.asList(
23             "MonoAlphabet", "HomophonyReplacement", "PolyalphabeticReplacement",
24             "PoligrammnayaReplacement", "VerticalPermutation", "BitRevers",
25             "VizhinerMethod", "XOR"
26         );
27     }
28
29     @Override
30     protected Parent getRootNode() {
31         Parent parent = null;
32         try {
33             parent = FXMLLoader.load(getClass().getResource("/fxml/MainWindow.fxml"));
34             return parent;
35         } catch (IOException ex) {
36             // TODO ...
37         }
38         return parent;
39     }
40 }
```

Листинг 14 – код модуля LoginUiCustomTest.java

```

1 package ui.custom;
2
3 import controller.Login;
4 import javafx.scene.control.Button;
5 import javafx.scene.control.PasswordField;
6 import javafx.scene.control.TextField;
7 import javafx.scene.layout.GridPane;
8 import javafx.scene.text.Text;
9 import javafx.stage.Stage;
10 import org.testfx.framework.junit.ApplicationTest;
11
12 import static controller.Login.GRID;
13 import static org.junit.Assert.assertEquals;
14 import static org.loadui.testfx.GuiTest.find;
15 import static org.loadui.testfx.GuiTest.waitUntil;
16 import static org.loadui.testfx.controls.impl.VisibleNodesMatcher.visible;
17
18 /**
19  * Created by Алексей on 25.12.2016.
20  */
21 public abstract class LoginUiCustomTest extends ApplicationTest {
22     protected Text resultAuthorization;
23     protected Button authorization;
24     protected Button registry;
25     protected TextField userName;
26     protected PasswordField password;
27     protected Stage stage;
28     protected GridPane gridPane;
29
30     @Override
31     public void start(Stage stage) throws Exception {
32         this.stage = stage;
33         new Login().start(stage);
34         gridPane = GRID;
35         resultAuthorization = find("#resultAuthorization");
36         authorization = find("#authorization");
37         userName = find("#login");
38         password = find("#password");
39         registry = find("#checkIn");
40     }
41
42     protected void openRegistryWindowAndTest() {
43         clickOn(registry);
44         assertEquals(gridPane.isDisable(), true);
45         waitUntil("#registryPane", visible());
46     }
47
48     protected void closeRegistryWindowAndTest() {
49         clickOn("#back");
50         assertEquals(gridPane.isDisable(), false);
51     }
52 }

```



Листинг 15 – код модуля LoginUiCloseTest.java

```
1 package ui.login;
2
3 import javafx.scene.paint.Color;
4 import org.junit.Test;
5 import ui.custom.LoginUiCustomTest;
6
7 import static controller.Login.sessionFactory;
8 import static org.junit.Assert.assertEquals;
9 import static org.testfx.api.FxAssert.verifyThat;
10 import static org.testfx.matcher.base.NodeMatchers.hasText;
11
12 /**
13  * Created by Алексей on 25.12.2016.
14  */
15 public class LoginUiCloseTest extends LoginUiCustomTest {
16     @Test
17     public void threeWrongAuthorizationTest() {
18         for (int i = 0; i < 3; i++) {
19             clickOn(userName).write("Aleksey");
20             clickOn(password).write("tttt");
21             clickOn(authorization).sleep(100);
22             verifyThat(resultAuthorization, hasText("Не верный пароль"));
23             assertEquals(resultAuthorization.getFill(), Color.FIREBRICK);
24             userName.clear();
25             password.clear();
26         }
27         assertEquals(stage.isShowing(), false);
28         assertEquals(sessionFactory.isClosed(), true);
29     }
30 }
```

Листинг 16 – код модуля LoginUiTest.java

```

1 package ui.login;
2
3 import javafx.scene.paint.Color;
4 import org.junit.Before;
5 import org.junit.Test;
6 import ui.custom.LoginUiCustomTest;
7
8 import static controller.Login.sessionFactory;
9 import static org.junit.Assert.assertEquals;
10 import static org.loadui.testfx.GuiTest.waitUntil;
11 import static org.loadui.testfx.controls.impl.VisibleNodesMatcher.visible;
12 import static org.testfx.api.FxAssert.verifyThat;
13 import static org.testfx.matcher.base.NodeMatchers.hasText;
14
15 /**
16  * Created by Алексей on 20.12.2016.
17  */
18 public class LoginUiTest extends LoginUiCustomTest {
19
20     @Before
21     public void checkOpenSession() {
22         assertEquals(sessionFactory.isClosed(), false);
23     }
24
25     @Test
26     public void openMainWindowTest() {
27         assertEquals(sessionFactory.isClosed(), false);
28         clickOn(userName).write("Aleksey");
29         clickOn(password).write("yui");
30         clickOn(authorization);
31         verifyThat(resultAuthorization, hasText("Пароль верный"));
32         assertEquals(resultAuthorization.getFill(), Color.GREEN);
33         waitUntil("#AnchorPane", visible());
34     }
35
36     @Test
37     public void wrongAuthorizationTest() {
38         assertEquals(sessionFactory.isClosed(), false);
39         clickOn(userName).write("Aleksey");
40         clickOn(password).write("retdfgdfh");
41         clickOn(authorization);
42         verifyThat(resultAuthorization, hasText("Не верный пароль"));
43         assertEquals(resultAuthorization.getFill(), Color.FIREBRICK);
44     }
45
46     @Test
47     public void openRegistryWindowTest() {
48         openRegistryWindowAndTest();
49         closeRegistryWindowAndTest();
50     }
51
52     @Test
53     public void thisUserDoesntExistTest() {
54         assertEquals(sessionFactory.isClosed(), false);

```

```
55     clickOn(userName).write("ТестНик");
56     clickOn(password).write("tttt");
57     clickOn(authorization);
58     verifyThat(resultAuthorization, hasText("Такого пользователя не существует"));
59     assertEquals(resultAuthorization.getFill(), Color.FIREBRICK);
60 }
61 }
```

Листинг 17 – код модуля IntegrationCipherTest.java

```

1 package unit.integration;
2
3 import encryptionMethods.base.SubstitutionCipher;
4 import encryptionMethods.bitrevers.BitReversCipher;
5 import encryptionMethods.monoAlphabet.MonoAlphabetCipher;
6 import org.junit.Assert;
7 import org.junit.Before;
8 import org.junit.BeforeClass;
9 import org.junit.Test;
10
11 /**
12  * Created by Алексей on 22.12.2016.
13  */
14 public class IntegrationCipherTest {
15     private static SubstitutionCipher<Integer> monoAlphabet;
16     private static SubstitutionCipher<String> bitRevers;
17     private String actualText;
18
19     @BeforeClass
20     public static void test() {
21         monoAlphabet = new MonoAlphabetCipher();
22         bitRevers = new BitReversCipher();
23     }
24
25     @Before
26     public void createCipherClass() {
27         actualText = "специальный текст для тестов";
28     }
29
30     @Test
31     public void integrationCipherEqualsTest() {
32         monoAlphabet.calculationPrivateAlphabet(4);
33         bitRevers.calculationPrivateAlphabet("32451");
34         String expectedText = monoAlphabet.encodeText(actualText);
35         expectedText = bitRevers.decodeText(expectedText);
36         expectedText = bitRevers.encodeText(expectedText);
37         expectedText = monoAlphabet.decodeText(expectedText);
38         Assert.assertEquals(expectedText, actualText);
39     }
40
41     @Test
42     public void integrationCipherNotEqualsTest() {
43         monoAlphabet.calculationPrivateAlphabet(4);
44         bitRevers.calculationPrivateAlphabet("32451");
45         String testText = monoAlphabet.encodeText(actualText);
46         testText = bitRevers.decodeText(testText);
47         bitRevers.calculationPrivateAlphabet("52341");
48         String expectedText = bitRevers.encodeText(testText);
49         Assert.assertNotEquals(expectedText, testText);
50     }
51
52     @Test
53     public void integrationDeepCipherEqualsTest() {
54         monoAlphabet.calculationPrivateAlphabet(2);

```

```

55     String expectedText = monoAlphabet.encodeText(actualText);
56
57     bitRevers.calculationPrivateAlphabet("53241");
58     expectedText = bitRevers.encodeText(expectedText);
59     Assert.assertNotEquals(expectedText, actualText);
60
61     monoAlphabet.calculationPrivateAlphabet(3);
62     expectedText = monoAlphabet.decodeText(expectedText);
63
64     bitRevers.calculationPrivateAlphabet("24531");
65     expectedText = bitRevers.decodeText(expectedText);
66     Assert.assertNotEquals(expectedText, actualText);
67
68     expectedText = bitRevers.encodeText(expectedText);
69     expectedText = monoAlphabet.encodeText(expectedText);
70     Assert.assertNotEquals(expectedText, actualText);
71
72     bitRevers.calculationPrivateAlphabet("53241");
73     expectedText = bitRevers.decodeText(expectedText);
74
75     monoAlphabet.calculationPrivateAlphabet(2);
76     expectedText = monoAlphabet.decodeText(expectedText);
77
78     Assert.assertEquals(expectedText, actualText);
79 }
80 }

```

Листинг 18 – код модуля MainWindowUiIntegrationTest.java

```

1 package ui.mainwindow;
2
3 import controller.MainWindowController;
4 import javafx.scene.control.TextArea;
5 import org.junit.*;
6 import ui.custom.CustomMainWindowUiTest;
7
8 import java.util.Arrays;
9
10 /**
11  * Created by Алексей on 25.12.2016.
12  */
13 public class MainWindowUiIntegrationTest extends CustomMainWindowUiTest {
14     private TextArea textArea;
15     String actualText;
16
17     @BeforeClass
18     public static void mainWindowSettings() {
19         MainWindowController.nameMethods = Arrays.asList(
20             methodNames.get(0), methodNames.get(5)
21         );
22     }
23
24     @Before
25     public void findTextArea() {
26         textArea = find("#textArea");
27         actualText = "специальный текст для тестов";
28         robot.clickOn(textArea).write(actualText);
29     }
30
31     @After
32     public void clearTextArea() {
33         textArea.clear();
34     }
35
36     @Test
37     public void integrationCipherEqualsTest() {
38         robot.clickOn("#encodeMenu").clickOn("#encodeMonoAlphabet")
39             .clickOn("#txtFieldDialog").write('4').clickOn("#okDialog");
40         robot.clickOn("#decodeMenu").clickOn("#decodeBitRevers")
41             .clickOn("#txtFieldDialog").write("32451").clickOn("#okDialog");
42         robot.clickOn("#encodeMenu").clickOn("#encodeBitRevers")
43             .clickOn("#txtFieldDialog").write("32451").clickOn("#okDialog");
44         robot.clickOn("#decodeMenu").clickOn("#decodeMonoAlphabet")
45             .clickOn("#txtFieldDialog").write('4').clickOn("#okDialog");
46         Assert.assertEquals(textArea.getText(), actualText);
47     }
48
49     @Test
50     public void integrationCipherNotEqualsTest() {
51         robot.clickOn("#encodeMenu").clickOn("#encodeMonoAlphabet")
52             .clickOn("#txtFieldDialog").write('4').clickOn("#okDialog");
53         robot.clickOn("#decodeMenu").clickOn("#decodeBitRevers")
54             .clickOn("#txtFieldDialog").write("32451").clickOn("#okDialog");

```

```

55     String testText = textArea.getText();
56     robot.clickOn("#decodeMenu").clickOn("#decodeBitRevers")
57         .clickOn("#txtFieldDialog").write("52341").clickOn("#okDialog");
58     Assert.assertNotEquals(textArea.getText(), testText);
59 }
60
61 @Test
62 public void integrationDeepCipherEqualsTest() {
63     robot.clickOn("#encodeMenu").clickOn("#encodeMonoAlphabet")
64         .clickOn("#txtFieldDialog").write("2").clickOn("#okDialog");
65     robot.clickOn("#encodeMenu").clickOn("#encodeBitRevers")
66         .clickOn("#txtFieldDialog").write("53241").clickOn("#okDialog");
67     Assert.assertNotEquals(textArea.getText(), actualText);
68     robot.clickOn("#decodeMenu").clickOn("#decodeMonoAlphabet")
69         .clickOn("#txtFieldDialog").write('3').clickOn("#okDialog");
70     robot.clickOn("#decodeMenu").clickOn("#decodeBitRevers")
71         .clickOn("#txtFieldDialog").write("24531").clickOn("#okDialog");
72     Assert.assertNotEquals(textArea.getText(), actualText);
73     robot.clickOn("#encodeMenu").clickOn("#encodeBitRevers")
74         .clickOn("#txtFieldDialog").write("24531").clickOn("#okDialog");
75     robot.clickOn("#encodeMenu").clickOn("#encodeMonoAlphabet")
76         .clickOn("#txtFieldDialog").write('3').clickOn("#okDialog");
77     Assert.assertNotEquals(textArea.getText(), actualText);
78     robot.clickOn("#decodeMenu").clickOn("#decodeBitRevers")
79         .clickOn("#txtFieldDialog").write("53241").clickOn("#okDialog");
80     robot.clickOn("#decodeMenu").clickOn("#decodeMonoAlphabet")
81         .clickOn("#txtFieldDialog").write('2').clickOn("#okDialog");
82     Assert.assertEquals(textArea.getText(), actualText);
83 }
84 }

```