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 — Болотин Д. И., Ивашин А.В.

Преподаватель к.т.н., ст. преп. — Воробьев К. А.

Рыбинск 2016

Содержание

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 пользователю требуется пройти авторицазию или регистрацию, после чего ему доступны 2 метода шифрования. Во время работы можно сохранить/загрузить текстовый файл.

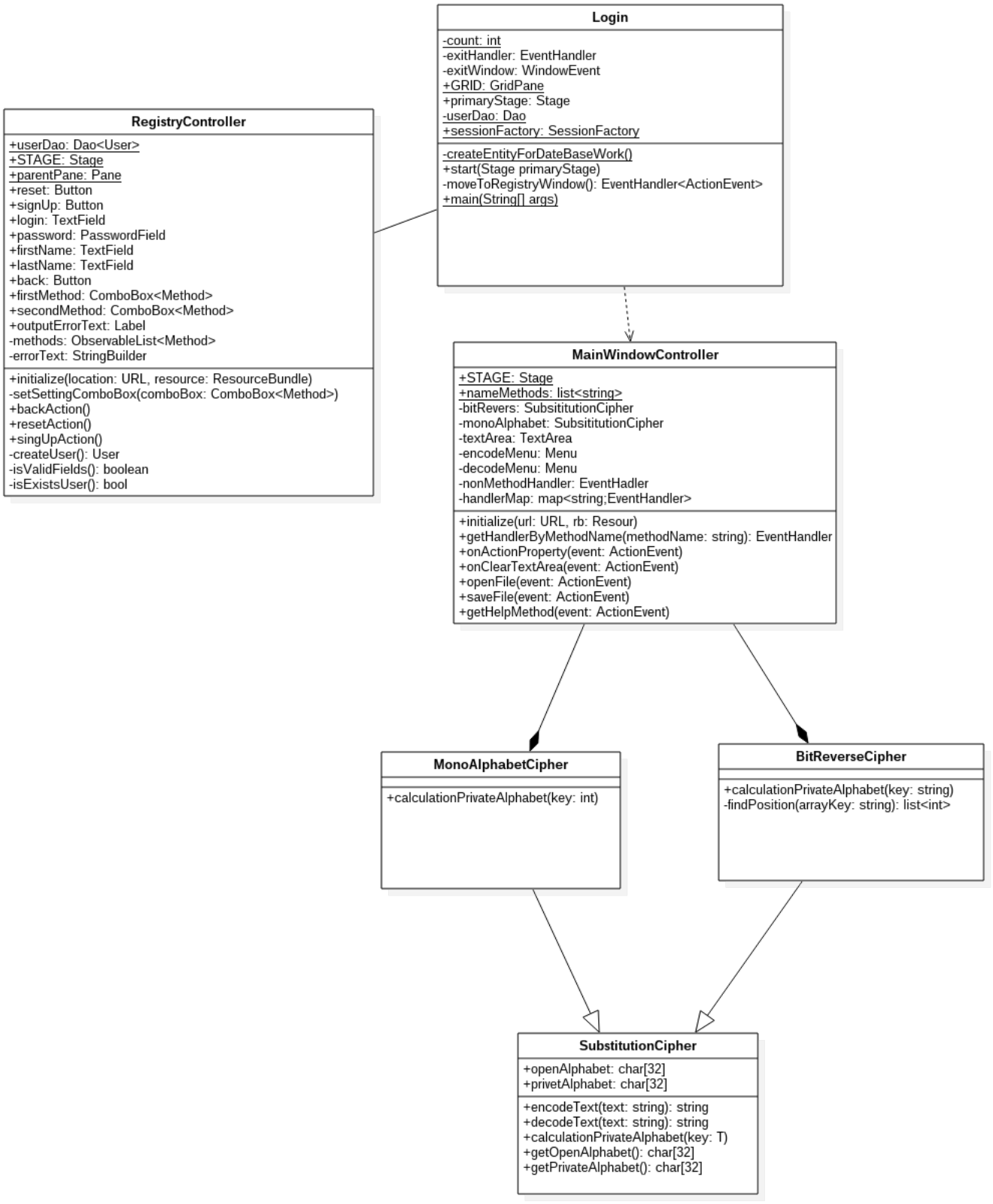

Рис. 1 – Диаграмма классов проекта

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

Алфавит текста состоит из строчных букв русского алфавита и пробела кроме букв 'ё' и 'й'.

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 2, на ней упущены имена методов и атрибутов.

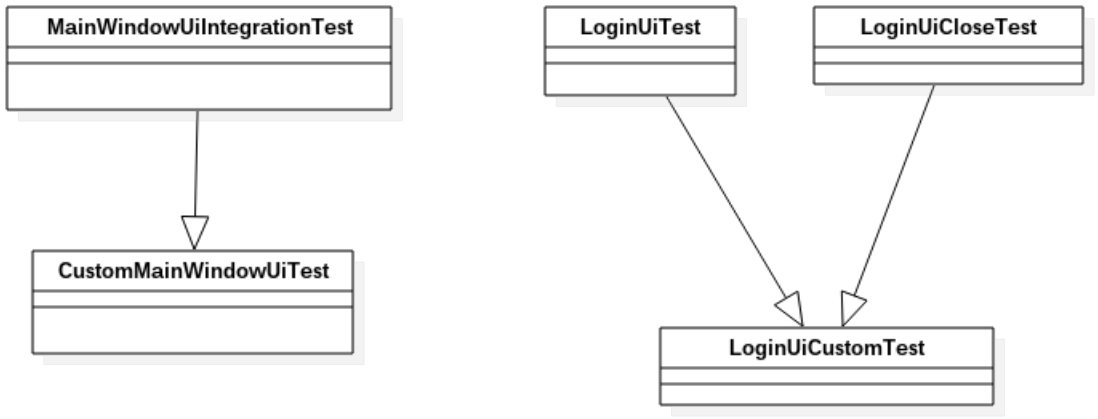

Рис. 2 – Диаграмма тестирующих классов

4. Тестирование взаимодействия класса Login

Это главный класс программы. Он отвечает за окно авторизации. Пользователь должен пройти авторизацию или зарегистрироваться. Если после пятой попытки не удалось авторизиваться, то программа закрывается.

4.1. Закрытие программы при неверной авторизации

С помощью TestFx моделируется попытка неудачной авторизации (Листинг 1).

Листинг 1 – Тестирование неверной авторизации

```
public class LoginUiCloseTest extends LoginUiCustomTest {
    @Test
    public void threeWrongAuthorizationTest() {
        for (int i = 0; i < 3; i++) {
            clickOn(userName).write("Aleksey");
            clickOn(password).write("tttt");
            clickOn(authorization).sleep(100);
            verifyThat(resultAuthorization, hasText("Не верный пароль"));
            assertEquals(resultAuthorization.getFill(), Color.FIREBRICK);
            userName.clear();
            password.clear();
        }
        assertEquals(stage.isShowing(), false);
        assertEquals(sessionFactory.isClosed(), true);
    }
}
```

4.2. Реакция программы при неверном пароле

В окне должна появиться надпись "Неверный пароль!"красного цвета. Этот и последующие тесты класса Login описаны в классе LoginUiTest (Листинг 2).

Листинг 2 – Тестирование ввода неверного пароля

```
    @Test
    public void wrongAuthorizationTest() {
        assertEquals(sessionFactory.isClosed(), false);
        clickOn(userName).write("Aleksey");
        clickOn(password).write("retdfgdfh");
        clickOn(authorization);
        verifyThat(resultAuthorization, hasText("Не верный пароль"));
        assertEquals(resultAuthorization.getFill(), Color.FIREBRICK);
    }
```

4.3. Реакция программы при не существующем логине

Должно появиться сообщение "Такого пользователя не существует"(Листинг 3).

Листинг 3 – Тестирование ввода неверного пароля

```
    @Test
```

```
    public void wrongAuthorizationTest() {
        assertEquals(sessionFactory.isClosed(), false);
        clickOn(userName).write("Aleksey");
        clickOn(password).write("retdfgdfh");
        clickOn(authorization);
        verifyThat(resultAuthorization, hasText("Не верный пароль"));
        assertEquals(resultAuthorization.getFill(), Color.FIREBRICK);
    }
```

4.4. Реакция программы при верной авторизации

При верной авторизации должно исчезнуть начальное окно и появиться основное окно программы (Листинг 4).

Листинг 4 – Тестирование правильной авторизации

```
    protected void openRegistryWindowAndTest() {
        clickOn(registry);
        assertEquals(gridPane.isDisable(), true);
        waitUntil("#registryPane", visible());
    }

    protected void closeRegistryWindowAndTest() {
        clickOn("#back");
        assertEquals(gridPane.isDisable(), false);
    }

    @Test
    public void openMainWindowTest() {
        assertEquals(sessionFactory.isClosed(), false);
        clickOn(userName).write("Aleksey");
        clickOn(password).write("yui");
        clickOn(authorization);
        verifyThat(resultAuthorization, hasText("Пароль верный"));
        assertEquals(resultAuthorization.getFill(), Color.GREEN);
        waitUntil("#AnchorPane", visible());
    }
```

4.5. Тестирование перехода к окну регистрации

При нажатии на кнопку “Регистация” , должно появится соответствующее окно, а окно для авторизации исчезнуть (Листинг 5).

Листинг 5 – Тестирование перехода к окну регистрации

```
    @Test
    public void openRegistryWindowTest() {
        openRegistryWindowAndTest();
        closeRegistryWindowAndTest();
    }
```

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 ключь одного из методов, то результат должен не совпасть с исходным. Далее проведём тест, при котором зашуфруем несколько раз обоими методами с разнами ключами и расшуфруем 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 = "специальныи 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

Этот класс отвечает за главное окно программы и 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 сомом деле), посредством моделирвания работы пользователя с gui. Оно представлено в листинге 7. Предыдущее тестирование направлено на проверку корректности классов шифрования и их взаимодействия, а такущее преимущественно на корректность работы MainWindow с данными классами.

Листинг 7 – класс MainWindowUiIntegrationTest

```
public class MainWindowUiIntegrationTest extends CustomMainWindowUiTest {
    private TextArea textArea;
    String actualText;

    @BeforeClass
    public static void mainWindowSettings() {
        MainWindowController.nameMethods = Arrays.asList(
                methodNames.get(0), methodNames.get(5)
        );
    }

    @Before
    public void findTextArea() {
        textArea = find("#textArea");
        actualText = "специальныи текст для тестов";
        robot.clickOn(textArea).write(actualText);
    }

    @After
    public void clearTextArea() {
        textArea.clear();
    }

    @Test
    public void integrationCipherEqualsTest() {
        robot.clickOn("#encodeMenu").clickOn("#encodeMonoAlphabet")
                .clickOn("#txtFieldDialog").write('4').clickOn("#okDialog");
        robot.clickOn("#decodeMenu").clickOn("#decodeBitRevers")
                .clickOn("#txtFieldDialog").write("32451").clickOn("#okDialog");
        robot.clickOn("#encodeMenu").clickOn("#encodeBitRevers")
                .clickOn("#txtFieldDialog").write("32451").clickOn("#okDialog");
        robot.clickOn("#decodeMenu").clickOn("#decodeMonoAlphabet")
                .clickOn("#txtFieldDialog").write('4').clickOn("#okDialog");
        Assert.assertEquals(textArea.getText(), actualText);
    }

    @Test
    public void integrationCipherNotEqualsTest() {
        robot.clickOn("#encodeMenu").clickOn("#encodeMonoAlphabet")
                .clickOn("#txtFieldDialog").write('4').clickOn("#okDialog");
        robot.clickOn("#decodeMenu").clickOn("#decodeBitRevers")
                .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  }
```

Выводы

В ходе работы с помошью библиотек JUnit и TestFX было разработано интеграционное тестироваие проекта. Тесты составлялись из рассчёта на то, что всякое непосредственное взаимодеёствие классов программы должно быть покрыто. В результате тестирования была обнаружена ошибка, не замеченная при модульном тестровании: в базовом классе SubstitutionCipher поле privetAlphabet (закрытый алфавит - правила сопоставления символов при шифровании) было объявлено как static, т.е. все наследники этого класса имели один закрытый алфавит на всех, актуальный только для класса, который последний обновил это поле. Такое поведение классов шифрования недопустимо, но не было замечено на предыдущем этапе. Других ошибок обнаружено не было.

Приложение 1. Исходные коды тестируемых классов

Листинг 8 – код модуля SubstitutionCipher.java

```
package encryptionMethods.base;

/**
 * Created by Алексей on 16.12.2016.
 */
public abstract class SubstitutionCipher<T> {
    // Откртый алфавит
    protected final char[] openAlphabet = new char[32];
    // Закрытый алфавит
    protected final char[] privateAlphabet = new char[32];

    public SubstitutionCipher() {
        createOpenAlphabet();
    }

    private void createOpenAlphabet() {
        openAlphabet[0] = '\u0020';
        for (int i = 0; i < 9; i++) {
            openAlphabet[i + 1] = (char) ('а' + i);
        }
        for (int i = 10; i < 32; ++i) {
            openAlphabet[i] = (char) ('а' + i);
        }
    }

    public abstract void calculationPrivateAlphabet(T key);

    /**
     * Закодировать текст
     *
     * @param originalText текст
     * @return закодированный текст
     */
    public String encodeText(String originalText) {
        originalText = originalText.replaceAll("\n", " ").toLowerCase();
        char[] text = originalText.toCharArray();
        char[] result = new char[text.length];
        int index;
        for (int i = 0; i < text.length; i++) {
            index = contains(openAlphabet, text[i]);
            if (index < 0) return "Недопустимый символ. Шифрование не возможно";
            result[i] = privateAlphabet[index];
        }
        return String.valueOf(result);
    }

    /**
     * Раскодировать текст
     *
     * @param originalText текст
     * @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

```java
package encryptionMethods.monoAlphabet;

import encryptionMethods.base.SubstitutionCipher;

/**
 * Класс для кодирования текста моноалфавитным методом Created by Алексей on
 * 25.03.2016.
 */
public class MonoAlphabetCipher extends SubstitutionCipher<Integer> {

    public MonoAlphabetCipher() {
        super();
    }

    /**
     * Создать закрытый алфавит
     *
     * @param key ключ смещения
     */
    @Override
    public void calculationPrivateAlphabet(Integer key) {
        for (int i = 0; i < 32; i++) {
            privateAlphabet[i] = openAlphabet[Math.floorMod(i + key, 32)];
        }
    }
}
```

Листинг 10 – код модуля BitReversCipher.java

```
package encryptionMethods.bitrevers;

import encryptionMethods.base.SubstitutionCipher;

/**
 * Класс для кодирования текста методом побитовой перестановки Created by
 * Алексей on 25.03.2016.
 */
public class BitReversCipher extends SubstitutionCipher<String> {
    public BitReversCipher() {
        super();
    }

    /**
     * Вычислить закрытый алфавит по заданому ключу
     *
     * @param key ключ
     */
    @Override
    public void calculationPrivateAlphabet(String key) {
        int[] arrayKey = findPosition(key);
        for (int i = 0; i < 32; i++) {
            StringBuilder stI = new StringBuilder(Integer.toBinaryString(i));
            for (int j = stI.length(); j < 5; j++) {
                stI.insert(0, "0");
            }
            StringBuilder charAlph = new StringBuilder();
            char[] mass = stI.toString().toCharArray();
            for (int anArrayKey : arrayKey) {
                charAlph.append(mass[anArrayKey]);
            }
            int exitI = Integer.parseInt(charAlph.toString(), 2);
            privateAlphabet[i] = openAlphabet[exitI];
        }
    }

    /**
     * Найти в каких позициях произошла перестановка
     *
     * @param arrayKey
     * @return
     */
    private int[] findPosition(String arrayKey) {
        int[] result = new int[arrayKey.length()];
        result[0] = arrayKey.indexOf("1");
        result[1] = arrayKey.indexOf("2");
        result[2] = arrayKey.indexOf("3");
        result[3] = arrayKey.indexOf("4");
        result[4] = arrayKey.indexOf("5");
        return result;
    }
}
```

Листинг 11 – код модуля MainWindowController.java

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package controller;

import encryptionMethods.base.SubstitutionCipher;
import encryptionMethods.bitrevers.BitReversCipher;
import encryptionMethods.monoAlphabet.MonoAlphabetCipher;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.stage.FileChooser;
import javafx.stage.Stage;
import javafx.stage.StageStyle;
import javafx.stage.WindowEvent;
import org.hibernate.SessionFactory;
import utils.FileWorker;
import utils.alert.ErrorAlert;
import utils.alert.InformAlert;

import java.io.File;
import java.io.IOException;
import java.net.URL;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.ResourceBundle;
import java.util.function.Consumer;
import java.util.function.Predicate;

import static utils.UtilFunctions.isNullString;

/**
 * FXML Controller class
 *
 * @author Алексей
 */
public class MainWindowController implements Initializable {

    public static SessionFactory sessionFactory;
    public static Stage stage;
    public static List<String> nameMethods;
    public MenuItem decodeMonoAlphabet;

    @FXML
    private TextArea textArea;
    @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 ситуции 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");
```

```
            buttons.setAlignment(Pos.CENTER);
            Button buttonOk = new Button("Ok");
            buttonOk.setId("okDialog");
            buttonOk.setOnAction((ActionEvent evt) -> {
                bitRevers.calculationPrivateAlphabet(txtField.getText());
                dialog.close();
                String codeText = bitRevers
                        .encodeText(textArea.getText().replaceAll("\n", " "));
                textArea.clear();
                textArea.setText(codeText);
            });
            Button buttonEx = new Button("Cancel");
            buttonEx.setId("cancelDialog");
            buttonEx.setOnAction(evt -> {
                dialog.close();
            });
            buttons.getChildren().addAll(buttonOk, buttonEx);
            Label label = new Label("Введите ключ");
            label.setId("labelDialog");
            box.getChildren().addAll(label, txtField, buttons);
            Scene scene = new Scene(box, 300, 100);
            dialog.setScene(scene);
            dialog.show();
        });

        handlerMap.put("decodeMonoAlphabet", event -> {
            Stage dialog = new Stage();
            dialog.initStyle(StageStyle.UTILITY);
            dialog.setTitle("Окно ввода ключа");
            VBox box = new VBox();
            box.setAlignment(Pos.CENTER);
            HBox buttons = new HBox();
            TextField txtField = new TextField();
            txtField.setId("txtFieldDialog");
            buttons.setAlignment(Pos.CENTER);
            Button buttonOk = new Button("Ok");
            buttonOk.setId("okDialog");
            buttonOk.setOnAction((ActionEvent evt) -> {
                int key = Integer.valueOf(txtField.getText());
                monoAlphabet.calculationPrivateAlphabet(key);
                dialog.close();
                String decodeText = monoAlphabet
                        .decodeText(textArea.getText().replaceAll("\n", " "));
                textArea.clear();
                textArea.setText(decodeText);
            });
            Button buttonEx = new Button("Cancel");
            buttonEx.setId("cancelDialog");
            buttonEx.setOnAction(evt -> {
                dialog.close();
            });
            buttons.getChildren().addAll(buttonOk, buttonEx);
            Label label = new Label("Введите ключ");
            label.setId("labelDialog");
            box.getChildren().addAll(label, txtField, buttons);
```

```
            Scene scene = new Scene(box, 300, 100);
            dialog.setScene(scene);
            dialog.show();
        });

        handlerMap.put("decodeBitRevers", event -> {
            Stage dialog = new Stage();
            dialog.initStyle(StageStyle.UTILITY);
            dialog.setTitle("Окно ввода ключа");
            VBox box = new VBox();
            box.setAlignment(Pos.CENTER);
            HBox buttons = new HBox();
            TextField txtField = new TextField();
            txtField.setId("txtFieldDialog");
            buttons.setAlignment(Pos.CENTER);
            Button buttonOk = new Button("Ok");
            buttonOk.setId("okDialog");
            buttonOk.setOnAction((ActionEvent evt) -> {
                bitRevers.calculationPrivateAlphabet(txtField.getText());
                dialog.close();
                String decodeText = bitRevers
                        .decodeText(textArea.getText().replaceAll("\n", " "));
                textArea.clear();
                textArea.setText(decodeText);
            });
            Button buttonEx = new Button("Cancel");
            buttonEx.setId("cancelDialog");
            buttonEx.setOnAction(evt -> {
                dialog.close();
            });
            buttons.getChildren().addAll(buttonOk, buttonEx);
            Label label = new Label("Введите ключ");
            label.setId("labelDialog");
            box.getChildren().addAll(label, txtField, buttons);
            Scene scene = new Scene(box, 300, 100);
            dialog.setScene(scene);
            dialog.show();
        });
        //</editor-fold>
    }

    private void initializeStage() {
        if (stage == null) {
            stage = new Stage();
            stage.setTitle("Добро пожаловать, Неизвестный");
        }
        ;
        stage.setOnCloseRequest(we -> sessionFactory.close());
    }


    private EventHandler<ActionEvent> getHandlerByMethodName(String methodName) {
        if (handlerMap.containsKey(methodName)) {
            return handlerMap.get(methodName);
        }
```

```
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(
```

```
                "При Моноалфавитной замене каждой букве открытого текста ставится соотве
                        + "закрытого текста из этого же алфавита. y_i=(K_1 X_i+K_2 ) mod
                        + "K_1 и K_2– это константы. K_1=1,K_2– это смещение \n"
                        + "сиволов закрытого алфавита относительно открытого алфавита, \
                        + "если K_1=1,K_2=3, то это так называемый код Цезаря. X_i– это
                        + "i символа открытого алфавита Y_i– это код i символа закрытого
        );
        txtAreaHelp.appendText("Побитовая перестановка\n");
        txtAreaHelp.appendText(
                "Несколько более сложной является побитовая перестановка, при которой в
                        + "соответствии с вектором перестановки изменяются позиции разря
                        + "символов открытого текста, обычно берутся ASCII коды. \n"
        );

        HBox buttons = new HBox();
        buttons.setAlignment(Pos.CENTER);
        Button buttonOk = new Button("Ok");
        buttonOk.setOnAction((ActionEvent evt) -> {
            dialog.close();
        });

        buttons.getChildren().addAll(buttonOk);
        box.getChildren().addAll(txtAreaHelp, buttons);
        Scene scene = new Scene(box, 600, 200);
        dialog.setScene(scene);
        dialog.show();
    }

}
```

Листинг 12 – код модуля Login.java

```
package controller;

import database.dao.Dao;
import database.entity.User;
import database.service.DaoFactory;
import database.service.DataBaseService;
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXMLLoader;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.HBox;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
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;
```

```
            }
            actionTarget.setFill(Color.GREEN);
            actionTarget.setText("Пароль верный");
            Parent root = null;
            Stage mainWindowStage = new Stage();
            try {
                MainWindowController.stage = mainWindowStage;
                MainWindowController.sessionFactory = sessionFactory;
                MainWindowController.nameMethods = user.getMethods();
                String firstName = isNullString(user.getFirstName()) ? "Неизвестный" : u
                mainWindowStage.setTitle("Добро пожаловать, " + firstName);

                root = FXMLLoader.load(this.getClass()
                        .getResource("/fxml/MainWindow.fxml")
                );

                primaryStage.close(); // закрытие формы авторизации
                Scene scene = new Scene(root, 400, 400);

                mainWindowStage.setScene(scene);
                mainWindowStage.show();
            } catch (IOException ex) {
                Logger.getLogger(Login.class.getName()).log(Level.SEVERE, null, ex);
            }
        });

        Scene scene = new Scene(GRID, 500, 275);
        primaryStage.setScene(scene);

        primaryStage.show();
    }

    private EventHandler<ActionEvent> moveToRegistryWindow() {
        return event -> {
            Parent root = null;
            try {
                GRID.setDisable(true);
                root = FXMLLoader.load(this.getClass()
                        .getResource("/fxml/Registry.fxml"));
                Scene scene = new Scene(root, 600.0D, 400.0D);
                Stage registryStage = new Stage();
                registryStage.setScene(scene);
                RegistryController.STAGE = registryStage;
                RegistryController.parentPane = GRID;
                RegistryController.userDao = userDao;
                registryStage.show();
            } catch (IOException e) {
                e.printStackTrace();
            }
        };
    }

    /**
     * @param args the command line arguments
     */
```

```
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

```
package ui.custom;

import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import org.loadui.testfx.GuiTest;
import org.testfx.api.FxRobot;

import java.io.IOException;
import java.util.Arrays;
import java.util.List;

/**
 * Created by Алексей on 25.12.2016.
 */
public class CustomMainWindowUiTest extends GuiTest {
    protected static FxRobot robot;
    protected static List<String> methodNames;

    static {
        robot = new FxRobot();

        methodNames = Arrays.asList(
                "MonoAlphabet", "HomophonyReplacement", "PolyalphabeticReplacement",
                "PoligrammnayaReplacement", "VerticalPermutation", "BitRevers",
                "VizhinerMethod", "XOR"
        );
    }

    @Override
    protected Parent getRootNode() {
        Parent parent = null;
        try {
            parent = FXMLLoader.load(getClass().getResource("/fxml/MainWindow.fxml"));
            return parent;
        } catch (IOException ex) {
            // TODO ...
        }
        return parent;
    }
}
```

Листинг 14 – код модуля LoginUiCustomTest.java

```
package ui.custom;

import controller.Login;
import javafx.scene.control.Button;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import org.testfx.framework.junit.ApplicationTest;

import static controller.Login.GRID;
import static org.junit.Assert.assertEquals;
import static org.loadui.testfx.GuiTest.find;
import static org.loadui.testfx.GuiTest.waitUntil;
import static org.loadui.testfx.controls.impl.VisibleNodesMatcher.visible;

/**
 * Created by Алексей on 25.12.2016.
 */
public abstract class LoginUiCustomTest extends ApplicationTest {
    protected Text resultAuthorization;
    protected Button authorization;
    protected Button registry;
    protected TextField userName;
    protected PasswordField password;
    protected Stage stage;
    protected GridPane gridPane;

    @Override
    public void start(Stage stage) throws Exception {
        this.stage = stage;
        new Login().start(stage);
        gridPane = GRID;
        resultAuthorization = find("#resultAuthorization");
        authorization = find("#authorization");
        userName = find("#login");
        password = find("#password");
        registry = find("#checkIn");
    }

    protected void openRegistryWindowAndTest() {
        clickOn(registry);
        assertEquals(gridPane.isDisable(), true);
        waitUntil("#registryPane", visible());
    }

    protected void closeRegistryWindowAndTest() {
        clickOn("#back");
        assertEquals(gridPane.isDisable(), false);
    }
}
```

Листинг 15 – код модуля LoginUiCloseTest.java

```
package ui.login;

import javafx.scene.paint.Color;
import org.junit.Test;
import ui.custom.LoginUiCustomTest;

import static controller.Login.sessionFactory;
import static org.junit.Assert.assertEquals;
import static org.testfx.api.FxAssert.verifyThat;
import static org.testfx.matcher.base.NodeMatchers.hasText;

/**
 * Created by Алексей on 25.12.2016.
 */
public class LoginUiCloseTest extends LoginUiCustomTest {
    @Test
    public void threeWrongAuthorizationTest() {
        for (int i = 0; i < 3; i++) {
            clickOn(userName).write("Aleksey");
            clickOn(password).write("tttt");
            clickOn(authorization).sleep(100);
            verifyThat(resultAuthorization, hasText("Не верный пароль"));
            assertEquals(resultAuthorization.getFill(), Color.FIREBRICK);
            userName.clear();
            password.clear();
        }
        assertEquals(stage.isShowing(), false);
        assertEquals(sessionFactory.isClosed(), true);
    }
}
```

Листинг 16 – код модуля LoginUiTest.java

```
package ui.login;

import javafx.scene.paint.Color;
import org.junit.Before;
import org.junit.Test;
import ui.custom.LoginUiCustomTest;

import static controller.Login.sessionFactory;
import static org.junit.Assert.assertEquals;
import static org.loadui.testfx.GuiTest.waitUntil;
import static org.loadui.testfx.controls.impl.VisibleNodesMatcher.visible;
import static org.testfx.api.FxAssert.verifyThat;
import static org.testfx.matcher.base.NodeMatchers.hasText;

/**
 * Created by Алексей on 20.12.2016.
 */
public class LoginUiTest extends LoginUiCustomTest {

    @Before
    public void checkOpenSession() {
        assertEquals(sessionFactory.isClosed(), false);
    }

    @Test
    public void openMainWindowTest() {
        assertEquals(sessionFactory.isClosed(), false);
        clickOn(userName).write("Aleksey");
        clickOn(password).write("yui");
        clickOn(authorization);
        verifyThat(resultAuthorization, hasText("Пароль верный"));
        assertEquals(resultAuthorization.getFill(), Color.GREEN);
        waitUntil("#AnchorPane", visible());
    }

    @Test
    public void wrongAuthorizationTest() {
        assertEquals(sessionFactory.isClosed(), false);
        clickOn(userName).write("Aleksey");
        clickOn(password).write("retdfgdfh");
        clickOn(authorization);
        verifyThat(resultAuthorization, hasText("Не верный пароль"));
        assertEquals(resultAuthorization.getFill(), Color.FIREBRICK);
    }

    @Test
    public void openRegistryWindowTest() {
        openRegistryWindowAndTest();
        closeRegistryWindowAndTest();
    }

    @Test
    public void thisUserDoesntExistTest() {
        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

```
package unit.integration;

import encryptionMethods.base.SubstitutionCipher;
import encryptionMethods.bitrevers.BitReversCipher;
import encryptionMethods.monoAlphabet.MonoAlphabetCipher;
import org.junit.Assert;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;

/**
 * Created by Алексей on 22.12.2016.
 */
public class IntegrationCipherTest {
    private static SubstitutionCipher<Integer> monoAlphabet;
    private static SubstitutionCipher<String> bitRevers;
    private String actualText;

    @BeforeClass
    public static void test() {
        monoAlphabet = new MonoAlphabetCipher();
        bitRevers = new BitReversCipher();
    }

    @Before
    public void createCipherClass() {
        actualText = "специальныи текст для тестов";
    }

    @Test
    public void integrationCipherEqualsTest() {
        monoAlphabet.calculationPrivateAlphabet(4);
        bitRevers.calculationPrivateAlphabet("32451");
        String expectedText = monoAlphabet.encodeText(actualText);
        expectedText = bitRevers.decodeText(expectedText);
        expectedText = bitRevers.encodeText(expectedText);
        expectedText = monoAlphabet.decodeText(expectedText);
        Assert.assertEquals(expectedText, actualText);
    }

    @Test
    public void  integrationCipherNotEqualsTest() {
        monoAlphabet.calculationPrivateAlphabet(4);
        bitRevers.calculationPrivateAlphabet("32451");
        String testText = monoAlphabet.encodeText(actualText);
        testText = bitRevers.decodeText(testText);
        bitRevers.calculationPrivateAlphabet("52341");
        String expectedText = bitRevers.encodeText(testText);
        Assert.assertNotEquals(expectedText, testText);
    }

    @Test
    public void  integrationDeepCipherEqualsTest() {
        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

```
package ui.mainwindow;

import controller.MainWindowController;
import javafx.scene.control.TextArea;
import org.junit.*;
import ui.custom.CustomMainWindowUiTest;

import java.util.Arrays;

/**
 * Created by Алексей on 25.12.2016.
 */
public class MainWindowUiIntegrationTest extends CustomMainWindowUiTest {
    private TextArea textArea;
    String actualText;

    @BeforeClass
    public static void mainWindowSettings() {
        MainWindowController.nameMethods = Arrays.asList(
                methodNames.get(0), methodNames.get(5)
        );
    }

    @Before
    public void findTextArea() {
        textArea = find("#textArea");
        actualText = "специальныи текст для тестов";
        robot.clickOn(textArea).write(actualText);
    }

    @After
    public void clearTextArea() {
        textArea.clear();
    }

    @Test
    public void integrationCipherEqualsTest() {
        robot.clickOn("#encodeMenu").clickOn("#encodeMonoAlphabet")
                .clickOn("#txtFieldDialog").write('4').clickOn("#okDialog");
        robot.clickOn("#decodeMenu").clickOn("#decodeBitRevers")
                .clickOn("#txtFieldDialog").write("32451").clickOn("#okDialog");
        robot.clickOn("#encodeMenu").clickOn("#encodeBitRevers")
                .clickOn("#txtFieldDialog").write("32451").clickOn("#okDialog");
        robot.clickOn("#decodeMenu").clickOn("#decodeMonoAlphabet")
                .clickOn("#txtFieldDialog").write('4').clickOn("#okDialog");
        Assert.assertEquals(textArea.getText(), actualText);
    }

    @Test
    public void integrationCipherNotEqualsTest() {
        robot.clickOn("#encodeMenu").clickOn("#encodeMonoAlphabet")
                .clickOn("#txtFieldDialog").write('4').clickOn("#okDialog");
        robot.clickOn("#decodeMenu").clickOn("#decodeBitRevers")
                .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  }
```