

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

1. Общее описание тестируемой системы	3
2. Общее описание тестируемых классов	5
2.1. Ограничения для шифруемого/расшифруемого текста	6
3. Общее описание тестирующих классов	7
4. Тестирование взаимодействия класса Login	8
4.1. Закрытие программы при неверной авторизации	8
4.2. Реакция программы при неверном пароле	8
4.3. Реакция программы при не существующем логине	8
4.4. Реакция программы при верной авторизации	9
4.5. Тестирование перехода к окну регистрации	9
5. Тестирование взаимодействия классов MonoAlphabetCipher и BitReverseCipher	10
6. Тестирование взаимодействия класса MainWindow	12
7. Выводы	14
8. Приложения	15

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).

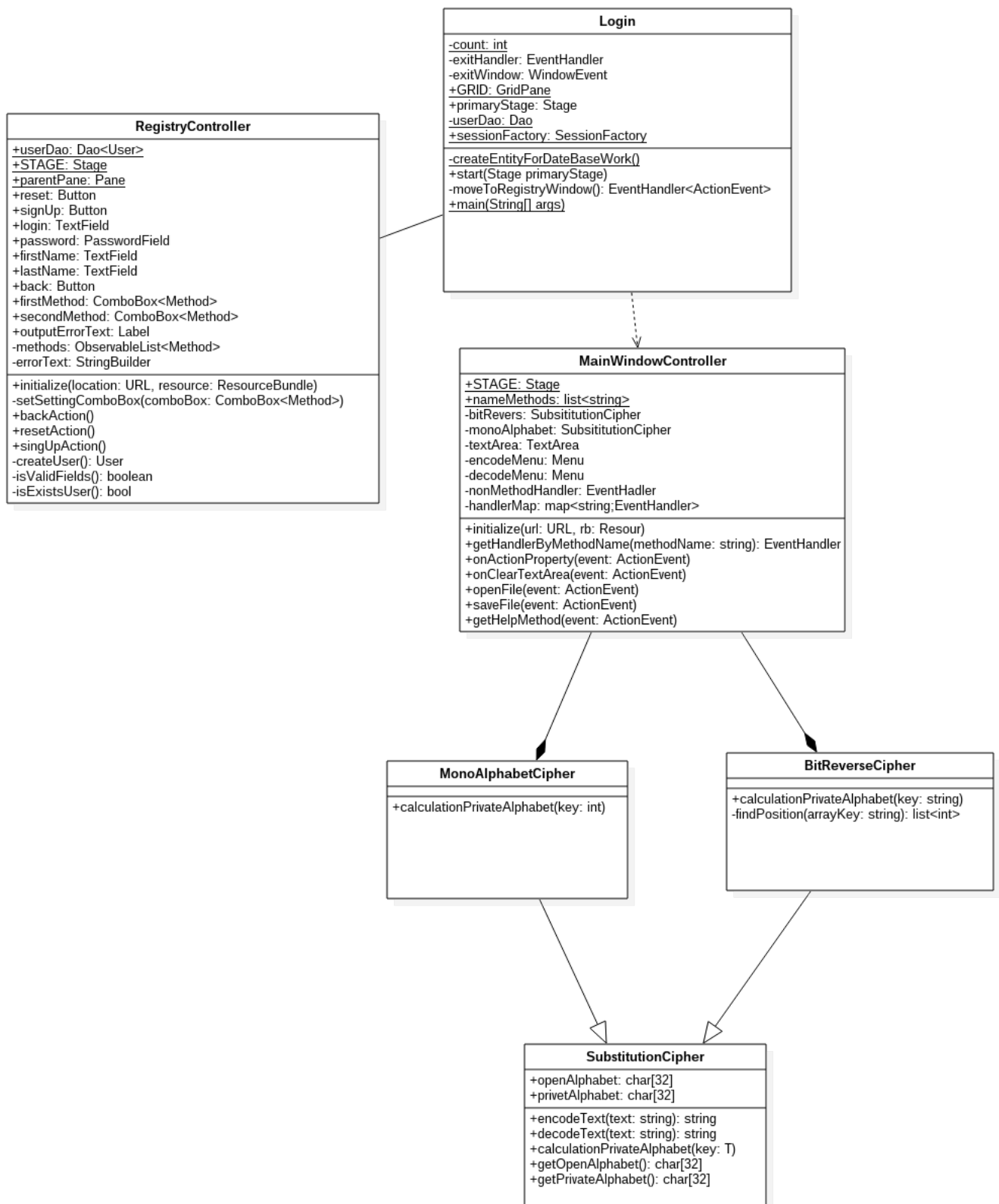


Рис. 1 – Диаграмма классов проекта

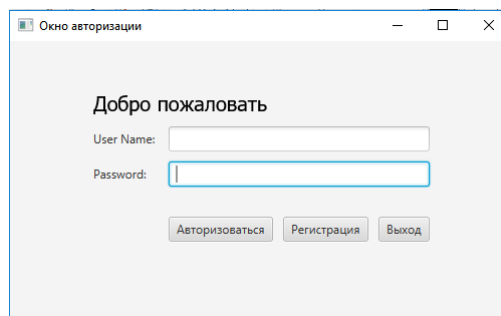


Рис. 2 – Окно авторизации

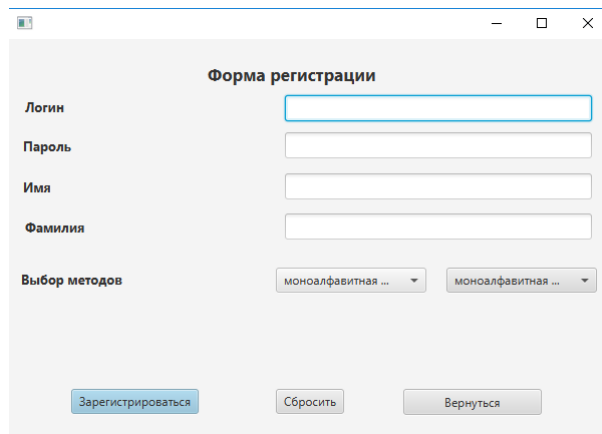


Рис. 3 – Окно регистрации

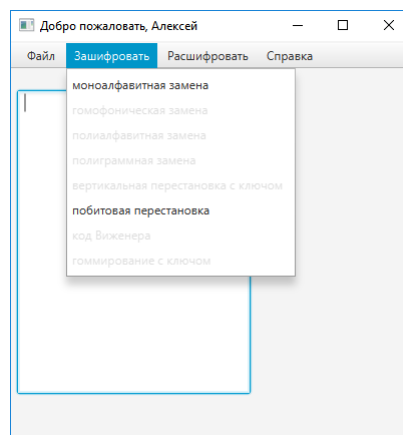


Рис. 4 – Главное окно программы

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

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 5, на ней упущены имена методов и атрибутов.

Рис. 5 – Тут будет диаграмма тестирующих классов, честно:)

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 авторизоваться, то программа закрывается.

4.1. Заккрытие программы при неверной авторизации

С помощью TestFx моделируется попытка неудачной авторизации (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

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

Листинг 2 – Тестирование ввода неверного пароля

```
1 @Test
2 public void wrongAuthorizationTest () {
3     assertEquals(sessionFactory.isClosed(), false);
4     clickOn(userName).write("Aleksey");
5     clickOn(password).write("retdfgdfh");
6     clickOn(authorization);
7     verifyThat(resultAuthorization, hasText("Не верный пароль"));
8     assertEquals(resultAuthorization.getFill(), Color.FIREBRICK);
9 }
```

4.3. Реакция программы при не существующем логине

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

Листинг 3 – Тестирование ввода неверного пароля

```
1 @Test
```



```

2      public void wrongAuthorizationTest () {
3          assertEquals(sessionFactory.isClosed(), false);
4          clickOn(userName).write("Aleksey");
5          clickOn(password).write("retdfgdfh");
6          clickOn(authorization);
7          verifyThat(resultAuthorization, hasText("Не верный пароль"));
8          assertEquals(resultAuthorization.getFill(), Color.FIREBRICK);
9      }

```

4.4. Реакция программы при верной авторизации

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

Листинг 4 – Тестирование ввода неверного пароля

```

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 ключь 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

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 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 работы с помощью библиотек JUnit и TestFX было разработано интеграционное тестирование проекта. 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 `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.

8. Приложения

Это старые приложения, я их обновлю.

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 static char[] getOpenAlphabet() {
75          return openAlphabet;
76      }
77
78      public static 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 MonoAlphabetCipherTest.java

```

1 import encryptionMethods.base.SubstitutionCipher;
2 import encryptionMethods.monoAlphabet.MonoAlphabetCipher;
3 import org.junit.After;
4 import org.junit.Assert;
5 import org.junit.Before;
6 import org.junit.Test;
7
8 import static org.junit.Assert.assertArrayEquals;
9 import static org.junit.Assert.assertEquals;
10
11 /**
12  * Created by Алексей on 19.12.2016.
13  */
14 public class MonoAlphabetCipherTest {
15     SubstitutionCipher monoAlphabetCipher;
16
17     @Before
18     public void beforeTest() {
19         monoAlphabetCipher = new MonoAlphabetCipher();
20     }
21
22     @After
23     public void afterTest() {
24         monoAlphabetCipher = null;
25     }
26
27     @Test(expected = NullPointerException.class)
28     public void testNullCalculationPrivateAlphabet() {
29         monoAlphabetCipher.calculationPrivateAlphabet(null);
30     }
31
32     @Test(expected = ClassCastException.class)
33     public void testWrongArgumentCalculationPrivateAlphabet() {
34         monoAlphabetCipher.calculationPrivateAlphabet("test");
35     }
36
37     @Test
38     public void testCalculationPrivateAlphabet() {
39         char[] actuals = {' ', 'а', 'б', 'в', 'г', 'д', 'е', 'ж', 'з', 'и', 'к', 'л', 'м', 'н', 'о', 'п', 'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ъ', 'ы', 'ь', 'э', 'ю', 'я'};
40         monoAlphabetCipher.calculationPrivateAlphabet(0);
41         assertArrayEquals(monoAlphabetCipher.getPrivateAlphabet(), actuals);
42         assertArrayEquals(monoAlphabetCipher.getOpenAlphabet(), actuals);
43         assertArrayEquals(monoAlphabetCipher.getOpenAlphabet(), monoAlphabetCipher.getOpenAlphabet());
44     }
45
46     @Test
47     public void testCalculationPrivateAlphabetCaesarCipher() {
48         char[] actuals = {'в', 'г', 'д', 'е', 'ж', 'з', 'и', 'к', 'л', 'м', 'н', 'о', 'п', 'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ъ', 'ы', 'ь', 'э', 'ю', 'я'};
49         monoAlphabetCipher.calculationPrivateAlphabet(3);
50         assertArrayEquals(monoAlphabetCipher.getPrivateAlphabet(), actuals);
51     }
52
53     @Test
54     public void testEncodeText() {

```

```

55     monoAlphabetCipher.calculationPrivateAlphabet(3);
56     assertEquals(monoAlphabetCipher.encodeText("тест"), "хифх");
57 }
58
59 @Test
60 public void testEncodeAndDecode() {
61     monoAlphabetCipher.calculationPrivateAlphabet(30);
62     String encode = monoAlphabetCipher.encodeText("тест");
63     assertEquals("тест", monoAlphabetCipher.decodeText(encode));
64 }
65
66 @Test
67 public void testWrongCharInEncodeAndDecodeText() {
68     monoAlphabetCipher.calculationPrivateAlphabet(1);
69     Assert.assertEquals(monoAlphabetCipher.encodeText("tttt"), "Недопустимый символ. Превышен лимит");
70     Assert.assertEquals(monoAlphabetCipher.encodeText("й"), "Недопустимый символ. Превышен лимит");
71     Assert.assertEquals(monoAlphabetCipher.encodeText("ё"), "Недопустимый символ. Превышен лимит");
72     Assert.assertEquals(monoAlphabetCipher.encodeText("-"), "Недопустимый символ. Превышен лимит");
73     Assert.assertEquals(monoAlphabetCipher.encodeText("+"), "Недопустимый символ. Превышен лимит");
74     Assert.assertEquals(monoAlphabetCipher.decodeText("tttt"), "Недопустимый символ. Превышен лимит");
75     Assert.assertEquals(monoAlphabetCipher.decodeText("й"), "Недопустимый символ. Превышен лимит");
76     Assert.assertEquals(monoAlphabetCipher.decodeText("ё"), "Недопустимый символ. Превышен лимит");
77     Assert.assertEquals(monoAlphabetCipher.decodeText("-"), "Недопустимый символ. Превышен лимит");
78     Assert.assertEquals(monoAlphabetCipher.decodeText("+"), "Недопустимый символ. Превышен лимит");
79 }
80 }

```

Листинг 12 – код модуля BitReversCipherTest.java

```

1 import encryptionMethods.base.SubstitutionCipher;
2 import encryptionMethods.bitrevers.BitReversCipher;
3 import org.junit.After;
4 import org.junit.Assert;
5 import org.junit.Before;
6 import org.junit.Test;
7
8 import static org.junit.Assert.assertArrayEquals;
9 import static org.junit.Assert.assertEquals;
10
11 /**
12  * Created by Алексей on 19.12.2016.
13  */
14 public class BitReversCipherTest {
15     SubstitutionCipher bitReversCipher;
16
17     @Before
18     public void beforeTest() {
19         bitReversCipher = new BitReversCipher();
20     }
21
22     @After
23     public void afterTest() {
24         bitReversCipher = null;
25     }
26
27     @Test
28     public void testCalculationPrivateAlphabetEqualsOpenAlphabet() {
29         char[] actuals = {' ', 'а', 'б', 'в', 'г', 'д', 'е', 'ж', 'з', 'и', 'к', 'л', 'м', 'н', 'о', 'п', 'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ъ', 'ы', 'ь', 'э', 'ю', 'я'};
30         bitReversCipher.calculationPrivateAlphabet("12345");
31         assertArrayEquals(bitReversCipher.getPrivateAlphabet(), actuals);
32         assertArrayEquals(bitReversCipher.getOpenAlphabet(), actuals);
33         assertArrayEquals(bitReversCipher.getOpenAlphabet(), bitReversCipher.getOpenAlphabet());
34     }
35
36     @Test
37     public void testCalculationPrivateAlphabet() {
38         char[] actuals = {' ', 'п', 'з', 'ш', 'г', 'ф', 'м', 'ь', 'б', 'т', 'к', 'ъ', 'е', 'и', 'а', 'о', 'н', 'с', 'д', 'л', 'у', 'х', 'ч', 'щ', 'ц', 'ж', 'я', 'ы', 'ь', 'э', 'ю'};
39         bitReversCipher.calculationPrivateAlphabet("54321");
40         assertArrayEquals(bitReversCipher.getPrivateAlphabet(), actuals);
41     }
42
43     @Test
44     public void testFirstElementPrivateAlphabetAtAnyKey() {
45         bitReversCipher.calculationPrivateAlphabet("54321");
46         assertEquals(bitReversCipher.getPrivateAlphabet()[0], ' ');
47         bitReversCipher.calculationPrivateAlphabet("53241");
48         assertEquals(bitReversCipher.getPrivateAlphabet()[0], ' ');
49         bitReversCipher.calculationPrivateAlphabet("13425");
50         assertEquals(bitReversCipher.getPrivateAlphabet()[0], ' ');
51         bitReversCipher.calculationPrivateAlphabet("31245");
52         assertEquals(bitReversCipher.getPrivateAlphabet()[0], ' ');
53     }
54 }

```

```

55  @Test
56  public void testEncode() {
57      bitReversCipher.calculationPrivateAlphabet("54321");
58      assertEquals(bitReversCipher.encodeText("тест"), "имси");
59  }
60
61  @Test
62  public void testEncodeDecode() {
63      bitReversCipher.calculationPrivateAlphabet("54321");
64      String encode = bitReversCipher.encodeText("тест");
65      assertEquals(bitReversCipher.decodeText(encode), "тест");
66  }
67
68  @Test(expected = IndexOutOfBoundsException.class)
69  public void testWrongValueForCalculationPrivateAlphabet() {
70      bitReversCipher.calculationPrivateAlphabet("00000");
71  }
72
73  @Test(expected = ClassCastException.class)
74  public void testWrongArgumentForCalculationPrivateAlphabet() {
75      bitReversCipher.calculationPrivateAlphabet(1213234);
76  }
77
78  @Test
79  public void testWrongCharInEncodeAndDecodeText() {
80      bitReversCipher.calculationPrivateAlphabet("54321");
81      Assert.assertEquals(bitReversCipher.encodeText("tttt"), "Недопустимый символ. Шифр");
82      Assert.assertEquals(bitReversCipher.encodeText("й"), "Недопустимый символ. Шифр");
83      Assert.assertEquals(bitReversCipher.encodeText("ё"), "Недопустимый символ. Шифр");
84      Assert.assertEquals(bitReversCipher.encodeText("-"), "Недопустимый символ. Шифр");
85      Assert.assertEquals(bitReversCipher.encodeText("+"), "Недопустимый символ. Шифр");
86      Assert.assertEquals(bitReversCipher.decodeText("tttt"), "Недопустимый символ. Ра");
87      Assert.assertEquals(bitReversCipher.decodeText("й"), "Недопустимый символ. Расши");
88      Assert.assertEquals(bitReversCipher.decodeText("ё"), "Недопустимый символ. Расши");
89      Assert.assertEquals(bitReversCipher.decodeText("-"), "Недопустимый символ. Расши");
90      Assert.assertEquals(bitReversCipher.decodeText("+"), "Недопустимый символ. Расши");
91  }
92 }

```