ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

СОГЛАСОВАНО
Доцент департамента
Программной инженерии
факультета компьютерных наук, к.т.н.

_____ /Ахметсафина Р. З.
«__» _____ 2015 г.

УТВЕРЖДЕНО
Академический руководитель
образовательной программы
«Программная инженерия»

_____ /Шилов В. В.
«__» _____ 2015 г.

АНИМАТОР КОДИРОВАНИЯ И ДЕКОДИРОВАНИЯ КОДОВ РИДА-МАЛЛЕРА

Текст программы

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.503200-01 12 01-1-ЛУ

Исполнитель:
студент группы 301 ПИ
_____ /Наседкин А. В.
«__» _____ 2015 г.

2015

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата
RU.17701729.503200-01 12 01-1-ЛУ				

УТВЕРЖДЕНО
RU.17701729.503200-01 12 01-1-ЛУ

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата
RU.17701729.503200-01 12 01-1				

АНИМАТОР КОДИРОВАНИЯ И ДЕКОДИРОВАНИЯ КОДОВ РИДА-МАЛЛЕРА

Текст программы

RU.17701729.503200-01 12 01-1

Листов 48

2015

СОДЕРЖАНИЕ

1. ТЕКСТ ПРОГРАММЫ 3
1.1. Класс BinaryFiniteField.java 3
1.2. Класс BitMatrix.java 6
1.3. Класс RMMatrix.java 9
1.4. Класс RMCode.java 13
1.5. Класс TransmitChannel.java 17
1.6. Класс Util.java 18
1.7. Класс RMView.java 21
1.8. Класс RMCodeSystem.java 44

\

1. ТЕКСТ ПРОГРАММЫ

Программа состоит из 8 классов.

Текст программы на исходном языке находится в архиве program_text.rar в директории doc на информационном носителе типа компакт-диск в связи с большим количеством строк кода.

1.1. Класс BinaryFiniteField.java

```
package model;

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

public class BinaryFiniteField {

   /**
    * Returns sum of two values in finite field F_2
    *
    * @param a first value
    * @param b second value
    * @return sum of two values in finite field F_2
    */
   public static boolean add(boolean a, boolean b) {
      return a ^ b;
   }

   /**
    * Returns sum of two vectors in finite field F_2
    *
    * @param a first vector
    * @param b second vector
    * @return sum of two vectors in finite field F_2
    */
   public static List<Boolean> add(List<Boolean> a, List<Boolean> b) {
      if (a.size() != b.size()) {
         throw new IllegalArgumentException("Both vectors must be same size");
      }

      List<Boolean> result = new ArrayList<>(a.size());
      for (int i = 0; i < a.size(); i ++) {
         result.add(add(a.get(i), b.get(i)));
      }

      return result;
   }
```

```
/**
 * Returns multiplication of two values in finite field F_2
 *
 * @param a first value
 * @param b second value
 * @return multiplication of two values in finite field F_2
 */
public static boolean multiply(boolean a, boolean b) {
    return a & b;
}

/**
 * Returns multiplication of two vectors in finite field F_2
 *
 * @param a first vector
 * @param b second vector
 * @return multiplication of two vectors in finite field F_2
 */
public static List<Boolean> multiply(List<Boolean> a, List<Boolean> b) {
    if (a.size() != b.size()) {
        throw new IllegalArgumentException("Both vectors must be same size");
    }

    List<Boolean> result = new ArrayList<>(a.size());
    for (int i = 0; i < a.size(); i ++) {
        result.add(multiply(a.get(i), b.get(i)));
    }

    return result;
}

/**
 * Returns multiplication of vector and scalar in finite field F_2
 *
 * @param a vector
 * @param b scalar
 * @return multiplication of vector and scalar in finite field F_2
 */
public static List<Boolean> multiply(List<Boolean> a, boolean b) {
    List<Boolean> result = new ArrayList<>(a.size());
    for (boolean bit : a) {
        result.add(multiply(bit, b));
    }

    return result;
}
```

```
    /**
     * Returns scalar product of two vectors in finite field F_2
     *
     * @param a first vector
     * @param b second vector
     * @return scalar product of two vectors in finite field F_2
     */
    public static boolean scalarProduct(List<Boolean> a, List<Boolean> b) {
        if (a.size() != b.size()) {
            throw new IllegalArgumentException("Both vectors must be same size");
        }

        boolean result = false;
        for (int i = 0; i < a.size(); i ++) {
            result = add(result, multiply(a.get(i), b.get(i)));
        }

        return result;
    }

    /**
     * Returns inverse value of a single bit
     *
     * @param a bit to be inverted
     * @return inverse value of a bit
     */
    public static Boolean invert(Boolean a) {
        return a ^ Boolean.TRUE;
    }

    /**
     * Returns multiplicative inverse of binary vector in finite field F_2
     *
     * @param a binary vector
     * @return multiplicative inverse of binary vector in finite field F_2
     */
    public static List<Boolean> invert(List<Boolean> a) {
        List<Boolean> result = new ArrayList<>();
        for (boolean bit : a) {
            result.add(invert(bit));
        }

        return result;
    }
}
```

1.2. Класс BitMatrix.java

```
package model;

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

public class BitMatrix {

   /**
    * Collection of bits represented as bit matrix
    */
   protected List<List<Boolean>> collection;

   /**
    * Constructs an object of model.BitMatrix
    */
   public BitMatrix() {
      collection = new ArrayList<>();
   }

   /**
    * Constructs a deep copy of a BitMatrix object
    *
    * @param data matrix to be copied
    */
   public BitMatrix(BitMatrix data) {
      collection = new ArrayList<>();
      for (int i = 0; i < data.getRowNumber(); i++) {
         collection.add(new ArrayList<>(data.getRow(i)));
      }
   }

   /**
    * Returns number of matrix rows
    *
    * @return number of rows
    */
   public int getRowNumber() {
      return collection.size();
   }

   /**
    * Returns matrix row by index
    *
    * @param index index of the row
    * @return certain row of the matrix
```

```
 */
public List<Boolean> getRow(int index) {
   return collection.get(index);
}

/**
 * Returns matrix column by index
 *
 * @param index index of the column
 * @return certain column of the matrix
 */
public List<Boolean> getColumn(int index) {
   List<Boolean> column = new ArrayList<>();

   for (List<Boolean> row : collection) {
      column.add(row.get(index));
   }

   return column;
}

/**
 * Adds a copy of argument row in the end of the matrix
 *
 * @param row row to be added
 */
public void addRow(List<Boolean> row) {
   collection.add(new ArrayList<>(row));
}

/**
 * Adds a copy of argument row as a last column of the matrix
 *
 * @param column row to be added as a column
 */
public void addColumn(List<Boolean> column) {
   List<Boolean> result = new ArrayList<>(column);

   if (result.size() != collection.size()) {
      throw new IllegalArgumentException("Column height does not match matrix height");
   }

   for (int i = 0; i < collection.size(); i ++) {
      collection.get(i).add((result.get(i)));
   }
}
```

```
    /**
     * Returns digit sequence string without redundant zero digits in the beginning
     *
     * @return digit sequence
     */
    public String toString() {
        String result = "";
        for (int i = 0; i < getRowNumber(); i++) {
            for (int j = 0; j < getRow(i).size(); j ++) {
                result += getRow(i).get(j) ? "1" : "0";
            }
            result += "\n";
        }

        return result;
    }

}
```

1.3. Класс RMMatrix.java

```
package model;

import util.Util;

import java.util.*;

public class RMMatrix extends BitMatrix {

    /**
     * Combinations of row indexes (used in forming the generator matrix)
     */
    List<List<Integer>> combination = new ArrayList<>();

    /**
     * Matrix width
     */
    private int width;

    /**
     * RM code length parameter
     */
    private int m;

    /**
     * Constructs a generator matrix of RM(r, m) code
     *
     * @param r order parameter
     * @param m block length parameter
     */
    public RMMatrix(int r, int m) {
        if (m < 2) {
            throw new IllegalArgumentException("M parameter should be greater than 1");
        }
        if (r >= m) {
            throw new IllegalArgumentException("Code order should be under length determinator");
        }
        if (r < 0) {
            throw new IllegalArgumentException("Code order should be non-negative");
        }
        this.m = m;
        width = (int)Math.pow(2, m);

        // first row of n ones
        addRow(Collections.nCopies(width, true));
```

Изм.	Лист	№ докум.	Подп.	Дата
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
        combination.add(Collections.nCopies(1, 0));

        // m rows which columns represent value sequences
        if (r == 0) {
            return;
        }
        List<Boolean> row = new ArrayList<>(width);
        for (int i = width / 2; i >= 1; i /= 2) {
            // (i = n / q; i >= 1; i /= q) or (i = 1; i < n; i *= q)
            for (int j = 0; j < width; j ++) {
                row.add(((j / i) % 2) != 0);
                // (j / i + 1) % q or (j / i) % q
            }
            addRow(row);
            row.clear();
            combination.add(Collections.nCopies(1, combination.size()));
        }

        // add multiplication matrix based on combination of rows (1, m)
        for (int i = 2; i <= r; i ++) {
            List<List<Integer>> combinationMI = Util.combination(m, i);
            for (List<Integer> combinationItem : combinationMI) {
                List<Boolean> multiplication = new ArrayList<>(getRow(combinationItem.get(0)));
                for (int k = 1; k < i; k ++) {
                    multiplication = BinaryFiniteField.multiply(multiplication,
getRow(combinationItem.get(k)));
                }
                addRow(multiplication);
            }
            for (List<Integer> combinationItem : combinationMI) {
                // append local combinationMI to combination
                combination.add(new ArrayList<>(combinationItem));
            }
        }
    }

    /**
     * Returns matrix width
     *
     * @return matrix width
     */
    public int getWidth() {
        return width;
    }

    /**
     * Returns list of index combinations
```

```
     * @return combinations
     */
    public List<List<Integer>> getCombination() {
        return combination;
    }

    /**
     * Returns characteristic vectors of a generator matrix row
     * They are used (as check-sums) in major decoding algorithm
     * @param index row index
     * @return matrix of characteristic vectors
     */
    public BitMatrix getCharacteristicVectors(int index) {
        if (index < 1) {
            throw new IllegalArgumentException();
        }
        BitMatrix result = new BitMatrix();

        // get monomial index which are not involved in a row combination
        Set<Integer> combinationSet = new HashSet<>(combination.get(index));
        List<Integer> monomialIndex = new ArrayList<>();
        for (int i = 1; i <= m; i ++) {
            if (! combinationSet.contains(i)) {
                monomialIndex.add(i);
            }
        }

        // get combination of bool function inputs
        BitMatrix monomialCombination = new BitMatrix();
        List<Boolean> row = new ArrayList<>(width);
        final int variety = (int)Math.pow(2, monomialIndex.size());
        for (int i = variety / 2; i >= 1; i /= 2) {
            for (int j = 0; j < variety; j ++) {
                row.add((j / i % 2) != 0);
            }
            monomialCombination.addRow(row);
            row.clear();
        }

        // produce combination of characteristic vectors
        for (int i = 0; i < variety; i ++) {
            List<Boolean> inputs = monomialCombination.getColumn(i);
            List<Boolean> multiplication = new ArrayList<>(
                    !inputs.get(0)  ?  getRow(monomialIndex.get(0))  :
BinaryFiniteField.invert(getRow(monomialIndex.get(0))));
            for (int j = 1; j < inputs.size(); j ++) {
                multiplication = BinaryFiniteField.multiply(multiplication,
```

Изм.	Лист	№ докум.	Подп.	Дата
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
                !inputs.get(j)  ?  getRow(monomialIndex.get(j))  :
BinaryFiniteField.invert(getRow(monomialIndex.get(j))));
            }
            result.addRow(multiplication);
        }

        return result;
    }
}
```

Изм.	Лист	№ докум.	Подп.	Дата
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

1.4. Класс RMCode.java

```
package model;

import util.Util;

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

public class RMCode {

    /**
     * Generator matrix of Reed-Muller code RM(r, m)
     */
    private RMMatrix generatorMatrix;

    /**
     * Code rate (message length / block length)
     */
    private double rate;

    /**
     * The minimal Hamming distance between code words
     */
    private int distance;

    /**
     * Maximum number of errors that code can correct while decoding
     */
    private int error;

    /**
     * Array of decoding matrix order rows (yR)
     */
    private String[][] yByOrderR;

    /**
     * Constructs an object of RM code
     *
     * @param r order parameter
     * @param m block length parameter
     */
    public RMCode(int r, int m) {
        generatorMatrix = new RMMatrix(r, m);
        rate = generatorMatrix.getRowNumber() / Math.pow(2, m);
        distance = (int)Math.pow(2, m - r);
```

```
    error = (int)Math.pow(2, m - r - 1) - 1;
    if (error < 0) error = 0;
}

/**
 * Returns generator matrix
 *
 * @return generator matrix
 */
public RMMatrix getGeneratorMatrix() {
    return generatorMatrix;
}

/**
 * Returns code rate
 * @return code rate
 */
public double getRate() {
    return rate;
}

/**
 * Returns minimal Hamming distance of code
 * @return code distance
 */
public int getDistance() {
    return distance;
}

/**
 * Returns maximum error number that code can correct
 * @return maximum error correction value
 */
public int getMaxErrorCorrection() {
    return error;
}

/**
 * Returns yR according to message and order indexes
 *
 * @param message message index
 * @param order   order index
 * @return yR
 */
public String getOrderY(int message, int order) {
    return yByOrderR[message][order];
}
```

Изм.	Лист	№ докум.	Подп.	Дата
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
   /**
    * Encodes given message
    * @param data message
    * @return encoded matrix of messages
    */
   public BitMatrix encode(BitMatrix data) {
      if (generatorMatrix == null) {
         throw new IllegalStateException("Generator matrix has not been initialized");
      }
      if (data == null) {
         throw new IllegalArgumentException("Data to encode is empty");
      }

      BitMatrix result = new BitMatrix();
      for (int i = 0; i < data.getRowNumber(); i++) {
         if (data.getRow(i).size() != generatorMatrix.getRowNumber()) {
            throw new IllegalArgumentException("Word length does not match generator matrix
height");
         }
         List<Boolean>                codeWord                =                new
ArrayList<>(BinaryFiniteField.multiply(generatorMatrix.getRow(0), data.getRow(i).get(0)));
         for (int j = 1; j < generatorMatrix.getRowNumber(); j++) {
            codeWord                =                BinaryFiniteField.add(codeWord,
BinaryFiniteField.multiply(generatorMatrix.getRow(j), data.getRow(i).get(j)));
         }
         result.addRow(codeWord);
      }

      return result;
   }

   /**
    * Decodes encoded message
    * @param data encoded message
    * @return matrix of decoded messages
    */
   public BitMatrix decode(BitMatrix data) {
      if (generatorMatrix == null) {
         throw new IllegalStateException("Generator matrix has not been initialized");
      }
      if (data == null) {
         throw new IllegalArgumentException("Data to decode is empty");
      }

      BitMatrix decoded = new BitMatrix();
      yByOrderR = new String[data.getRowNumber()][generatorMatrix.getRowNumber()];
```

```
        for (int i = 0; i < data.getRowNumber(); i++) {
            // for each received encoded word
            List<Boolean> yR = new ArrayList<>(data.getRow(i));
            List<Boolean> coefficient = new ArrayList<>();
            List<Boolean> My = new ArrayList<>(Collections.nCopies(generatorMatrix.getWidth(),
Boolean.FALSE));
            for (int j = generatorMatrix.getRowNumber() - 1; j > 0; j--) {
                // add yR to decoding parameter
                yByOrderR[i][j] = "";
                for (int k = 0; k < generatorMatrix.getWidth(); k++) {
                    yByOrderR[i][j] += yR.get(k) ? '1' : '0';
                }
                // for each non-first row of generator matrix
                BitMatrix characteristic = generatorMatrix.getCharacteristicVectors(j);
                List<Boolean> dotProductValues = new ArrayList<>();
                for (int k = 0; k < characteristic.getRowNumber(); k++) {
                    dotProductValues.add(BinaryFiniteField.scalarProduct(characteristic.getRow(k),
yR));
                }
                coefficient.add(Util.getMajorBit(dotProductValues));
                // multiply each coefficient by its corresponding row and add the resulting vectors
                My                                =                                BinaryFiniteField.add(My,
BinaryFiniteField.multiply(generatorMatrix.getRow(j), coefficient.get(coefficient.size() - 1)));
                // reduce inner degree
                if                (generatorMatrix.getCombination().get(j).size()                !=
generatorMatrix.getCombination().get(j - 1).size()) {
                    yR = BinaryFiniteField.add(yR, My);
                    My      =      new      ArrayList<>(Collections.nCopies(generatorMatrix.getWidth(),
Boolean.FALSE));
                }
            }
            yR = BinaryFiniteField.add(yR, My);
            // add yR to decoding parameter
            yByOrderR[i][0] = "";
            for (int k = 0; k < generatorMatrix.getWidth(); k++) {
                yByOrderR[i][0] += yR.get(k) ? '1' : '0';
            }
            coefficient.add(Util.getMajorBit(yR));
            Collections.reverse(coefficient);
            decoded.addRow(coefficient);
        }

        return decoded;
    }

}
```

Изм.	Лист	№ докум.	Подп.	Дата
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

1.5. Класс TransmitChannel.java

```
package model;

import java.util.HashSet;
import java.util.List;
import java.util.Random;
import java.util.Set;

public class TransmitChannel {

   /**
    * Transmits message through channel with errors
    *
    * @param data  message to be transmitted
    * @param error maximum number of errors to be placed in a single message block
    */
   public static void transmitMessage(BitMatrix data, int error) {
      Random generator = new Random();

      for (int i = 0; i < data.getRowNumber(); i++) {
         List<Boolean> row = data.getRow(i);
         int currentError = generator.nextInt(error + 1);
         Set<Integer> errorIndex = new HashSet<>();
         while (errorIndex.size() < currentError) {
            int currentIndex = generator.nextInt(row.size());
            if (!errorIndex.contains(currentIndex)) {
               errorIndex.add(currentIndex);
            }
         }
         for (Integer index : errorIndex) {
            row.set(index, BinaryFiniteField.invert(row.get(index)));
         }
      }
   }
}
```

1.6. Класс Util.java

```
package util;

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

public class Util {

   /**
    * Proceed all combinations without repeating of C(n, k)
    *
    * @param n maximum integer in a sequence
    * @param k number of integers to be retreived from sequence in a subsequence
    * @return list of combinations
    */
   public static List<List<Integer>> combination(int n, int k) {
      if (k > n) {
         throw new IllegalArgumentException();
      }
      if (k < 1 || n < 1) {
         throw new IllegalArgumentException();
      }

      List<List<Integer>> result = new ArrayList<>();
      List<Integer> combination = new ArrayList<>(k);
      for (int i = 1; i <= k; i ++) {
         combination.add(i);
      }
      if (k == n) {
         result.add(new ArrayList<>(combination));
         return result;
      }

      int p = k;
      while (p >= 1) {
         result.add(new ArrayList<>(combination));
         if (combination.get(k - 1) == n) {
            p --;
         } else {
            p = k;
         }
         if (p >= 1) {
            for (int i = k; i >= p; i --) {
               combination.set(i - 1, combination.get(p - 1) + i - p + 1);
            }
         }
```

Изм.	Лист	№ докум.	Подп.	Дата
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
        }

        return result;
    }

    /**
     * Returns a bit with major occurrences in a bit sequence
     *
     * @param a bit sequence
     * @return major occurred bit
     */
    public static boolean getMajorBit(List<Boolean> a) {
        int isZeroMajor = 0;

        for (boolean bit : a) {
            if (! bit) {
                isZeroMajor++;
                continue;
            }
            isZeroMajor--;
        }

        if (isZeroMajor == 0) {
            throw new RuntimeException("Number of ones equals to number of zeros in dot product
values");
        }

        return isZeroMajor < 0;
    }

    /**
     * Converts array of bytes into array of bits
     * @param bytes byte array
     * @return bit array
     */
    public static boolean[] byteArrayToBitArray(byte[] bytes) {
        boolean[] bits = new boolean[bytes.length * 8];
        for (int i = 0; i < bytes.length * 8; i++) {
            if ((bytes[i / 8] & (1 << (7 - (i % 8)))) > 0)
                bits[i] = true;
        }
        return bits;
    }

    /**
     * Converts bit array into byte array
     * @param bits bit array
```

Изм.	Лист	№ докум.	Подп.	Дата
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
     * @return byte array
     */
    public static byte[] bitArrayToByteArray(boolean[] bits) {
        if (bits.length % 8 != 0) {
            throw new IllegalArgumentException();
        }

        byte[] bytes = new byte[bits.length / 8];
        String singleByte = "";
        for (int i = 0; i < bits.length; i++) {
            singleByte += bits[i] ? "1" : "0";
            if (i % 8 == 7) {
                bytes[i / 8] = (byte) Integer.parseInt(singleByte, 2);
                singleByte = "";
            }
        }

        return bytes;
    }
}
```

1.7. Класс RMView.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 view;

import model.BinaryFiniteField;
import model.BitMatrix;
import model.RMCode;
import model.TransmitChannel;
import viewmodel.RMCodeSystem;

import javax.swing.*;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import javax.swing.text.BadLocationException;
import javax.swing.text.DefaultHighlighter;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.nio.charset.Charset;
import java.util.ArrayList;
import java.util.List;

public class RMView extends javax.swing.JFrame {

    private final double maxSleepTime = 3000;

    private final String charset = Charset.defaultCharset().name();

    private RMCodeSystem codeSystem;

    private int m = 3;
    private int r = 1;

    private BitMatrix converted;
    private BitMatrix encoded;
    private BitMatrix transmitted;
    private BitMatrix decoded;

    private double sleepTime = maxSleepTime / 2;
    private volatile boolean paused = false;
```

```
// Variables declaration - do not modify
private javax.swing.JLabel RMLabel;
private javax.swing.JLabel blockLabel;
private javax.swing.JTextField blockText;
private javax.swing.JButton convertButton;
private javax.swing.JLabel convertedMessageLabel;
private javax.swing.JScrollPane convertedMessageScroll;
private javax.swing.JTextArea convertedMessageText;
private javax.swing.JButton decodeButton;
private javax.swing.JLabel decodedLabel;
private javax.swing.JScrollPane decodedScroll;
private javax.swing.JLabel decodedStringLabel;
private javax.swing.JTextField decodedStringText;
private javax.swing.JTextArea decodedText;
private javax.swing.JLabel decodingProcessLabel;
private javax.swing.JScrollPane decodingProcessScroll;
private javax.swing.JTextArea decodingProcessText;
private javax.swing.JLabel distanceLabel;
private javax.swing.JTextField distanceText;
private javax.swing.JButton encodeButton;
private javax.swing.JLabel encodedLabel;
private javax.swing.JScrollPane encodedScroll;
private javax.swing.JTextArea encodedText;
private javax.swing.JLabel errorLabel;
private javax.swing.JTextField errorText;
private javax.swing.JLabel generatorLabel;
private javax.swing.JScrollPane generatorScroll;
private javax.swing.JTextArea generatorText;
private javax.swing.JTextField inputMessage;
private javax.swing.JLabel inputMessageLabel;
private javax.swing.JLabel lengthLabel;
private javax.swing.JTextField lengthText;
private javax.swing.JLabel messageLabel;
private javax.swing.JTextField messageLengthText;
private javax.swing.JPanel messagePanel;
private javax.swing.JLabel orderLabel;
private javax.swing.JTextField orderText;
private javax.swing.JLabel rateLabel;
private javax.swing.JTextField rateText;
private javax.swing.JScrollPane rowScroll;
private javax.swing.JTextArea rowText;
private javax.swing.JLabel signLabel;
private javax.swing.JSlider slider;
private javax.swing.JLabel sliderLabel;
private javax.swing.JButton transmitButton;
private javax.swing.JLabel transmittedLabel;
private javax.swing.JScrollPane transmittedScroll;
```

```
    private javax.swing.JLabel transmittedStringLabel;
    private javax.swing.JTextField transmittedStringText;
    private javax.swing.JTextArea transmittedText;
    // End of variables declaration


    /**
     * Creates new form RMView
     */
    public RMView() {
        initComponents();
        initHandlers();

        inputMessage.setText("Hello, world!");
        lengthText.setText("" + m);
        orderText.setText("" + r);

        initCode();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(RMView.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(RMView.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        } catch (IllegalAccessException ex) {
```

```
java.util.logging.Logger.getLogger(RMView.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(RMView.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new RMView().setVisible(true);
            }
        });
    }

    private void reset() {
        // set buttons enabling
        convertButton.setEnabled(true);
        encodeButton.setEnabled(false);
        transmitButton.setEnabled(false);
        decodeButton.setEnabled(false);

        // flush text
        convertedMessageText.setText("");
        encodedText.setText("");
        transmittedText.setText("");
        decodedText.setText("");
        rowText.setText("");
        transmittedStringText.setText("");
        decodedStringText.setText("");
        decodingProcessText.setText("");

        // remove highlights
        encodedText.getHighlighter().removeAllHighlights();
        transmittedText.getHighlighter().removeAllHighlights();
        decodedText.getHighlighter().removeAllHighlights();
        rowText.getHighlighter().removeAllHighlights();
        generatorText.getHighlighter().removeAllHighlights();

        // flush matrices
        converted = encoded = transmitted = decoded = null;
    }

    private void disableControls() {
```

```
        lengthText.setEnabled(false);
        orderText.setEnabled(false);

        convertButton.setEnabled(false);
        encodeButton.setEnabled(false);
        transmitButton.setEnabled(false);
        decodeButton.setEnabled(false);
    }

    private void sleepThread() {
        for (int i = 0; i < 10; i++) {
            while (paused) {
            }
            try {
                Thread.currentThread().sleep((int) sleepTime / 10);
            } catch (InterruptedException ie) {
                // do nothing
            }
        }
    }

    private void addHighlight(JTextArea area, int p0, int p1, Color c) {
        try {
            area.getHighlighter().addHighlight(p0,                p1,                new
DefaultHighlighter.DefaultHighlightPainter(c));
        } catch (BadLocationException ble) {
            // do nothing
        }
    }

    private void initHandlers() {
        lengthText.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                initCode();
            }
        });
        orderText.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                initCode();
            }
        });
        convertButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if (inputMessage.getText().length() == 0) {
```

```
                JOptionPane.showMessageDialog(messagePanel, "Please input message to
proceed",
                        "Incorrect input message", JOptionPane.ERROR_MESSAGE);
                return;
            }
            reset();
            converted = codeSystem.prepareMessageToEncodeProcess(inputMessage.getText(),
charset);
            convertedMessageText.setText(converted.toString());
            // highlight message start
            convertedMessageText.getHighlighter().removeAllHighlights();
            addHighlight(convertedMessageText, 0, convertedMessageText.getText().indexOf('1')
+ 1, Color.yellow);
            convertedMessageText.setCaretPosition(0);
            encodeButton.setEnabled(true);
        }
    });
    slider.addChangeListener(new ChangeListener() {
        @Override
        public void stateChanged(ChangeEvent e) {
            sleepTime = maxSleepTime * (1 - (double) slider.getValue() / (slider.getMaximum() -
slider.getMinimum()));
            paused = sleepTime == maxSleepTime;
        }
    });
    encodeButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (converted == null) {
                JOptionPane.showMessageDialog(messagePanel, "Input message should be
converted into bit stream before encoding",
                        "Incorrect state", JOptionPane.ERROR_MESSAGE);
                return;
            }
            disableControls();
            encodedText.setText("");

            new Thread(new Runnable() {
                @Override
                public void run() {
                    encoded = codeSystem.encode(converted);
                    String[] convertedRows = converted.toString().split("\n");
                    String[] encodedRows = encoded.toString().split("\n");

                    for (int i = 0; i < convertedRows.length; i++) {
                        convertedMessageText.getHighlighter().removeAllHighlights();
                        addHighlight(convertedMessageText, i * (convertedRows[i].length() + 1),
```

```
                    i * (convertedRows[i].length() + 1) + convertedRows[i].length(),
Color.yellow);
                convertedMessageText.setCaretPosition(i * (convertedRows[i].length() + 1));
                // set row
                rowText.setText("");
                rowText.getHighlighter().removeAllHighlights();
                generatorText.getHighlighter().removeAllHighlights();
                for (int j = 0; j < convertedRows[i].length(); j++) {
                   char bit = convertedRows[i].charAt(j);
                   rowText.append(bit + (j == convertedRows[i].length() - 1 ? "" : "\n"));
                   // highlight xor rows
                   if (bit == '1') {
                      addHighlight(rowText, 2 * j, 2 * j + 1, Color.yellow);
                      addHighlight(generatorText,
                            j * (codeSystem.getCode().getGeneratorMatrix().getWidth() + 1),
                            j * (codeSystem.getCode().getGeneratorMatrix().getWidth() + 1) +
codeSystem.getCode().getGeneratorMatrix().getWidth(),
                            Color.yellow);
                   }
                }
                // append encoded row
                encodedText.getHighlighter().removeAllHighlights();
                encodedText.append(encodedRows[i] + (i == convertedRows.length - 1 ? "" :
"\n"));
                addHighlight(encodedText, i * (encodedRows[i].length() + 1),
                    i * (encodedRows[i].length() + 1) + encodedRows[i].length(),
Color.green);
                // set caret
                encodedText.setCaretPosition(encodedText.getText().length() - 1);
                rowText.setCaretPosition(0);
                generatorText.setCaretPosition(0);
                sleepThread();
             }
             convertedMessageText.getHighlighter().removeAllHighlights();
             convertedMessageText.setCaretPosition(0);
             encodedText.getHighlighter().removeAllHighlights();
             encodedText.setCaretPosition(0);
             rowText.setText("");
             rowText.getHighlighter().removeAllHighlights();
             generatorText.getHighlighter().removeAllHighlights();
             // enable controls
             convertButton.setEnabled(true);
             encodeButton.setEnabled(true);
             transmitButton.setEnabled(true);
             lengthText.setEnabled(true);
             orderText.setEnabled(true);
          }
```

```
            }).start();
        }
    });
    transmitButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (encoded == null) {
                JOptionPane.showMessageDialog(messagePanel, "Converted message should be
encoded before transmitting",
                        "Incorrect state", JOptionPane.ERROR_MESSAGE);
                return;
            }
            disableControls();
            transmittedText.setText("");
            transmittedText.getHighlighter().removeAllHighlights();
            decoded = null;
            decodedText.setText("");
            decodedStringText.setText("");
            decodingProcessText.setText("");

            new Thread(new Runnable() {
                @Override
                public void run() {
                    transmitted = new BitMatrix(encoded);
                    TransmitChannel.transmitMessage(transmitted,
codeSystem.getCode().getMaxErrorCorrection());

                    for (int i = 0; i < transmitted.getRowNumber(); i++) {
                        encodedText.getHighlighter().removeAllHighlights();
                        addHighlight(encodedText,                    i                    *
(codeSystem.getCode().getGeneratorMatrix().getWidth() + 1),
                                i   *   (codeSystem.getCode().getGeneratorMatrix().getWidth()   +   1)   +
codeSystem.getCode().getGeneratorMatrix().getWidth(),
                                Color.yellow);
                        encodedText.setCaretPosition(i                                                    *
(codeSystem.getCode().getGeneratorMatrix().getWidth() + 1));
                        // first append
                        for (int j = 0; j < transmitted.getRow(i).size(); j++) {
                            transmittedText.append(transmitted.getRow(i).get(j) ? "1" : "0");
                        }
                        if (i != transmitted.getRowNumber() - 1) transmittedText.append("\n");
                        // then highlight
                        for (int j = 0; j < transmitted.getRow(i).size(); j++) {
                            if (transmitted.getRow(i).get(j) ^ encoded.getRow(i).get(j)) {
                                addHighlight(transmittedText,                    i                    *
(codeSystem.getCode().getGeneratorMatrix().getWidth() + 1) + j,
```

```
                                i * (codeSystem.getCode().getGeneratorMatrix().getWidth() + 1) + j
+ 1,
                                Color.red);
                        }
                    }
                    transmittedText.setCaretPosition(transmittedText.getText().length() - 1);
                    sleepThread();
                }
                encodedText.getHighlighter().removeAllHighlights();
                encodedText.setCaretPosition(0);
                transmittedText.setCaretPosition(0);

transmittedStringText.setText(codeSystem.getValidMessageFromBitMatrix(transmitted,
charset));
                // enable controls
                convertButton.setEnabled(true);
                encodeButton.setEnabled(true);
                transmitButton.setEnabled(true);
                decodeButton.setEnabled(true);
                lengthText.setEnabled(true);
                orderText.setEnabled(true);
            }
        }).start();
    }
});
decodeButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        if (transmitted == null) {
            JOptionPane.showMessageDialog(messagePanel, "Encoded message should be
transmitted thought channel before decoding",
                    "Incorrect state", JOptionPane.ERROR_MESSAGE);
            return;
        }
        disableControls();
        decodedText.setText("");
        decodingProcessText.setText("");

        new Thread(new Runnable() {
            @Override
            public void run() {
                decoded = codeSystem.decode(transmitted);
                String[] transmittedRows = transmitted.toString().split("\n");
                String[] decodedRows = decoded.toString().split("\n");
```

Изм.	Лист	№ докум.	Подп.	Дата
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
            String[][] characteristicVectors = new
String[codeSystem.getCode().getGeneratorMatrix().getRowNumber()][codeSystem.getCode().ge
tGeneratorMatrix().getWidth()];
            for (int i = 1; i < codeSystem.getCode().getGeneratorMatrix().getRowNumber();
i++) {
               characteristicVectors[i] =
codeSystem.getCode().getGeneratorMatrix().getCharacteristicVectors(i).toString().split("\n");
            }

            for (int i = 0; i < transmittedRows.length; i++) {

transmittedText.setCaretPosition((codeSystem.getCode().getGeneratorMatrix().getWidth() + 1)
* (i + 1) - 1);
               for (int j = decodedRows[i].length() - 1; j >= 0; j--) {
                  // add decoded message
                  generatorText.getHighlighter().removeAllHighlights();
                  addHighlight(generatorText, j *
(codeSystem.getCode().getGeneratorMatrix().getWidth() + 1),
                        (j + 1) * (codeSystem.getCode().getGeneratorMatrix().getWidth() + 1),
Color.yellow);
                  String decoded = "";
                  for (int k = 0; k < decodedRows[i].length(); k++) {
                     if (k <= j) decoded += "-";
                     else decoded += decodedRows[i].charAt(k);
                  }
                  // add dot products

decodingProcessText.setText(formDecodingProcessMessage(transmittedRows[i], decoded, j + 1,
characteristicVectors[j], i, j));
                  sleepThread();
               }
               decodedText.getHighlighter().removeAllHighlights();
               decodedText.append(decodedRows[i]);
               if (i != transmittedRows.length - 1) decodedText.append("\n");
               addHighlight(decodedText, i * (decodedRows[i].length() + 1), i *
(decodedRows[i].length() + 1) + decodedRows[i].length(), Color.green);
               decodedText.setCaretPosition(decodedText.getText().length() - 1);
            }
            transmittedText.setCaretPosition(0);
            decodedText.getHighlighter().removeAllHighlights();
            generatorText.getHighlighter().removeAllHighlights();
            decodedText.setCaretPosition(0);
            decodingProcessText.setText("");

decodedStringText.setText(codeSystem.getValidMessageAfterDecoding(decoded, charset));
            // enable controls
            convertButton.setEnabled(true);
```

```
                encodeButton.setEnabled(true);
                transmitButton.setEnabled(true);
                decodeButton.setEnabled(true);
                lengthText.setEnabled(true);
                orderText.setEnabled(true);
            }
        }).start();
    }
});
}

private String formDecodingProcessMessage(String transmittedRow, String decodedMessage,
int bitNumber, String[] characteristicVectors, int message, int order) {
    String result = "Decoding code block:\n" + transmittedRow;
    if (decodedMessage != null) {
        result += "\nDecoded message:\n" + decodedMessage;
    }
    if (bitNumber > -1) {
        result += "\nDecoding bit number: " + bitNumber;
    }
    if (characteristicVectors != null) {
        if (characteristicVectors[0] != null) {
            result += "\nDot products:";
            for (String vector : characteristicVectors) {
                List<Boolean> characteristic = new ArrayList<>();
                for (int j = 0; j < transmittedRow.length(); j++) {
                    characteristic.add(vector.charAt(j) != '0');
                }
                String currentYR = codeSystem.getCode().getOrderY(message, order);
                List<Boolean> currentYRBool = new ArrayList<>();
                for (int i = 0; i < currentYR.length(); i++) {
                    currentYRBool.add(currentYR.charAt(i) != '0');
                }
                result += "\n(" + vector + ").(" + (currentYR) + ") = " +
                        (BinaryFiniteField.scalarProduct(currentYRBool, characteristic) ? "1" : "0");
            }
        } else {
            result += "\nFirst bit determinator:\n" + codeSystem.getCode().getOrderY(message,
0);
        }
    }

    return result;
}

private void initCode() {
    try {
```

Изм.	Лист	№ докум.	Подп.	Дата
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
            if (lengthText.getText().length() == 0) lengthText.setText("0");
            if (orderText.getText().length() == 0) orderText.setText("0");
            m = Integer.parseInt(lengthText.getText());
            r = Integer.parseInt(orderText.getText());
            if (m > 6)
                throw new IllegalArgumentException("In order to provide smooth animation length determinator should be under 7");
            codeSystem = new RMCodeSystem(new RMCode(r, m));

        } catch (NumberFormatException nfe) {
            JOptionPane.showMessageDialog(this, "Please, input correct number", "Incorrect code parameters",
                    JOptionPane.ERROR_MESSAGE);
            return;
        } catch (IllegalArgumentException iae) {
            JOptionPane.showMessageDialog(this, iae.getMessage(), "Incorrect code parameters",
                    JOptionPane.ERROR_MESSAGE);
            return;
        }
        RMCode code = codeSystem.getCode();
        blockText.setText("" + code.getGeneratorMatrix().getWidth());
        messageLengthText.setText("" + code.getGeneratorMatrix().getRowNumber());
        distanceText.setText("" + code.getDistance());
        rateText.setText("" + String.format("%.2f", code.getRate()));
        errorText.setText("" + code.getMaxErrorCorrection());
        generatorText.setText(code.getGeneratorMatrix().toString());
        reset();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        messagePanel = new javax.swing.JPanel();
        inputMessageLabel = new javax.swing.JLabel();
        inputMessage = new javax.swing.JTextField();
        convertedMessageLabel = new javax.swing.JLabel();
        convertedMessageScroll = new javax.swing.JScrollPane();
        convertedMessageText = new javax.swing.JTextArea();
        encodedScroll = new javax.swing.JScrollPane();
        encodedText = new javax.swing.JTextArea();
        encodedLabel = new javax.swing.JLabel();
```

```
transmittedScroll = new javax.swing.JScrollPane();
transmittedText = new javax.swing.JTextArea();
transmittedLabel = new javax.swing.JLabel();
decodedLabel = new javax.swing.JLabel();
decodedScroll = new javax.swing.JScrollPane();
decodedText = new javax.swing.JTextArea();
convertButton = new javax.swing.JButton();
encodeButton = new javax.swing.JButton();
transmitButton = new javax.swing.JButton();
decodeButton = new javax.swing.JButton();
transmittedStringLabel = new javax.swing.JLabel();
transmittedStringText = new javax.swing.JTextField();
decodedStringLabel = new javax.swing.JLabel();
decodedStringText = new javax.swing.JTextField();
RMLabel = new javax.swing.JLabel();
lengthLabel = new javax.swing.JLabel();
orderLabel = new javax.swing.JLabel();
lengthText = new javax.swing.JTextField();
orderText = new javax.swing.JTextField();
blockLabel = new javax.swing.JLabel();
messageLabel = new javax.swing.JLabel();
blockText = new javax.swing.JTextField();
messageLengthText = new javax.swing.JTextField();
distanceLabel = new javax.swing.JLabel();
distanceText = new javax.swing.JTextField();
rateLabel = new javax.swing.JLabel();
rateText = new javax.swing.JTextField();
errorLabel = new javax.swing.JLabel();
errorText = new javax.swing.JTextField();
generatorLabel = new javax.swing.JLabel();
rowScroll = new javax.swing.JScrollPane();
rowText = new javax.swing.JTextArea();
signLabel = new javax.swing.JLabel();
generatorScroll = new javax.swing.JScrollPane();
generatorText = new javax.swing.JTextArea();
slider = new javax.swing.JSlider();
sliderLabel = new javax.swing.JLabel();
decodingProcessLabel = new javax.swing.JLabel();
decodingProcessScroll = new javax.swing.JScrollPane();
decodingProcessText = new javax.swing.JTextArea();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("Reed-Muller code animator");
setLocationByPlatform(true);
setResizable(false);
```

```
messagePanel.setBorder(javax.swing.BorderFactory.createEtchedBorder(javax.swing.border.Etc
hedBorder.RAISED));

        inputMessageLabel.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
        inputMessageLabel.setText("Input message to encode:");

        inputMessage.setFont(new java.awt.Font("Monospaced", 0, 12)); // NOI18N

inputMessage.setText("ABCDEFGHIJKLMNOPQRSTUVWXYZАБВГДЕЁЖЗИЙКЛМНОП
РСТУФХЦЧШЩЪЫЬЭЮЯ0123456789");

        convertedMessageLabel.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
        convertedMessageLabel.setText("Converted message:");

        convertedMessageText.setEditable(false);
        convertedMessageText.setColumns(20);
        convertedMessageText.setFont(new java.awt.Font("Monospaced", 0, 12)); // NOI18N
        convertedMessageText.setRows(5);
        convertedMessageText.setPreferredSize(null);
        convertedMessageScroll.setViewportView(convertedMessageText);

        encodedText.setEditable(false);
        encodedText.setColumns(20);
        encodedText.setFont(new java.awt.Font("Monospaced", 0, 12)); // NOI18N
        encodedText.setRows(5);
        encodedText.setPreferredSize(null);
        encodedScroll.setViewportView(encodedText);

        encodedLabel.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
        encodedLabel.setText("Encoded message bit stream:");

        transmittedText.setEditable(false);
        transmittedText.setColumns(20);
        transmittedText.setFont(new java.awt.Font("Monospaced", 0, 12)); // NOI18N
        transmittedText.setRows(5);
        transmittedText.setPreferredSize(null);
        transmittedScroll.setViewportView(transmittedText);

        transmittedLabel.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
        transmittedLabel.setText("Transmitted bit stream:");

        decodedLabel.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
        decodedLabel.setText("Decoded bit stream:");

        decodedText.setEditable(false);
        decodedText.setColumns(20);
```

```
        decodedText.setFont(new java.awt.Font("Monospaced", 0, 12)); // NOI18N
        decodedText.setRows(5);
        decodedText.setPreferredSize(null);
        decodedScroll.setViewportView(decodedText);

        convertButton.setText("Convert message");
        convertButton.setPreferredSize(new java.awt.Dimension(146, 23));

        encodeButton.setText("Encode data");

        transmitButton.setText("Transmit encoded message");

        decodeButton.setText("Decode received data");

        transmittedStringLabel.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
        transmittedStringLabel.setText("Transmitted message through channel:");

        transmittedStringText.setEditable(false);
        transmittedStringText.setFont(new java.awt.Font("Monospaced", 0, 12)); // NOI18N

        decodedStringLabel.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
        decodedStringLabel.setText("Decoded message:");

        decodedStringText.setEditable(false);
        decodedStringText.setFont(new java.awt.Font("Monospaced", 0, 12)); // NOI18N

        javax.swing.GroupLayout messagePanelLayout = new
javax.swing.GroupLayout(messagePanel);
        messagePanel.setLayout(messagePanelLayout);
        messagePanelLayout.setHorizontalGroup(

messagePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(messagePanelLayout.createSequentialGroup()
                    .addContainerGap()

.addGroup(messagePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING)
                        .addComponent(transmittedStringText)
                        .addComponent(inputMessage)
                        .addComponent(decodedStringText)
                        .addGroup(messagePanelLayout.createSequentialGroup()

.addGroup(messagePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING)
                            .addComponent(inputMessageLabel)
                            .addGroup(messagePanelLayout.createSequentialGroup()
```

```
.addGroup(messagePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRA
ILING, false)
                                        .addComponent(convertButton,
javax.swing.GroupLayout.Alignment.LEADING,    javax.swing.GroupLayout.DEFAULT_SIZE,
168, Short.MAX_VALUE)
                                        .addComponent(convertedMessageScroll,
javax.swing.GroupLayout.Alignment.LEADING)
                                        .addComponent(convertedMessageLabel,
javax.swing.GroupLayout.Alignment.LEADING))
                                    .addGap(18, 18, 18)

.addGroup(messagePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING, false)
                                        .addComponent(encodedLabel)
                                        .addComponent(encodedScroll,
javax.swing.GroupLayout.DEFAULT_SIZE, 168, Short.MAX_VALUE)
                                        .addComponent(encodeButton,
javax.swing.GroupLayout.DEFAULT_SIZE,          javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                                    .addGap(18, 18, 18)

.addGroup(messagePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING, false)
                                        .addComponent(transmittedScroll)
                                        .addComponent(transmittedLabel)
                                        .addComponent(transmitButton,
javax.swing.GroupLayout.DEFAULT_SIZE, 168, Short.MAX_VALUE))
                                    .addGap(18, 18, 18)

.addGroup(messagePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING, false)
                                        .addComponent(decodedLabel)
                                        .addComponent(decodedScroll,
javax.swing.GroupLayout.DEFAULT_SIZE, 168, Short.MAX_VALUE)
                                        .addComponent(decodeButton,
javax.swing.GroupLayout.DEFAULT_SIZE,          javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)))
                                .addComponent(transmittedStringLabel)
                                .addComponent(decodedStringLabel))
                            .addGap(0, 0, Short.MAX_VALUE)))
                    .addContainerGap())
        );
        messagePanelLayout.setVerticalGroup(

messagePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(messagePanelLayout.createSequentialGroup()
```

```
                .addContainerGap()
                .addComponent(inputMessageLabel)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(inputMessage,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    30,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(messagePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE)
                        .addComponent(convertedMessageLabel)
                        .addComponent(encodedLabel)
                        .addComponent(transmittedLabel)
                        .addComponent(decodedLabel))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(messagePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING, false)
                        .addComponent(decodedScroll,
javax.swing.GroupLayout.DEFAULT_SIZE, 200, Short.MAX_VALUE)
                        .addComponent(transmittedScroll)
                        .addComponent(convertedMessageScroll)
                        .addComponent(encodedScroll))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(messagePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE)
                        .addComponent(convertButton,
javax.swing.GroupLayout.PREFERRED_SIZE,      javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(encodeButton)
                        .addComponent(transmitButton)
                        .addComponent(decodeButton))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(transmittedStringLabel)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(transmittedStringText,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    30,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
                    .addComponent(decodedStringLabel)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(decodedStringText,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    30,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );

        RMLabel.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
        RMLabel.setText("Reed Muller code parameters:");
        RMLabel.setToolTipText("");

        lengthLabel.setText("Length determinator m:");

        orderLabel.setText("Code order r:");

        blockLabel.setText("Block length:");

        messageLabel.setText("Message length:");

        blockText.setEditable(false);

        messageLengthText.setEditable(false);

        distanceLabel.setText("Minimum distance:");

        distanceText.setEditable(false);

        rateLabel.setText("Code rate:");

        rateText.setEditable(false);

        errorLabel.setText("Max error correction:");

        errorText.setEditable(false);

        generatorLabel.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
        generatorLabel.setText("Encoding block and code generator matrix:");


rowScroll.setHorizontalScrollBarPolicy(javax.swing.ScrollPaneConstants.HORIZONTAL_SCR
OLLBAR_NEVER);

        rowText.setEditable(false);
        rowText.setColumns(20);
```

```
        rowText.setRows(5);
        rowScroll.setViewportView(rowText);

        signLabel.setFont(new java.awt.Font("Monospaced", 0, 20)); // NOI18N
        signLabel.setText("⊙");

        generatorText.setEditable(false);
        generatorText.setColumns(20);
        generatorText.setRows(5);
        generatorScroll.setViewportView(generatorText);

        sliderLabel.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
        sliderLabel.setText("Animation velocity:");

        decodingProcessLabel.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
        decodingProcessLabel.setText("Decoding process:");

        decodingProcessText.setEditable(false);
        decodingProcessText.setColumns(20);
        decodingProcessText.setRows(5);
        decodingProcessScroll.setViewportView(decodingProcessText);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                        .addComponent(messagePanel,
javax.swing.GroupLayout.PREFERRED_SIZE,      javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGroup(layout.createSequentialGroup()
                            .addGap(10, 10, 10)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                        .addComponent(lengthLabel)
                                        .addComponent(orderLabel)
                                        .addComponent(blockLabel)
                                        .addComponent(messageLabel)
                                        .addComponent(distanceLabel)
                                        .addComponent(rateLabel)
```

```
                                .addComponent(errorLabel))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                .addComponent(errorText,
javax.swing.GroupLayout.PREFERRED_SIZE,                                 46,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                                .addComponent(distanceText)
                                .addComponent(messageLengthText)
                                .addComponent(blockText)
                                .addComponent(lengthText,
javax.swing.GroupLayout.Alignment.LEADING)
                                .addComponent(orderText,
javax.swing.GroupLayout.Alignment.LEADING)
                                .addComponent(rateText,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE,                                 46,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                            .addGap(18, 18, 18)
                            .addComponent(rowScroll,
javax.swing.GroupLayout.PREFERRED_SIZE,                                 50,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                            .addComponent(signLabel)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                            .addComponent(generatorScroll,
javax.swing.GroupLayout.PREFERRED_SIZE,                                 166,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGroup(layout.createSequentialGroup()
                            .addComponent(RMLabel)
                            .addGap(33, 33, 33)
                            .addComponent(generatorLabel)))
                    .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(decodingProcessScroll)
                        .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                .addComponent(decodingProcessLabel)
                                .addGroup(layout.createSequentialGroup()
                                    .addComponent(sliderLabel)
```

```
                                        .addGap(18, 18, 18)
                                        .addComponent(slider,
javax.swing.GroupLayout.PREFERRED_SIZE,                                 151,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                                    .addGap(0, 0, Short.MAX_VALUE)))))
                .addContainerGap(16, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addContainerGap()
                    .addComponent(messagePanel,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(15, 15, 15)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addGroup(layout.createSequentialGroup()
                            .addGap(29, 29, 29)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                                .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                        .addComponent(lengthLabel)
                                        .addComponent(lengthText,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                        .addComponent(orderLabel)
                                        .addComponent(orderText,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                        .addComponent(blockLabel)
                                        .addComponent(blockText,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                        .addComponent(messageLabel)
                                        .addComponent(messageLengthText,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                        .addComponent(distanceLabel)
                                        .addComponent(distanceText,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                        .addComponent(rateText,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addComponent(rateLabel))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                        .addComponent(errorLabel)
                                        .addComponent(errorText,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                                    .addComponent(rowScroll)
                                    .addComponent(generatorScroll)))
                                .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                        .addComponent(generatorLabel)
                                        .addComponent(sliderLabel))
                                    .addComponent(RMLabel))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addGroup(layout.createSequentialGroup()
                                        .addGap(15, 15, 15)
                                        .addComponent(decodingProcessLabel)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
                                .addComponent(decodingProcessScroll))
                            .addGroup(layout.createSequentialGroup()
                                .addGap(96, 96, 96)
                                .addComponent(signLabel)
                                .addGap(0, 0, Short.MAX_VALUE))))
                        .addComponent(slider,  javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>

}
```

1.8. Класс RMCodeSystem.java

```
package viewmodel;

import model.BitMatrix;
import model.RMCode;
import util.Util;

import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.List;

public class RMCodeSystem {

    /**
     * RMCode instance
     */
    private RMCode code;

    /**
     * Constructs RMCode boundary
     *
     * @param code RMCode object
     */
    public RMCodeSystem(RMCode code) {
        this.code = code;
    }

    /**
     * Returns RMCode instance
     *
     * @return
     */
    public RMCode getCode() {
        return code;
    }


    public BitMatrix decode(BitMatrix data) {
        return code.decode(data);
    }

    public BitMatrix encode(BitMatrix prepared) {
        return code.encode(prepared);
    }

    /**
```

```
    * Returns model.BitMatrix of message blocks with specified beginning
    * of the message (in order to fit size of input message to code block length
    * by adding zeros in the beginning and one 1 bit)
    *
    * @param message message to encode
    * @param charset encoding
    * @return list of message blocks starting with exceed zeros and one 1
    */
   public BitMatrix prepareMessageToEncodeProcess(String message, String charset) {
      BitMatrix data = new BitMatrix();

      // get bit sequence from input text
      boolean[] bitSequence;
      try {
         bitSequence = Util.byteArrayToBitArray(message.getBytes(charset));
      } catch (UnsupportedEncodingException uee) {
         bitSequence = Util.byteArrayToBitArray(message.getBytes());
      }

      int messageLength = bitSequence.length; // plus one 1 bit meaning beginning of the message value
      int messageWordLength = code.getGeneratorMatrix().getRowNumber();
      // number of zero bit to be added in the beginning of message in order to match word size
      int exceedBit = (messageWordLength - (messageLength + 1) % messageWordLength) % messageWordLength;
      // message length + 1 because of exceed 1 at the beginning

      // transfer bit stream into sparse matrix of fixed code word size
      List<Boolean> messageWord = new ArrayList<>(messageWordLength);
      // -exceedBit - 1 because of exceed 1 at the beginning
      for (int i = -exceedBit - 1; i < messageLength; i++) {
         // check for starting one 1
         messageWord.add(i == -1 ? Boolean.TRUE : (i < 0 ? Boolean.FALSE : bitSequence[i]));
         if (messageWord.size() == messageWordLength) {
            data.addRow(messageWord);
            messageWord.clear();
         }
      }

      return data;
   }

   /**
    * Returns valid decoded message formed by removing exceed zero bits
    * in the beginning with one 1 (as beginning of the message)
    *
    * @param data    decoded message
```

Изм.	Лист	№ докум.	Подп.	Дата
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
 * @param charset encoding
 * @return String decoded text
 */
public String getValidMessageAfterDecoding(BitMatrix data, String charset) {
   boolean[] bits = new boolean[0];
   int bitIndex = 0;
   int redundantBit = 0;
   int decodedLength = data.getRowNumber() * data.getRow(0).size();
   boolean redundant = true;
   for (int i = 0; i < data.getRowNumber(); i++) {
      for (int j = 0; j < data.getRow(i).size(); j++) {
         if (redundant) {
            if (!data.getRow(i).get(j)) {
               redundantBit++;
               continue;
            }
            redundant = false;
            bits = new boolean[decodedLength - redundantBit - 1];
            continue;
         }
         bits[bitIndex++] = data.getRow(i).get(j);
      }
   }

   String result;
   try {
      result = new String(Util.bitArrayToByteArray(bits), charset);
   } catch (UnsupportedEncodingException uee) {
      result = new String(Util.bitArrayToByteArray(bits));
   }

   return result;
}

/**
 * Returns valid message formed by adding exceed zero bits
 * in the beginning of first message in order to fit byte amount
 *
 * @param data matrix of bit vectors
 * @param charset message encoding
 * @return converted message
 */
public String getValidMessageFromBitMatrix(BitMatrix data, String charset) {
   if (data == null || data.getRowNumber() == 0) {
      throw new IllegalArgumentException();
   }
```

```
        int messageSize = data.getRowNumber() * data.getRow(0).size();
        // number of zero bits to be added at the beginning of the message
        // in order to match the message length of the code
        int exceedBit = (8 - messageSize % 8) % 8;
        int bitIndex = 0;

        boolean[] bits = new boolean[messageSize + exceedBit];
        // pre fill the message with exceeding zero bits
        for (; bitIndex < exceedBit; bitIndex++) {
           bits[bitIndex] = false;
        }
        // fill the bit sequence with data matrix values
        for (int i = 0; i < data.getRowNumber(); i++) {
           for (int j = 0; j < data.getRow(i).size(); j++) {
              bits[bitIndex++] = data.getRow(i).get(j);
           }
        }

        String result;
        try {
           result = new String(Util.bitArrayToByteArray(bits), charset);
        } catch (UnsupportedEncodingException uee) {
           result = new String(Util.bitArrayToByteArray(bits));
        }

        return result;
     }

}
```

Лист регистрации изменений									
Изм.	Номера листов				Всего листов (страниц) в докум.	№ докум.	Входящий № сопроводительного докум. и дата	Подп.	Дата
	измененных	замененных	новых	аннулированных					

Изм.	Лист	№ докум.	Подп.	Дата
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата