#### SDESGUI.java

SDESGUI.java负责实现ASCII码和二进制加解密的初始界面

界面控件的实现：方便用户操作:

public SDESGUI() {

        setTitle("S-DES 加密/解密");

        setSize(400, 250);

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setLocationRelativeTo(null);

        JPanel mainPanel = new JPanel();

        mainPanel.setLayout(new GridLayout(5, 2, 10, 10));

        JLabel keyLabel = new JLabel("密钥 (10位二进制):");

        keyField = new JTextField(10);

        JLabel inputLabel = new JLabel("输入文本 :");

        inputField = new JTextField(20);

        JLabel outputLabel = new JLabel("输出文本:");

        encryptButton = new JButton("加密");

        decryptButton = new JButton("解密");

        outputField = new JTextField();

        outputField.setEditable(false);

        binaryRadio = new JRadioButton("二进制", true);

        asciiRadio = new JRadioButton("ASCII");

        ButtonGroup radioGroup = new ButtonGroup();

        radioGroup.add(binaryRadio);

        radioGroup.add(asciiRadio);

        mainPanel.add(keyLabel);

        mainPanel.add(keyField);

        mainPanel.add(inputLabel);

        mainPanel.add(inputField);

        mainPanel.add(encryptButton);

        mainPanel.add(decryptButton);

        mainPanel.add(outputLabel);

        mainPanel.add(outputField);

        mainPanel.add(binaryRadio);

        mainPanel.add(asciiRadio);

        add(mainPanel, BorderLayout.CENTER);

**设置监视器，实现GUI界面的各项功能：**

encryptButton.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                String key = keyField.getText();

                String input = inputField.getText();

                if (binaryRadio.isSelected()) {

                    String ciphertext = S\_DES.encrypt(input);

                    outputField.setText(ciphertext);

                } else if (asciiRadio.isSelected()) {

                    // 将ASCII明文转换为二进制字符串

                    StringBuilder binaryPlaintext = new StringBuilder();

                    for (char c : input.toCharArray()) {

                        binaryPlaintext.append(String.format("%8s", Integer.toBinaryString(c)).replace(' ', '0'));

                    }

                    String ciphertext = S\_DES.encrypt(binaryPlaintext.toString());

                    // 将二进制密文转换回ASCII字符串

                    StringBuilder asciiCiphertext = new StringBuilder();

                    for (int i = 0; i < ciphertext.length(); i += 8) {

                        String byteStr = ciphertext.substring(i, i + 8);

                        int byteValue = Integer.parseInt(byteStr, 2);

                        asciiCiphertext.append((char) byteValue);

                    }

                    outputField.setText(asciiCiphertext.toString());

                }

            }

        });

        decryptButton.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                String key = keyField.getText();

                String input = inputField.getText();

                String plaintext = "";

                if (binaryRadio.isSelected()) {

                    plaintext = S\_DES.decrypt(input);

                } else if (asciiRadio.isSelected()) {

                    // 将ASCII密文转换为二进制字符串

                    StringBuilder binaryCiphertext = new StringBuilder();

                    for (char c : input.toCharArray()) {

                        binaryCiphertext.append(String.format("%8s", Integer.toBinaryString(c)).replace(' ', '0'));

                    }

                    plaintext = S\_DES.decrypt(binaryCiphertext.toString());

                    // 将二进制明文转换回ASCII字符串

                    StringBuilder asciiPlaintext = new StringBuilder();

                    for (int i = 0; i < plaintext.length(); i += 8) {

                        String byteStr = plaintext.substring(i, i + 8);

                        int byteValue = Integer.parseInt(byteStr, 2);

                        asciiPlaintext.append((char) byteValue);

                    }

                    plaintext = asciiPlaintext.toString();

                }

                outputField.setText(plaintext);

            }

        });

    }

#### S\_DES.java

S\_DES.java中，在S\_DES类中实现了Simple-DES算法，分别对外提供二进制数组，二进制字符，ASCLL字符的加解密及破解。

**数组的初始化：**

static int P4[] = new int[] { 2, 4, 3, 1 };

    static int P8[] = new int[] { 6, 3, 7, 4, 8, 5, 10, 9 };

    static int P10[] = new int[] { 3, 5, 2, 7, 4, 10, 1, 9, 8, 6 };

    static int IP[] = new int[] { 2, 6, 3, 1, 4, 8, 5, 7 };

    static int IP\_1[] = new int[] { 4, 1, 3, 5, 7, 2, 8, 6 };

    static int EP[] = new int[] { 4, 1, 2, 3, 2, 3, 4, 1 };

    static int S1[][] = { { 1, 0, 3, 2 }, { 3, 2, 1, 0 }, { 0, 2, 1, 3 }, { 3, 1, 0, 2 } };

    static int S2[][] = { { 0, 1, 2, 3 }, { 2, 3, 1, 0 }, { 3, 0, 1, 2 }, { 2, 1, 0, 3 } };

    static int subkey1, subkey2 = 0;

**算法的实现：**

static int applyPermutation(int a, int b[], int c) {

        int x = 0;

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

            x <<= 1;

            x |= (a >> (c - b[i])) & 1;

        }

        return x;

    }

    static int applyEP(int a, int b) {

        int t, t0, t1, x1, x2, y1, y2;

        t = applyPermutation(a, EP, 4);

        t0 = (t >> 4) & 0xf;

        t1 = t & 0xf;

        x1 = ((t0 & 0x8) >> 2) | (t0 & 1);

        y1 = (t0 >> 1) & 0x3;

        x2 = ((t1 & 0x8) >> 2) | (t1 & 1);

        y2 = (t1 >> 1) & 0x3;

        t0 = S1[x1][y1];

        t1 = S2[x2][y2];

        t = applyPermutation((t0 << 2) | t1, P4, 4);

        return t;

    }

    static int applyF(int a, int subkey) {

        int left = (a >> 4) & 0xf;

        int right = a & 0xf;

        return ((left ^ applyEP(right, subkey)) << 4) | right;

    }

    static void generateSubkeys(String key) {

        int x = Integer.parseInt(key, 2);

        x = applyPermutation(x, P10, 10);

        int leftKey, rightKey = 0;

        leftKey = (x >> 5) & 0x1f;

        rightKey = x & 0x1f;

        leftKey = ((leftKey & 0xf) << 1) | ((leftKey & 0x10) >> 4);

        rightKey = ((rightKey & 0xf) << 1) | ((rightKey & 0x10) >> 4);

        subkey1 = applyPermutation((leftKey << 5) | rightKey, P8, 10);

        leftKey = ((leftKey & 0x07) << 2) | ((leftKey & 0x18) >> 3);

        rightKey = ((rightKey & 0x07) << 2) | ((rightKey & 0x18) >> 3);

        subkey2 = applyPermutation((leftKey << 5) | rightKey, P8, 10);

    }

**加密算法和解密算法的实现：**

 static String encrypt(String plaintext) {

        int temp = Integer.parseInt(plaintext, 2);

        temp = applyPermutation(temp, IP, 8);

        temp = applyF(temp, subkey1);

        temp = ((temp & 0xf) << 4) | ((temp >> 4) & 0xf);

        temp = applyF(temp, subkey2);

        temp = applyPermutation(temp, IP\_1, 8);

        String ciphertext = Integer.toBinaryString(temp);

        while (ciphertext.length() < 8) {

            ciphertext = "0" + ciphertext;

        }

        return ciphertext;

    }

    static String decrypt(String ciphertext) {

        int temp = Integer.parseInt(ciphertext, 2);

        temp = applyPermutation(temp, IP, 8);

        temp = applyF(temp, subkey2);

        temp = ((temp & 0xf) << 4) | ((temp >> 4) & 0xf);

        temp = applyF(temp, subkey1);

        temp = applyPermutation(temp, IP\_1, 8);

        String plaintext = Integer.toBinaryString(temp);

        while (plaintext.length() < 8) {

            plaintext = "0" + plaintext;

        }

        return plaintext;

}

**根据用户选择的功能分别进行二进制加解密和ASCII码加解密：**

public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        System.out.println("请选择操作：");

        System.out.println("1. 加密");

        System.out.println("2. 解密");

        System.out.println("3. ASCII加密");

        System.out.println("4. ASCII解密");

        int choice = scan.nextInt();

        scan.nextLine();  // 消耗掉换行符

        if (choice == 1) {

            System.out.println("请输入10位二进制密钥：");

            String key = scan.nextLine();

            System.out.println("请输入8位二进制明文：");

            String plaintext = scan.nextLine();

            generateSubkeys(key);

            String ciphertext = encrypt(plaintext);

            System.out.println("加密后的密文: " + ciphertext);

        } else if (choice == 2) {

            System.out.println("请输入10位二进制密钥：");

            String key = scan.nextLine();

            System.out.println("请输入8位二进制密文：");

            String ciphertext = scan.nextLine();

            generateSubkeys(key);

            String plaintext = decrypt(ciphertext);

            System.out.println("解密后的明文: " + plaintext);

        } else if (choice == 3) {

            System.out.println("请输入10位二进制密钥：");

            String key = scan.nextLine();

            System.out.println("请输入ASCII明文：");

            String plaintext = scan.nextLine();

            // 将ASCII明文转换为二进制字符串

            StringBuilder binaryPlaintext = new StringBuilder();

            for (char c : plaintext.toCharArray()) {

                binaryPlaintext.append(String.format("%8s", Integer.toBinaryString(c)).replace(' ', '0'));

            }

            generateSubkeys(key);

            String ciphertext = encrypt(binaryPlaintext.toString());

            // 将二进制密文转换为ASCII字符串

            StringBuilder asciiCiphertext = new StringBuilder();

            for (int i = 0; i < ciphertext.length(); i += 8) {

                String byteStr = ciphertext.substring(i, i + 8);

                int byteValue = Integer.parseInt(byteStr, 2);

                asciiCiphertext.append((char) byteValue);

            }

            System.out.println("加密后的密文: " + asciiCiphertext.toString());

        } else if (choice == 4) {

            System.out.println("请输入10位二进制密钥：");

            String key = scan.nextLine();

            System.out.println("请输入ASCII密文：");

            String ciphertext = scan.nextLine();

            generateSubkeys(key);

            // 将ASCII密文转换为二进制字符串

            StringBuilder binaryCiphertext = new StringBuilder();

            for (char c : ciphertext.toCharArray()) {

                binaryCiphertext.append(String.format("%8s", Integer.toBinaryString(c)).replace(' ', '0'));

            }

            String plaintextBinary = decrypt(binaryCiphertext.toString());

            // 将二进制明文转换为ASCII字符串

            StringBuilder plaintext = new StringBuilder();

            for (int i = 0; i < plaintextBinary.length(); i += 8) {

                String byteStr = plaintextBinary.substring(i, i + 8);

                int byteValue = Integer.parseInt(byteStr, 2);

                plaintext.append((char) byteValue);

            }

            System.out.println("解密后的明文: " + plaintext.toString());

        } else {

            System.out.println("无效的选择。");

        }

    }

}

#### SDESBreakTherad.java

**SDESBreakTherad.java运用多线程破解的思想，提升了暴力破解的效率：**

public class SDESBreakerThreadPool {

    private static final String ciphertext = "11111011"; // 替换为你的密文

    private static final String plaintext = "01010101"; // 替换为你的明文

    private static final int NUM\_THREADS = 4; // 定义线程数量

    private static final int SUBKEY\_SPACE\_SIZE = 1024; // 定义密钥空间大小

    public static void main(String[] args) {

        ExecutorService executor = Executors.newFixedThreadPool(NUM\_THREADS);

        for (int i = 0; i < NUM\_THREADS; i++) {

            final int startKey = i \* (SUBKEY\_SPACE\_SIZE / NUM\_THREADS);

            final int endKey = (i + 1) \* (SUBKEY\_SPACE\_SIZE / NUM\_THREADS);

            executor.submit(new Runnable() {

                @Override

                public void run() {

                    crackKeys(startKey, endKey);

                }

            });

        }

        executor.shutdown();

        try {

            executor.awaitTermination(Long.MAX\_VALUE, TimeUnit.SECONDS);

        } catch (InterruptedException e) {

            e.printStackTrace();

        }

    }

    private static void crackKeys(int startKey, int endKey) {

        for (int key = startKey; key < endKey; key++) {

            String keyString = String.format("%10s", Integer.toBinaryString(key)).replace(' ', '0');

            S\_DES.generateSubkeys(keyString); // 生成子密钥

            String decrypted = S\_DES.decrypt(ciphertext); // 使用当前密钥解密密文

            if (decrypted.equals(plaintext)) { // 检查是否解密后的明文与给定明文匹配

                System.out.println("找到正确的密钥：" + keyString);

                //System.exit(0); // 找到密钥后退出程序

            }

            else{

                System.out.println("ffff");

            }

        }

    }

}

#### CollisionDetection.java

**因为S-DES算法的密钥空间较小，所以采用遍历的方法来寻找碰撞密钥和碰撞次数：**

public class CollisionDetection {

    public static void main(String[] args) {

        String givenPlaintext = "01010101"; // 给定的明文

        String givenCiphertext = "11111010"; // 对应的密文

        int numKeysToGenerate = 1024; // 要生成的密钥数量

        int collisionCount = 0; // 记录碰撞的次数

        List<String> collisionKeys = new ArrayList<>(); // 存储碰撞密钥的列表

        for (int i = 0; i < numKeysToGenerate; i++) {

            String key = generateRandomKey(); // 生成随机密钥

            S\_DES.generateSubkeys(key); // 生成子密钥

            String ciphertext = S\_DES.encrypt(givenPlaintext); // 使用当前密钥加密给定的明文

            if (ciphertext.equals(givenCiphertext)) {

                System.out.println("找到碰撞密钥：");

                System.out.println("给定的明文：" + givenPlaintext);

                System.out.println("给定的密文：" + givenCiphertext);

                System.out.println("碰撞密钥：" + key);

                collisionKeys.add(key); // 将碰撞密钥添加到列表中

                collisionCount++; // 增加碰撞次数

            }

        }

        if (collisionCount == 0) {

            System.out.println("未找到碰撞密钥。");

        } else {

            System.out.println("总共找到 " + collisionCount + " 个碰撞密钥：");

            for (String collisionKey : collisionKeys) {

                System.out.println("碰撞密钥：" + collisionKey);

            }

        }

    }

    private static String generateRandomKey() {

        Random random = new Random();

        StringBuilder key = new StringBuilder();

        for (int i = 0; i < 10; i++) {

            int randomBit = random.nextInt(2);

            key.append(randomBit);

        }

        return key.toString();

    }

}