Practice 9: HASH FUNCTION

9.1 OVERVIEW

9.1.1 Introduction

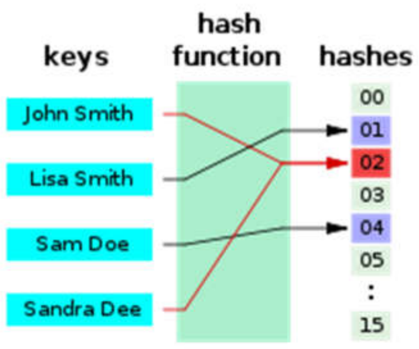
- Lab 9: Hash Function
- Practice time: class: 3 study hours, self-study: 3 study hours.
- Requirements: Students using Netbeans Software

9.1.2 Objective

- This course provides students with knowledge of cryptographic algorithms and how they are used in today's world.
- The content emphasizes the principles, topics, approaches, and problem solving related to the underlying technologies and architectures of the field.

9.2 CONTENTS

9.2.1 Basic knowledge



A hash function is any function that can be used to map data of arbitrary size to fixed-size values. Hash values, hash codes, digests, or just hashes are the names given to the results of a hash function. The values are often used to index a hash table, a fixed-size table. It is referred to as hashing or scatter storage addressing when a hash function is used to index a hash table.

Three tasks might be regarded to be accomplished by a hash function:

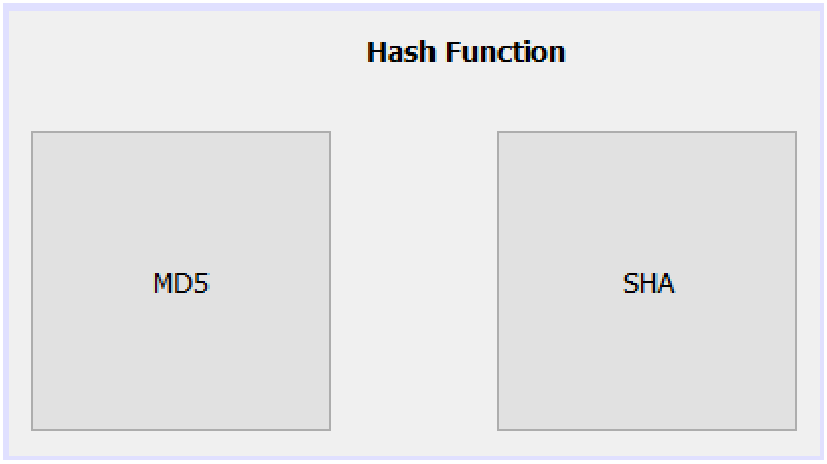
– Convert variable-length keys into fixed length (usually machine word length or less) values, by folding them by words or other units using a parity-preserving operator like ADD or XOR.
– Scramble the bits of the key so that the resulting values are uniformly distributed over the keyspace.
– Map the key values into ones less than or equal to the size of the table.

9.2.2 MD5

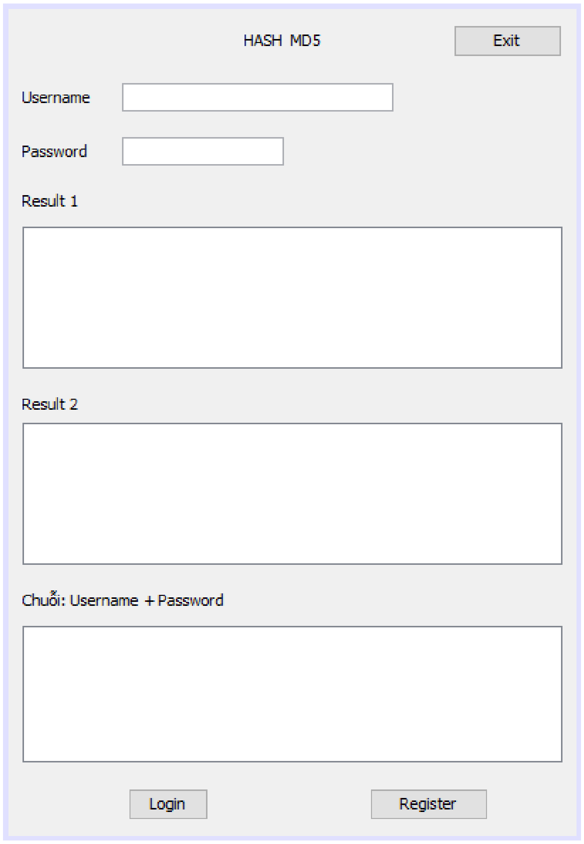
The MD5 (Message-Digest Algorithm 5) is a cryptographically broken but still widely used hash function producing a 128-bit hash value.

Implement the MD5 hash algorithm with the following requirements:
– Allow users to enter username and password.
– Use MD5 algorithm to hash username and password and save it to File.
– Use username and password to log in, authenticate with the file with username and password recorded.

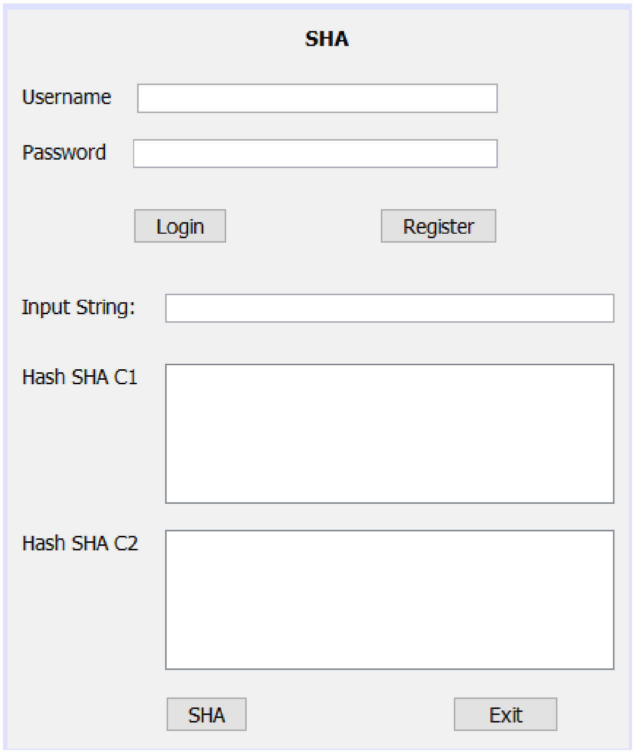
❖ **Step 1: Design Form:**
- **1.1 Main Form:**



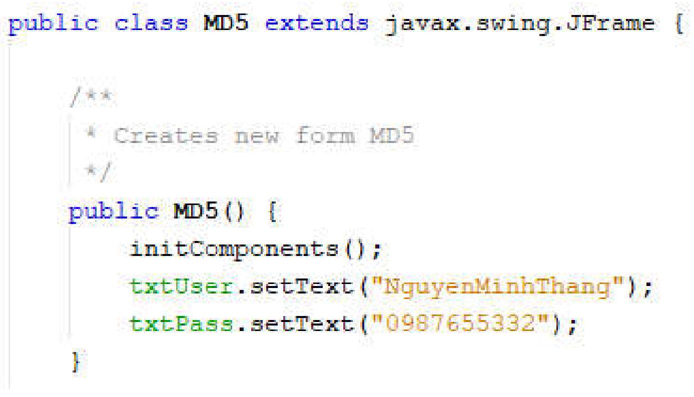
- ### 1.2 MD5 Fom:



- ### 1.3 SHA Form:



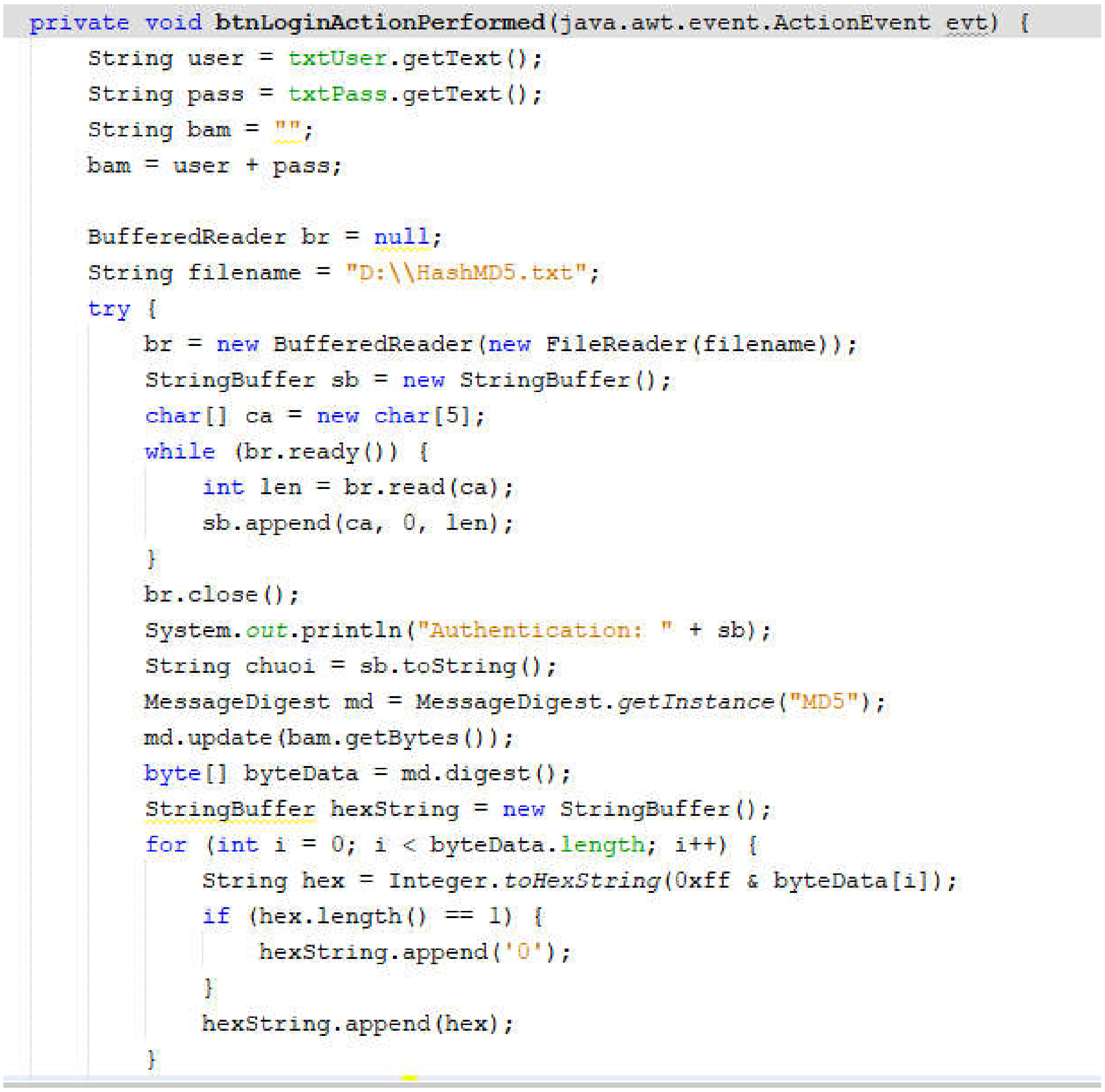
❖ **Step 2: rite code for initialization function:**



```
public class MD5 extends javax.swing.JFrame {

    /**
     * Creates new form MD5
     */
    public MD5() {
        initComponents();
        txtUser.setText("NguyenMinhThang");
        txtPass.setText("0987655332");
    }
```

❖ **Step 3: Write an event handler function for MD5 Form:**

3.1 Button Login:



```
private void btnLoginActionPerformed(java.awt.event.ActionEvent evt) {
    String user = txtUser.getText();
    String pass = txtPass.getText();
    String bam = "";
    bam = user + pass;

    BufferedReader br = null;
    String filename = "D:\\HashMD5.txt";
    try {
        br = new BufferedReader(new FileReader(filename));
        StringBuffer sb = new StringBuffer();
        char[] ca = new char[5];
        while (br.ready()) {
            int len = br.read(ca);
            sb.append(ca, 0, len);
        }
        br.close();
        System.out.println("Authentication: " + sb);
        String chuoi = sb.toString();
        MessageDigest md = MessageDigest.getInstance("MD5");
        md.update(bam.getBytes());
        byte[] byteData = md.digest();
        StringBuffer hexString = new StringBuffer();
        for (int i = 0; i < byteData.length; i++) {
            String hex = Integer.toHexString(0xff & byteData[i]);
            if (hex.length() == 1) {
                hexString.append('0');
            }
            hexString.append(hex);
        }
```

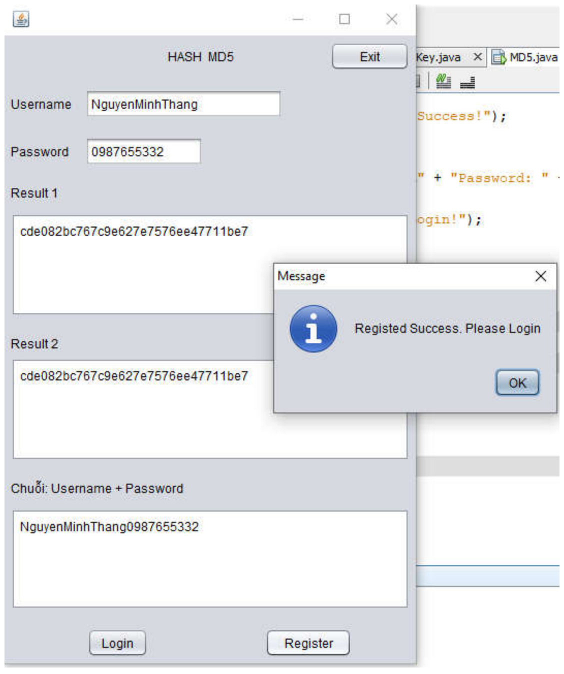
3.2 Button Resgister:

```
private void btnRegisterActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String user = txtUser.getText();
        String pass = txtPass.getText();
        String bam = "";
        bam = user + pass;

        MessageDigest md = MessageDigest.getInstance("MD5");
        md.update(bam.getBytes());
        byte[] byteData = md.digest();

        StringBuffer sb = new StringBuffer();
        for (int i = 0; i < byteData.length; i++) {
            sb.append(Integer.toString((byteData[i] & 0xff) + 0x100, 16).substring(1));
        }
        System.out.println("Digest(in hex format):: " + sb.toString());
        txtHash1.setText(sb.toString());

        StringBuffer hexString = new StringBuffer();
        for (int i = 0; i < byteData.length; i++) {
            String hex = Integer.toHexString(0xff & byteData[i]);
            if (hex.length() == 1) {
                hexString.append('0');
            }
            hexString.append(hex);
        }
        System.out.println("Digest(in hex format):: " + hexString.toString());
        txtHash2.setText(hexString.toString());
        txtString.setText(bam.toString());

        BufferedWriter bw = null;
        String filename = "D:\\HashMD5.txt";
        bw = new BufferedWriter(new FileWriter(filename));
        bw.write(hexString.toString());
        bw.close();
        JOptionPane.showMessageDialog(null, "Registed Success. Please Login");
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(null, "Error Hash User and Pass: " + ex);
    }

}
```

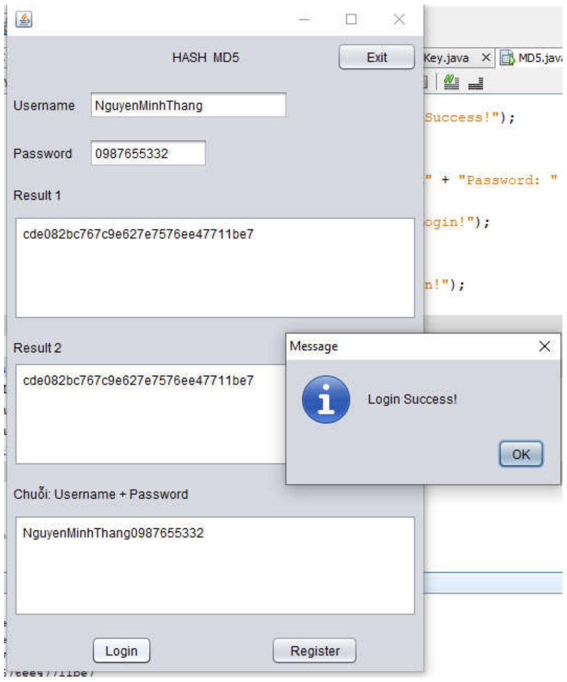
3.4 Button Exit

```
private void btnExitActionPerformed(java.awt.event.ActionEvent evt) {
    Main main = new Main();
    main.setLocationRelativeTo(null);
    main.~~show~~();
    this.dispose();
}
```

Result:



```
Username: NguyenMinhThang
Password: 0987655332
Result 1: cde082bc767c9e627e7576ee47711be7
Result 2: cde082bc767c9e627e7576ee47711be7
Chuỗi: Username + Password: NguyenMinhThang0987655332
Registed Success. Please Login
```



```
Username: NguyenMinhThang
Password: 0987655332
Result 1: cde082bc767c9e627e7576ee47711be7
Result 2: cde082bc767c9e627e7576ee47711be7
Chuỗi: Username + Password: NguyenMinhThang0987655332
Login Success!
```

9.2.3 SHA

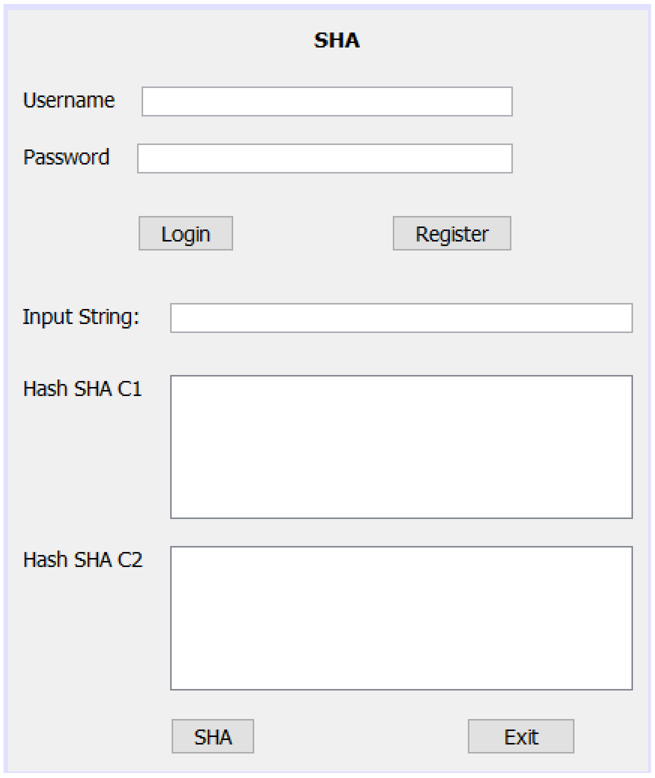
SHA-0: A retronym used to refer to the 160-bit hash function's first iteration, which was first published in 1993 under the name "SHA." It had a "significant flaw" that wasn't reported, therefore it was pulled from circulation soon after release. It was replaced with the slightly updated SHA-1.

SHA-1: A hash function of 160 bits that is similar to the older MD5 technique. The National Security Agency (NSA) created this to be a component of the digital signature algorithm. After SHA-1's cryptographic flaws were found, the standard was abandoned for the majority of cryptographic applications after 2010.

SHA-2: The SHA-256 and SHA-512 families are two comparable hash algorithms, each with a different block size. Word size is different between both; SHA-512 utilizes 64-bit words whereas SHA-256 uses 32-bit words. As well as full versions of each standard, there are truncated variants known as SHA-224, SHA-384, SHA-512/224, and SHA-512/256. The NSA also developed these.

SHA-3: A hash function originally known as Keccak that was selected in 2012 following an open contest among non-NSA designers. It supports the same hash lengths as SHA-2 and has a very different internal structure from the other members of the SHA family.

❖ **Design SHA Form:**

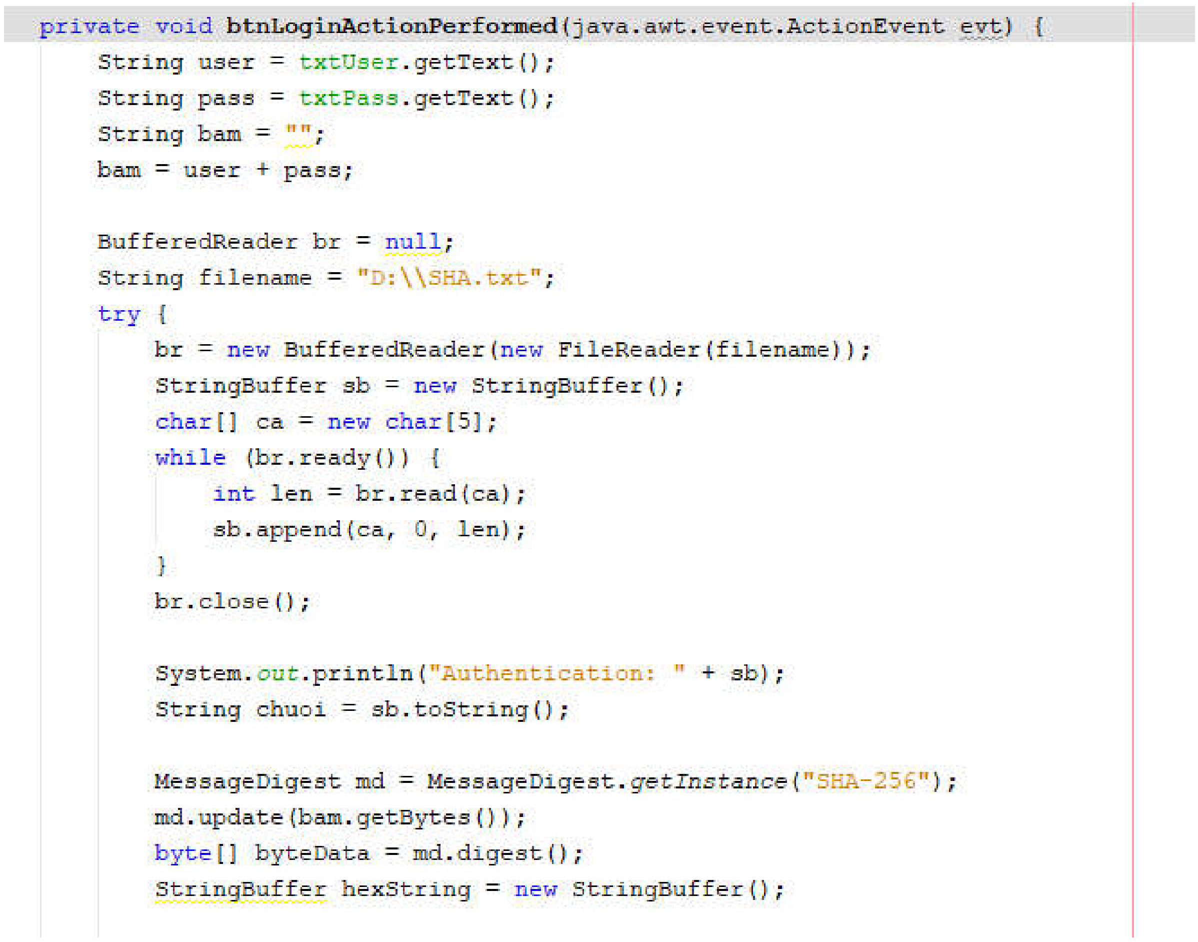


❖ **Step 2: write code for initialization function:**

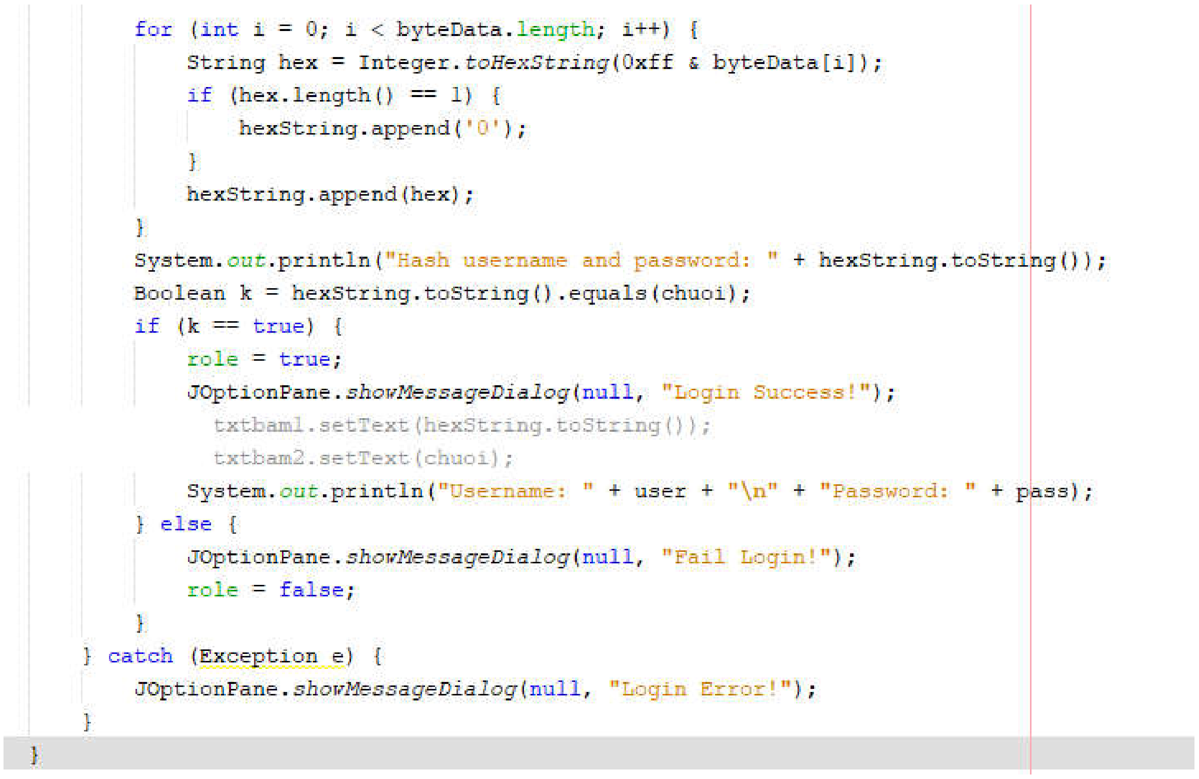
```
public class SHA extends javax.swing.JFrame {

    boolean role;

    /**
     * Creates new form SHA
     */
    public SHA() {
        initComponents();
        role = false;
        txtUser.setText("NguyenMinhThang");
        txtPass.setText("123456");
        txtString.setText("informationSecurity");
    }
```

❖ **Step 3: Write an event handler function for MD5 Form:**

3.1 Button Login:

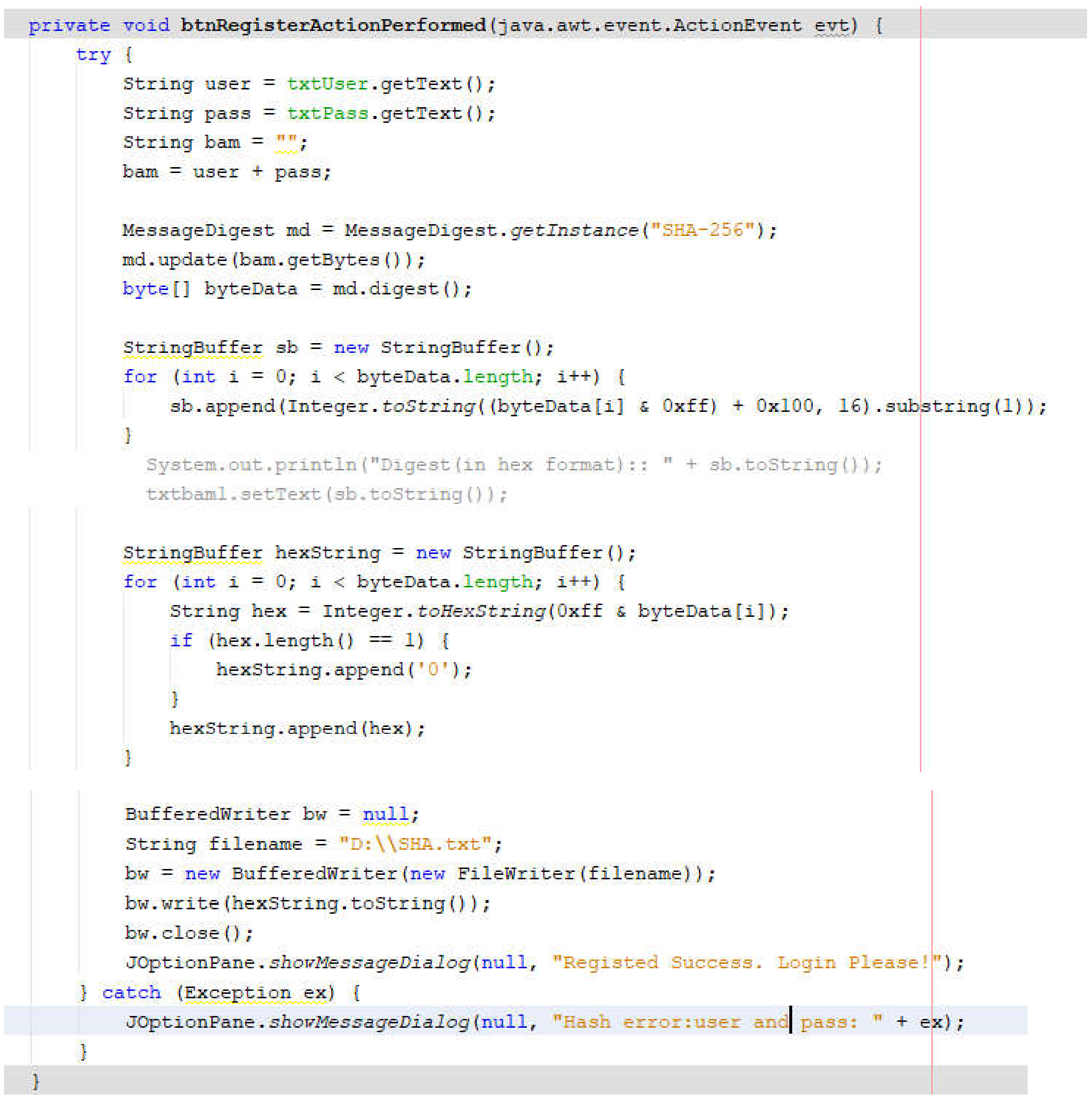


```
private void btnLoginActionPerformed(java.awt.event.ActionEvent evt) {
    String user = txtUser.getText();
    String pass = txtPass.getText();
    String bam = "";
    bam = user + pass;

    BufferedReader br = null;
    String filename = "D:\\SHA.txt";
    try {
        br = new BufferedReader(new FileReader(filename));
        StringBuffer sb = new StringBuffer();
        char[] ca = new char[5];
        while (br.ready()) {
            int len = br.read(ca);
            sb.append(ca, 0, len);
        }
        br.close();

        System.out.println("Authentication: " + sb);
        String chuoi = sb.toString();

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        md.update(bam.getBytes());
        byte[] byteData = md.digest();
        StringBuffer hexString = new StringBuffer();
```



```
        for (int i = 0; i < byteData.length; i++) {
            String hex = Integer.toHexString(0xff & byteData[i]);
            if (hex.length() == 1) {
                hexString.append('0');
            }
            hexString.append(hex);
        }
        System.out.println("Hash username and password: " + hexString.toString());
        Boolean k = hexString.toString().equals(chuoi);
        if (k == true) {
            role = true;
            JOptionPane.showMessageDialog(null, "Login Success!");
              txtbam1.setText(hexString.toString());
              txtbam2.setText(chuoi);
            System.out.println("Username: " + user + "\n" + "Password: " + pass);
        } else {
            JOptionPane.showMessageDialog(null, "Fail Login!");
            role = false;
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Login Error!");
    }
}
```

3.2 Button Resgister:



```
private void btnRegisterActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String user = txtUser.getText();
        String pass = txtPass.getText();
        String bam = "";
        bam = user + pass;

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        md.update(bam.getBytes());
        byte[] byteData = md.digest();

        StringBuffer sb = new StringBuffer();
        for (int i = 0; i < byteData.length; i++) {
            sb.append(Integer.toString((byteData[i] & 0xff) + 0x100, 16).substring(1));
        }
          System.out.println("Digest(in hex format):: " + sb.toString());
          txtbaml.setText(sb.toString());

        StringBuffer hexString = new StringBuffer();
        for (int i = 0; i < byteData.length; i++) {
            String hex = Integer.toHexString(0xff & byteData[i]);
            if (hex.length() == 1) {
                hexString.append('0');
            }
            hexString.append(hex);
        }

        BufferedWriter bw = null;
        String filename = "D:\\SHA.txt";
        bw = new BufferedWriter(new FileWriter(filename));
        bw.write(hexString.toString());
        bw.close();
        JOptionPane.showMessageDialog(null, "Registed Success. Login Please!");
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(null, "Hash error:user and pass: " + ex);
    }
}
```

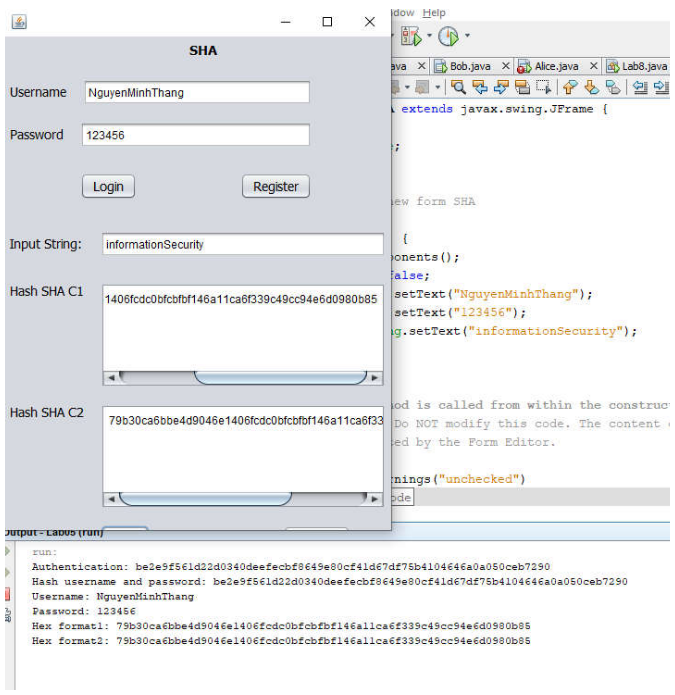
3.4 Button SHA

```java
private void btnSHAActionPerformed(java.awt.event.ActionEvent evt) {
    if (role == true) {
        try {
            String chuoi = "";
            chuoi = txtString.getText();
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            md.update(chuoi.getBytes());
            byte[] byteData = md.digest();
            StringBuffer sb = new StringBuffer();
            for (int i = 0; i < byteData.length; i++) {
                sb.append(Integer.toString((byteData[i] & 0xff) + 0x100, 16).substring(1));
            }
            System.out.println("Hex format1: " + sb.toString());
            txtSHA1.setText(sb.toString());
            StringBuffer hexString = new StringBuffer();
            for (int i = 0; i < byteData.length; i++) {
                String hex = Integer.toHexString(0xff & byteData[i]);
                if (hex.length() == 1) {
                    hexString.append('0');
                }
                hexString.append(hex);
            }
            System.out.println("Hex format2: " + hexString.toString());
            txtSHA2.setText(hexString.toString());
        } catch (NoSuchAlgorithmException ex) {
            Logger.getLogger(SHA.class.getName()).log(Level.SEVERE, null, ex);
        }
    } else {
        JOptionPane.showMessageDialog(null, "Login Please!");
    }
}
```

3.5 Button Exit

```java
private void btnExitActionPerformed(java.awt.event.ActionEvent evt) {
    Main main = new Main();
    main.setLocationRelativeTo(null);
    main.~~show~~();
    this.dispose();
}
```

Result:



Username: NguyenMinhThang
Password: 123456
Input String: informationSecurity
Hash SHA C1: 1406fcdc0bfcbfbf146a11ca6f339c49cc94e6d0980b85
Hash SHA C2: 79b30ca6bbe4d9046e1406fcdc0bfcbfbf146a11ca6f33

```
run:
Authentication: be2e9f561d22d0340deefecbf8649e80cf41d67df75b4104646a0a050ceb7290
Hash username and password: be2e9f561d22d0340deefecbf8649e80cf41d67df75b4104646a0a050ceb7290
Username: NguyenMinhThang
Password: 123456
Hex format1: 79b30ca6bbe4d9046e1406fcdc0bfcbfbf146a11ca6f339c49cc94e6d0980b85
Hex format2: 79b30ca6bbe4d9046e1406fcdc0bfcbfbf146a11ca6f339c49cc94e6d0980b85
```