

Lab 6: Hashing and Digests

Due: 10/27/09 by 11:59PM

The hash function converts data of variable sizes to a small value of data. The hash value is often referred to as a message digest or simply digest. As we have already discussed, there are many benefits and uses to hash functions (e.g. in digital signatures, password files). Two of the most common digest algorithms are MD5 and SHA-1. The Secure Hash Algorithm (SHA) was published by NIST; the algorithm has been, and is being, constantly improved. The Message Digest Algorithm 5 (MD5) is another widely used hash function.

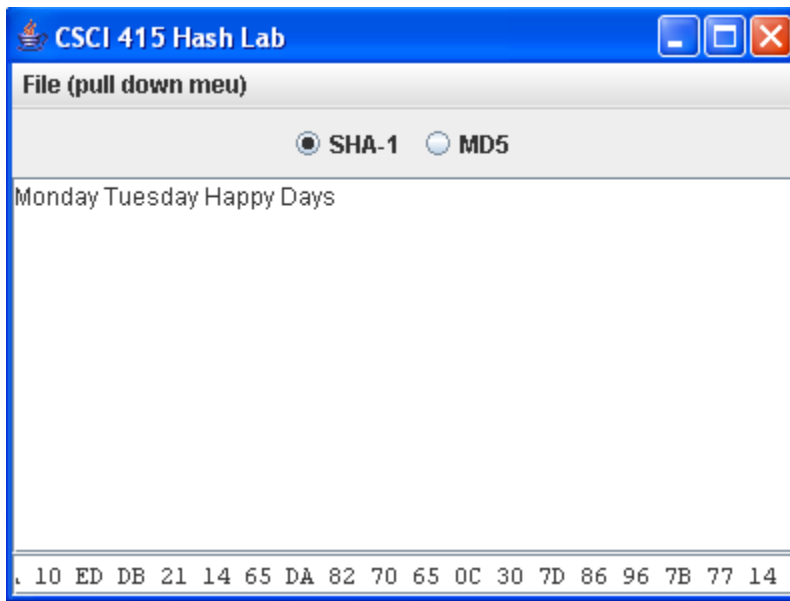
A message digest has two essential properties¹:

- If one bit or several bits of the data are changed, then the message digest also changes.
- A forger who is in possession of a given message cannot construct a fake message that has the same message digest as the original.

Objectives - to:

1. gain an understanding of the nature of hashing
2. complete a program that enable us to witness hashing in real time

Output



¹ adapted from **Core Java™ Volume II—Advanced Features, Eighth Edition**, by Cay S. Horstmann; Gary Cornell

Laboratory Activities

Complete the code¹ below and be sure to complete the **commenting**:

```
import java.io.*;
import java.security.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

// This program computes the hash function (SHA-1 or MD5) of a text or
file

public class MessageHash {
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                JFrame frame = new _____();

                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setVisible(true);
            }
        });
    }
}

// The MessageDigestFrame contains:
// radio buttons for the 2 hash functions
// the text area
// the text field that displays the hash
// the pull down menu to compute the digest of either the file or text

class MessageDigestFrame extends JFrame {
    public MessageDigestFrame() {
        setTitle("CSCI 415 Hash Lab");
        setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);

        JPanel panel = new JPanel();
        ButtonGroup group = _____);
        addRadioButton(panel, "SHA-1", group);
        addRadioButton(panel, _____);

        add(panel, BorderLayout.NORTH);
        add(new JScrollPane(message), BorderLayout.CENTER);
        add(digest, BorderLayout.SOUTH);
        digest.setFont(new Font("Monospaced", Font.PLAIN, 12));

        setAlgorithm("SHA-1");

        JMenuBar menuBar = new JMenuBar();
        JMenu menu = new JMenu("File (pull down menu)");
        JMenuItem fileDigestItem = new JMenuItem("File digest");
        fileDigestItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent event) {
                loadFile();
            }
        });
        menu.add(fileDigestItem);
        JMenuItem textDigestItem = new JMenuItem("Text digest");
        textDigestItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent event) {
                String m = message.getText();
            }
        });
    }
}
```

```

        computeDigest(____.getBytes());
    }
});
menu.add(textDigestItem);
menuBar.add(menu);
setJMenuBar(menuBar);
}

// c: the container into which the button is placed
// name: of the algorithm
// g: the button group

public void addRadioButton(Container ____, final String name,
ButtonGroup _____) {
    ActionListener listener = _____() {
        public void actionPerformed(ActionEvent event) {
            setAlgorithm(_____);
        }
    };
    JRadioButton b = new JRadioButton(name, g.getButtonCount()
== 0);
    c.add(b);
    _____.add(b);
    _____.addActionListener(listener);
}

// algorithm to compute digest
// alg: name of algorithm

public void setAlgorithm(String _____) {
    try {
        currentAlgorithm = MessageDigest.getInstance(_____);
        digest.setText("");
    } catch (NoSuchAlgorithmException e) {
        digest.setText("" + e);
    }
}

// to load a file and compute its message digest

public void loadFile() {
    JFileChooser chooser = new JFileChooser();
    chooser.setCurrentDirectory(new File("."));

    int r = chooser.showOpenDialog(this);
    if (r == JFileChooser.APPROVE_OPTION) {
        try {
            String name =
chooser.getSelectedFile().getAbsolutePath();
            computeDigest(loadBytes(name));
        } catch (IOException e) {
            JOptionPane.showMessageDialog(null, e);
        }
    }
}

// name: the name of the file
// return an array with the bytes in the file (which are loaded
here)

public byte[] loadBytes(String _____) throws IOException {
    FileInputStream in = null;

```

```
        in = new FileInputStream(______);
        try {
            ByteArrayOutputStream buffer = new
ByteArrayOutputStream();
            int ch;
            while ((ch = in.read()) != -1)
                buffer.write(ch);
            return buffer.toByteArray();
        } finally {
            in.close();
        }
    }

    // b: the bytes for which the message digest should be computed
    // computes and displays the message digest of an array of bytes

    public void computeDigest(byte[] b) {
        currentAlgorithm.reset();
        currentAlgorithm.update(______);
        byte[] hash = currentAlgorithm.digest();
        String d = "";
        for (int i = 0; i < hash.length; i++) {
            int v = hash[i] & 0xFF;
            if (v < 16)
                d += "0";
            d += Integer.toString(v, ____).toUpperCase() + " ";
        }
        digest.setText(d);
    }

    private JTextArea message = new JTextArea();
    private JTextField digest = new JTextField();
    private MessageDigest currentAlgorithm;
    private static final int DEFAULT_WIDTH = 400;
    private static final int DEFAULT_HEIGHT = 300;
}
```