##ResultSystem.java  
import java.awt.\*;  
import java.awt.event.\*;  
import javax.swing.\*;  
import java.time.LocalTime;  
import java.time.format.DateTimeFormatter;  
import java.io.\*;  
import javax.swing.table.\*;  
  
public class ResultSystem extends JFrame  
{  
    public enum UserType  
    {  
        TEACHER,  
        STUDENTRESULT;  
    }  
     
    private JLabel timeLabel;  
     
    public ResultSystem()  
    {  
        setTitle(&quot;Student Result System&quot;);  
        setLayout(new FlowLayout());  
        ImageIcon icon = new ImageIcon(&quot;favicon.ico&quot;);  
     
        JButton teacherButton = new JButton(&quot;Teacher&quot;);  
        teacherButton.addActionListener(new UserTypeButtonListener(UserType.TEACHER));  
        JButton studentResultButton = new JButton(&quot;Student Result&quot;);  
        studentResultButton.addActionListener(new  
UserTypeButtonListener(UserType.STUDENTRESULT));  
         
        timeLabel = new JLabel();  
        updateTimeLabel();  
         
        add(timeLabel);  
        add(teacherButton);  
        add(studentResultButton);  
        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
        setSize(450, 200);  
        setVisible(true);  
        setIconImage(icon.getImage());  
        Timer timer = new Timer(1000, new ActionListener()  
  
        {  
            public void actionPerformed(ActionEvent e)  
            {  
                updateTimeLabel();  
            }  
        });  
        timer.start();  
    }  
     
    private void updateTimeLabel()  
    {  
        LocalTime currentTime = LocalTime.now();  
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern(&quot;hh:mm:ss a&quot;);  
        String formattedTime = currentTime.format(formatter);  
        timeLabel.setText(formattedTime);  
    }  
public class ResultCreationWindow extends JFrame  
{  
    private JLabel resultLabel;  
    private JTextField resultField;  
    private JTable resultTable;  
    private DefaultTableModel tableModel;  
    private JScrollPane tableScrollPane;  
    private JButton saveButton;  
    private JButton showAnswersButton;  
    public ResultCreationWindow()  
    {  
        setTitle(&quot;Create Results&quot;);  
        setLayout(new FlowLayout());  
        resultLabel = new JLabel(&quot;Enter result: &quot;);  
        resultField = new JTextField(30);  
        tableModel = new DefaultTableModel(new String[] {&quot;Serial&quot;, &quot;Result&quot;}, 0);  
        resultTable = new JTable(tableModel);  
        tableScrollPane = new JScrollPane(resultTable);  
        try {  
            BufferedReader reader = new BufferedReader(new FileReader(&quot;results.txt&quot;));  
            String line = reader.readLine();  
            int serial = 1;  
            while (line != null)  
            {  
                tableModel.addRow(new String[] {String.valueOf(serial), line});  
                line = reader.readLine();  
                serial++;  
            }  
  
            reader.close();  
        } catch (IOException ex) {  
            ex.printStackTrace();  
        }  
        saveButton = new JButton(&quot;Save&quot;);  
        saveButton.addActionListener(new SaveButtonListener());  
        showAnswersButton = new JButton(&quot;Show Answers&quot;);  
        showAnswersButton.addActionListener(new ShowAnswersButtonListener());  
        add(resultLabel);  
        add(resultField);  
        add(saveButton);  
        add(showAnswersButton);  
        add(tableScrollPane);  
        setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);  
        setSize(800, 600);  
        setVisible(true);  
    }  
    private class SaveButtonListener implements ActionListener  
    {  
        public void actionPerformed(ActionEvent e)  
        {  
            String question = resultField.getText();  
            try {  
                BufferedWriter writer = new BufferedWriter(new FileWriter(&quot;results.txt&quot;,  
true));  
                writer.append(question + &quot;\n&quot;);  
                writer.close();  
            } catch (IOException ex) {  
                ex.printStackTrace();  
            }  
            resultField.setText(&quot;&quot;);  
            int serial = tableModel.getRowCount() + 1;  
            tableModel.addRow(new String[] {String.valueOf(serial), question});  
        }  
    }  
    private class ShowAnswersButtonListener implements ActionListener  
    {  
        public void actionPerformed(ActionEvent e)  
        {  
            new ShowAnswersWindow();  
        }  
    }  
  
}  
  
public class ShowAnswersWindow extends JFrame  
{  
    private JTextArea answerArea;  
    private JButton previousButton;  
    private JButton nextButton;  
    private File[] answerFiles;  
    private int currentIndex;  
    public ShowAnswersWindow()  
    {  
        setTitle(&quot;Show Answers&quot;);  
        setLayout(new BorderLayout());  
        answerArea = new JTextArea(10, 30);  
        answerArea.setEditable(false);  
        add(new JScrollPane(answerArea), BorderLayout.CENTER);  
        JPanel bottomPanel = new JPanel();  
        bottomPanel.setLayout(new BoxLayout(bottomPanel, BoxLayout.X\_AXIS));  
        previousButton = new JButton(&quot;Previous&quot;);  
        previousButton.addActionListener(new PreviousButtonListener());  
        nextButton = new JButton(&quot;Next&quot;);  
        nextButton.addActionListener(new NextButtonListener());  
        bottomPanel.add(previousButton);  
        bottomPanel.add(nextButton);  
        add(bottomPanel, BorderLayout.SOUTH);  
        File answersFolder = new File(&quot;answers&quot;);  
        answerFiles = answersFolder.listFiles(new FilenameFilter()  
        {  
            @Override  
            public boolean accept(File dir, String name)  
            {  
                return name.endsWith(&quot;.txt&quot;);  
            }  
        });  
        if (answerFiles.length &gt; 0)  
        {  
            currentIndex = 0;  
            loadAnswerFile(answerFiles[currentIndex]);  
        }  
        setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);  
        setSize(400, 300);  
        setVisible(true);  
  
    }  
    private void loadAnswerFile(File answerFile)  
    {  
        try  
        {  
            BufferedReader reader = new BufferedReader(new FileReader(answerFile));  
            String line = reader.readLine();  
            while (line != null)  
            {  
                answerArea.append(line + &quot;\n&quot;);  
                line = reader.readLine();  
            }  
            reader.close();  
            setTitle(&quot;Show Answers - &quot; + answerFile.getName());  
        } catch (IOException ex) {  
            ex.printStackTrace();  
        }  
    }  
    private class PreviousButtonListener implements ActionListener  
    {  
        public void actionPerformed(ActionEvent e)  
        {  
            if (currentIndex &gt; 0)  
            {  
                currentIndex--;  
                answerArea.setText(&quot;&quot;);  
                loadAnswerFile(answerFiles[currentIndex]);  
            }  
        }  
    }  
    private class NextButtonListener implements ActionListener  
    {  
        public void actionPerformed(ActionEvent e)  
        {  
            if (currentIndex &lt; answerFiles.length - 1)  
            {  
                currentIndex++;  
                answerArea.setText(&quot;&quot;);  
                loadAnswerFile(answerFiles[currentIndex]);  
            }  
        }  
    }  
}  
public class TeacherPasswordWindow extends JFrame  
{  
  
    private JLabel passwordLabel;  
    private JPasswordField passwordField;  
    private JButton submitButton;  
    private String password;  
    public TeacherPasswordWindow()  
    {  
      setTitle(&quot;Teacher Password&quot;);  
      setLayout(new FlowLayout());  
      passwordLabel = new JLabel(&quot;Enter password: &quot;);  
      passwordField = new JPasswordField(10);  
      submitButton = new JButton(&quot;Submit&quot;);  
      submitButton.addActionListener(new SubmitButtonListener());  
      add(passwordLabel);  
      add(passwordField);  
      add(submitButton);  
      setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);  
      setSize(350, 100);  
      setVisible(true);  
    }  
    private class SubmitButtonListener implements ActionListener  
    {  
        public void actionPerformed(ActionEvent e)  
        {  
            char[] enteredPassword = passwordField.getPassword();  
            password = new String(enteredPassword);  
            if (password.equals(&quot;admin&quot;)) {  
              new ResultCreationWindow();  
            } else {  
              JOptionPane.showMessageDialog(null, &quot;Incorrect password. Please try  
again.&quot;);  
            }  
        }  
    }  
}  
public class StudentResultWindow extends JFrame  
{  
    private JLabel questionLabel;  
    private JTextArea questionArea;  
    private JLabel answerLabel;  
    private JTextArea answerField;  
    private JLabel nameLabel;  
    private JTextField nameField;  
  
    private JLabel idLabel;  
    private JTextField idField;  
    private JButton submitButton;  
    public StudentResultWindow()  
    {  
        setTitle(&quot;Take Result&quot;);  
        setLayout(new FlowLayout());  
        questionLabel = new JLabel(&quot;Question: &quot;);  
        questionArea = new JTextArea(10, 40);  
        questionArea.setEditable(false);  
        try {  
            BufferedReader reader = new BufferedReader(new FileReader(&quot;questions.txt&quot;));  
            String line = reader.readLine();  
            while (line != null) {  
                questionArea.append(line + &quot;\n&quot;);  
                line = reader.readLine();  
            }  
            reader.close();  
        } catch (IOException ex) {  
            ex.printStackTrace();  
        }  
        answerLabel = new JLabel(&quot;Answer: &quot;);  
        answerField = new JTextArea(10, 40);  
        nameLabel = new JLabel(&quot;Name: &quot;);  
        nameField = new JTextField(20);  
        idLabel = new JLabel(&quot;ID: &quot;);  
        idField = new JTextField(20);  
        submitButton = new JButton(&quot;Submit&quot;);  
        submitButton.addActionListener(new SubmitButtonListener());  
        add(nameLabel);  
        add(nameField);  
        add(idLabel);  
        add(idField);  
        add(questionLabel);  
        add(questionArea);  
        add(answerLabel);  
        add(answerField);  
        add(submitButton);  
        setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);  
        setSize(570, 470);  
        setVisible(true);  
  
    }  
    private class SubmitButtonListener implements ActionListener  
    {  
        public void actionPerformed(ActionEvent e)  
        {  
            String answer = answerField.getText();  
            String name = nameField.getText();  
            String id = idField.getText();  
            File answersFolder = new File(&quot;answers&quot;);  
            if (!answersFolder.exists()) {  
                answersFolder.mkdir();  
            }  
            File answersFile = new File(answersFolder, id + &quot;.txt&quot;);  
            try {  
                BufferedWriter writer = new BufferedWriter(new FileWriter(answersFile,  
true));  
                writer.append(name + &quot;\n\n&quot; + id + &quot;\n\n&quot; + answer + &quot;\n&quot;);  
                writer.close();  
            } catch (IOException ex) {  
                ex.printStackTrace();  
            }  
            answerField.setText(&quot;&quot;);  
            nameField.setText(&quot;&quot;);  
            idField.setText(&quot;&quot;);  
        }  
    }  
}  
    private class UserTypeButtonListener implements ActionListener  
    {  
        private UserType userType;  
        public UserTypeButtonListener(UserType userType)  
        {  
            this.userType = userType;  
        }  
        public void actionPerformed(ActionEvent e)  
        {  
            switch (userType) {  
                case TEACHER:  
                    new TeacherPasswordWindow();  
                    break;  
                case STUDENTRESULT:  
                    new StudentResultWindow();  
  
                    break;  
            }  
        }  
    }  
}