import java.awt.\*;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.HashSet;

import java.util.Set;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

public class Projecto2 extends *JFrame* {

    private JLabel label;

    private JPanel painelPrincipal;

    private GridBagConstraints gbc;

    // Campos de entrada

    private JTextField campoCliente;

    private JTextField campoDestino;

    private JTextField campoTempo;

    // Tabela e modelo

    private JTable tabela;

    private DefaultTableModel tabelaModelo;

    public Projecto2() {

        setTitle("Aplicativo Comunicação");

        setSize(400, 300);

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setLocationRelativeTo(null);

        // Painel lateral

        JPanel barraLateral = new JPanel();

        barraLateral.setLayout(new BoxLayout(barraLateral, BoxLayout.Y\_AXIS));

        barraLateral.setBackground(new Color(50, 50, 50));

        barraLateral.setPreferredSize(new Dimension(150, getHeight()));

        JButton btnInicio = new JButton("Inicio");

        JButton btnMarcar = new JButton("Marcação");

        JButton btnConsultar = new JButton("Consultar");

        for (JButton btn : new JButton[]{btnInicio, btnMarcar, btnConsultar}) {

            btn.setMaximumSize(new Dimension(300, 30));

            btn.setAlignmentX(Component.CENTER\_ALIGNMENT);

            barraLateral.add(Box.createRigidArea(new Dimension(0, 15)));

            barraLateral.add(btn);

        }

        // Painel principal

        painelPrincipal = new JPanel(new GridBagLayout());

        gbc = new GridBagConstraints();

        label = new JLabel("Bem-Vindo!", SwingConstants.CENTER);

        label.setFont(new Font("Arial", Font.BOLD, 24));

        gbc.gridx = 0;

        gbc.gridy = 0;

        gbc.weightx = 1.0;

        gbc.weighty = 1.0;

        gbc.anchor = GridBagConstraints.CENTER;

        painelPrincipal.add(label, gbc);

        // Ações dos botões

        btnInicio.addActionListener(e -> {

            painelPrincipal.removeAll();

            label.setText("Página Inicial");

            painelPrincipal.add(label, gbc);

            painelPrincipal.revalidate();

            painelPrincipal.repaint();

        });

        btnMarcar.addActionListener(e -> mostrarFormularioMarcar());

        btnConsultar.addActionListener(e -> mostrarFormularioConsulta());

        getContentPane().setLayout(new BorderLayout());

        getContentPane().add(barraLateral, BorderLayout.WEST);

        getContentPane().add(painelPrincipal, BorderLayout.CENTER);

    }

    private void mostrarFormularioConsulta() {

        painelPrincipal.removeAll();

        painelPrincipal.setLayout(new GridBagLayout());

        GridBagConstraints gc = new GridBagConstraints();

        JLabel lblCliente = new JLabel("Cliente:");

        JTextField campoConsultaCliente = new JTextField(10);

        JButton btnConsultar = new JButton("Consultar");

        JButton btnGerarFactura = new JButton("Gerar Factura");

        JButton btnEliminar = new JButton("Eliminar"); // Novo botão

        gc.insets = new Insets(10, 10, 10, 10);

        gc.gridx = 0;

        gc.gridy = 0;

        painelPrincipal.add(lblCliente, gc);

        gc.gridx = 1;

        painelPrincipal.add(campoConsultaCliente, gc);

        gc.gridx = 2;

        painelPrincipal.add(btnConsultar, gc);

        gc.gridx = 3;

        painelPrincipal.add(btnGerarFactura, gc);

        gc.gridx = 4;

        painelPrincipal.add(btnEliminar, gc); // Adiciona botão "Eliminar"

        // Tabela

        String[] colunas = {"Cliente", "Destino", "Tempo (s)", "Região", "Valor Total"};

        tabelaModelo = new DefaultTableModel(colunas, 0);

        tabela = new JTable(tabelaModelo);

        JScrollPane scrollPane = new JScrollPane(tabela);

        scrollPane.setPreferredSize(new Dimension(750, 300));

        gc.gridx = 0;

        gc.gridy = 1;

        gc.gridwidth = 5;

        gc.fill = GridBagConstraints.HORIZONTAL;

        painelPrincipal.add(scrollPane, gc);

        // CONSULTAR

        btnConsultar.addActionListener(e -> {

            String nomeCliente = campoConsultaCliente.getText().trim();

            tabelaModelo.setRowCount(0);

            if (nomeCliente.isEmpty()) {

                JOptionPane.showMessageDialog(this, "Digite o nome do cliente para consultar.", "Aviso", JOptionPane.WARNING\_MESSAGE);

                return;

            }

            try (BufferedReader br = new BufferedReader(new FileReader("Clientes.csv"))) {

                String linha;

                while ((linha = br.readLine()) != null) {

                    String[] partes = linha.split(",");

                    if (partes.length >= 3 && partes[0].equalsIgnoreCase(nomeCliente)) {

                        String cliente = partes[0];

                        String destino = partes[1];

                        String tempo = partes[2];

                        String regiao = detectarRegiao(destino);

                        String valorTotal = calcularValorTotal(destino, tempo);

                        tabelaModelo.addRow(new Object[]{cliente, destino, tempo, regiao, valorTotal});

                    }

                }

            } catch (IOException ex) {

                JOptionPane.showMessageDialog(this, "Erro ao ler o arquivo: " + ex.getMessage(), "Erro", JOptionPane.ERROR\_MESSAGE);

            }

        });

        // GERAR FACTURA

        btnGerarFactura.addActionListener(e -> {

            String nomeCliente = campoConsultaCliente.getText().trim();

            if (nomeCliente.isEmpty()) {

                JOptionPane.showMessageDialog(this, "Digite o nome do cliente para gerar a factura.", "Aviso", JOptionPane.WARNING\_MESSAGE);

                return;

            }

            File factura = new File("Factura\_" + nomeCliente + ".txt");

            try (PrintWriter pw = new PrintWriter(new FileWriter(factura))) {

                pw.println("FACTURA");

                pw.println("EMPRESA DE COMINICAÇÃO US\n");

                pw.println("CLIENTE: " + nomeCliente + "\n");

                pw.printf("%-20s %-10s %-15s %-10s%n", "DESTINO", "TEMPO", "REGIÃO", "VALOR");

                double totalGeral = 0;

                for (int i = 0; i < tabelaModelo.getRowCount(); i++) {

                    String destino = tabelaModelo.getValueAt(i, 1).toString();

                    String tempo = tabelaModelo.getValueAt(i, 2).toString();

                    String regiao = tabelaModelo.getValueAt(i, 3).toString();

                    String valorStr = tabelaModelo.getValueAt(i, 4).toString();

                    double valor = Double.parseDouble(valorStr.replace("€", "").replace(",", ".").replace("ECV", "").trim());

                    totalGeral += valor;

                    pw.printf("%-20s %-10s %-15s %-10.2f%n", destino, tempo, regiao, valor);

                }

                pw.printf("%nTOTAL PAGO: %.2f €%n", totalGeral);

                JOptionPane.showMessageDialog(this, "Factura gerada com sucesso: " + factura.getAbsolutePath(), "Sucesso", JOptionPane.INFORMATION\_MESSAGE);

            } catch (IOException ex) {

                JOptionPane.showMessageDialog(this, "Erro ao gerar a factura: " + ex.getMessage(), "Erro", JOptionPane.ERROR\_MESSAGE);

            }

        });

        // ELIMINAR

        btnEliminar.addActionListener(e -> {

            String nomeCliente = campoConsultaCliente.getText().trim();

            if (nomeCliente.isEmpty()) {

                JOptionPane.showMessageDialog(this, "Digite o nome do cliente para eliminar.", "Aviso", JOptionPane.WARNING\_MESSAGE);

                return;

            }

            int[] linhasSelecionadas = tabela.getSelectedRows();

            // Se houver seleção na tabela, remover apenas essas linhas

            if (linhasSelecionadas.length > 0) {

                int confirm = JOptionPane.showConfirmDialog(this,

                    "Tem certeza que deseja eliminar os registos selecionados?",

                    "Confirmar Eliminação", JOptionPane.YES\_NO\_OPTION);

                if (confirm != JOptionPane.YES\_OPTION) return;

                Set<String> linhasParaRemover = new HashSet<>();

                for (int linha : linhasSelecionadas) {

                    String cliente = tabelaModelo.getValueAt(linha, 0).toString();

                    String destino = tabelaModelo.getValueAt(linha, 1).toString();

                    String tempo = tabelaModelo.getValueAt(linha, 2).toString();

                    linhasParaRemover.add(cliente + "," + destino + "," + tempo);

                }

                // Atualiza CSV

                File inputFile = new File("Clientes.csv");

                File tempFile = new File("temp.csv");

                try (BufferedReader reader = new BufferedReader(new FileReader(inputFile));

                    PrintWriter writer = new PrintWriter(new FileWriter(tempFile))) {

                    String linha;

                    while ((linha = reader.readLine()) != null) {

                        if (!linhasParaRemover.contains(linha.trim())) {

                            writer.println(linha);

                        }

                    }

                } catch (IOException ex) {

                    JOptionPane.showMessageDialog(this, "Erro ao eliminar: " + ex.getMessage(), "Erro", JOptionPane.ERROR\_MESSAGE);

                    return;

                }

                if (inputFile.delete() && tempFile.renameTo(inputFile)) {

                    JOptionPane.showMessageDialog(this, "Registos selecionados eliminados com sucesso.", "Sucesso", JOptionPane.INFORMATION\_MESSAGE);

                } else {

                    JOptionPane.showMessageDialog(this, "Erro ao atualizar o ficheiro.", "Erro", JOptionPane.ERROR\_MESSAGE);

                }

                // Remove da tabela

                for (int i = linhasSelecionadas.length - 1; i >= 0; i--) {

                    tabelaModelo.removeRow(linhasSelecionadas[i]);

                }

            } else {

                // Eliminar todos os registos do cliente

                int confirm = JOptionPane.showConfirmDialog(this,

                    "Nenhuma linha foi selecionada.\nDeseja eliminar TODOS os registos do cliente \"" + nomeCliente + "\"?",

                    "Confirmar Eliminação", JOptionPane.YES\_NO\_OPTION);

                if (confirm != JOptionPane.YES\_OPTION) return;

                File inputFile = new File("Clientes.csv");

                File tempFile = new File("temp.csv");

                try (BufferedReader reader = new BufferedReader(new FileReader(inputFile));

                    PrintWriter writer = new PrintWriter(new FileWriter(tempFile))) {

                    String linha;

                    while ((linha = reader.readLine()) != null) {

                        String[] dados = linha.split(",");

                        if (dados.length >= 1 && !dados[0].equalsIgnoreCase(nomeCliente)) {

                            writer.println(linha);

                        }

                    }

                } catch (IOException ex) {

                    JOptionPane.showMessageDialog(this, "Erro ao eliminar: " + ex.getMessage(), "Erro", JOptionPane.ERROR\_MESSAGE);

                    return;

                }

                if (inputFile.delete() && tempFile.renameTo(inputFile)) {

                    JOptionPane.showMessageDialog(this, "Todos os registos do cliente foram eliminados com sucesso.", "Sucesso", JOptionPane.INFORMATION\_MESSAGE);

                } else {

                    JOptionPane.showMessageDialog(this, "Erro ao atualizar o ficheiro.", "Erro", JOptionPane.ERROR\_MESSAGE);

                }

                // Remove da tabela

                for (int i = tabelaModelo.getRowCount() - 1; i >= 0; i--) {

                    if (tabelaModelo.getValueAt(i, 0).toString().equalsIgnoreCase(nomeCliente)) {

                        tabelaModelo.removeRow(i);

                    }

                }

            }

        });

        painelPrincipal.revalidate();

        painelPrincipal.repaint();

    }

    private void mostrarFormularioMarcar() {

        painelPrincipal.removeAll();

        JPanel painelFormulario = new JPanel(new GridBagLayout());

        painelFormulario.setBorder(BorderFactory.createEmptyBorder(30, 0, 0, 0));

        GridBagConstraints fc = new GridBagConstraints();

        fc.insets = new Insets(10, 20, 10, 20);

        campoCliente = new JTextField(10);

        campoDestino = new JTextField(10);

        campoTempo = new JTextField(5);

        fc.gridx = 0;

        painelFormulario.add(criarBloco("Inserir Cliente", campoCliente), fc);

        fc.gridx = 1;

        painelFormulario.add(criarBloco("Inserir Destino", campoDestino), fc);

        fc.gridx = 2;

        painelFormulario.add(criarBloco("Tempo (s)", campoTempo), fc);

        fc.gridx = 3;

        JButton btnInserir = new JButton("Inserir");

        btnInserir.setPreferredSize(new Dimension(100, 30));

        painelFormulario.add(btnInserir, fc);

        // Tabela e modelo

        String[] colunas = {"Cliente", "Destino", "Tempo (s)"};

        tabelaModelo = new DefaultTableModel(colunas, 0);

        tabela = new JTable(tabelaModelo);

        JScrollPane scrollPane = new JScrollPane(tabela);

        scrollPane.setPreferredSize(new Dimension(700, 200));

        // Ação do botão Inserir

        btnInserir.addActionListener(e -> {

            String cliente = campoCliente.getText();

            String destino = campoDestino.getText();

            String tempo = campoTempo.getText();

            if (!cliente.isEmpty() && !destino.isEmpty() && !tempo.isEmpty()) {

                // Adiciona à tabela

                tabelaModelo.addRow(new Object[]{cliente, destino, tempo});

                // Grava no ficheiro CSV

                gravarLinhaCSV("Clientes.csv", new String[]{cliente, destino, tempo});

                // Limpa os campos

                campoCliente.setText("");

                campoDestino.setText("");

                campoTempo.setText("");

            } else {

                JOptionPane.showMessageDialog(this, "Preencha todos os campos!", "Aviso", JOptionPane.WARNING\_MESSAGE);

            }

        });

        // <<< IMPORTAÇÃO AUTOMÁTICA DO CSV AO ABRIR O FORMULÁRIO >>>

        importarCSVDoCaminho("Clientes.csv");

        gbc.gridx = 0;

        gbc.gridy = 0;

        gbc.anchor = GridBagConstraints.NORTH;

        painelPrincipal.add(painelFormulario, gbc);

        gbc.gridy = 1;

        gbc.anchor = GridBagConstraints.CENTER;

        painelPrincipal.add(scrollPane, gbc);

        painelPrincipal.revalidate();

        painelPrincipal.repaint();

    }

    private void gravarLinhaCSV(String nomeArquivo, String[] dados) {

        try (FileWriter fw = new FileWriter(nomeArquivo, true);

            BufferedWriter bw = new BufferedWriter(fw);

            PrintWriter out = new PrintWriter(bw)) {

            out.println(String.join(",", dados));

        } catch (IOException e) {

            JOptionPane.showMessageDialog(this, "Erro ao gravar no arquivo: " + e.getMessage(), "Erro", JOptionPane.ERROR\_MESSAGE);

        }

    }

    private void importarCSVDoCaminho(String nomeArquivo) {

        File arquivo = new File(nomeArquivo);

        if (!arquivo.exists()) return;

        try (BufferedReader br = new BufferedReader(new FileReader(nomeArquivo))) {

            String linha;

            while ((linha = br.readLine()) != null) {

                System.out.println("Linha lida: " + linha);  // <--- Veja no console

                String[] colunas = linha.split(",");

                if (colunas.length >= 3) {

                    tabelaModelo.addRow(new Object[]{colunas[0], colunas[1], colunas[2]});

                }

            }

        } catch (IOException e) {

            JOptionPane.showMessageDialog(this, "Erro ao ler o arquivo: " + e.getMessage(), "Erro", JOptionPane.ERROR\_MESSAGE);

        }

    }

    private String detectarRegiao(String destino) {

        if (destino.startsWith("+") || destino.startsWith("00")) {

            return "Internacional";

        } else if (destino.startsWith("2") || destino.startsWith("3") ||

                destino.startsWith("5") || destino.startsWith("9")) {

            return "Nacional";

        }

        return "Desconhecida"; // ou "" se preferir vazio

    }

    private String calcularValorTotal(String destino, String tempoStr) {

        try {

            double tempo = Double.parseDouble(tempoStr);

            double precoPorSegundo;

            if (destino.startsWith("+") || destino.startsWith("00")) {

                precoPorSegundo = 1.05;

            } else if (destino.startsWith("2") || destino.startsWith("3")) {

                precoPorSegundo = 0.24;

            } else if (destino.startsWith("8") || destino.startsWith("9")) {

                precoPorSegundo = 0.43;

            } else {

                return "Indefinido";

            }

            double total = tempo \* precoPorSegundo;

            return String.format("%.2f", total) + " €";

        } catch (NumberFormatException e) {

            return "Erro";

        }

    }

    private JPanel criarBloco(String titulo, JTextField campoTexto) {

        JPanel painel = new JPanel();

        painel.setLayout(new BoxLayout(painel, BoxLayout.Y\_AXIS));

        JLabel labelTitulo = new JLabel(titulo);

        labelTitulo.setFont(new Font("Arial", Font.BOLD, 12));

        labelTitulo.setAlignmentX(Component.CENTER\_ALIGNMENT);

        JPanel inputPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));

        JLabel labelN = new JLabel("Nº:");

        inputPanel.add(labelN);

        inputPanel.add(campoTexto);

        painel.add(labelTitulo);

        painel.add(Box.createRigidArea(new Dimension(0, 5)));

        painel.add(inputPanel);

        return painel;

    }

    public static void main(String[] args) {

        SwingUtilities.invokeLater(() -> {

            Projecto2 janela = new Projecto2();

            janela.setVisible(true);

        });

    }

}