import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.\*;

import java.util.ArrayList;

public class ToDoListApp extends JFrame implements ActionListener {

private ArrayList<String> tasks;

private DefaultListModel<String> taskModel;

private JList<String> taskList;

private JTextField taskField;

private JButton addButton, updateButton, deleteButton, saveButton, loadButton;

// Constructor

public ToDoListApp() {

tasks = new ArrayList<>();

taskModel = new DefaultListModel<>();

// Set up the frame

setTitle("To-Do List");

setSize(400, 400);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new BorderLayout());

// Task input field

taskField = new JTextField();

add(taskField, BorderLayout.NORTH);

// Task list

taskList = new JList<>(taskModel);

JScrollPane scrollPane = new JScrollPane(taskList);

add(scrollPane, BorderLayout.CENTER);

// Panel for buttons

JPanel panel = new JPanel();

panel.setLayout(new GridLayout(1, 5));

// Buttons

addButton = new JButton("Add");

updateButton = new JButton("Update");

deleteButton = new JButton("Delete");

saveButton = new JButton("Save");

loadButton = new JButton("Load");

panel.add(addButton);

panel.add(updateButton);

panel.add(deleteButton);

panel.add(saveButton);

panel.add(loadButton);

// Add panel to the frame

add(panel, BorderLayout.SOUTH);

// Add action listeners for buttons

addButton.addActionListener(this);

updateButton.addActionListener(this);

deleteButton.addActionListener(this);

saveButton.addActionListener(this);

loadButton.addActionListener(this);

}

// Event handling for buttons

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == addButton) {

String task = taskField.getText();

if (!task.isEmpty()) {

tasks.add(task);

taskModel.addElement(task);

taskField.setText("");

}

} else if (e.getSource() == updateButton) {

int selectedIndex = taskList.getSelectedIndex();

if (selectedIndex != -1) {

String updatedTask = taskField.getText();

if (!updatedTask.isEmpty()) {

tasks.set(selectedIndex, updatedTask);

taskModel.set(selectedIndex, updatedTask);

taskField.setText("");

}

}

} else if (e.getSource() == deleteButton) {

int selectedIndex = taskList.getSelectedIndex();

if (selectedIndex != -1) {

tasks.remove(selectedIndex);

taskModel.remove(selectedIndex);

}

} else if (e.getSource() == saveButton) {

saveTasksToFile();

} else if (e.getSource() == loadButton) {

loadTasksFromFile();

}

}

// Save tasks to a file

private void saveTasksToFile() {

try (FileWriter writer = new FileWriter("tasks.txt")) {

for (String task : tasks) {

writer.write(task + "\n");

}

JOptionPane.showMessageDialog(this, "Tasks saved successfully!");

} catch (IOException ex) {

JOptionPane.showMessageDialog(this, "Error saving tasks!");

}

}

// Load tasks from a file

private void loadTasksFromFile() {

try (BufferedReader reader = new BufferedReader(new FileReader("tasks.txt"))) {

tasks.clear();

taskModel.clear();

String line;

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

tasks.add(line);

taskModel.addElement(line);

}

JOptionPane.showMessageDialog(this, "Tasks loaded successfully!");

} catch (IOException ex) {

JOptionPane.showMessageDialog(this, "Error loading tasks!");

}

}

// Main method

public static void main(String[] args) {

ToDoListApp app = new ToDoListApp();

app.setVisible(true);

}

}