**Aim:**

To develop a **Phone Book Application** using Java Swing that allows users to manage their contact information effectively. The application provides functionalities such as adding, editing, deleting, searching, filtering, importing, exporting, and sorting contacts. The aim is to create a user-friendly graphical interface where users can store and organize contact details (name, phone, email, location, and category) and perform various operations for efficient contact management.

### Algorithm for Phone Book Application

1. \*\*Start Application\*\*

- Initialize the Java Swing GUI.

- Set up the main window with table, buttons, and input fields.

2. \*\*Load Contacts from File\*\*

- Read contacts from the file `contacts.txt`.

- Parse contact information and store in an array list.

3. \*\*Display Contacts in Table\*\*

- Populate the JTable with the loaded contacts.

4. \*\*Add Contact\*\*

- On "Add Contact" button click:

- Open a dialog to input new contact details.

- Validate the input (check phone number and email format).

- Add new contact to the list and update the table.

- Save the updated list to the file.

5. \*\*Edit Contact\*\*

- On "Edit Contact" button click:

- Check if a contact is selected.

- Open the dialog pre-filled with selected contact details.

- Update the contact with new values after validation.

- Save changes and refresh the table.

6. \*\*Delete Contact\*\*

- On "Delete Contact" button click:

- Check if a contact is selected.

- Remove the selected contact from the list.

- Save changes and update the table.

7. \*\*Search/Filter Contacts\*\*

- On "Search" or text input in the search field:

- Filter contacts by name and/or category.

- Update the table to display matching results.

8. \*\*Sort Contacts\*\*

- On "Sort by Name" button click:

- Sort contacts alphabetically by name.

- Refresh the table to reflect the sorted list.

9. \*\*Export Contacts to CSV\*\*

- On "Export to CSV" button click:

- Write all contacts to a CSV file.

10. \*\*Import Contacts from CSV\*\*

- On "Import from CSV" button click:

- Open a file dialog and read contacts from a selected CSV file.

- Add imported contacts to the list and update the table.

- Save the new list to the file.

11. \*\*Save Contacts to File\*\*

- After any add, edit, delete, or import operation:

- Save the updated list of contacts to `contacts.txt`.

12. \*\*End Application\*\*

- Exit when the window is closed.

1 Imports  
  
import javax.swing.\*;

import javax.swing.border.EmptyBorder;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.JTableHeader;

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import java.util.ArrayList;

import java.util.Comparator;

2 Load Contacts  
  
 private void loadContactsToTable() {

}

private void showAddContactDialog() {

JDialog dialog = createContactDialog("Add Contact", null);

dialog.setVisible(true);

}

3 Edit contact

private void editContact() {

int selectedRow = contactTable.getSelectedRow();

if (selectedRow == -1) {

showMessage("Select a contact to edit.");

return;

}

Contact contact = contacts.get(selectedRow);

JDialog dialog = createContactDialog("Edit Contact", contact);

dialog.setVisible(true);

}

4 Delete Contact   
  
private void deleteContact() {

int selectedRow = contactTable.getSelectedRow();

if (selectedRow != -1) {

contacts.remove(selectedRow);

saveContacts();

loadContactsToTable();

showMessage("Contact deleted successfully!");

} else {

showMessage("Select a contact to delete.");

}

}

5 Search Contact   
  
private void searchContacts() {

String searchTerm = searchField.getText().toLowerCase();

String selectedCategory = (String) categoryFilter.getSelectedItem();

model.setRowCount(0);

for (Contact contact : contacts) {

boolean matches = contact.getName().toLowerCase().contains(searchTerm);

boolean categoryMatches = selectedCategory.equals("All") || contact.getCategory().equals(selectedCategory);

if (matches && categoryMatches) {

model.addRow(new Object[]{contact.getName(), contact.getPhone(), contact.getEmail(), contact.getLocation(), contact.getCategory()});

}

}

}

6 Reset search & filter contact  
private void resetSearch() {

searchField.setText("");

categoryFilter.setSelectedIndex(0);

loadContactsToTable();

}

private void filterContacts(String searchTerm) {

String selectedCategory = (String) categoryFilter.getSelectedItem();

model.setRowCount(0);

for (Contact contact : contacts) {

boolean matches = contact.getName().toLowerCase().contains(searchTerm.toLowerCase());

boolean categoryMatches = selectedCategory.equals("All") || contact.getCategory().equals(selectedCategory);

if (matches && categoryMatches) {

model.addRow(new Object[]{contact.getName(), contact.getPhone(), contact.getEmail(), contact.getLocation(), contact.getCategory()});

}

}

}

7 Sort & Save Contact   
  
 private void sortContacts() {

contacts.sort(Comparator.comparing(Contact::getName));

loadContactsToTable();

showMessage("Contacts sorted by name.");

}

private void saveContacts() {

try (BufferedWriter writer = new BufferedWriter(new FileWriter("contacts.txt"))) {

for (Contact contact : contacts) {

writer.write(contact.getName() + "|" + contact.getPhone() + "|" + contact.getEmail() + "|" + contact.getLocation() + "|" + contact.getCategory());

writer.newLine();

}

} catch (IOException e) {

showMessage("Failed to save contacts.");

}

}

Main Class  
  
class Contact {

private String name;

private String phone;

private String email;

private String location;

private String category;

public Contact(String name, String phone, String email, String location, String category) {

this.name = name;

this.phone = phone;

this.email = email;

this.location = location;

this.category = category;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getLocation() {

return location;

}

public void setLocation(String location) {

this.location = location;

}

public String getCategory() {

return category;

}

public void setCategory(String category) {

this.category = category;

}

}

Outputs  


  
  
