**Implement the following projects which focus on different aspects of Java programming, including:**

Object-Oriented Programming principles

Exception handling

File operations

Data structures

Input validation

User interface design

10 Address Book

Implement an address book that stores and manages contact information using object oriented programming in Java. Key points: - Use HashMap for storage - Implement search functionality - Allow contact editing/deletion

Edit

I'll help you create an Address Book application that implements contact management using OOP principles and HashMap for storage.

Address Book Application

Click to open code

This Address Book implementation includes several key features and OOP concepts:

1. Class Structure:
   * Contact: Stores contact information
   * AddressBook: Manages contact operations
   * ContactNotFoundException: Custom exception
   * InvalidInputException: Custom exception
   * AddressBookApp: Main class with UI
2. Features:
   * Add, update, delete contacts
   * Search functionality (searches all fields)
   * Data persistence using file storage
   * Input validation
   * User-friendly interface
3. OOP Principles Used:
   * Encapsulation: Private fields with getters/setters
   * Exception handling
   * Serialization for data persistence
   * Clean separation of concerns
4. Data Storage:
   * HashMap for efficient contact lookup
   * File-based persistence
   * Serialization for object storage

To run the program:

1. Save the code in AddressBookApp.java
2. Compile: javac AddressBookApp.java
3. Run: java AddressBookApp

import java.io.\*;

import java.util.\*;

// Contact class to store individual contact information

class Contact implements Serializable {

private String name;

private String phone;

private String email;

private String address;

public Contact(String name, String phone, String email, String address) {

this.name = name;

this.phone = phone;

this.email = email;

this.address = address;

}

// Getters and setters

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 getAddress() { return address; }

public void setAddress(String address) { this.address = address; }

@Override

public String toString() {

return String.format("""

Name: %s

Phone: %s

Email: %s

Address: %s

""", name, phone, email, address);

}

}

// Custom exceptions

class ContactNotFoundException extends Exception {

public ContactNotFoundException(String message) {

super(message);

}

}

class InvalidInputException extends Exception {

public InvalidInputException(String message) {

super(message);

}

}

// AddressBook class to manage contacts

class AddressBook {

private final Map<String, Contact> contacts;

private static final String DATA\_FILE = "contacts.dat";

public AddressBook() {

contacts = new HashMap<>();

loadContacts();

}

// Add new contact

public void addContact(Contact contact) throws InvalidInputException {

validateContact(contact);

contacts.put(contact.getName().toLowerCase(), contact);

saveContacts();

}

// Update existing contact

public void updateContact(String name, Contact updatedContact)

throws ContactNotFoundException, InvalidInputException {

if (!contacts.containsKey(name.toLowerCase())) {

throw new ContactNotFoundException("Contact not found: " + name);

}

validateContact(updatedContact);

contacts.put(name.toLowerCase(), updatedContact);

saveContacts();

}

// Delete contact

public void deleteContact(String name) throws ContactNotFoundException {

if (contacts.remove(name.toLowerCase()) == null) {

throw new ContactNotFoundException("Contact not found: " + name);

}

saveContacts();

}

// Search contacts

public List<Contact> searchContacts(String searchTerm) {

searchTerm = searchTerm.toLowerCase();

List<Contact> results = new ArrayList<>();

for (Contact contact : contacts.values()) {

if (contact.getName().toLowerCase().contains(searchTerm) ||

contact.getPhone().contains(searchTerm) ||

contact.getEmail().toLowerCase().contains(searchTerm) ||

contact.getAddress().toLowerCase().contains(searchTerm)) {

results.add(contact);

}

}

return results;

}

// Get all contacts

public List<Contact> getAllContacts() {

return new ArrayList<>(contacts.values());

}

// Validate contact information

private void validateContact(Contact contact) throws InvalidInputException {

if (contact.getName().trim().isEmpty()) {

throw new InvalidInputException("Name cannot be empty");

}

if (!contact.getPhone().matches("\\d{10}")) {

throw new InvalidInputException("Phone number must be 10 digits");

}

if (!contact.getEmail().matches("^[A-Za-z0-9+\_.-]+@(.+)$")) {

throw new InvalidInputException("Invalid email format");

}

}

// Save contacts to file

private void saveContacts() {

try (ObjectOutputStream oos = new ObjectOutputStream(

new FileOutputStream(DATA\_FILE))) {

oos.writeObject(contacts);

} catch (IOException e) {

System.err.println("Error saving contacts: " + e.getMessage());

}

}

// Load contacts from file

@SuppressWarnings("unchecked")

private void loadContacts() {

try (ObjectInputStream ois = new ObjectInputStream(

new FileInputStream(DATA\_FILE))) {

Map<String, Contact> loadedContacts = (Map<String, Contact>) ois.readObject();

contacts.putAll(loadedContacts);

} catch (FileNotFoundException e) {

System.out.println("No existing contacts file found. Starting fresh.");

} catch (IOException | ClassNotFoundException e) {

System.err.println("Error loading contacts: " + e.getMessage());

}

}

}

// Main class with user interface

public class AddressBookApp {

private static final Scanner scanner = new Scanner(System.in);

private static final AddressBook addressBook = new AddressBook();

public static void main(String[] args) {

while (true) {

try {

displayMenu();

int choice = Integer.parseInt(scanner.nextLine());

switch (choice) {

case 1 -> addContact();

case 2 -> updateContact();

case 3 -> deleteContact();

case 4 -> searchContacts();

case 5 -> viewAllContacts();

case 6 -> {

System.out.println("Goodbye!");

return;

}

default -> System.out.println("Invalid choice. Please try again.");

}

} catch (NumberFormatException e) {

System.out.println("Please enter a valid number.");

}

System.out.println("\nPress Enter to continue...");

scanner.nextLine();

}

}

private static void displayMenu() {

System.out.println("""

Address Book Menu

1. Add Contact

2. Update Contact

3. Delete Contact

4. Search Contacts

5. View All Contacts

6. Exit

Enter your choice: """);

}

private static void addContact() {

try {

System.out.println("\nEnter Contact Details:");

System.out.print("Name: ");

String name = scanner.nextLine();

System.out.print("Phone (10 digits): ");

String phone = scanner.nextLine();

System.out.print("Email: ");

String email = scanner.nextLine();

System.out.print("Address: ");

String address = scanner.nextLine();

Contact contact = new Contact(name, phone, email, address);

addressBook.addContact(contact);

System.out.println("Contact added successfully!");

} catch (InvalidInputException e) {

System.out.println("Error: " + e.getMessage());

}

}

private static void updateContact() {

try {

System.out.print("\nEnter name of contact to update: ");

String name = scanner.nextLine();

System.out.println("\nEnter new details:");

System.out.print("Phone (10 digits): ");

String phone = scanner.nextLine();

System.out.print("Email: ");

String email = scanner.nextLine();

System.out.print("Address: ");

String address = scanner.nextLine();

Contact updatedContact = new Contact(name, phone, email, address);

addressBook.updateContact(name, updatedContact);

System.out.println("Contact updated successfully!");

} catch (ContactNotFoundException | InvalidInputException e) {

System.out.println("Error: " + e.getMessage());

}

}

private static void deleteContact() {

try {

System.out.print("\nEnter name of contact to delete: ");

String name = scanner.nextLine();

addressBook.deleteContact(name);

System.out.println("Contact deleted successfully!");

} catch (ContactNotFoundException e) {

System.out.println("Error: " + e.getMessage());

}

}

private static void searchContacts() {

System.out.print("\nEnter search term: ");

String searchTerm = scanner.nextLine();

List<Contact> results = addressBook.searchContacts(searchTerm);

if (results.isEmpty()) {

System.out.println("No contacts found.");

} else {

System.out.println("\nSearch Results:");

results.forEach(contact -> {

System.out.println("------------------------");

System.out.println(contact);

});

}

}

private static void viewAllContacts() {

List<Contact> allContacts = addressBook.getAllContacts();

if (allContacts.isEmpty()) {

System.out.println("\nNo contacts in address book.");

} else {

System.out.println("\nAll Contacts:");

allContacts.forEach(contact -> {

System.out.println("------------------------");

System.out.println(contact);

});

}

}

}

**o/p**

No existing contacts file found. Starting fresh.

Address Book Menu

1. Add Contact

2. Update Contact

3. Delete Contact

4. Search Contacts

5. View All Contacts

6. Exit

Enter your choice:

1

Enter Contact Details:

Name: Dr.Sm.Badhusha

Phone (10 digits): 8123456634

Email: badhusha.sm@gmail.com

Address: Prof/CSE, AIET

Contact added successfully!

Press Enter to continue...

Address Book Menu

1. Add Contact

2. Update Contact

3. Delete Contact

4. Search Contacts

5. View All Contacts

6. Exit

Enter your choice:

5

All Contacts:

------------------------

Name: Dr.Sm.Badhusha

Phone: 8123456634

Email: badhusha.sm@gmail.com

Address: Prof/CSE, AIET

Press Enter to continue...

Address Book Menu

1. Add Contact

2. Update Contact

3. Delete Contact

4. Search Contacts

5. View All Contacts

6. Exit

Enter your choice:

2

Enter name of contact to update: Dr. S.Mohideen Badhusha

Enter new details:

Phone (10 digits): 8122572250

Email: badhusha.sm@gmail.com

Address: Sr.Prof/CSE, Aiet

Error: Contact not found: Dr. S.Mohideen Badhusha

Press Enter to continue...

Address Book Menu

1. Add Contact

2. Update Contact

3. Delete Contact

4. Search Contacts

5. View All Contacts

6. Exit

Enter your choice:

5

All Contacts:

------------------------

Name: Dr.Sm.Badhusha

Phone: 8123456634

Email: badhusha.sm@gmail.com

Address: Prof/CSE, AIET

Press Enter to continue...

Address Book Menu

1. Add Contact

2. Update Contact

3. Delete Contact

4. Search Contacts

5. View All Contacts

6. Exit

Enter your choice:

1

Enter Contact Details:

Name: Ram

Phone (10 digits): 1234567898

Email: sjdjd.sm@gmail.com

Address: 56, mm street

Contact added successfully!

Press Enter to continue...

Address Book Menu

1. Add Contact

2. Update Contact

3. Delete Contact

4. Search Contacts

5. View All Contacts

6. Exit

Enter your choice:

5

All Contacts:

------------------------

Name: Dr.Sm.Badhusha

Phone: 8123456634

Email: badhusha.sm@gmail.com

Address: Prof/CSE, AIET

------------------------

Name: Ram

Phone: 1234567898

Email: sjdjd.sm@gmail.com

Address: 56, mm street

Press Enter to continue...

Address Book Menu

1. Add Contact

2. Update Contact

3. Delete Contact

4. Search Contacts

5. View All Contacts

6. Exit

Enter your choice:

2

Enter name of contact to update: Ram

Enter new details:

Phone (10 digits): 9999999999

Email: ty.sm@gmial.com

Address: tt street, Erode

Contact updated successfully!

Press Enter to continue...

5

Address Book Menu

1. Add Contact

2. Update Contact

3. Delete Contact

4. Search Contacts

5. View All Contacts

6. Exit

Enter your choice:

5

All Contacts:

------------------------

Name: Dr.Sm.Badhusha

Phone: 8123456634

Email: badhusha.sm@gmail.com

Address: Prof/CSE, AIET

------------------------

Name: Ram

Phone: 9999999999

Email: ty.sm@gmial.com

Address: tt street, Erode