Java Refresher Design Document

Kevin Fang

Overview

The Contact Manager is a class for managing a list of contacts. It has the following features:

- Add new contacts
- View all contacts
- Search for specific contacts
- Edit contact details
- Delete contacts
- Save and load contacts from a file
- Runtime reporting for tracking operation durations

Class and Method Descriptions

Class: ContactMakers

The `ContactMakers` class is the main controller, handling user interactions and running contact-related operations. It uses an `ArrayList` to manage/store the contacts, which can adjust when contacts are being manipulated.

Instance Variables:

- `ArrayList<Contacts> contacts`: Stores all the contact objects. This dynamic list grows or shrinks as needed.
- `Scanner scanner`: Used for taking user input from the console.
- `String runtimeFile`: the default filename for storing runtime performance reports.
- `boolean reportRuntime`: Flag indicating whether to generate runtime reports.
- `boolean verbose_mode`: Extra output for debugging and information purposes.

Main Method (`main`):

- Provides a menu-driven user interface for managing contacts.
- Processes command-line options:
  - `-v` enables verbose mode for debugging.
  - `-a <filename>` specifies the file to load contacts from.
  - `-O <filename>` enables runtime reporting and specifies the file to write the report to.

Features:

1. **Add Contact (`addContact`)**:
    - Prompts the user for the contact's name, phone number, email, and other optional details like alt phone, birthday, and social media.
    - Adds the new contact to the list.
    - Verbose mode displays a success message upon adding a contact.
2. **View Contacts (`viewContact`)**:
    - Displays all the contacts stored in the list.
    - Throws `AddressBookEmptyException` if no contacts exist.
    - Verbose mode indicates when all contacts are listed.
3. **Search Contact (`searchContact`)**:
    - Prompts the user for a contact name to search.
    - If found, the contact details are displayed; if not, `ContactNotFoundException` is thrown.
    - Verbose mode confirms when a contact is successfully found.
4. **Edit Contact (`editContact`)**:
    - Searches for a contact by name and allows the user to update the contact's details such as phone number, email, mailing address, birthday, and social media profile.
    - Throws `ContactNotFoundException` if the contact is not found.
    - Verbose mode confirms when a contact is successfully edited.
5. **Remove Contact (`removeContact`)**:
    - Searches for a contact by name and removes it from the list.
    - Throws `ContactNotFoundException` if the contact is not found.
    - Verbose mode confirms when a contact is successfully removed.
6. **Save Contacts to File (`saveContactsToFile`)**:
    - Saves the current list of contacts to a specified file in CSV format.
    - Verbose mode confirms the file-saving operation.
7. **Load Contacts from File (`loadContactsFromFile`)**:
    - Loads contacts from a specified file and adds them to the list.
    - Throws `InvalidFileFormatException` if the file format is incorrect or the content is invalid.
    - Verbose mode confirms successful loading of contacts.
8. **Runtime Reporting (`reportOperationTime`)**:
    - Tracks the time taken for key operations (adding, viewing, searching, editing, removing contacts).
    - If `reportRuntime` is enabled, the execution time is written to a file.

Custom Exceptions

1. **AddressBookEmptyException**:
    - Thrown when trying to view or manipulate contacts, but the address book is empty.
2. **ContactNotFoundException**:
    - Thrown when a contact cannot be found in the list during search, edit, or delete operations.
3. **InvalidFileFormatException**:
    - Thrown when the file format does not match the expected structure during file loading.

Class: Contacts

The `Contacts` class represents individual contacts with their personal details.

Attributes:

- `String name`: The contact's name.
- `String phoneNumber`: The contact's phone number.
- `String altPhoneNumber`: Alternative phone number (optional).
- `String email`: Main email address.
- `String altEmail`: Alternative email address (optional).
- `String mailingAddress`: Physical mailing address (optional).
- `String birthday`: The contact’s birth date (optional).
- `Strong socialMedia`: Social media profile (optional).

Methods:

- **Constructor:** Initializes a contact with all the provided attributes.
- **Getter Methods:** Retrieve details like `getName()`, `getPhoneNumber()`, `getEmail()`, etc.
- **Setter Methods:** Allow modification of contact details such as `setName()`, `setPhoneNumber()`, `setEmail()`, `setAddress()`, `setBirthday()`, and `setMedia()`.
- **toString()**: Provides a formatted string representation of the contact’s details.
- **toCSV()**: Returns a CSV-formatted string for saving the contact to a file.