Introduction

Hello, I am Idorenyin Ekotebe, and I am going to take you through the Person Management System. This system will facilitate collection, storing, updating, and deletion of a Person's records; Just to name but a few. You will not have to search through manual pages to look for a person's records, through use of a person's id, records can be retrieved. This system shall also 100% enable you to store records to a maximum number of a randomly generated value, but there is an option for you to customize the size of your database. So let's dig right into it, and discover the potentials, and capabilities of this system.

Define All Classes and Packages

The System has four packages:

1. **module04.oop.main** - This package contains classes responsible for managing the main functionality of the program. Here are the classes:
    A. Main.java - This class serves as the entry point of the program and orchestrates user interaction.
    B. PersonManager.java - handles user input and manages the operations on the collection of persons.
    C. StackOfPersons.java - This class implements a stack data structure to store and manage Person objects.
    D. TestStackOfPersons.java - A testing class that demonstrates the functionality of StackOfPersons.java
2. **module04.oop.models** - The models package encompasses classes that define the data structure of a Person and related attributes e.g. Admin.java;
    A. Admin.java - This class models an administrator entity responsible for managing the application.
    B. Person.java - This class is the core entity representing an individual in the system.
    C. Funds.java - represents financial information associated with a person.
    D. Location.java - Location class manages or contains a person's address.
    E. Name.java - This class encapsulates a person's name.
    F. Contact.java – This class contains contact information for a person.
3. **module04.oop.randompms** - This package provides utility classes for generating random data.
    A. MyRandomAdminData.java - Generates random data for creating Admin objects.
    B. MyRandomPersonData.java - Provides methods to generate random person data.

Program Functionality

1. Add a Person

The program allows users to add a new person to the system. A unique ID is automatically generated for each person.

2. **Display All Persons**

   Users can view a list of all persons currently in the system, including their details.

3. **Update a Person**
   Individual attributes of a person, such as name, location, contact, and funds, can be updated through this functionality.
4. **Delete a Person**
   Users have the ability to remove a person from the system by providing their unique ID.
5. **Search for a Person by ID**
   This feature enables users to search for a person using their unique ID, providing quick access to specific records.
6. **Bulk**
   This feature allows users to choose to add a bulk number of persons to the system for testing or demonstration purposes.
7. **Count**
   The system provides a count of the total number of persons currently in the system.
8. **Exit**
   Users will have the option to exit the program at any time.

Chapter 10 Subtopics Involved

1. **Class Abstraction and Encapsulation**
   The project extensively employs class abstraction and encapsulation to ensure that data and methods are logically grouped within each class, providing a clear and structured design.
2. **Thinking in Objects**
   The project emphasizes thinking in terms of objects, with each class representing a real-world entity, such as a person or administrator.
3. **Class Relationships**
   Classes within the project have well-defined relationships, with StackOfPersons managing a collection of Person objects, and Person containing attributes such as Name, Location, Contact, and Funds. Aggregation, Composition, and Association are the relationships factored in the design. No inheritance relationship has been utilized.
4. **Case Study: Designing a Class for Stacks**
   The StackOfPersons class serves as an illustrative case study, demonstrating the effective design and implementation of a stack data structure.

Conclusion

By utilizing class abstraction, encapsulation, and thoughtful design, the program offers a scalable and efficient solution for handling collections of persons.