Project 0: ASEAN Phonebook (Java Version)

Objectives

This project aims to:

- Provide a basic understanding of how object-oriented programming works in Java.
- Practice the use of documentation strings and comments to easily trace what a program is doing in a specific operation.
- Understand the contents of a UML Class diagram and implement it based on what was written.
- Create custom test methods and judge the validity of one’s program.

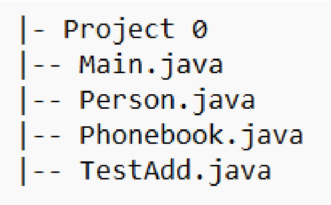
Overview

You are given a half-finished Philippine Phonebook program where it has several classes and methods with missing implementation. Your job in this project is to complete the implementation of the Philippine Phonebook program whose behavior and functionalities similarly follow everything illustrated in the document attached with this one.

The purpose of this document is to guide you on which functions to finish first and how each of these functions should behave.

Project File Structure

Firstly, download the `Project0.zip` file and extract the contents. Once this is done, it should have the following file structure:



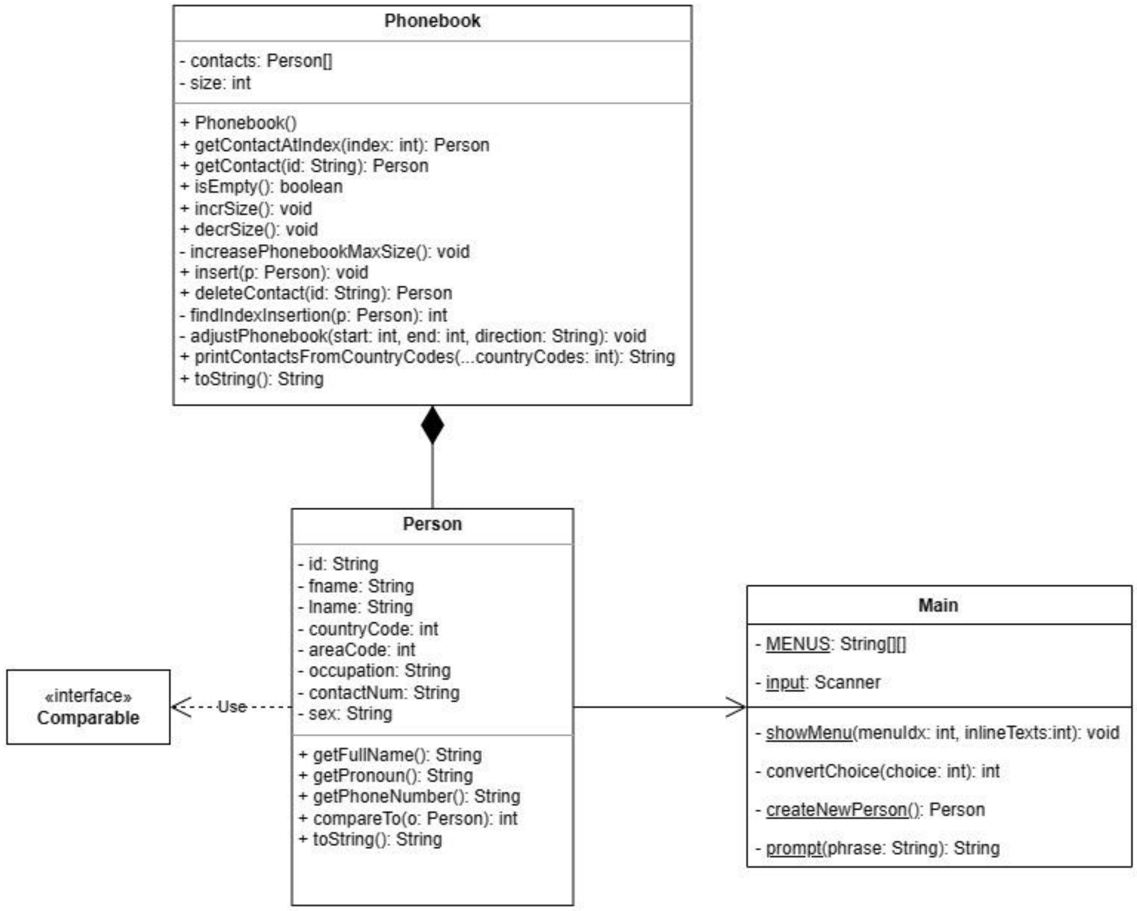
```
|- Project 0
|-- Main.java
|-- Person.java
|-- Phonebook.java
|-- TestAdd.java
```

Figure 1. Project File Structure for Project 0 – Phonebook.

You can use the test files to check the validity of your ASEAN Phonebook program. You can also expand the number of tests and see if it works.

UML Class Diagram

Refer to the class diagram below about the contents of each class. Note that you should read the `Phonebook Functionality.pdf` and ascertain which methods belong to which functionality.



Phonebook
- contacts: Person[]
- size: int
+ Phonebook()
+ getContactAtIndex(index: int): Person
+ getContact(id: String): Person
+ isEmpty(): boolean
+ incrSize(): void
+ decrSize(): void
- increasePhonebookMaxSize(): void
+ insert(p: Person): void
+ deleteContact(id: String): Person
- findIndexInsertion(p: Person): int
- adjustPhonebook(start: int, end: int, direction: String): void
+ printContactsFromCountryCodes(...countryCodes: int): String
+ toString(): String
Person
- id: String
- fname: String
- lname: String
- countryCode: int
- areaCode: int
- occupation: String
- contactNum: String
- sex: String
+ getFullName(): String
+ getPronoun(): String
+ getPhoneNumber(): String
+ compareTo(o: Person): int
+ toString(): String
«interface»
Comparable
Use
Main
- MENUS: String[][]
- input: Scanner
- showMenu(menuIdx: int, inlineTexts:int): void
- convertChoice(choice: int): int
- createNewPerson(): Person
- prompt(phrase: String): String

Figure 2. UML Class Diagram of the Phonebook Program.

Sample Output

Refer to the attached file, `Phonebook Functionality.pdf`, for the specific outputs of every functionality.

Note: The “Search by Surname” functionality is now replaced by “Search by ID”.

Limitations and Warnings

You are forbidden in using any function, classes, and interfaces that are not yet discussed in your lecture session.

You are also forbidden in creating additional functions/methods inside the `Main.java, Person.java, and Phonebook.java` files. The signature or definition of the function(s) and/or methods should also not be modified. This includes but not limited to:

- Class attributes and names.
- Function name
- Function parameters and data types
- Function return type
- Class/Function documentation string

Violating this note will result in significant sanctions imposed by your instructor.

Any changes to the original Test methods, whether accidental or intentional, will result to a score of 0 on this project without additional considerations. ***However, you may add your own set of "Tests" without editing the contents of the existing Test methods. This is to foster your necessity in creating test methods for the programs that you will develop in the future.***

For any student with similar code submissions, ignoring variable/function/class name changes, white spacing, etc., your score will be divided upon how many other students is found to share such code similarity. Since you decide to submit the same output, then it is reasonable that the score will be divided upon each, and everyone involved in such practice.

Submission

You must submit a compressed file containing all required Java files (no ".javac" files) for this project. It should also include a text file named **"GROUPS.txt"** listing the group members. Additionally, invite me to your GitHub repository, where I will review the final commit made **before February 17, 2025.** This means that your work must be submitted **no later than February 16, 2025**. Absolutely no deadline extensions will be granted, and failed last-second submissions will not be accepted.