CSE 271 Lab 13 – Object Streams and Binary Files
Spring 2022
Assigned: 4/28/2022
Due: 5/1/2022

Introduction:

In this lab, we will utilize Object Streams, both input and output, to read and write information from and to a binary file, respectively. Create a project in your Eclipse IDE called **Lab13**. You are going to create a **Contact.java** and **AddressBook.java** class which follow the specifications below.

Contact Class:

Create a class named **Contact.java** that can store a single contact's information. The **Contact** class should store the contact's firstName, lastName, phoneNumber, email, and address as instance properties. All instance properties are of type String. Define an empty (default) constructor that initializes all instance properties with empty Strings. Write toString(), equals(Object o), and getter and setter methods for all instance properties. The toString() method forms a String in the following format: "Garrett Goodman, 100 Main St, Miami, FL 30309, testEmail@miamioh.edu, 123-456-7890". Two contacts are considered equal when they have the same firstName, lastName, and phoneNumber. Hint, you need to make sure **Contact** objects are serializable.

AddressBook Class:

Create a class named **AddressBook.java** with the main() method. It has an instance property of **ArrayList<Contact>** to store a list of **Contact** objects. It offers the following options, in an infinite loop, to the user.

Address Book Operations:
1) Add
2) Remove
3) Save
4) Load
5) Display All
6) Search
7) Exit

Select an option (number): **2**

The options are: (the bolded blue color 2 in the above figure is the user input)

1. Add a contact to the ArrayList.
    a. Ask (prompt the user) for all the contact information and take input from the keyboard. Then, create a **Contact** object and add it to the ArrayList.
2. Remove a contact from the ArrayList.
    a. Ask (prompt the user) for the phoneNumber and remove the corresponding contact from the ArrayList. The phoneNumber is unique. Therefore, when adding a new contact, you must ensure that there are no duplicate phone numbers.
3. Save all contacts to the file (**AddressBook.bin**).
4. Load all contacts from the file (**AddressBook.bin**).
5. Display all contacts using the toString() method in the **Contact** class.

6. Search for a specific contact and display it.
    a. Take a String input from the keyboard to search by.
    b. The search should find all contacts where any of the instance properties that contains the search String should be displayed. For example, if "**abc**" is the search String, then any contact where either the firstName, lastName, phoneNumber, email, or address contains the search String "**abc**" should be displayed. You can use the toString() method in the **Contact** class to display contacts. Note that the search String is case sensitive.
7. Exit from the program.
    a. Save all the contacts from the ArrayList to the file and terminate the program.

Your program should keep running while performing these operations and only terminate if the user selects the Exit (7) option from the menu. In that case, you should save all the contacts from the ArrayList to the file (before you terminate the program). When you restart your program, you should read in all the contacts from the file. The first time you run the program (the file does not exist), you should create an empty file called **AddressBook.bin** using the Paths and Files class as discussed in lecture.

Additional Notes:

- There is no formatting guideline for this lab. Please use appropriate prompts (printed text) when you ask for input and display the correct corresponding information.
- You need to offer the options in an infinite loop until the user selects the Exit (7) option.
- This is a console-based application, do not create a GUI for this lab.
- You should use an ObjectInputStream to read the Contact ArrayList from the binary file and an ObjectOutputStream to write the Contact ArrayList to the file. Similarly, you may use these objects to read and write each object individually to the file instead. The choice is yours. We are **NOT** using RandomAccessFile objects for this lab.
- Add appropriate try-catch exception handling when performing file operations.
- Make sure to include JavaDoc for all classes and methods including parameter and return descriptions.

Submission Instructions:

After you have completed the lab assignment, locate your source code (**Contact.java and AddressBook.java**) in your workspace and submit it to the corresponding Lab 13 Canvas folder.

Rubric:

Task	Grade
Contact.java	
Contact class correctly implemented	10
AddressBook.java	
Add operation implemented	10
Remove operation implemented	10
Save operation implemented using ObjectOutputStream	10
Load operation implemented using ObjectInputStream	10
Display All operation implemented	10
Search operation implemented (finds all contacts containing the search String)	10
Exit operation implemented	10
Reads from the file when the program starts	5
Creates an empty file if the file does not exist	5
Adequate JavaDoc for all methods and classes and followed the Miami University coding guidelines	10
Total	**100**