Concordia University
COMP 249 – Fall 2013
Object-Oriented Programming II
Assignment #1

Goal: The goal of the first assignment is to help you review some of the main topics covered in previous courses, including classes, loops, arrays, arrays of objects, as well as to test your algorithmic thinking with simple problems.

Context: You are the lead designer in a software company and you have been assigned with the task of designing and implementing a Contact Manager. A Contact Manager is an application that stores business and personal contacts and offers a set of functionalities, such as the addition of a new contact, the update of an existing one, search for contacts with custom search criteria and many more cool features.
The company aims to conquer this market by producing the application for multiple platforms and environments, including a desktop application, a mobile application and a web application component. Your design solution will be common for all products and as a result your role is very important for the success of this project. You should take under serious consideration the following facts:

- Your design solution will be used by other development teams in the company that will be responsible for the implementation of the desktop application, mobile application and web application component, respectively. As a result, your code should be very well documented (with javadoc and inline comments) and easy to understand.
- Your design solution should be easily extensible to future requirements (without having to change existing code). Every now and then, a new social network comes into play and the users have to update their contacts with new information. Your design should be flexible and easily extensible in order to ensure the diachronic success of the product.

Requirements:
The requirements team has gathered the following requirements for the Contact Manager application:

Each **Contact** has the following attributes:
- Last name (String)
- First name (String)
- Middle name (String)
- Address list (an array of Address objects)
- Phone number list (an array of PhoneNumber objects)
- Social network account list (an array of SocialNetworkAccount objects)

Note: Two contacts are considered equal if they have the same first, last and middle name.

Each **Address** has the following attributes:
- type [Home, Work, Other] (Hint: use Enum types)
- Street name
- Street number
- Apartment number (could be null)
- City
- Postal Code
- Country

Note: Two addresses are considered equal if all of the aforementioned attributes are equal.

Each **PhoneNumber** has the following attributes:
- type [Home, Work, Mobile, Fax] (Hint: use Enum types)
- country code
- area code
- phone number
- extension (usually applicable for work phone numbers, could be null)

Note: Two phone numbers are considered equal if all of the aforementioned attributes are equal.

Each **SocialNetworkAccount** has the following attributes:
- type [Skype, Facebook, Google+, Twitter, LinkedIn] (Hint: use Enum types)
- account id/name

Note: Two social network accounts are considered equal if all of the aforementioned attributes are equal.

The class **ContactManager** will be used to store the contacts in an array of **Contact** objects.

Task #1

Design and implement the required classes.
- For each class, create an appropriate constructor that initializes the values of the attributes. (Hint: attributes of array type should be initialized as empty arrays allowing to store 10 objects at maximum)
- Create appropriate methods in class Contact for the addition of
    a) An address **addAddress()**
    b) A phone number **addPhoneNumber()**
    c) A social network account **addSocialNetworkAccount()**

to a specific contact. The methods should examine if the element passed as argument already exists in the corresponding array (Hint: create a separate private method that performs this check) and add it only if it is not already present. (Hint: the check should be performed using the corresponding **equals()** method)
- Create method **addContact()** for the addition of a new contact in class ContactManager. Again, the method should examine if the contact passed as argument already exists in the array and add it only if it is not already present. Create also method **numberOfContacts()** that returns the current number of stored contacts. (Hint: the check should be performed using the corresponding **equals()** method)
- Create a **main()** method in **Driver** class that creates and populates the attributes of five contacts and adds them in the ContactManager.

Task #2

For the first prototype, you will have to implement and test the correctness of the following features:

1. **Contact search**: Create a method **searchContacts()** in ContactManager that takes a String as argument and prints the contacts whose first or last name **contains** that string (i.e., the string given as argument should be part of the first or last name of a contact). The string comparison should be case insensitive. (Hint: use method **contains()** or **matches()** from class String)
2. **Find my contacts in Social Network X**: Create a method **findContactsInSocialNetwork()** in ContactManager that takes as argument the type of a Social Network and prints the contacts having an account for this specific Social Network. (Hint: make an additional method in class Contact that checks whether the array of social network accounts contains an account corresponding to the type passed as argument)
3. **Contact sorting**: Create a method **sortContacts()** that returns all contacts stored in the ContactManager sorted lexicographically/alphabetically in ascending order based on their last name. In the case two contacts have the same last name they will be sorted based on their first name in ascending order. (Hint: use method **compareTo()** of class String to compare the contact names. You can use the insertion sort algorithm)

Submitting Assignment #1

- Zip the Eclipse project directory. External libraries (jar files) should be included in the project.
- Make sure that your source code is compliant with Java 1.6 (Right-click on your project and select **Properties**, Click on **Java Compiler** in the menu and make sure that the **Compiler compliance level** is equal to 1.6)
- Naming convention for zip file: The zip file should be named a#_studentID, where # is the number of the assignment studentID is your student ID number. For example, for the first assignment, student 123456 would submit a zip file named a1_123456.zip
- Submit your zip file at: https://fis.encs.concordia.ca/eas/ as **Programming Assignment** and submission **#1**. Assignments submitted to the wrong directory would be discarded and no replacement submission will be allowed.
- If you submit multiple times an assignment, only the **LAST** version will be graded and all others will be disregarded.

Evaluation Criteria for Assignment #1 (10 points)

Task #1	5 points
Class, constructor, accessor method creation	1 point
add***() methods in class Contact & ContactManager	1.5 points
equals(), toString() implementation	1.5 points
Proper creation and use of enumerated types	1 point
Task #2	5 points
Contact search	1 point
Find contacts in Social Network X	1 point
Contact sorting	3 points