CSc 110 Lab 9: Classes and Objects

Objectives:

At the conclusion of Lab 9 you should be able to:

- Create a class,
- Declare an new instance of a class,
- Use an instance of a class (object).

Submission: At the conclusion of this lab submit your updated Contact.java file (including updated comments that adhere to the coding convention), to the Lab09 submission site on Connex.

Exercise 1:

- Download the file: Contact.java
- Compile the file.
- Then, open it for examination. Identify the following characteristics of a class:
  - Attributes and methods.
  - The constructor(s). Observer that they have: No return type; the same name as the class; are invoked when the 'new' keyword is used.
  - Identify the instance methods (that don't us the static keyword.)
- Review the Coding conventions document for our course and add suitable *class-description* comments and *method-header* comments throughout the Contact class.

Exercise 2:

- Download the file: TestContact.java into the same folder as Contact.java.
- Compile the file.
- Run the program.
- Add the following code to the main method:
  - `acquaint.setName("/*Type", "your name", "here*/");`
  - `friend2.changeEmail("/*Type your friend's email here*/");`
  - `System.out.println(friend1.getName());`

Exercise 3:

Enhance the Contact class, as follows:

- Create a **`toString()`** method: Normally, every class has a **`toString()`** method that allows printing an object's data attributes. It is invoked automatically anytime the + operator (for concatenation) is used in a String context or a print function is called. Add a **`toString()`** method into the Contact class. It can include the statement:
  `return getName() + "; " + kind + ": " + email;`
- Test the **`toString()`** method: Go back into main in TestContact and print each contact created so far. For example it is possible to print all the data for **`friend1`** by issuing the command
  `System.out.println(friend1);`

means the same as issuing the command:

```
System.out.println(friend1.toString());
```

Exercise 4:

- Add a new attribute, a private String variable called `alias`. It's default value should be an empty String.
- Make a new constructor that takes two string parameters, the first representing an email and the second representing an alias. Thus, a contact will now also have an alias.
- Add alias to the output of the **`toString()`** method.
- Assure that the Contact class compiles.
- Declare a new contact in your TestContact program and initialize it with an email and an alias to test the new method and attributes.
- Also, print the new contact.
- Update the documentation in your Contact.java class such that it follows the requirements of the CSC 110 Coding Conventions.

Discussion: What other methods that could be added to Contact?

`Exercise 5`

- Download ContactList.java into the same folder as Contact.java. (Observe that A ContactList is an array of Contacts.)
- Download and use TestContactList.java (rather than TestContact.java.)
- Examine and discuss with colleagues and your Teaching Assistant the line:
  `this.list[i].getEmail().equals(email)`
  that is a conditional in an if statement. What exactly is happening?

Take the time to understand the difference between instance methods and variables as opposed to static methods and variables.