# Unit 4 Assignment

## GENERAL

1. Save the work of each exercise in a separate Java source code file and name it *StuPrefixClassName.java* and follow the coding style as explained in Unit 1 Assignment.

2. Overall comment your program appropriately (file prolog comment, method prolog comment, and comments on the code). Pay attention to the standard stuff like coding style, meaningful identifier names, indention, and locations of braces.

3. When you’re done with an exercise, include a screenshot of the execution of the program in your assignment report document. This is to remind you to verify the execution result.

## EXERCISES

We will build a multi-class program, each class in its own source code file. All classes from this assignment should be created under the same Project in your IDE.

1. (Name this class StuPreContact) Find a Contact class in Part 1 of an Assignment4Code.txt for this assignment. Draw a UML class diagram for the given Contact class. Follow the class diagram notation introduced in the lecture.

Next put this code in a class called StuPreContact. You may comment in the main() (i.e. put it back in the source code) to test your class before moving to the next exercise. You need to update each occurrence of the class name.

After you’re done, comment out the main() before starting the next exercise.

1. (Name this class StuPreEmail) Add an Email class to your project based on the following specification:

|  |
| --- |
| **StuPreEmail** |
| - idGen : int = 1000  - id : int  - from : StuPreContact  - to : StuPreContact  - subject : String  - timeStamp : LocalDateTime |
| + StuPreEmail()  + StuPreEmail(from : StuPreContact, to : StuPreContact, subject : String)  + StuPreEmail(from : StuPreContact, to : StuPreContact, subject : String, time : LocalDateTime)  + getFrom() : StuPreContact  + getTo() : StuPreContact  + getSubject() : String  + getTimeStamp() : LocalDateTime  + setFrom(from : StuPreContact) : void  + setTo(to : StuPreContact) : void  + setSubject(subject : String) : void  + toString() : String |

* The static member idGen is used to generate a unique id for each new email object. Its current value should be used when creating a new email object and then incremented (to prepare for the next object). Hint: it needs to be used and updated in each constructor.
* Default constructor: constructs an email object with a unique id based on idGen. Use null value for the remaining data members.
* The three-parameter constructor: constructs an email object with a unique id based on idGen, and the specified from, to, and subject fields. The timestamp field will be set to the current local time: timeStamp = LocalDateTime.now();
* The four-parameter constructor: constructs an email object with a unique id based on idGen and the specified from, to, subject, and time fields
* getXXX() and setXXX() methods follow the general convention of such methods.
* toString(): returns a string representation of the calling Email object in this format:

Subject: *subject-str*

From: *from-contact-in-a-string*

To: *to-contact-in-a-string*

Time: *timestamp-in-a-string*

Be sure to use the toString() method of the StuPreContact class.

Use this segment of code to convert the timeStamp data member (a LocalDateTime object) to a string:

// format: Three-letter-month-name, dd, yyyy, hh:mm AM-or-PM

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("LLL dd, yyyy, hh:mm a");

String timeStr = timeStamp.format(formatter);

You will use two new classes in this exercise: LocalDateTime and DateTimeFormatter.

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

The instructions given above should be enough, but feel free to research those two classes.

<https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/time/LocalDateTime.html>

<https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/time/format/DateTimeFormatter.html>

You should have commented out the main() in the Contact class.

Set up a simple main() within this Email class like in the given Contact class. Test your class incrementally while you build it. For example, it’s possible to just create an empty class and then test it by creating a new object of it in the main(). Your main() should at least include the following:

1). Create a Contact object with your name and email address (email address may be a fake one, but the name should be yours).

2). Create a 2nd Contact object with the information of a friend of yours.

3). Create an Email object using the three-parameter constructor: from you to your friend with some subject string.

4). Create a 2nd Email object using the four-parameter constructor: from your friend to you with a different subject. The time argument should be *yourFirstEmailVar*.getTimeStamp().plusHours(1) which means 1 hour after the first email message. plusHours() is a method of LocalDateTime.

5). Print out the 1st email object.

6). Print out the 2nd email object.

Your IDE may issue a warning on the id field as it’s never used (assigned but never retrieved). Don’t worry about it.

Take a screenshot of the execution of your program and include it in the assignment report document.

Comment out this main() after you’re done.

1. (Name this class StuPreTestEmail) Finally add a TestEmail class with an empty main(). It will serve as the client program.

Find a static generateEmailList() method in Part 2 of the Assignment4Code.txt for this assignment. Add the generateEmailList() method to your TestEmail class. Be sure to modify the method following the notes in the text file.

Add the following to the main() of TestEmail:

1). Call generateEmailList() with argument 20. Save the returned result in a proper local variable. Hint: it should be an ArrayList of some type.

2). Print out the returned list.

3). Retrieve the email address associated with the from field of the 1st object (at index 0) stored in the returned list. Hint: first retrieve the object at index 0 of the result list, which is an Email object. Next call getXX() on this Email object to get a Contact object. Finally call getXXX() on this Contact object. Those may be done in one statement with chained method calls or in multiple statements.

4). Print out that email address in a message like “Search for email messages from/to *that\_email\_address*:”.

5). Search the returned list for that email address. Find and print any email from or to that address. You need to loop through the returned list, retrieve the email address from the from field and to field of each object, and compare them with the email address from step 3. This step should at least print out the very first email object in the list, since the email address is taken out of the first email object’s from field.

The print from step 2 may look like this if the toString() of your Email class returns a string without an ending newline character:

[Subject: Msg 0

From: *Name1* <*email1*>

To: *Name2* <*email2*>

Time: *time0*, Subject: Msg 1

From: *Name3* <*email3*>

To: *Name6* <*email6*>

Time: *time1*, Subject: Msg 2

…

Time: *time18*, Subject: Msg 19

From: *Name7* <*email7*>

To: *Name2* <*email2*>

Time: *time19*]

If there is an ending newline character, your print from step 2 may look like this instead. Either way is okay.

[Subject: Msg 0

From: *Name1* <*email1*>

To: *Name2* <*email2*>

Time: *time0*

, Subject: Msg 1

From: *Name3* <*email3*>

To: *Name6* <*email6*>

Time: *time1*

, Subject: Msg 2

…

Time: *time18*

, Subject: Msg 19

From: *Name7* <*email7*>

To: *Name2* <*email2*>

Time: *time19*]

Take a screenshot of the execution of your program (at least showing the email address used for searching and the search result from step 5) and include it in the assignment report document.

1. Reflection: answer those questions AFTER you’ve completed this assignment:
2. What’s the hardest part of this assignment for you? Please explain.
3. How may you change the Email class if you are to design it? Identify at least one change and briefly explain why you want to make such a change.

## SUBMISSION

Submit three .java files + one word/PDF document. Please put UML class diagram, screenshots and answers to the reflection question into your word/PDF document.

* Exercise 1: UML class diagram and a java file (main() commented out, if any).
* Exercise 2: a java file (main() commented out) and a screenshot of execution.
* Exercise 3: a java file and a required screenshot of execution.
* Exercise 4: Assignment reflection
* Check the completeness of your work against the rubric before turning it in.

## Rubric: Unit 4 Assignment

| **Criteria** | **Ratings** | | | **Pts** |
| --- | --- | --- | --- | --- |
| **Exercise 1 (Contact)** | 2 pts. Correct UML class diagram following the notation introduced in lecture. | 1 pts. One or more parts incorrect:  (1 pts) all members are listed;  (1 pts) correct notation (visibility marker, static member underlined, type of data member/param, return type of methods) | 0 pts. All steps are incorrect or no submission. | 2 |
| **Exercise 2**  **(Email)** | 11 pts. Correct class meeting all requirements and working with the class from exercise 1. | 10 ~ 1 pts. At least one requirement is incorrect or missing:  (3 pts) Six data members;  (3 pts) three constructors;  (1 pts) four getXXX();  (1 pts) three setXXX();  (1 pts) toString().  (2 pts) required main() code and the class works as described. | 0 pts. Incorrect or missed all requirements; no submission | 11 |
| **Exercise 3 (TestEmail)** | 4 pts. Correct class meeting all requirements and working with the classes from exercise 1 and 2. | 3 ~ 1 pts. At least one requirement is incorrect or missing:  (1 pts) call generateEmailList() and save returned result;  (1 pts) Retrieve and print the from email address of the specified object;  (1 pts) Loop through all objects in the returned list;  (1 pts) identify and print all email objects from or to the specified email address. | 0 pts. Incorrect or missed all requirements; no submission | 4 |
| **Exercise 4. Reflection** | 1 pts. Answered both questions. | 0.5 pts. Answered only one question. | 0 pts. Didn’t answer the questions or no submission. | 1 |
| **Required program name (StuPre part) + Style Points (Proper comments; meaning identifier names; consistent indentation)** | 2 pts. Correctly named program and proper style in all three areas of style points. | 1 pts. Problems in one of the following areas: program name, three areas of style points. | 0 pts. Problems in two or more of the following areas: program name, three areas of style points. | 2 |
|  |  |  | *Total Points* | 20 |