ÇANKAYA UNIVERSITY

SENG 201 Data and Game Structures

Programming Assignment 1

This assignment is designed to help your understanding of the concepts of linked list and stack data structures.

PART I

The IT department of a company decided to build its own simple version of an email server. They divide the emails into 3 categories, namely: INBOX, ARCHIVE and TRASH. You are asked to implement a system that manages emails.

a) Email Class

First, you must implement an Email Class. An email object should hold at least the following information:

- `subject : String`
- `id : Integer`
- `message : String`
- `time : Integer   //milliseconds`

The class methods should include a default constructor, and the `set…(), get…()` methods for each of the attributes. You can add additional helper attributes or methods to the class as you need.

b) ListOfEmails Class

Now, implement a `ListOfEmails` class. The list should store the `Email` objects as its data. You can make use of the **java.util.LinkedList** library (you don't need to implement the linked list). The `ListOfEmails` class should support the following methods:

- add(Email E): Adds a new email to the list.
- read(int id): Show current Email details with the given email id
- delete(int id): Delete the Email with the given email id, and return the corresponding Email object.
- showAll(boolean flag): Print all emails details in all categories. If flag is true, all emails are shown. If flag is false, **only unread** emails are shown.

c) EmailApplication Class

First, as mentioned above, the emails are divided into three categories, namely: INBOX, ARCHIVE and TRASH. Therefore, in your program, you should create three empty email lists, with these names: `Inbox, Archive, Trash`.

Then, your program should read input from the standard input line-by-line. Each line of input will contain:

N	New email arrived. Some examples:
	
	`> N`
	`> Hello from CMPE242`
	`> 1234`
	`> This is a welcome email from the course`
	`> 2364675`
	
	`> N`
	`> Homework 1 posted`
	`> 1237`
	`> This is your first homework for the course`
	`> 2264672`

R <id>	Read an email with the given id. Some examples:
	
	`> R 1234`
	`Email id: 1234`
	`Subject: Hello from CMPE242`
	`Body: This is a welcome email from the course`
	`Time received: 2364675`
	`Status: Read`
	
	`> R 1235`
	`No such email.`

A <id>	Archive the email with the given id. This should move the email from the Inbox to the Archive. For example:
	
	`> A 1234`
	`Email 1234 archived.`

D <id>	Delete the email. This should move the email from the Inbox to the Trash. Example:
	
	`> D 1234`
	`Email 1234 is deleted.`

S <Folder>	Show the contents of the email box. The parameter <Folder> can be one of: `Inbox, Archive, Trash.` Example:

	`> S Inbox`
	`Email   Subject             Body                      Time      Read`
	`1234    Hello from CMPE242. This is a welcome email. 2364675   No`
	`1237    Homework 1 posted.  This is your first ho... 2264675   Yes`
	`...`
	
	You should print the truncated form of the "subject" and "body" strings to 25 and 40 characters, respectively. Check the Java System.out.printf() method details.

`U <Folder>`	Show all unread emails in the folder in. The parameter `<Folder>` can be one of: `Inbox, Archive, Trash.`
`C <Folder>`	Clear the contents of the folder. The parameter `<Folder>` can be one of: `Inbox, Archive, Trash.` If the parameter is `Trash,` the program should empty it. If the parameter is `Inbox` or `Archive`, then the program should move all emails from the target folder to the `Trash.`

PART II

The second part of the homework is about stacks.

a) Calculator (Calc) Class

You task is to implement a basic calculator that supports integer operands like 1, 647, and -42 as well as the (binary) integer operators +, -, * and /. The style of arithmetic expressions our calculator will evaluate is also called a post-fix notation. Stacks are great for doing this job! Your task is to write a program that uses a Stack as we discussed in class. You can make use of the **java.util.Stack** library.

Your program should be called **Calc** and work as follows:

- The user enters input consists of **operands** and **operators** one by one.
- If the user enters a valid integer, you push that integer onto the stack.
- If the user enters a valid operator, you pop two integers off the stack, perform the requested operation, and push the result back onto the stack.
- If the user enters the E, exit the program.

Note that there are a number of error conditions that your program must deal with gracefully for full credit. This means that you'll have to print error messages to the user.

Examples

```
$ java Calc
10
20
+
Result is 30
15
-
Result is 15
A
Invalid operator
E
```

You should submit one zip file name as **“YourNameSurname_PA1.zip”** and it should contain all the java files you created in different folders: “Part1” and “Part2”.

IMPORTANT

IMPORTANT NOTES: Do not start your homework before reading these notes!!!

1. This assignment is due by 23:59 on Friday, November 3rd.

2. You should submit your homework to course webonline page before the deadline.

3. The below rules about late homework submissions apply. Please see the course syllabus for further discussion of the late homework policy as well as academic integrity.

4. You ARE NOT ALLOWED to modify the given method names. However, if necessary, you may define additional data members and member functions.

5. We will test your implementation by writing our own tester java files. For this reason, your classes' name MUST BE as shown in the homework description.

8. The submissions that do not obey these rules will not be graded.

9. To increase the efficiency of the grading process as well as the readability of your code, you have to follow the following instructions about the format and general layout of your program.

    - Indentation, indentation, indentation... The format of your code makes it readable!

    - This homework will be graded by your TA, Naz Dündar. Thus, you may ask them your homework related questions. You are also welcome to ask your course instructor Bora Çelikkale for help.