Assignment #3 [9 Marks]

Due Date	8 May 2022 23:59
Course	**[M1522.000600] Computer Programming**
Instructor	Jae W. Lee

- You are allowed to discuss with fellow students to understand the questions better, but your code must be **SOLELY YOURS**. You **MUST NOT** show your code to anyone else, or vice versa.
- We will use the automated copy detector to check the possible plagiarism of the code between the students. The copy checker is reliable so that it is highly likely to mark a pair of code as the copy even though two students quickly discuss the idea without looking at each other's code. Of course, we will evaluate the similarity of a pair compared to the overall similarity for the entire class.
- We will do the manual inspection of the code. In case we doubt that the code may be written by someone else (outside of the class like github), we reserve the right to request an explanation about the code. We will ask detailed Problems that cannot be answered if the code is not written by yourself.
- If one of the above cases happens, you will get 0 marks for the assignment and may get a further penalty. Please understand that we will apply these methods for the fairness of the assignment.
- Download and unzip “HW3.zip” file from the NeweTL. “HW3.zip” file contains skeleton codes for Question 1 (in the “problem1” directory) and Question 2 (in the “problem2” directory).
- When you submit, compress the “HW3” directory which contains “problem1” and “problem2” directories in a single zip file named “20XX-XXXXX.zip” (your student ID) and upload it to NeweTL as you submit the solution for the HW1, HW2. Contact the TA if you are not sure how to submit. Double-check if your final zip file is properly submitted. You will get 0 marks for the wrong submission format.
- Do not modify the overall directory structure after unzipping the file, and fill in the code in appropriate files. It is okay to add new directories or files if needed.
- Do not use external libraries. If you are unsure, contact TAs to clarify.
- Java Collections Framework is **allowed**.

Contents

Question 1. AssetManage Application [4 Marks]
1-1. Basic Methods for AssetManage [0.5] : OOP Basic, Collections
1-2. Asset Searching with Conditions [1] : OOP Basic, Collections
1-3. Asset Buying and Trading [1.5] : OOP Basic, Collections
1-4. Asset Sharing [1] : OOP Basic, Collections

Question 2. Social Network Service (SNS) [5 Marks]
2-1. Authenticate [1] : OOP Basic, Collections
2-2. Post a User Article [1] : OOP Basic, Collections, Encapsulation, Inheritance
2-3. Recommend Friends' Posts [1.5] : OOP Basic, Collections, Encapsulation
2-4. Search Posts [1.5] : OOP Basic, Collections, Encapsulation

Submission Guidelines

1. You should submit your code on NeweTL
2. After you extract the zip file, you must have a "HW3" directory. The submission directory structure should be as shown in the table below.
3. You can create additional directories or files in each "src" directory.
4. You can add additional methods or classes, but do not remove or change signatures of existing methods.
5. Compress the "HW3" directory and name the file "20XX-XXXXX.zip" (your student ID).

Submission Directory Structure (Directories or Files can be added)

- Inside the "HW3" directory, there should be "problem1" and "problem2" directories.

Directory Structure of Problem 1	Directory Structure of Problem 2

problem1/
└── src/
  └── Asset.java
  └── AssetManage.java
  └── Lab.java
  └── Test.java
	problem2/
	└── src/
	  └── App.java
	  └── BackEnd.java
	  └── FrontEnd.java
	  └── Post.java
	  └── ServerResourceAccessible.java
	  └── Test.java
	  └── User.java
	  └── UserInterface.java
	  └── View.java

Question 1: AssetManage Application [4 Marks]

Objectives: Implement an asset management application to register and trade assets.

Descriptions: You are an equipment manager of SNU. You are asked to implement a system for maintaining the status of assets being used at SNU. This application can add new assets, search for existing assets, register labs, and trace the trade between labs. A new asset is first added to the total list of assets. Then, the owner of the asset is designated when some lab buys it.

You will need to implement various methods in the AssetManage.java, Asset.java, and Lab.java classes in src directory as instructed below.

- Asset and Lab classes are provided in Asset.java and Lab.java, respectively; you **can** change these classes if necessary.
- Test cases are introduced in Test.java. Note: we provide the basic test cases and we will use a richer set of test cases for evaluation. We strongly encourage you to add more diverse test cases to make sure your application works as expected.

Question 1-1: Basic Methods for AssetManage Class [0.5 Mark]

Objectives: Implement the following methods of the AssetManage class: addAsset, findAsset addLab, and findLab.

Descriptions: The AssetManage class manages the information of all the assets and labs. As a start, you will implement the following four basic methods to add/find assets/labs inside the AssetManage class.

To implement these methods, you will need to decide on the appropriate collections (e.g., List, Set, Map, etc.) to store Asset objects and Lab objects, and add a few lines of code to declare and initialize the collections. For these methods, it is guaranteed that every argument given is not null.

- public boolean addAsset(int id, String item, int price, String location) creates an Asset object with the id, item, price, and location and saves it in the collection.
    - If the Asset object with the same id was already registered, do not create an Asset object and return false. The id of an asset should have a non-negative value. If a negative id is given, return false.
    - The price of an asset, price, is in the range of [0, 100000]. If the given price is out of this range, do not create an asset object and return false.
    - Return true if a new Asset object is created and stored successfully.
- public boolean addLab(String labname) creates a Lab object with the labname and saves it in the collection.

    - If there already exists a lab with the given labname, do not create a Lab object, and return false. If a null or empty string is given for the labname, return false.
    - Return true if a new Lab object is created and stored successfully.
- public Asset findAsset(int id) returns an Asset object with the given id.
    - Return null if there is no asset with the given id.
- public Lab findLab(String labname) returns a Lab object with the given labname.
    - Return null if there is no Lab with the given labname.

Question 1-2: Asset Searching with Conditions [1 Mark]

Objectives: Implement the findAssetsWithConditions method of the AssetManage class.

Descriptions: Implement the method to search for the assets matching the given conditions.

- Implement the public List<Asset> findAssetsWithConditions(int minprice, int maxprice, String item, String location) method.
- The method returns a List of Asset objects matching **all** of the given conditions.
    - A price condition is given as follows. Filter only assets of which price is in the range of [minprice, maxprice]. If the user wants to skip this condition, wildcard option can be used: both minprice = -1 and maxprice = -1. Using only one side is not allowed (e.g. minprice = -1 and maxprice = 5000 not allowed)
    - An item and location conditions select assets based on asset name and location, respectively. Also, the wildcard option can be used in both cases, and the keyword of the wildcard is **All** in string.
    - If there is no asset matching all these conditions, return an empty list.
    - The output list should be sorted in an ascending order of the asset id.
        - e.g. [10, 300, 5000, 5100]

Question 1-3: Asset Buying and Trading [1.5 Mark]

Objectives: Implement the buyNewAsset, tradeBtwLabs methods of the AssetManage class.

Descriptions: You will now implement two methods: (1) to buy an asset and (2) to trade an asset between two labs.

- public boolean buyNewAsset(Lab lab, int id)
    - This method tracks a lab buying an asset. Note that it is only used when an asset has no owner. (e.g. owners.size() == 0)
    - Do nothing and return false in one of the following cases.
        - The asset with id does not exist, or the lab is given null.
        - If the asset is owned by the specific lab (owners.size() is not 0).
        - When the lab cannot buy this asset due to its insufficient balance.
    - Otherwise, return true with the following modifications.

        - Add the asset to assetInventory of the lab.
        - Subtract the asset's price from the lab's balance.
        - Add the lab to the asset's owners.
- public boolean tradeBtwLabs(Lab buyer, Lab seller, int id)
    - This method tracks an asset trading between two labs. Note that it can only be used when an asset has at least one owner. (e.g. owner: seller)
    - In case the number of the owner of an asset was only one before the trade, the transaction price is the same as the asset's price. However, for the assets with more than one owner, the transaction price is defined as the asset's price divided by the total number of owners. (e.g. if there were two owners, then the transaction amount is price/2) If the transaction price is not an integer number due to division, floor it to an integer.
    - Do nothing and return false in one of the following cases.
        - The asset with id does not exist.
        - The seller or buyer is given null.
        - If the asset is 1) not owned by any lab or 2) it is not owned by the seller or 3) it is owned by the buyer already.
        - When the lab cannot buy this asset due to its insufficient balance.
    - Otherwise, return true with the following modifications.
        - Add the asset to assetInventory of the buyer.
        - Remove the asset from assestInventory of the seller.
        - Subtract the transaction price from the buyer's balance.
        - Add the transaction price to the seller's balance.
        - Add buyer to the asset's owners and remove the seller from the asset's owners.

Question 1-4: Asset Sharing [1 Mark]

Objectives: Implement the assetOnShare method of the AssetManage class.

Descriptions: You will implement a method to track the sharing status of an asset between multiple labs.

- public boolean assetOnShare(Lab sharer, int id)
    - This method tracks multiple labs sharing an asset. Note that it can only be used when an asset has at least one owner.
    - In case the number of the sharing owners of an asset was initially N and then, it becomes N+1 if our new sharer is added to share it. After that, the sharing price will be the asset's price divided by the total number of its owners, N+1. If the sharing price is not an integer number due to division, floor it to an integer. Modify the balance of all the labs sharing the item to meet this new expenditure. (e.g. both the new/existing labs that own the asset)
    - Do nothing and return false in one of the following cases.
        - The asset with id does not exist.

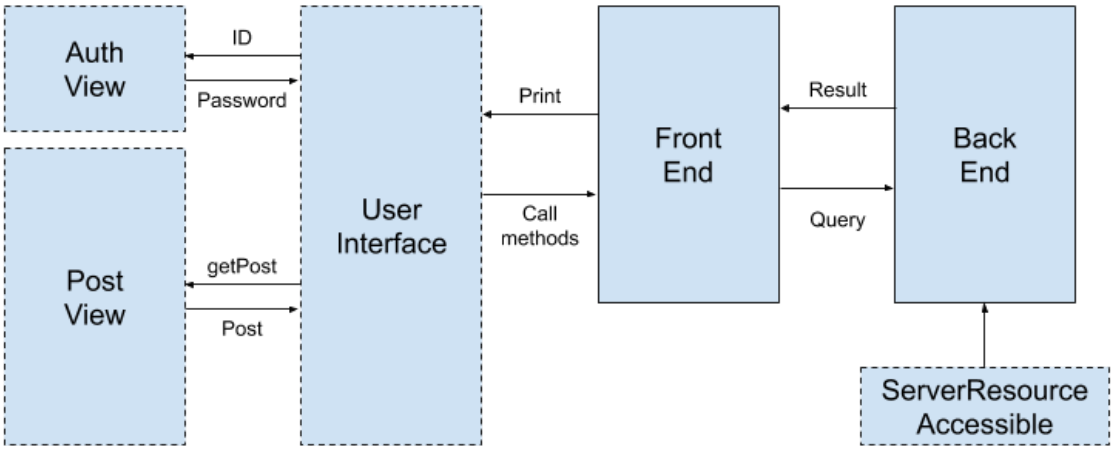
        - The sharer is given null.
        - If the asset is 1) not owned by any lab or 2) it is owned by the sharer already.
        - When the lab cannot buy this asset due to its insufficient balance.
    - Otherwise, return true with the following modifications.
        - Add the asset to assetInventory of the sharer.
        - Subtract sharing price from the sharer's balance.
        - Add the amount of difference occurring from the owners' number (from N to N+1) to previously owning labs.
        - Add sharer to asset's owners.

Question 2: Social Network Service (SNS) [5 Marks]

Objectives: Implement a simple Social Network Service (SNS).

Descriptions: We are going to build a simple console-based SNS system. A user can write posts, search them, and view the recommended friends' posts.

Before you jump into the implementations, you may first want to understand the overall structure of this application. The below figure shows the overall architecture of our SNS application.



The application mainly consists of 3 classes: `UserInterface`, `FrontEnd`, and `BackEnd` (along with a few additional classes supporting these main classes).

- The `UserInterface` class provides interfaces to users.
    - It does the authentication at the beginning of the application through the `AuthView` class.
    - After the login, the `PostView` class gets a command input from the user by `String getUserInput(String prompt)` method and calls the corresponding method of the `FrontEnd` class.
    - The results of the commands, provided by the `FrontEnd` class **should be displayed using the print methods** of `UserInterface` in an appropriate form.
        - `UserInterface.println` : Equivalent to `System.out.println`
        - `UserInterface.print` : Equivalent to `System.out.print`
- The `FrontEnd` class parses user commands/posts, and queries the `BackEnd` class (by calling the appropriate methods) to process them.
- The `BackEnd` class receives the queries from the `FrontEnd` class, processes them, and returns the results.
    - The `BackEnd` class inherits the `ServerResourceAccessible` class.
    - The `ServerResourceAccessible` class has the member attribute named `serverStorageDir`, which specifies the path to the data directory. All the necessary

data (e.g., user information, friends information, user posts) are stored as files under this directory. More details will be explained later.
    - Note that the `serverStorageDir` value ends with "/".
    - In subsequent descriptions, we will describe the data directory path as the $(DATA_DIRECTORY).
    - In the given skeleton, the data directory is set to "data/", but it may be changed to another path in the final testing. Do not assume that the data directory is always "data/". Use the `serverStorageDir` attribute to retrieve the data path.
- The program should do nothing in case of any exception (e.g., non-existence of the user id).
- Default test cases specified in this document are provided in `Test.java`. You can compare your result from `Test.java` with the expected outputs in this document. Note: we provide the basic test cases, and we will use a richer set of test cases for evaluation. We strongly encourage you to add more diverse test cases to make sure your application works as expected.

Great! You are now ready for the implementation of the cool SNS application.

- You need to change the `FrontEnd` and `BackEnd` classes to solve this problem. Please do not modify the rest of the classes.
- Do not modify the signatures of the `FrontEnd` class methods since we will use them for the final testing.
- There are 4 subproblems that are meant to be solved in order. If you proceed without solving the earlier problems, it may affect the later problems.

Question 2-1: Authenticate [1 Mark]

Objective: Implement the method `public boolean auth(String authInfo)` of the `FrontEnd` class to authenticate the user. You may need to make appropriate changes to other parts of the `FrontEnd` and `BackEnd` classes as well. In particular, it compares the input password with the password stored in the server to check its validity.

Description: Upon the program start, the console will ask for the user id and the password. In the given skeleton code, a user cannot log in even with a valid password. Now, we want to change it so that login can be possible with a valid password.

- The password of a user is stored at the path $(DATA_DIRECTORY)/(User ID)/password.txt; here, the (User ID) is the id of the user.
- Assume that all the names of the direct child directories of $(DATA_DIRECTORY) are valid user ids.

- Assume that every $(DATA_DIRECTORY)/(User ID) has a password.txt.The format of the password.txt is given in the following example (plain text without a newline). Suppose the password of the user ‘root’ is ‘pivot@eee’.

[File Format] $(DATA_DIRECTORY)/root/password.txt
pivot@eee

- For successful authentication, the input password and the stored password should be **identical** including white spaces.
- If the login fails, the program terminates.

Note that text in blue in the following example indicates the user input:

Console prompt (Login in this case)
------ Authentication ------ id : root passwd : pivot@eee ---------- root@sns.com post : Post contents recommend <number> : recommend <number> interesting posts search <keyword> : List post entries whose contents contain <keyword> exit : Terminate this program ---------- Command :

Console prompt (Fails Login in this case)
------ Authentication ------ id : root passwd : admin2 Failed Authentication.

Question 2-2: Post a User Article [1 Mark]

Objective: Implement the `public void post(List titleContentList)` method of the `FrontEnd` class to store the written post in the server. You may need to make appropriate changes to other parts of the `FrontEnd` and `BackEnd` classes as well.

Description: When a user inputs the “post” command to the console, he can start writing a post with the 1) title, 2) whether the post is for advertising or not, and 3) content. The content of the post ends when the user inputs “Enter” twice.

- Store the user’s post at the path $(DATA_DIRECTORY)/(User ID)/post/(Post ID).txt. (User ID) is the user’s id used for the login, and the (Post ID) is the nonnegative integer assigned uniquely to each and every post in the $(DATA_DIRECTORY). The newly assigned ID should be **1 + the largest post id in the entire posts in $(DATA_DIRECTORY)** or 0 in a case when $(DATA_DIRECTORY)/(User ID)/post/ is empty.
- Assume all the $(DATA_DIRECTORY)/(User ID) has a directory named post, and each post directory has at least one post.
- The format of the post file is given in the examples below.
- The variable “Advertising” can only be assigned “yes” or “no”.
- The content of the post should not include the trailing empty line.
- Hint: use the `LocalDateTime` class, and `LocalDateTime.now` to get the current date and time. Refer to the `Post` class implementation to format the date and time.

Let the user name be ‘root’, and the largest post id in the entire $(DATA_DIRECTORY) be 302. Then, the new post id is 302 + 1 = 303. Let the post date be 2022/04/08 10:01:02.

Console Prompt

```
Command : post
----------------------------------
New Post
* Title : my name is
* Advertising (yes/no) : yes
* Content :
> root and
> Nice to meet you!
>
----------------------------------
```

Then the post is saved to "$(DATA_DIRECTORY)/root/post/303.txt", as shown below. The date, title and advertising input fields are written in each new line, respectively. There is an **empty line after the advertising input**, and finally, the content is written.

[File Format] $(DATA_DIRECTORY)/root/post/303.txt

```
2022/04/08 10:01:02
my name is
yes
```

root and
Nice to meet you!

Question 2-3: Recommend Friends' Posts [1.5 Mark]

Objective: Implement the `public void recommend(int N)` method of the `FrontEnd` class to print the "advertised" and latest posts of the user's friends. You may need to make appropriate changes to other parts of the `FrontEnd` and `BackEnd` classes as well.

Description: Our SNS service recommends a user the latest posts of her friends. When the user inputs the "recommend <N>" command to the console, up to N latest posts of the friends should be displayed. However, an advertisement post has a higher priority than the latest posts.

- The list of the user's friends is stored at the path "$(DATA_DIRECTORY)/(User ID)/friend.txt". The format of the friend.txt is given as the following example. Suppose the user "root" has 3 friends, "admin", "redis", and "temp".
- You need to look at all the posts of the friends and print up to N posts prioritized by the criteria of 1) advertising and 2) latest creation date.
- How do we decide which N posts to recommend?
    - First, filter the **advertising posts** from all the posts of the friends. Then we have two types of lists: 1) advertised, and 2) non-advertised.
    - Sort both types of posts by the creation date specified in a post file in descending order (from latest to oldest).
    - Select the first N posts from the sorted list of **advertising posts**.
    - If N is bigger than the total number of advertising posts, the remainder should be selected in non-advertised posts.
    - If there are less than N posts in the whole sorted list, recommend all posts in the list.
- Assume the creation date and time of each post is unique. No two posts have the same creation date and time.
- Assume all the friend ids on the friend.txt are valid, and the corresponding folders exist in the $(DATA_DIRECTORY).
- The post should be printed as a format of the `toString` method of the `Post` class, not the `getSummary` method, with the `UserInterface.println` method.

[File Format] $(DATA_DIRECTORY)/root/friend.txt
admin redis temp

In the example below, the command ‘recommend’ displays 1 latest posts of “admin”, “redis”, “temp” users.

Console Prompt (Authenticated with ‘root’)
Command : recommend 1 --- id: 330 created at: 2019/08/13 00:00:00 title: her corespondent conscienceless cobitidae aneurysmal content: my where agonized why he are a balistes why detort her then aeronaut antithetically brachyurous a birdseye my where delegacy her carbonic the beslubber am why are egoism why where then is then awl claudius her decretive a am are columbia dracula cataclasm antiquity discernability brassavola big(a) arbitrage chasse basics chudder clem cockchafer chemisorptive carancha cassiterite bivalvia demythologization archegonial deadline delicious casein dialectics adeology exacum aere compunctious benignant aere dislike calligraphic concourse bowstring calque beaucoup epicedium crayon cosmic chytridiomycetes adventist epiphytic eelpout bewilderingly ahab capite abracadabra dampen accension cato badli barefaced calumny diverticulum compatibility depressive enactment dervish colostomy aminopyrine deperdition clarified arboriform disquisition decrepitude easternmost colloquy convertible cravat acousma dinginess ...

Question 2-4: Search Posts [1.5 Mark]

Objective: Implement the `public void search(String command)` method of the `FrontEnd` class to display up to 10 posts that contain at least one keyword. You may need to make appropriate changes to other parts of the `FrontEnd` and `BackEnd` classes as well.

Description: Our SNS service enables users to search for posts with multiple keywords. When the user inputs the “search” command along with a set of keywords, the console should display up to 10 posts containing the most number of keywords.

- The range of the search is the entire posts of all users (NOT friends only) in the $(DATA_DIRECTORY).
- The command string starts with “search” followed by keywords.

- Two keywords are separated with space(‘ ’). The newline should not be considered as a keyword.
- Duplicate keywords should be ignored. For example, the output of `search hi hi` should be identical to the output of `search hi`.
- You should count the number of occurrences of the **exact** keyword from the title and the content of the post.
- More specific details for the keyword matching:
    - It should be a case-sensitive comparison.
    - You should only count the word that is identical to the provided keyword. You don’t need to consider the word that has the given keyword as a substring.
- How do we decide which posts to show?
    - Sort the candidate posts based on two criteria. First, sort by the number of occurrences of the keywords in descending order. When multiple posts have the same number of occurrences, sort them by the number of words in the contents of the post in descending order.
    - No two posts have the same number of words and occurrences of the keywords.
    - Select up to 10 posts from the beginning of the sorted list.
- The posts should be displayed using the format provided by the `getSummary` method of the `Post` class, not the `toString` method.

Console Prompt
Command : search centavo id: 282, created at: 1988/03/28 00:00:00, title: astounding cavity cerussite her then id: 397, created at: 2016/03/31 00:00:00, title: corundom cudgels asphyxiating he dubitousness id: 371, created at: 1981/05/14 00:00:00, title: am baby-wise despumate anatomist breeched id: 368, created at: 1977/07/21 00:00:00, title: balanidae why everyone decretive corral id: 356, created at: 1988/08/05 00:00:00, title: are my blastopore a is id: 347, created at: 2010/06/15 00:00:00, title: balderdash begird why the arguementum id: 309, created at: 2008/03/26 00:00:00, title: boltonia my are derris canard id: 234, created at: 1995/03/22 00:00:00, title: deo ester he admired antiinflammatory id: 181, created at: 1972/12/18 00:00:00, title: delphinidae enets camisade why determinedly id: 169, created at: 2001/10/08 00:00:00, title: centavo you you carpellate chives