B99902079 李哲維

OOP homework #3

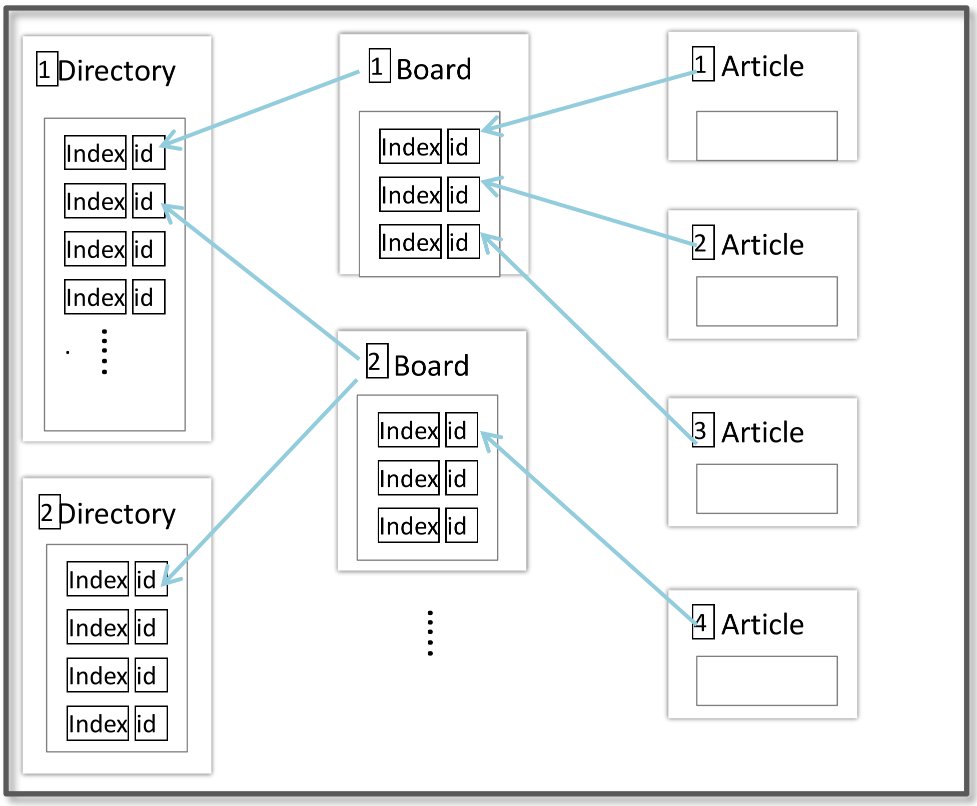
Report

The relations between classes

Directory class controls itself and ***board*** class.

When a ***board*** needs to be added to a ***directory***, we should call ***directory*** class to append it. And this ***directory*** wouldn't store the whole data of ***board*** but store some key information about the ***Board*** such as "board's id" to refer it when needed.

And there is same relation between ***Board*** class and ***Article*** class.



The advantages of this design

- <u>Easy to maintain classes</u>

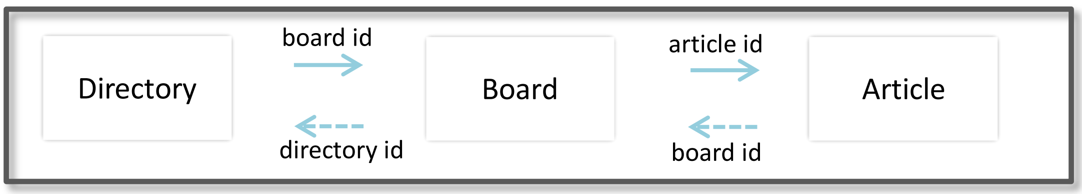
There are so many ***articles***, ***boards*** and ***directories*** in POOBBS and the same board could be appended to different ***directories***. If different classes store the data of each other, it would be very complex and difficult to maintain.

Thus, we store their id only which is unique. Even if one ***board*** appended to many ***directories*** is edited by someone, we can get the same result from other ***directories***.

- Easy to access data from different classes

When user is in ***directories***, he can do functions of ***directories*** or appended ***boards***; and it's easy to change from ***directory*** mode to ***board*** mode by board's id. Similar, user can do function of ***boards*** or appended ***articles*** and easily to access the target article.

Each mode change between these classes, we record the id of class. Thus, we can only retrieve id to go opposite direction.



- Friendly for debug

Because the data is stored at its class, if we find there is some problem in some ***directory***, we can just check the ***directory*** class and find bug there.

The disadvantages of this design

Because one wanting to view board must start from ***directory*** and to view ***article*** must start from ***board***, it may take more effort to access the classes sometimes.

Maintain code in repository

Whenever I make some functions in class, I will repo it to record the process.

And before to debug, I'll make a branch. If the bug is fixed, then merge it. By this way, it solves the problem that in order to debug, one adds a lot of commands and sometimes destroys the original version.

I write the codes in Eclipse while need to run it on Linux because I add some color of text words which work in Linux only. So I edit the program in Eclipse and test it in 217 workstation by pushing code from Eclipse and pulling to 217 workstation.

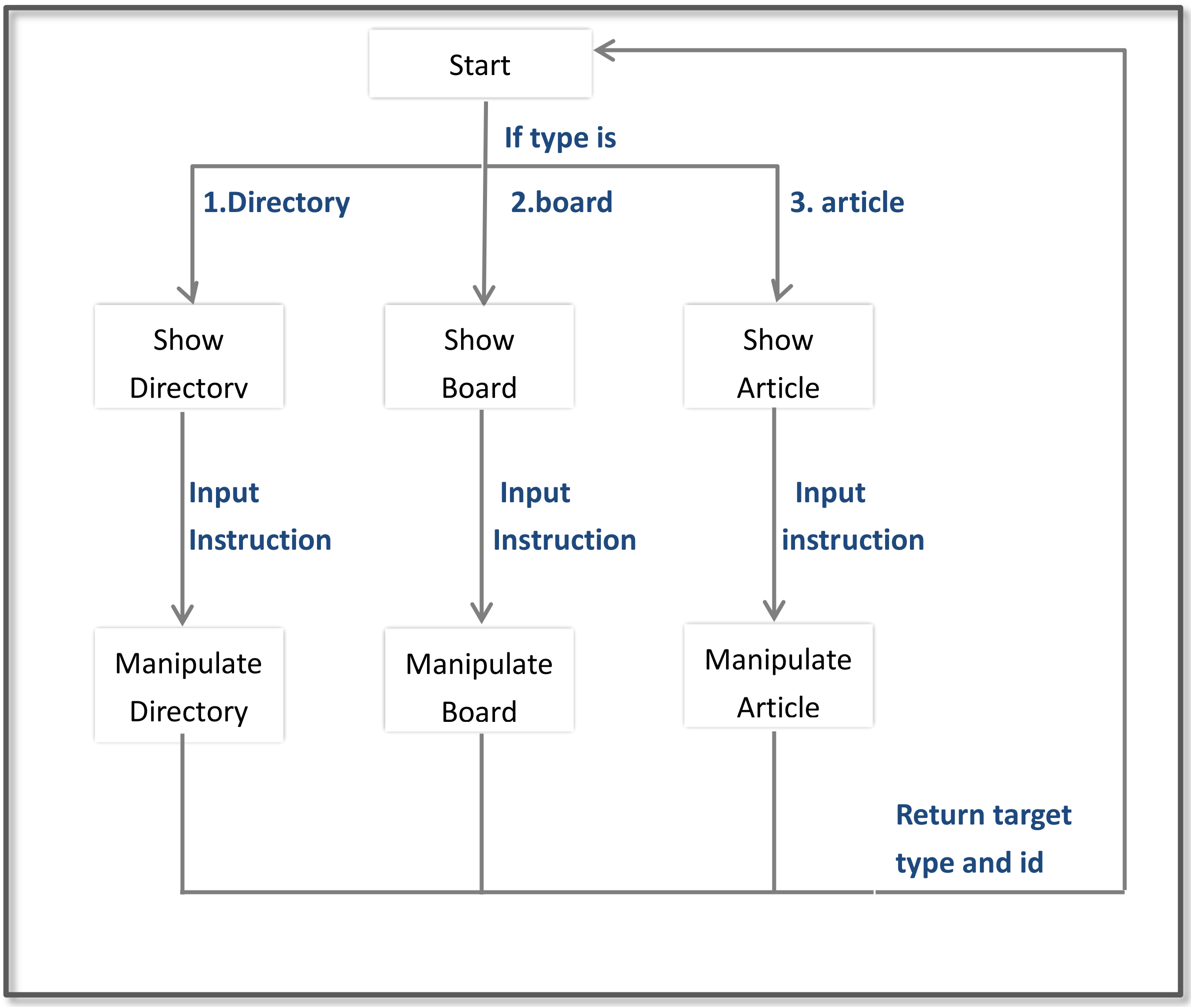
Bonus

- <u>Structure</u>

The main program called POOBBS.java has an easy structure. We can easily manipulate these classes.

Each instruction that user give will cause a flow as following chart, and it finally go back to start point and wait for the next instruction.

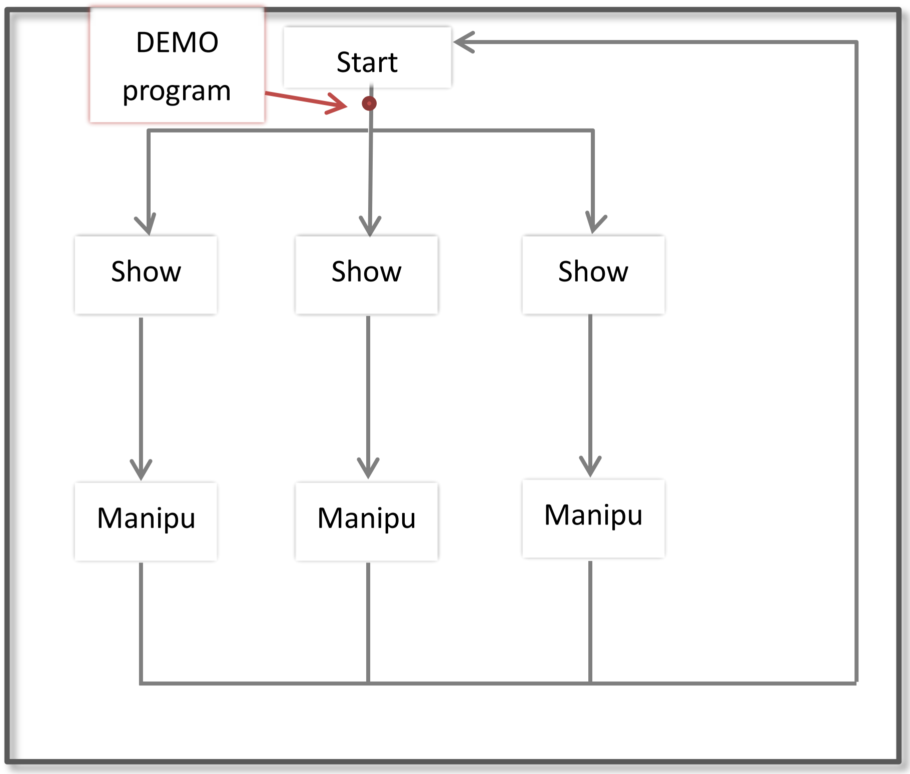
For example, when instruction tells the program we want to enter board, then it will show board in the next flow and wait for instruction of board.



- <u>Demo program</u>

  Due to this design, I can easily add demo flow like the following chart.

  The DEMO program is designed as that each loop it will print a demo line to user, so that the user can follow the command and know how it works.



- <u>Add color and clear screen in Linux</u>

  I add some color in demo version which makes interface clearer.

  And after each instruction, the screen will be cleared, just like the real BBS.