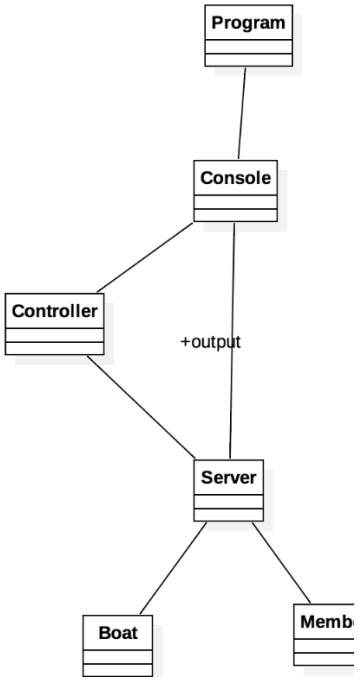
Object Oriented Analysis and Design

Workshop2 : Design

Guillaume Fumeaux

07 october 2015

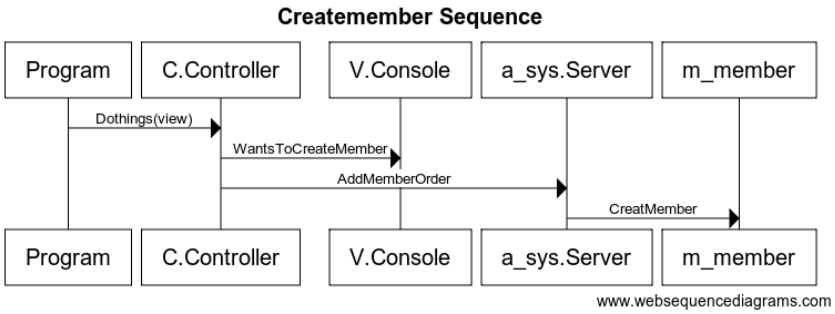
Class diagram



Review

1. You should indicates the direction and type of relation between classes.[1]
2. The diagram doesn't equals to the code.
    a. Program/Main instances server, console, controller and use a method of controller with view and server as parameter. The relationship type should be choose in consequence.[1]
    b. Member has a list of boats as attribute, it should have a relationship between them. This is typically a type of relationship.[2]
    c. I didn't find any relation between server and console in the code.
3. It could be nice to show in which package each class is. To show the MVC pattern.
4. The name of the instance can be mentioned with the multiplicity.[3]
5. Didn't understand the +output text.

Sequence diagram create member



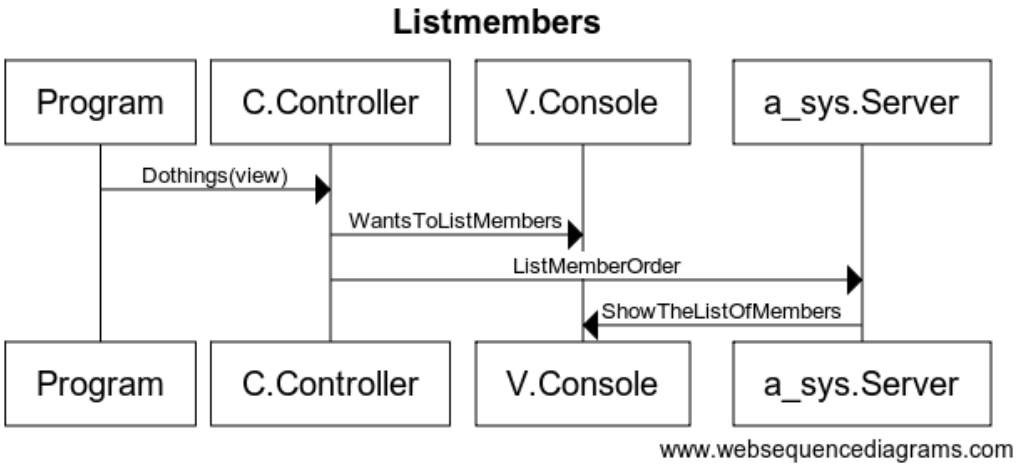
Review

1. The diagram doesn't corresponds to the code.
    a. After the program called dothings() from controller, controller call getEvent() from console, then wait for the user to select addmember. The console return the event to the controller. The controller call addmmeber() from server. There server create a member, ask the user to enter the name and

personal number. Finally he adds the member. Everything should appear on the sequence diagram, the intput from the user and the return from a function.

2. Don't forget to put the actor on the diagram, he performs some actions. [4]
3. When you create an object the arrow goes directly on the object. [4]
4. Use the correct function's name, the same as in the code. It should really represent the final code.[4]

Sequence diagram list member



Review

1. It should either represent CompactList, or VerboseList there is no Listmember in the code.
2. Same remark as before. You should really follow the code and show exactly what it does.[4]

Test runnable version

Bugs

1. Input doesn't work as it makes references to your folder "/Users/CaptainYan/Documents/workspace/WorkshopTwo/" just let data.txt that's enough
2. Any test is made on what the user enter, so the program crash every time we entry something false. The entries should be tested, or at least ignored if wrong.
3. When we modify a boat or a member, it would be good to have the possibility to not modify something. IE: I want to modify the name of the user, I wouldn't have to enter the personal number again. It can be done easily with just a test of if str.isEmpty();
4. To delete a boat you're asking to type the number of the boat to select which one, without showing any number.
5. You don't test if the personal number is already inside the data base. So it is possible to have 2 member with the same personal number, and then you are using it to find back the user. Either test if it already inside, or ask for the id of the user.
6. You transform the personal number of the member into double which add the .0 at the end of it. You should use int or let it as a string.
7. If we try to input, as there is no backup file, the program crash. File has the method file.exist("data.txt") to test before if it exists or not.
8. If an error is made when choosing the action, the program is shut downed, it would be better to go back to the menu instead, as the backup isn't automatic, you can lose everything.

Architecture

1. There is a model view separation, however it isn't respected. There is System.out.println everywhere in the code, whereas it should be only in the console. [5]
2. The input are taken by the server, it should be the console again. [5]
3. The console should be in charge of output and input. Usually a method of the console is called by the controller, and the console return the entry. The same for output. A console's method should be called with in parameter if some data should be showed. Console should have no error if controller and model are deleted.
   Controller should just transmit data between the view and the model. IE: controller ask the view to know which option has been selected, the view return the selection, the controller call the model to perform the option selected. Send to the view if some data have been send back by the model and have to be shown.
   The model shouldn't have any error if the controller and view are deleted. It should just perform what is asked by the controller, or send the data asked by the controller. You can have as in your code like a hierarchy with the model. Server asks member who asks boat.
   The class diagram should be design in consequence. [5]

Code

1. A new scanner is created each time, maybe one for the whole method would be enough, and create a method getName(), getPersonnalNb() would avoid to repeat the code multiple times.
2. The attribute boo should be renamed like run… Something explaining what it is used for.
3. Don't forget to put in capital different part in a name. setPersonalnumber should be setPersonalNumber. And some name like str could be named entry, input, entry2, input2
4. Some code is double, maybe it would be better to have one method called. IE: ChangBoatInformation() and DeleteBoat() are really similar.

Design

I didn't found anything to say about that. Maybe it is due to a lack of knowledge for me but I think all criteria are respected.

Questions:

As a developer would the diagrams help you and why/why not? As explained below, yes they help to have a general idea, but should be more precise.

What are the strong points of the design/implementation, what do you think is really good and why? The design is very well implemented. It respect the grasp pattern

What are the weaknesses of the design/implementation, what do you think should be changed and why? The console, who doesn't do his job at all. All input and output should be there.

Do you think the design/implementation has passed the grade 2 criteria?

If the view is corrected yes.

Bibliography

1. Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, Ch 16.4 16.11
2. Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, Ch 16.4
3. Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, Ch 16.15
4. Visual-paradigm, Sequence diagram, 08-10-2015, http://www.visual-paradigm.com/VPGallery/diagrams/Sequence.html
5. Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, Ch 13.7