Peer review - Andreas Anemyr

https://github.com/AndreasAnemyrLNU/1dv607/tree/master/Workshop-2-Design-NY

Disclaimer: The project is not yet finished, there is a lot of more or less empty classes and when running the program the `NotImplementedException` *is thrown more or less all the time. Andreas Anemyr is fully aware of this and wants feedback on what he has done so far in order to finish it before the final deadline.*

Architecture and Diagrams

UML class diagram only shows inheritance and not associations or dependencies. Therefore it is difficult to understand the relationships between different layers, like the model and view. Class diagram should also contain attributes and operations (1, p.250 section 16.1).

The dot at the beginning of the sequence diagram represents “a found message whose sender will not be specified” according to Larman (1, p.228 section) and it is correct to have it at the beginning of the diagram. But why have them in the switch statement when the participant are known/specified (2; 1, p.229 section 15.4)?

Code Quality

There is good naming in the code but since the use of many interfaces and inherits it would be help with more comments in order to understand the code. There is a good separation between model and view most of the time, but for example the model-class `Member` returns pre-formated strings which is not allowed according to MVC.

Design Quality

There is a lot of inheritance but it’s not always obvious why. For example, the `Member` inherits from `Crud` which in turn inherits from `View`. However, `Crud` has methods hiding some of the functionalities from `View`. So maybe another pattern, for example composite (1, p.452 section 26.8), would be a better solution. All the inheritance also makes it high coupling instead of low, as is preferable in the GRASP pattern (1, p.299 section 17.12). There is also a strange occurrence where the `IView` inherits from `ICRUD`. Then there is the `View` class that implements `IView` (and therefore also `ICRUD`). The `Crud` class inherits `View` class, that implements `IView` and `ICRUD`, and in turn implements `ICRUD` on its own. That means `Crud` both inherits a class with `ICRUD` implemented and implements `ICRUD` itself, again! This means that `Crud` has a lot of unnecessary code and functionality in it that it don’t use. This is an example of low cohesion, which the GRASP pattern want to avoid (1, p.314 section 17.14).

There is some premature optimisation (4) with interfaces, both `IClientCommunication` and `IMenu` only have only been implemented once each, which does not indicate a need for interfaces. There is also an interesting for-loop in `View.View.TitleOut()` where the for-loop is used to display a string twice. The easiest approach would probably just have been to store the string as a variable and write it twice.

The class `Cache` has 108 references to it which makes it really difficult to understand what the `Cache` actually do. Just looking at the code it looks like it caches more or less the whole program in one single object! Maybe the `Cache` can be splitted into smaller caches and also classes handling the state.

Inheritance or generalization/specialization, is in UML “shown with a solid line and fat triangular arrow” (1, p. 260 section 16.10). According to MSDN “A generalization relationship between a specialized type and a general type”(3). But can you really say that for example Boat is a type of Crud as suggested in the view classes?

Is the requirement of a unique member id correctly done?

Member does not have unique member ID unless social security number is considered an unique id.

As a developer would the diagrams help you and why/why not?

The diagram takes long time to understand and orient the meaning of different classes. Hard to find links between the various classes in the MVC structure

What are the strong points of the design/implementation, what do you think is really good and why?

Good separation between view and model. Nice user interface.

What are the weaknesses of the design/implementation, what do you think should be changed and why?

Design is difficult to understand. Unnecessary use of inheritance? Inheritance creates dead code when the class that inherits hide the code it inherited.

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

As the disclaimer said, the project is far from finished but Andreas Anemyr is aware of that.

References

1. Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062
2. Microsoft Developer Network, UML Sequence Diagrams: Reference, 2015, https://msdn.microsoft.com/en-us/library/dd409377.aspx
3. Microsoft Developer Network, UML Class Diagrams: Guidelines, 2015, https://msdn.microsoft.com/en-us/library/dd409416.aspx#Inheritance
4. Schubert Erlandsson J., Guest lecture at LNU, 2015, https://connect.sunet.se/p1x4fnql2zf/