Technical Peer review

Reviewing each other’s code (paired assignment)

In this assignment you are going to split your team up in two pairs of two students. Each pair is going to review a product of the other pair (for instance the scheduling part of the application, or the schedule website). Below you will be asked questions with regard to topics that are treated in OOD.

**What to do:**

1. Your tutor will split your group up into pairs of two.
2. Together with your tutor you decide what code base you will assess as a pair (code that you did not develop yourself).
3. You answer the questions below before the final meeting in week 15.
4. In the final meeting in week 15 you present/discuss your answers with the tutor and the other pair.

|  |  |  |
| --- | --- | --- |
| **Reviewer name 1** | Plamen Peev | |
| **Reviewer name 2** | Click or tap here to enter text. | |
| **Assessed code base** | Website - Employee side | |
| **Date** | 04-Jun-21 | |
| **Explain how classes are properly implemented in the target code (e.g. proper use of access modifiers [private, protected, public], proper use of static and other keywords when applicable, etc.)?** | | |
| MVC is clearly applied. Logic layer consists of 2 folders - classes and functionality. Properties inside classes have the correct access modifiers(private), helping methods inside classes are also private, so only code that is required in the presentation layer is exposed with public.All classes are devided in a particular file with a logic name, so that you can identify each class' purpose. | | |
| **Does the target code apply inheritance to generalize their code? Do you agree with their choice, elaborate your answer.** | | Yes |
| Due to our design choices, inheritance is applied in every class which holds certain CRUD functionality. The classes responsible for CRUD-ing employees,schedules, etc., all inherit from the Database class, which is responsible for communicating with the database. In my opinion, there is no other need for inheritance, except concerning the Database class. | | |
| **Does the target code apply Single responsibility to isolate individual responsibilities? Do you agree with their choice, elaborate your answer.** | | No |
| Single Responsibility is applied, but from my perspective, it isn't applied correctly. There classes, which are responsible for only 1 or 2 actions, but they can be combined with other classes which have a common functionality. For example, the following classes: "UpdateGeneralShiftPreference", "UpdateSpecificShiftPreference", ”SetSpecificShiftPreference", "SetSpecificShiftPreference". Only from the names of these classes, we see that they are functions,not classes themselves. All these classes can be combined in one class that could be something like "ShiftPreferencesManager". The same counts for Sick Reports and Dayoff Requests. In conclusion, Alexander separated every single function in a separate class, instead of combining the common functions in one general class. | | |
| **Does the target code apply the Open-closed principle to allow extension of behaviour without modification of existing classes in places where change/extension is expected? Do you agree with their choice, elaborate your answer.** | | Yes |
| An advantage of Alexander's design is that because he devided every function in a separate class, it is considerably easy to extend the functionalities without modifying other ones. Still, if changes were to be applied, it will take a lot of time to implement them, because if we continue defying a function as a class, at one point we will have too many files and we would be confused when and where do we need the specific function(in Alrexander's design -> function is equal to a class). | | |
| **Does the target code apply the Liskov substitution principle to take benefit of polymorphism? Explain where this happens, or if not, relate how your earlier answer about inheritance can allow communicating with the base class/interface.** | | Yes |
| As inheritance is used only for the Database class, I cannot conclude anything about the Liskov principle. We may assume that the principle is met,because of this inheritance, but we don't have any other indicator except that. | | |
| **When applicable, what other object-oriented design principles are applied in the target base (e.g. interface segregation, dependency inversion, etc.)?** | | |
| To be fair, I think Alexander used more functional programming than object-oriented. Just because he has classes, this doesn't mean it fully applies as object-oriented. They way classes are used, their only benefit is storing data from the database. There arent't management classes, dedicated for CRUD-ing a specific item. Rather, the CRUD for everything is split up in separate classes. I cannot see this as object-oriented | | |

|  |  |
| --- | --- |
| **Is the target code readable (clear naming convention, conscious use of white spaces, proper tab use (indentation)). What suggestions do you have, or what is particularly well done regarding readability.** | Yes |
| Particularly well done is the naming convention of the files and classes. It is very easy to navigate through the website. Except applying MVC, each folder has a logical name, as well as the files inside. | |
| **Below you have space for any other tips you want to share with the programmer of your target code?** | |
| My advice as a fellow colleague to Alexander is to pay more attention during lectures. He is a really smart guy, but his mistake is that he comforts himself with his past experience, therefore he is staying with the same knowledge without a considerable improvement. | |