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** | Ioan Popa | |
| **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.)?** | | |
| Code is split into classes that have the proper designated names and functionalities that relate to said functionalities. The software architecture follows the Data Access Layer format. The Logic Layer is split into folders. Access modifiers of the properties alongside the helping methods inside of the classes are declared as private. The only code that is "exposed" is the code that connects the Logic layer to the Presentation Layer and is used to dynamically load data. | | |
| **Does the target code apply inheritance to generalize their code? Do you agree with their choice, elaborate your answer.** | | Yes |
| In this instance inheritance was used when a class that has CRUD-ing functionality, inheriting from Database class. This class makes possible the connection to and from the database and help query data in a way that secures SRP. From my point of view, this type of inheritance is enough, judging the implementations and specifications that the website is capable of. | | |
| **Does the target code apply Single responsibility to isolate individual responsibilities? Do you agree with their choice, elaborate your answer.** | | No |
| The layout of the classes implies that each class has at its base, a method that acts like a single task. In simpler terms, the classes are divided by very small responsibilities, so small that classes such as "FindDayByDate", "FindDayByID" and other "Day" related classes (in this case) are too divided, even by SRP standards. | | |
| **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 |
| If going forward we decide to keep the current design choice for the classes, then the addition of new implementations will be a little easier. The downside to this is that, in the future, there will be an overwhelming amount of files to navigate. | | |
| **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 |
| In the case of our webiste, inheritance is only used in relation with the Database Class. This alone might be a strong enough argument to enuntiate a stabile use of the Liskov substitution principle. | | |
| **When applicable, what other object-oriented design principles are applied in the target base (e.g. interface segregation, dependency inversion, etc.)?** | | |
| The code written for the website follows a more practical vision, rather than a SOLID one. The architecture implies that classes have more of a "transitional" role rather than a logical role (not saying that logic and data management is not properly used). Keeping this in mind, this way of writing code does not adhere to the normal standards of the SOLID functionality. | | |

|  |  |
| --- | --- |
| **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 |
| Aside from the nice and readable way of styling the file architecture, the naming conventions of the classes do help a lot (keeping in mind my answer to the previews question regarding the way of splitting classes). This keeps the possibility of editing existing code fairly easy if you know what you want to add/update/delete! | |
| **Below you have space for any other tips you want to share with the programmer of your target code?** | |
| My advice for my friend and colleague Alexander is to always pursue what he wants. Even if his presence in classes was not felt, it wont be bad if from time to time he could show up to the lectures (but that is not a choice anymore). I hope that his decision of changing universities was the correct choice for him, and I hope that he will find there what he was not able to find here! | |