Technical Peer review

Reviewing each other’s code (paired assignment)

In this assignment you are asked to review parts of each other’s code on various aspects that have been covered in OOD.

**What to do:**

1. Your tutor will pair 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.

|  |  |  |
| --- | --- | --- |
| **Student name 1** | Esther Wolfs | |
| **Student name 2** | Soleil Umwiza | |
| **Assessed code base** | Media Bazaar code | |
| **Date** | 14-Dec-21 | |
| **Does the target code apply inheritance to generalize their code where applicable?** | | No |
| If not, where do you foresee possible cases for inheritance?  The employee class can be abstract and apply inheritance. This can be changed in the next iteration. | | |
| **Does the target code apply Single responsibility to isolate individual responsibilities?** | | Yes |
| If not, what classes would you propose that split up (elaborate about this)?  They have used manager classes to isolate individual responsibilities. | | |
| **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?** | | Yes |
| If not, where do you expect change/extension to happen, and how would you propose to facilitate this?  They use interfaces, so when you want to make adjustments you can add a new class that implements the interface, without having to change the classes. | | |
| **Does the target code apply the Liskov principle to take benefit of polymorphism?** | | No |
| If not, how can the target code change to communicate in the same way with child objects as you do with parent objects?  The classes don't use inheritance. When inheritance is added in the next iteration, these classes can take benefit of polymorphism. | | |
| **When applicable, what other object-oriented design principles are applied in the target base (e.g. interface segregation, dependency inversion, etc.)?** | | |
| Interface segregation is used, with different interaces for the management classes. | | |

|  |  |
| --- | --- |
| **Is the target code readable (clear naming convention, conscious use of white spaces, proper tab use (indentation)).** | Yes |
| if not, what could improve?  Yes, the code is very clear and readable. The names make sense and the use of white space is also correct. | |
| **Below you have space for any other tips you want to share with the programmer of your target code?** | |
| Try to make simple code, not too complicated and readable. Go through the code to make sure it's maintainable and everything is working. | |