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	Kristian Lachev
Student name 2	Ivan Marinchev
Assessed code base	Project System, Department System, Website
Date	01-Jun-20

Does the target code apply inheritance to generalize their code where applicable?	No/No
If not, where do you foresee possible cases for inheritance? No inheritance is applied in the code that I reviewed. The Department and the Statistics system could both benefit from inheritance, as both of those system could have future extensions that could be implemented easier by making use of inheritance. - Kristian Lachev The code does not apply inheritance as the current website was easier to make without it. For future iterration I could see that a "User" class with subclasses "Manager" and "Employee". - Ivan Marinchev	
Does the target code apply Single responsibility to isolate individual responsibilities?	No/No
If not, what classes would you propose that split up (elaborate about this)? I propose a new class for the Statistics System, as to not have everything in the Main form, the new class will get the information from the database and pass it into another class that manages the Statistics System from there it can be passed into the main form and displayed. About the Department System, I think that some of the methods in the DepartmentManager class are never used or are not used correctly. Currently the DepartmentManager class also interacts with the database, which breaks the Single responsibility pinciple. It would be better if we move all the methods that interact with the database into the DB class and use them to populate the list of Department objects that exists in the DepartmentManager class, from there we can use that list to display information. - Kristian Lachev As the website did not require the use of OOD principles there are no classes to be split but it will be implemented in the future iterration. - Ivan Marinchev	
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?	No/No
If not, where do you expect change/extension to happen, and how would you propose to facilitate this? Right now if you want to make an extension the code must be refactored, but that can be fixed by applying an interface which can serve as the link. By applying an interface the extension of the Department system can be done easily and new additions to it could be implemented without the need to refactor the code of the existing classes. As for the Statistics System, it can also be extended with an interface that can allow the addition of different types of statistics without the need of refactoring the old ones, thus saving time and making the code more open for future additions. - Kristian Lachev I could see the Open-closed principle being applied with the "User" class if a different type of user has to be added. - Ivan Marinchev	
Does the target code apply the Liskov principle to take benefit of polymorphism?	No/No
If not, how can the target code change to communicate in the same way with child objects as you do with parent objects? As of now the Statistics and Departments systems do not apply polymorphism but if they would, it would make the extencion of the code significantly easier. At the moment the classes communicate by association, thus when you make a change in communication you must refactor some code in order for it to work. - Kristian Lachev As of now the Liskov principle is not applied in the code but a possible future use of it could be when working with obejcts of the subclasses "Manager" and "Employee "of the class "User" - Ivan Marinchev	
When applicable, what other object-oriented design principles are applied in the target base (e.g. interface segregation, dependency inversion, etc.)?	

Currently both the Department System and the Statistics system could benefit from dependency segregation, as they connect to each other directly, but in the future if we decide to extend both of the systems it would require more work than if we had applied the dependency inversion principle. As of yet I cannot speak of the interface segregation principle as the code does not make use of interfaces and breaking them down into smaller, more specific ones would not be possible. - Kristian Lachev
As of now the website does not follow object-oriented design principles due to the given task not requiring it. - Ivan Marinchev

Is the target code readable (clear naming convention, conscious use of white spaces, proper tab use (indentation)).	Most of it/Yes
if not, what could improve? The variables and methods are named correctly using the standard naming convention. The things that are not named correctly are some of the buttons and other form objects that relate to the Statistics and Departments system. That can be easily fixed by just changing the names of the form objects in the bottom right box that says Properties. - Kristian Lachev The code provided uses a clear naming convention and is easily readable. - Ivan Marinchev	
Below you have space for any other tips you want to share with the programmer of your target code?	
I would say that the code for the Department and Statistic system in his current form could be improved immensely by making use of the Single Responsibily Principle, by making use of the classes that we already have. Doing that would benefit the people that will work with your code as it would become easier to navigate through it and the overall structure would become more sound. Other things that you may want to add are - Inheritance and Interfaces, as they wil extension of the code and it will require less refactoring when we need to bring new additions to it also it will help you work towards implementing the SOLID methodology. The last tip I have would be having better naming conventions for the buttons and other objects that exist in the form as that helps with navigation when scrolling through the code itself. - Kristian Lachev Only the use of obejct-oriented design and principles for the future iterations - Ivan Marinchev	