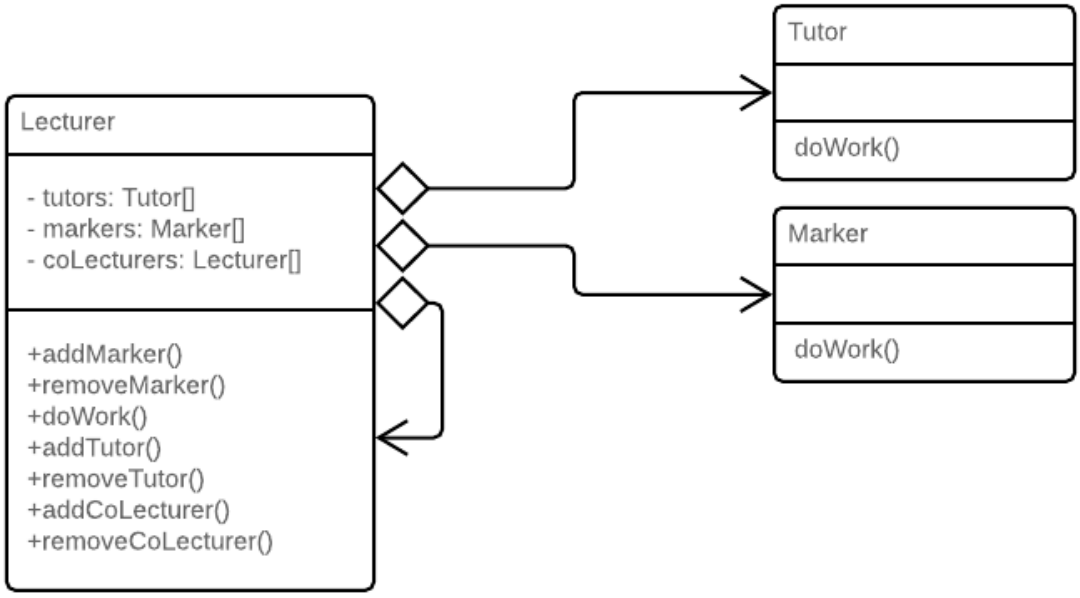
Design Pattern Exercise 2

In a university HR application, it is required to model staff management hierarchies. Lecturers can optionally manage tutors, markers or other lecturers.

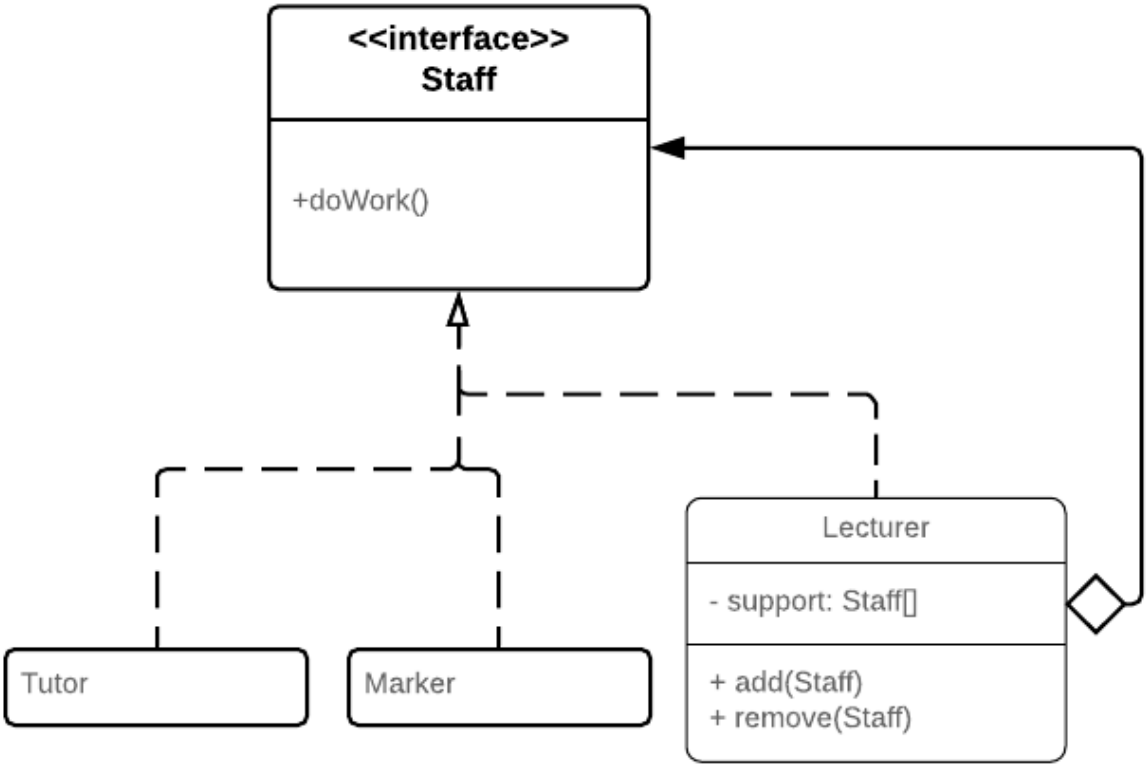


Identify and discuss issues with design with respect to SOLID principle violations or otherwise.

The Lecturer class is not closed for modification for the introduction of new staff types that it can manage. The class must be changed if there are new staff classes it must manage.

Different collections are used to store markers, coLecturers and tutors but in the context of the lecturer class, they are treated exactly the same. Methods which add and remove the different managed staff instances and calls to doWork() are redundant and can be generalized.

Suggest a refactor to this design with the application of a structural design pattern. Provide a uml diagram and a code sample of any changes made to the doWork() method in Lecturer.



As the staff hierarchy reflects a tree-like structure, the composite pattern can be applied. Tutors, Markers and Lecturers can be collected together as they all implement the staff interface. New classes which can be managed by lecturers can be added without changing the Lecturer class. The doWork() method is generalized in the Staff interface which allows the doWork() method in the Lecturer class to invoke doWork() on all of its managed staff instances.

Code Base: https://replit.com/@Snickdx/DP-Exercise-2-Solution