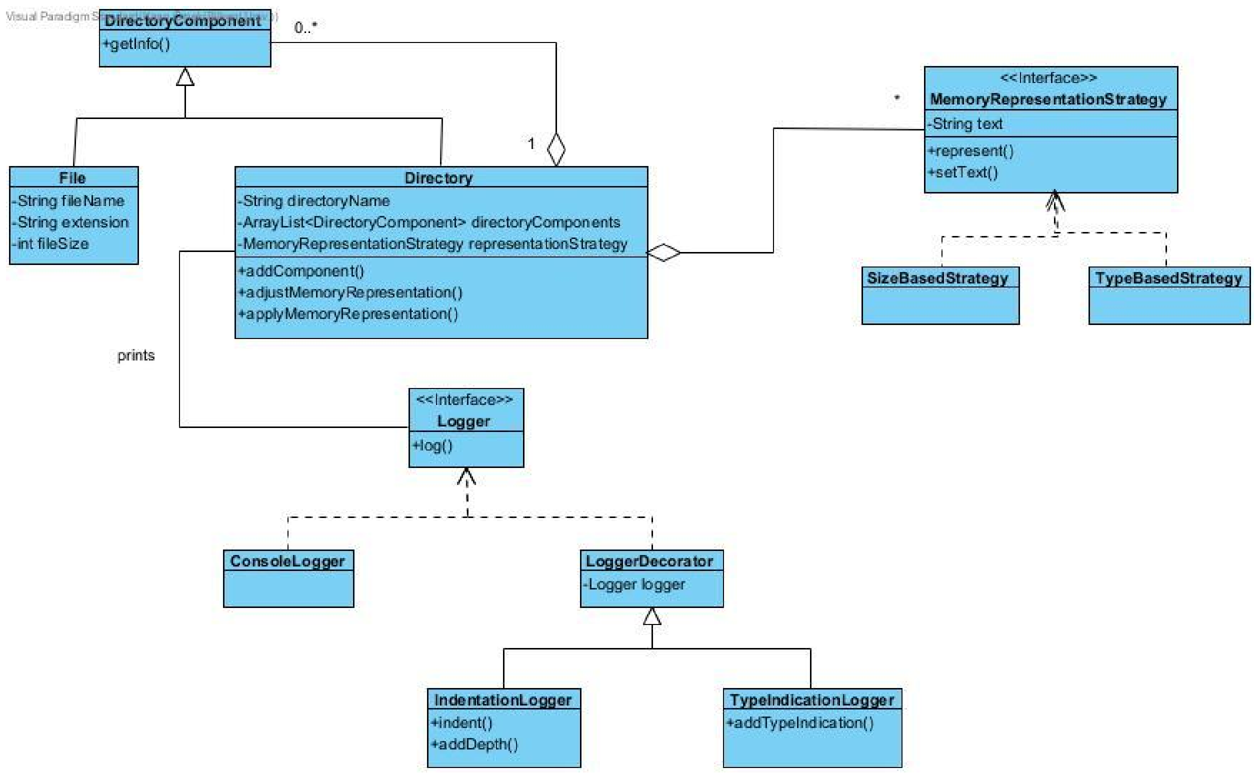
Mehmet Kaan Örnek 21901637 CS319 Design Pattern Homework

1. Class Diagram



2. Explanation of Design Patterns

In the homework, three different design patterns were used, which are Composite, Decorator and Strategy Patterns.

Composite Pattern was used on Directory, File and DirectoryComponent objects. Since the directories can include both directories and files and I had to store them in the same list, there should be a common superclass to ease the implementation. To use the Composite Pattern, I created another class named DirectoryComponent. Since the getInfo() method was common in both Directory and File objects, I moved that method to the DirectoryComponent. Directory and File extended the DirectoryComponent, so that it became their superclass. Therefore, by creating an ArrayList that contains DirectoryComponent objects, the problem was solved.

Decorator Pattern was used for indentation and type indication. The pattern consists of Logger, ConsoleLogger, LoggerDecorator, IndentationLogger and TypeIndicationLogger. Just like the example in the design pattern tutorial in the class, we are asked to implement two different log styles that can be used at the same time. Considering the company in real life, there might be more styles to add these two in the future. Therefore, I decided to implement it in the Decorator Pattern. I created the classes that were mentioned. The IndentationLogger has the indentation algorithm and TypeIndicationLogger has the type indication algorithm. There were no changes in the other classes to create the design pattern. Since they both can be considered as Logger objects, they can be nested (used at the same time).

Strategy Pattern was used to handle the desired representation of memory. The pattern consists of MemoryRepresentationStrategy, SizeBasedStrategy, TypeBasedStrategy. Since the desired representation might be different for different users, I decided to implement the Strategy Pattern. The pattern is used through separating size based representation and type based representation into two classes. Since there will be changes between the representations when needed, the strategy object stored in the Directory is the superclass of the two, MemoryRepresentationStrategy. Therefore, using the subclasses of it, the representation style becomes changeable.