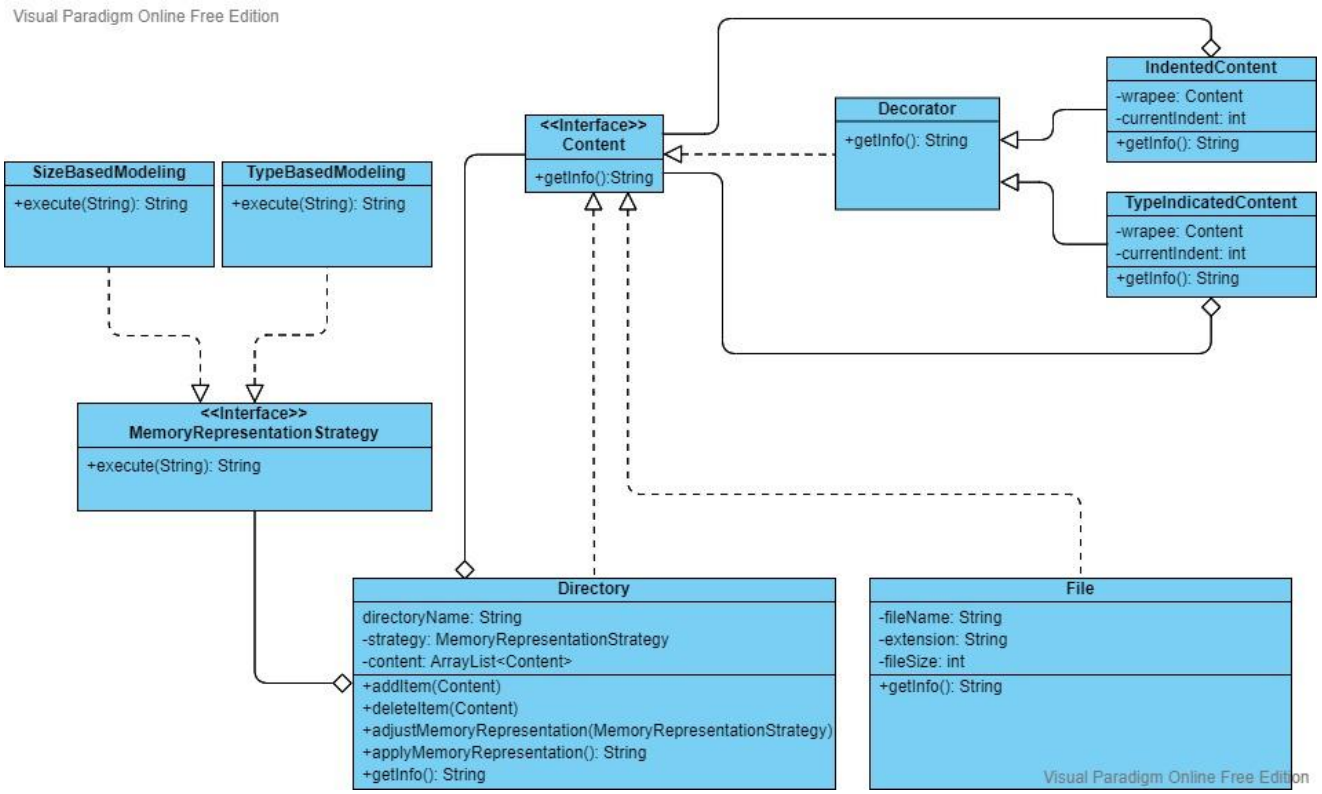
Bilkent University

Department of Computer Engineering

CS319 Design Patterns Homework

Mehmet Onur Uysal
22002609
Section 1

Class Diagram



For the first part of the problem I used the **Composite design pattern**. In order to implement this pattern, I created an interface called "Content" and made the Directory and the File classes implement this interface. Therefore, they were able to be classified as Content and had a common way to be stored. Directory class stores Content objects which can either be a File or a Directory so it includes other Directories or Files under the name of Content. I chose this design pattern because I wanted to compose the objects into a tree structure inside the Directory class.

For the second part, I used the **Decorator design pattern**. In order to implement this I created an abstract Decorator class which implements the Content interface. Then I added two decoration types as classes and inherited them from the Decorator class. These classes have a property called wrappee with type Content. These classes have their getInfo() functions which call the getInfo() function of their wrappee object, do decorations to the info and returns the decorated string value. These classes can be used indefinitely inside each other so an output of directory class' info can be decorated arbitrarily. I chose this pattern

because there was a task that can be done in different ways and different approaches could be used simultaneously and indefinitely. With this pattern I was able to attach reusable and extendible behaviours to the directory objects.

For the last part I used the **Strategy design pattern**. In order to implement this I created an interface called MemoryRepresentationStrategy which has a single function execute and made two other classes that implement this. Then I added a MemoryRepresentationStrategy property to the directory class. This property was storing the behaviour of the applyMemoryRepresentation() function. This function calls the execute function of the current MemoryRepresentationStrategy and returns the result. In order to adjust the current strategy of the directory, adjustMemoryRepresentation() function is used. It accepts a MemoryRepresentationStrategy object as argument and assigns it to the current strategy. I chose this strategy because there was a task that could be done with different strategies which should be independent of the implementation of the object and should be reusable.This pattern allowed me to define a family of algorithms which then I used for implementing the task for different objects.