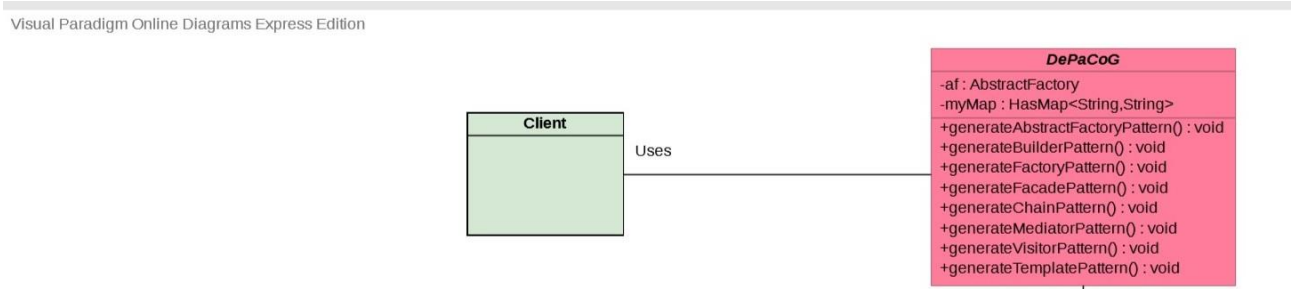
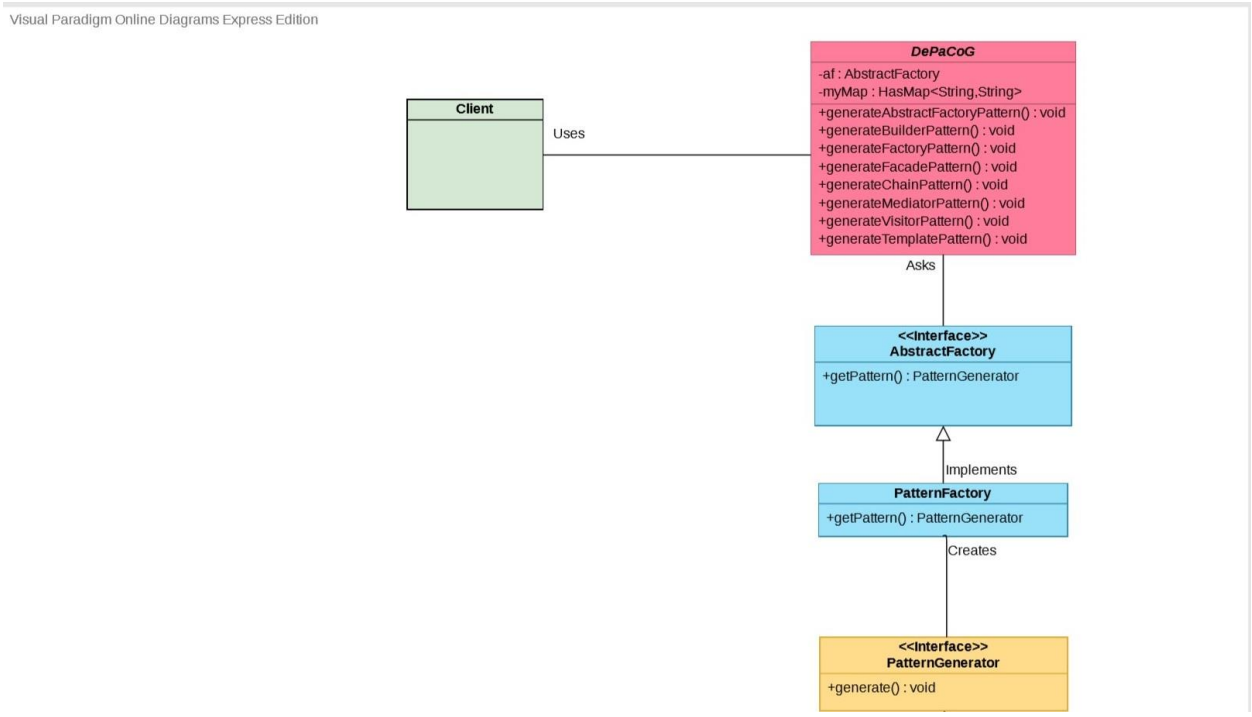
Documentation

In this homework, I created an object-oriented design and implementation of a program that generates the implemented code of the design patterns in the “The GoF Design Patterns Reference” book. My implementation model is using Abstract Factory, Builder and Facade design patterns. This model has a DePaCoG(Design Pattern Code Generator) class which acts as the facade, as it hides the complexities of the system. DePaCoG deals with the interfaces, not the implementation. Its purpose is to hide internal complexity behind a single interface that appears simple on the outside. Client can invoke DePaCoG to use its methods to create the desired design pattern code implementation.



DePaCoG has a reference to AbstractFactory interface. It asks this interface to get the desired design pattern implementation. PatternFactory implements the AbstractFactory interface to create the generator which will be responsible for generating the code. The purpose of the AbstractFactory interface is to achieve total abstraction.



Model has PatternBuilder class which implements the PatternGenerator interface. My initial plan was to have design patterns in different classes, but I decided to put them all in the builder class because I wanted to build different immutable objects using same object building process. Builder pattern aims to “Separate the construction of a complex object from its representation so that the same construction process can create different representations.”. PatternBuilder uses an additional class pBuilder which helps building desired User object with all mandatory attributes and combination of optional attributes, without losing the immutability.