Adapter

The adapter pattern is a design pattern that enables the collaboration of instances from classes that have incompatible interfaces. Its goal is to convert the interface of an existing class into another interface expected by the client, allowing them to work together. Three examples from the course have been provided to implement this pattern.I have a version of the first example (Document Adaptation), the second example (Class Adaptation for a Management class that uses stacks and then use linked lists) with two versions, and finally, a third example (adapting an interface for squares to handle rectangles).

The participants in this pattern are:

The interface: Introduces the signature of the methods of the object.

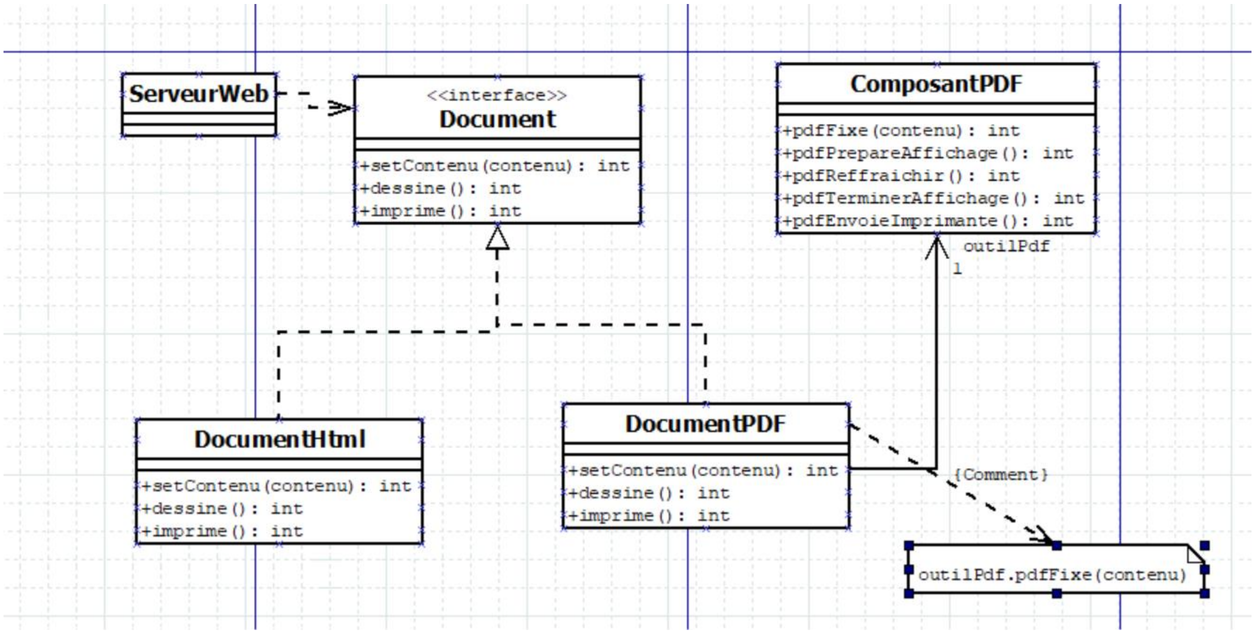
The client: Program or class that interacts with objects that respond to the interface.

The adapter: Implements the methods of the adapter by invoking the methods of the adapted.

The adapted: Object whose interface must be adapted to match the interface.

1. This concerns the implementation of adapting an interface that deals with HTML documents, but now having PDF components. Through this interface, we would like to process PDF documents as well. It is found in the “**document”** folder of the Adapter pattern in my repository.

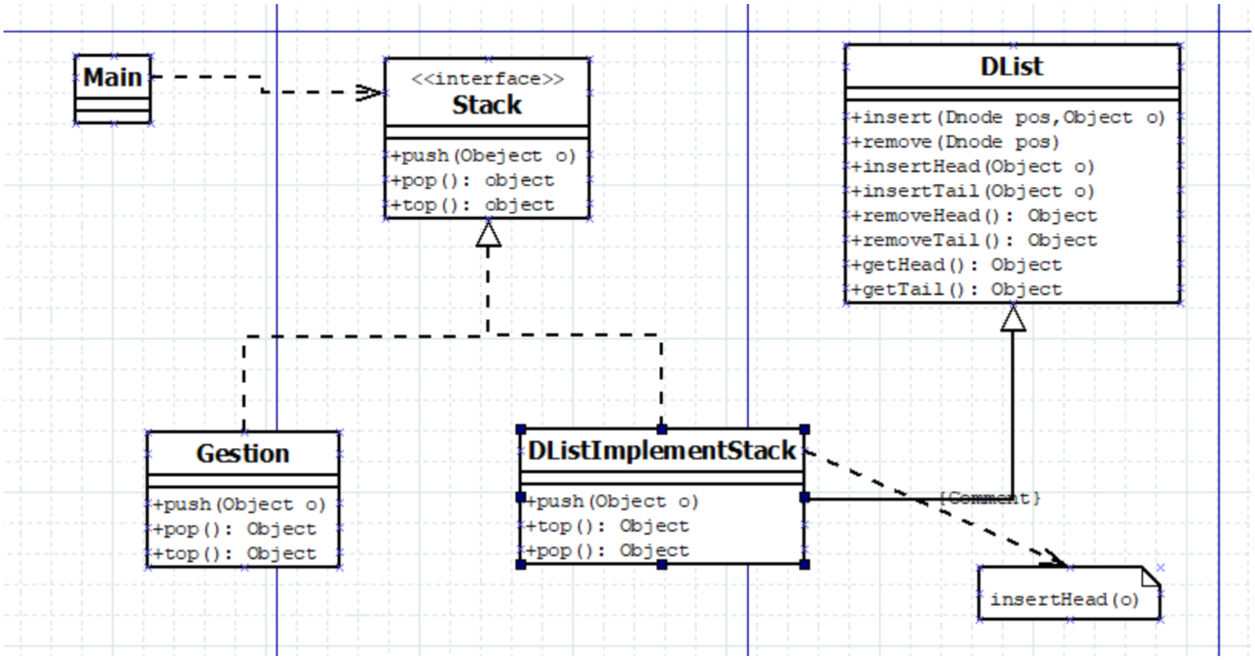
Model



2. In this case, we have a class DList that defines a doubly linked list, and another class (Gestion) that knows how to manipulate stacks. The goal is to use the DList class in

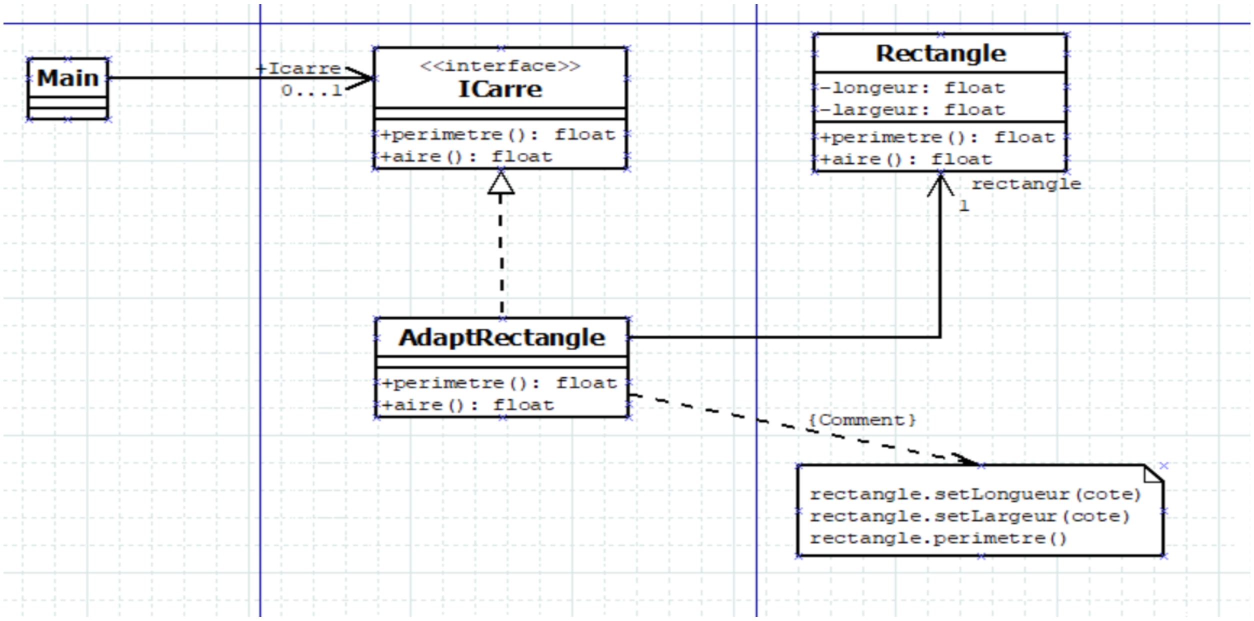
Gestion without modifying DList. Here, we choose to use inheritance and an interface on the Adapter class as our approach. The implementation of this class is in the "**Stack/association**" directory of the adapter pattern in my github repository.

Model



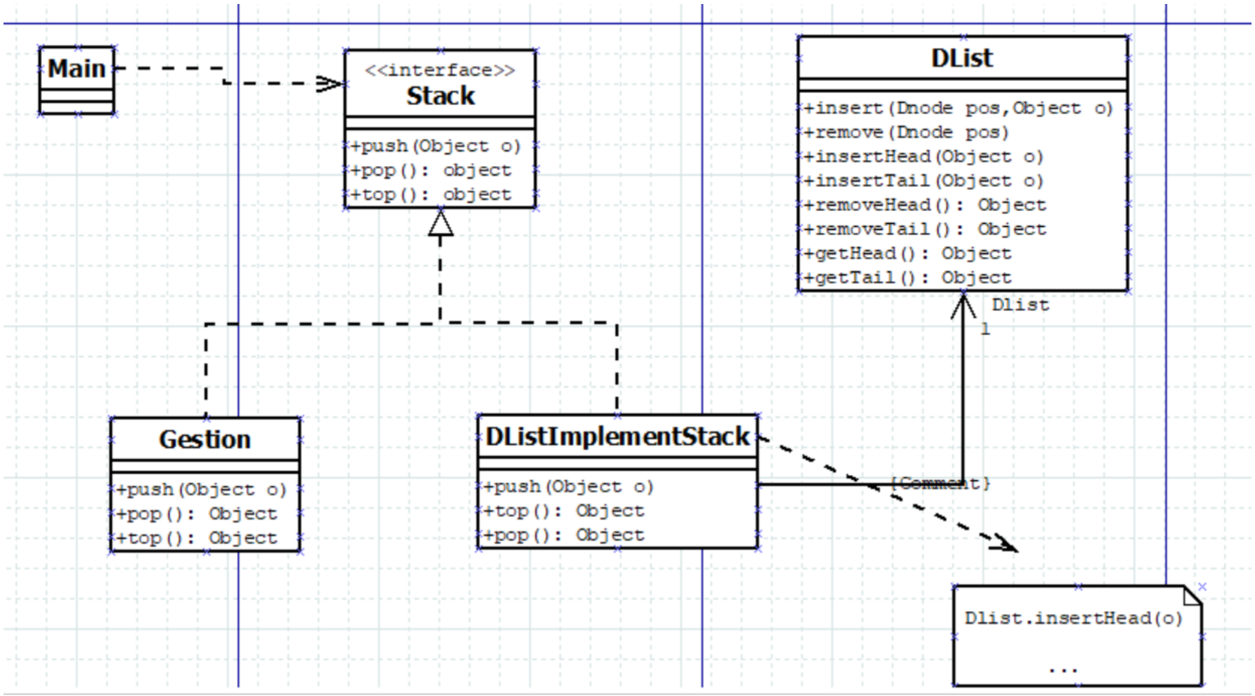
3. This involves adapting a program that uses a class capable of managing squares so that it can also handle rectangles. It is implemented in the "**square**" folder of the adapter pattern.

Model



4. Here, the goal was to implement a second version for Adapter 2(The Inheritance part). Instead of using inheritance, we opted for a relationship between the adapter class and the adapted class. It is implemented in the "**Stack/inheritance**" directory of the adapter pattern in my github repository.

Model



Main
<<interface>>
Stack
+push(Object o)
+pop(): object
+top(): object
DList
+insert(Dnode pos,Object o)
+remove(Dnode pos)
+insertHead(Object o)
+insertTail(Object o)
+removeHead(): Object
+removeTail(): Object
+getHead(): Object
+getTail(): Object
Dlist
1
Gestion
+push(Object o)
+pop(): Object
+top(): Object
DListImplementStack
+push(Object o)
+top(): Object
+pop(): Object
{Comment}
Dlist.insertHead(o)
...