INFO-0027: Programming Techniques

Project 2: File Explorer - Report

Goffart Maxime
180521

Joris Olivier
182113

Academic year 2020 - 2021

1 Software architecture

1.1 Creationnal patterns

We chose to use the factory method pattern to create the classes representing the different entities that the file manager manipulates (files, folders, aliases, and archives). We chose to use this pattern because all the entities are sharing the same properties (name, children, parent, etc). In this way, we can easily add new types of entities to the project and the task of choosing which entity to build is perfomed dynamically by the creator.

The advantages of using this pattern are that we can easily modify our code because the object creation is modular, we do not have code repetitions while instancing an object, and new types of concrete products can easily be added to the code.

The drawback is that the class hierarchy is a bit complex for his usage in our case.

We also use the singleton pattern for the concrete creator classes in order to be sure that they can not be instanciated more than one time and avoiding global variables.

The advantage of this pattern is that it ensures that there is only a single concrete creator for each corresponding entity.

The drawback of this pattern is that it can lead to break the single responsibility principle.

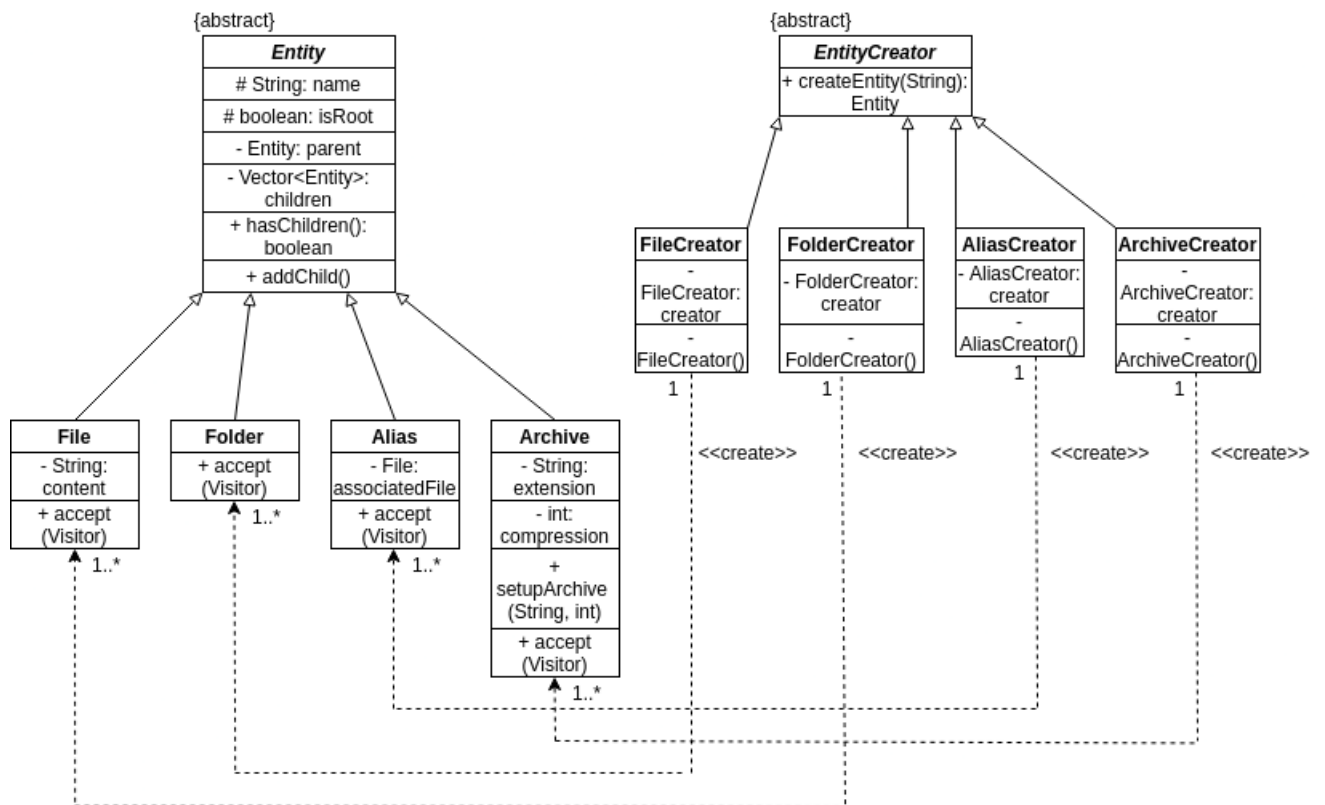The UML static diagram representing these patterns can be observed on the Figure 1

Figure 1: UML static diagram of the class related to creationnal patterns.

1.2 Structural patterns

We chose to use the adapter pattern because we do not have access to the source code relative to the `TextAreaManager` and `ExplorerSwingView` classes but we wanted to add some functionnalities relative to these classes : we wanted to manipulate the tree that represents the view of the file manager and texts.

The advantages of using this pattern is that it handles to reuse and adapt to our needs the interface implemented in the `graphics.jar` file without needing to have access to it and modify it.

The drawbacks are that the code complexity is increased. In fact, we can need many adaptation level to reach the desired type.

The UML static diagram representing this pattern can be observed on the Figure 2
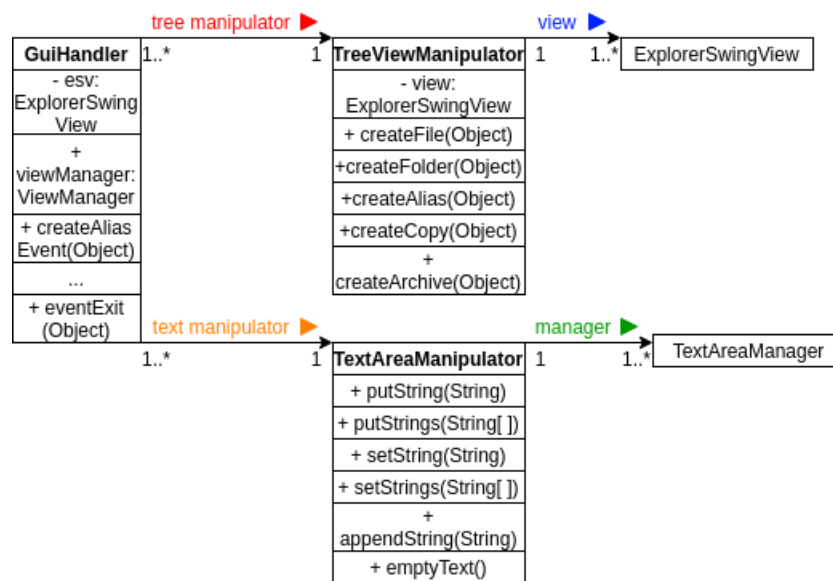

Figure 2: UML static diagram of the class related to structural patterns.

1.3 Behavioral patterns

We decided to use visitor pattern for the copy, the display and the building of entities (file, folder, alias, and archive) because, with this pattern, these new operations can easily be added and without modifying the code relative to these entities.

The advantage is that this pattern allows to pretty easily add such new operations.

The drawback is that the structure of composite objects is complex to modify and can lead to encapsulation problems.

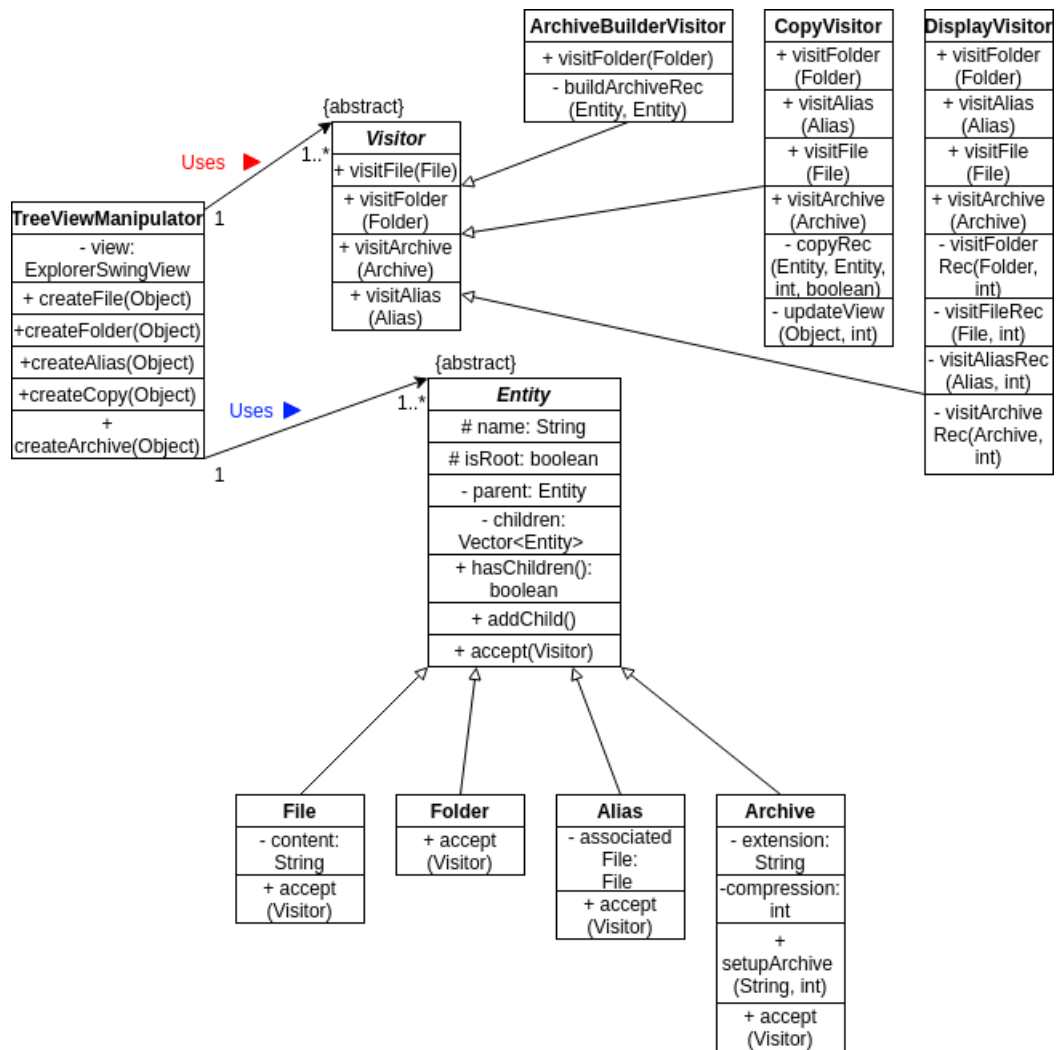The UML static diagram representing this pattern can be observed on the Figure 3

Figure 3: UML static diagram of the class related to behavioral patterns.

2   Feedback

Olivier spent about 12 hours on the project while Maxime, on his side, spent about ... hours on it.