# Design Document
## for the ML Generator

University of Zurich
Software Construction
FS2017

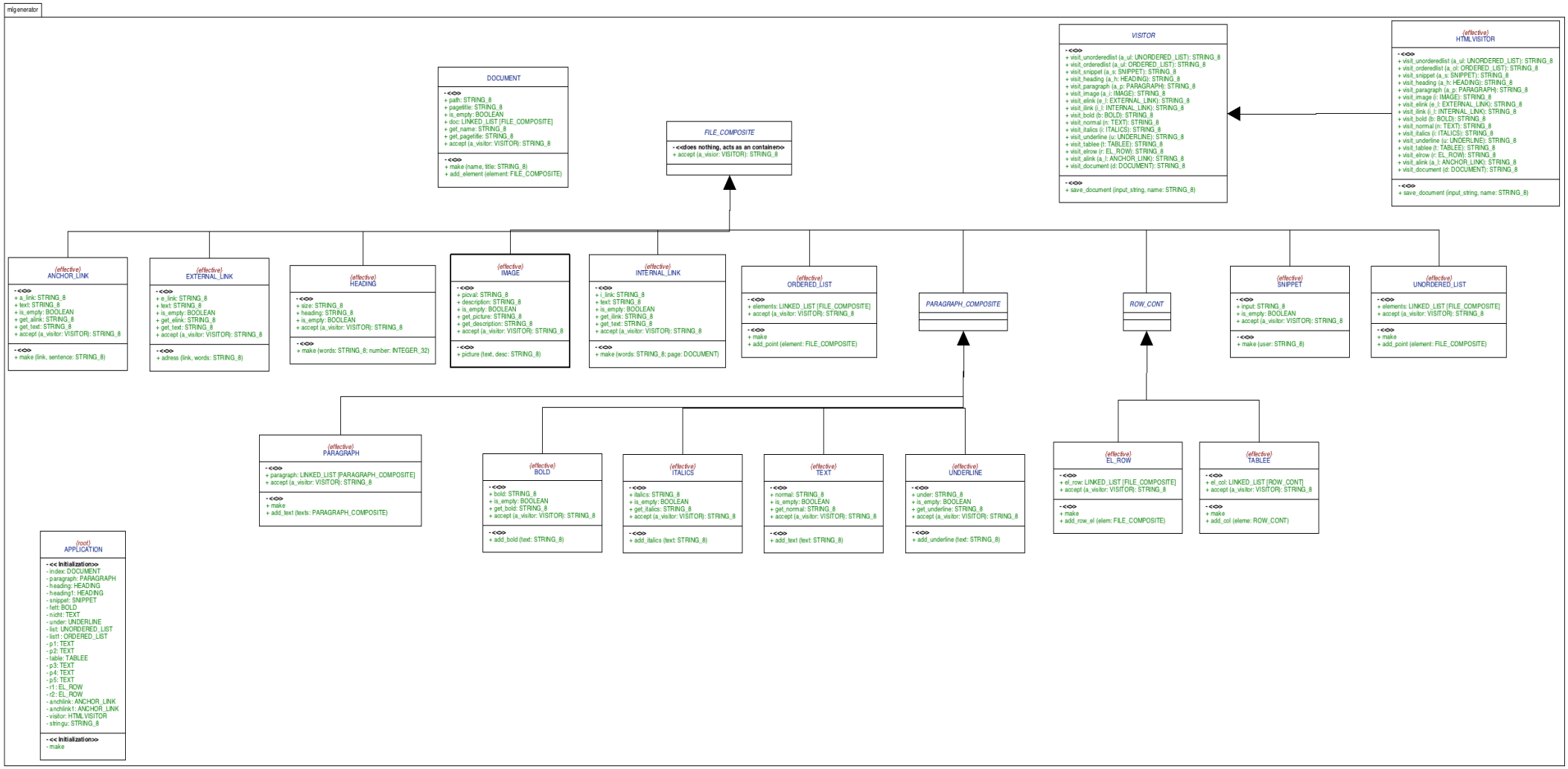
**Group 12**

Supervision: Raphael Matile

Authors:


| | |
|---|---|
| Nimra Ahmed | 16 723 934 |
| Yulia Brun | 16 700 536 |
| Aleksandra Markovic | 16 701 583 |
| Andjela Markovic | 16 701 575 |


Zurich, 12/3/17




**Class Diagram.** Source: EiffelStudio.

In the ML Generator the following design patterns are used:

**Visitor pattern**

The Visitor design pattern represents the operations to be performed on the markup language elements/classes of our objects. It behaves like a pointer and calls/visits the elements of the markup language document (the classes: UnorderedList, OrderedList, Snippet, Heading, Paragraph, Image, ExternalLink, InternalLink, Bold, Normal, Italics, Underline, Tablee, Elrow, AnchorLink, Document) and gets their output. The main purpose of our Visitor pattern is to coordinate all operations of the ML Generator and provide output after all operations have been executed. The advantage of the pattern is that Visitor operations can be extended without changing the elements they perform on, which makes our ML Generator adjustable for other markup languages.

**Composite pattern**

The Composite design pattern enables a tree structure of the library. We use it three times, once for the overall structure, once for Paragraph Structure, and once for the Table structure.

File Composite represents the abstraction of the components and behaves like a container for our elements/classes. Leaf objects of the pattern (File Composite) are Snippet, Heading, External Link, Internal Link, Image and Anchor Link as well as the Paragraph Composite. They implement the methods of the File Composite directly and are treated the same way as single elements of the same type of object. On the other side, List (both unordered or ordered), and RowCont, are also children of the File Composite but have the ability to have the whole File Composite as Child.

Paragraph Composite is a child element of File Composite (composite of the pattern) and has as leaf elements Bold, Italic, Underline and Normal. Child (Composite) of the Paragraph Composite is the Paragraph class which consists of the Paragraph Composite leaf elements. This allows the user to not only add text semantics to the whole paragraph but also to his desired words inside the paragraph.

A further composite pattern is the RowCont, which is here for the Table structure. RowCont has as leaf element ElRow and as composite Tablee that works like a folder. Composite objects implement methods to manipulate their children, elements of the markup language document, and implement methods of the File Composite by delegating them to the elements of the ML document. Composite elements of the pattern can or must contain (in some cases) leaf elements of the pattern.

The composite pattern allows clients to treat individual objects and compositions uniformly. We use this particular pattern to enable ML elements to contain other elements they can consist of and to enable the visitor pattern to operate on every element of the ML Generator.

# Changes made during the project

The Abstract Factory pattern is not used anymore, since Visitor has the ability to call/visit the elements/classes of the composite pattern it needs and provide the desired compound document by itself.

In the Application class is specified which document format (HTML in our case) the Visitor pattern is going to use (HTML Visitor) as an implementation issue for facilitated and more flexible use/implementation, since this was previously specified in the Document class. The downside is, that the user may try to use a Visitor we have not yet implemented.

The table class''Tablee'' is now a composite object of the composite pattern (File Composite) and not an ''alone standing'' class by itself, since it functions like a folder and consists of other table elements. It is possible to link on elements of the table but not the table itself and the table class can not contain a heading as an element anymore, but which can be created by the user by making the first row and bolding the input. This can be considered a downside.

There is a new Paragraph Composite considered as an implementation solution, consisting of bold, italic, underlined and normal text elements.

The Visitor pattern has the additional function ability to save the document at the end of the execution of the library and even choose the name under that the user wants to save it, which was newly implemented to this pattern.

The Anchor link class is implemented in a way that it can link on all document elements separate, besides the Text, Bold, Italics, Underline, Lists, Tables and Paragraph class which couldn't be implemented since there are no suitable''normal'' text tags and would not make sense for the other classes. A downside of here is that the ID tag always consists of the input to be linked at and therefore may get lengthy. It also contains whitespaces, which does not hinder the functioning wise but is actually not desired.

There is no Page Title class existing anymore, since Page Title has become a part of the Document class, implemented in a way that it is possible to have at exactly one Page Title.However, the tradeoff here is that we may require some element which may not be needed in others.