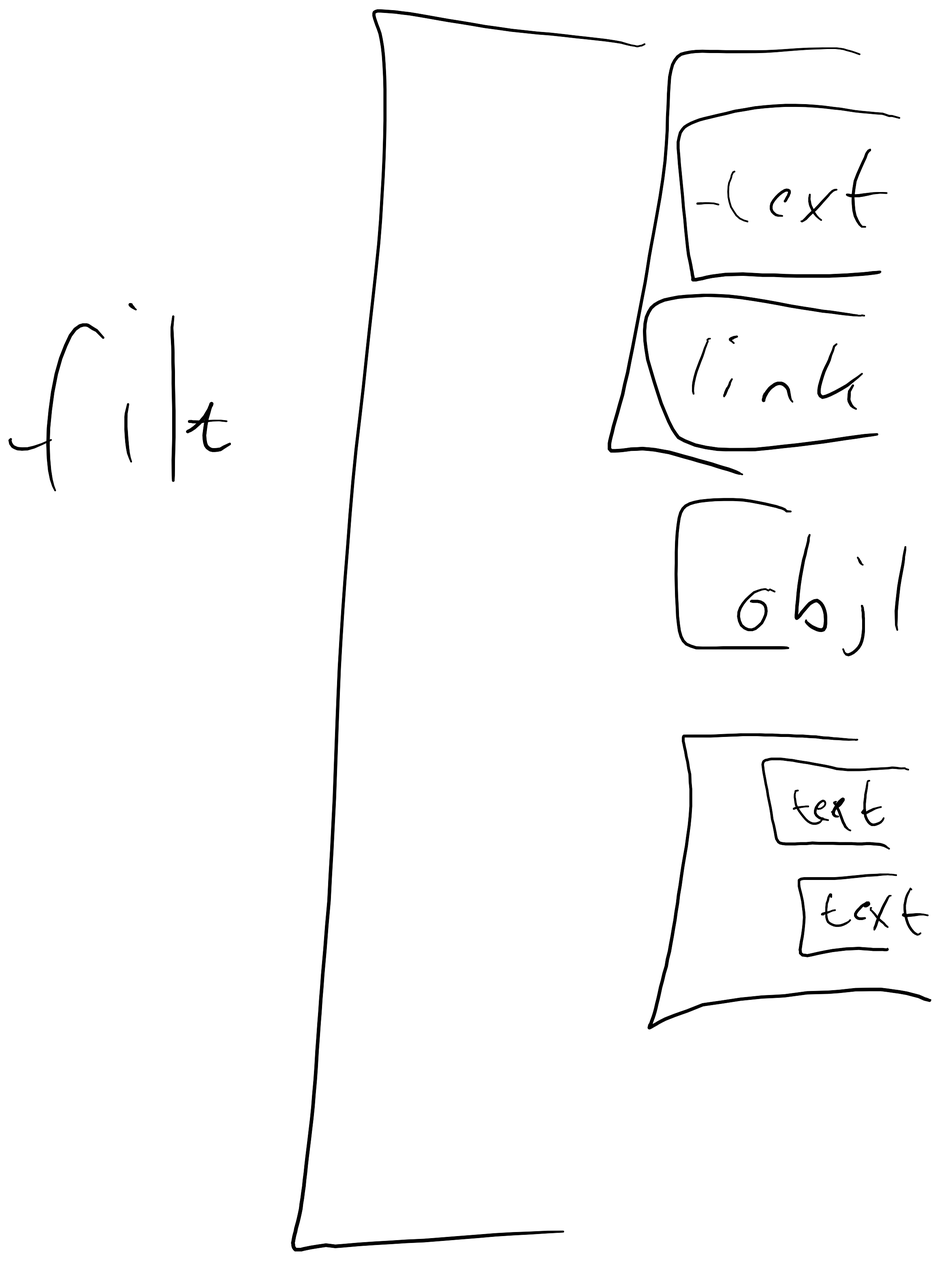
generate (filename, stringorfile, filepath)



class
TEXT
feature
content: STRING
ID: INTEGER
write (what: content)
do
content := what
end
make (initial: INTEGER)
do
content := initial
end
end -- TEXT

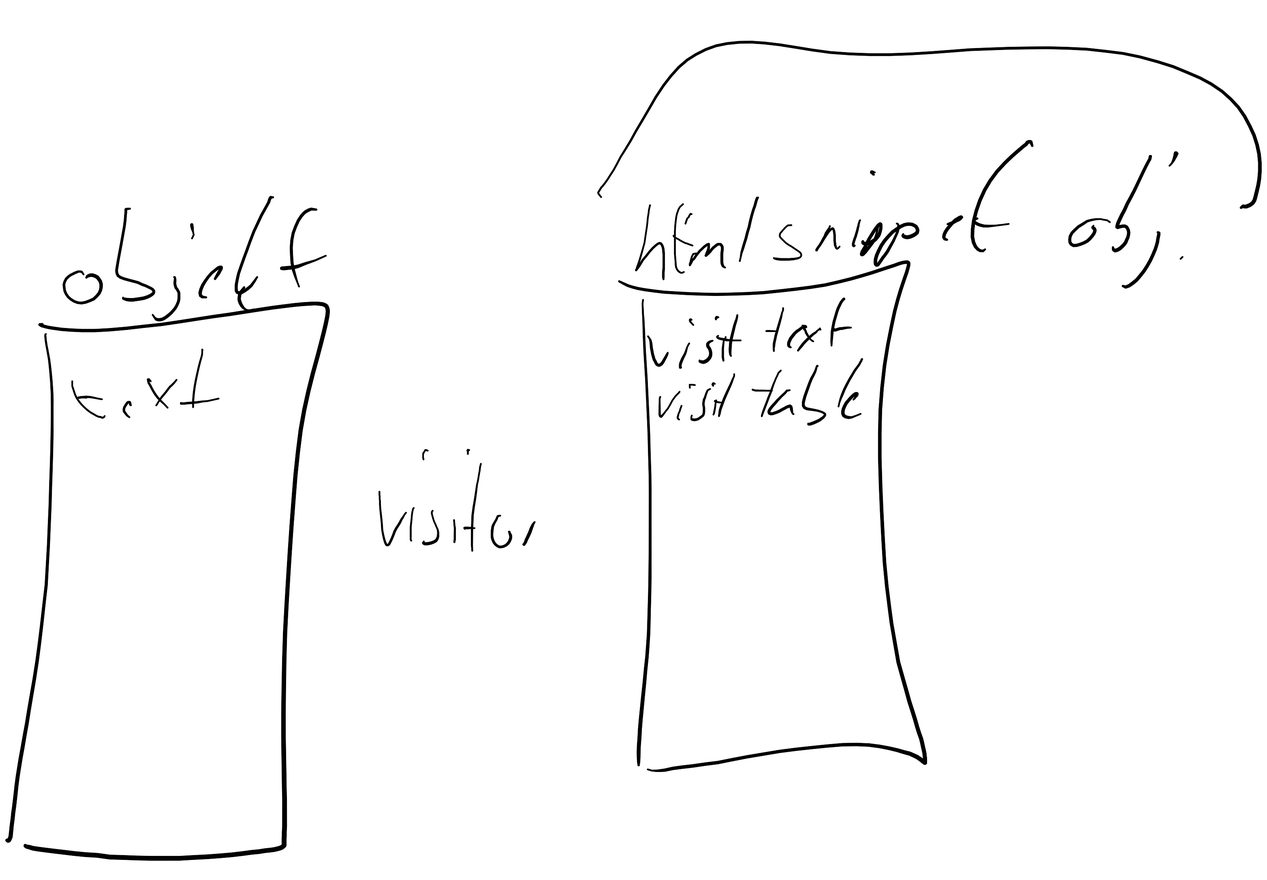
create text1.make (text here)
text1

content: "text here"

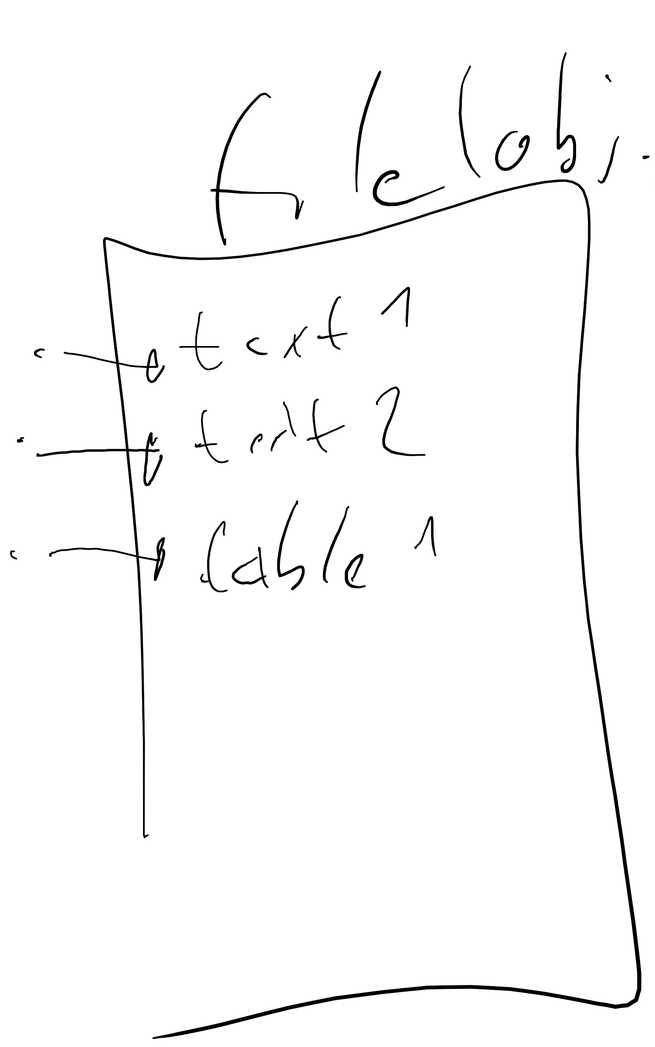
visitor

visit ...
visit txt
visit txt

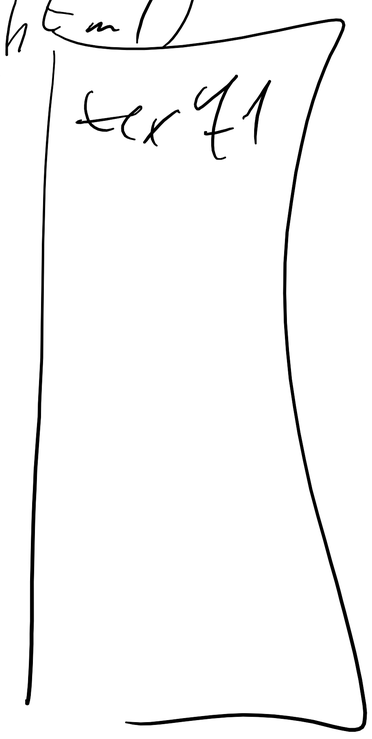
visit.dat
file [1



1.

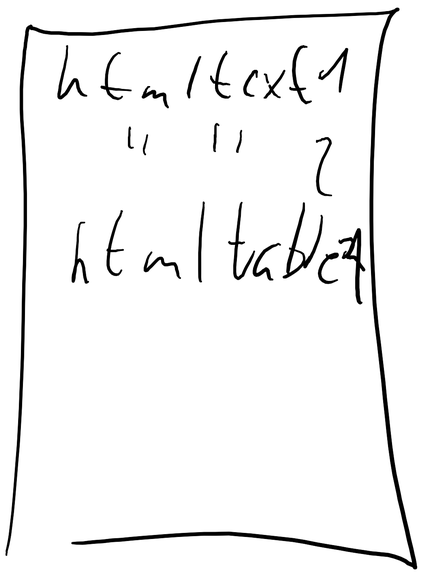


2.



3.

4.



3. htmlVisit text (text 1)
returns html text 1

4. htmlfile

5. generate (htmlfile)
fügt snippets zu endprodukt

Nicht vergessen: Tables enthalten objekte (text)

Flow:

1. Create instance of File (object)
2. Create instances of sub-level objects (text, table etc)
   these will be stored by pointers in the array of File
3. Call the Visitor with input to which markup-language
   it will be converted
4. Visitor creates a new *markuplanguage* file
5. Visitor goes through file array and visits each object
   with a visitor function/class/object that creates a new
   converted object of the markup language type. these objects
   are stored in the *markupLanguage* file again as pointers to
   the object in an array
6. Generate generates the output by going through the *markuplanguage*
   file.

Code:

File: Object that contains an Array, this
array is where the pointer to the lowest level
objects (text, table etc.) will be stored

Visitor: Function that defines what language it will be converted
to, maybe able to have Generate do this, needs
input what has to be converted aswell as the desired language

Generator: Final step that delivers the endproduct. Inputs will be
if wanted as a file: - file name
- destination
- what has to go there

if wanted as string: - what is content of the string

markup language file: Same as the file, only that the pointers
are pointing to "converted" objects.

sub-level object: text table etc. objects, contain content, unique ID,
type ID

IDEAS:

file classes have a function to edit order by showing the order of the titles of the object in the array.

ex.: [text 3, table 2, table 1, text1, text2]

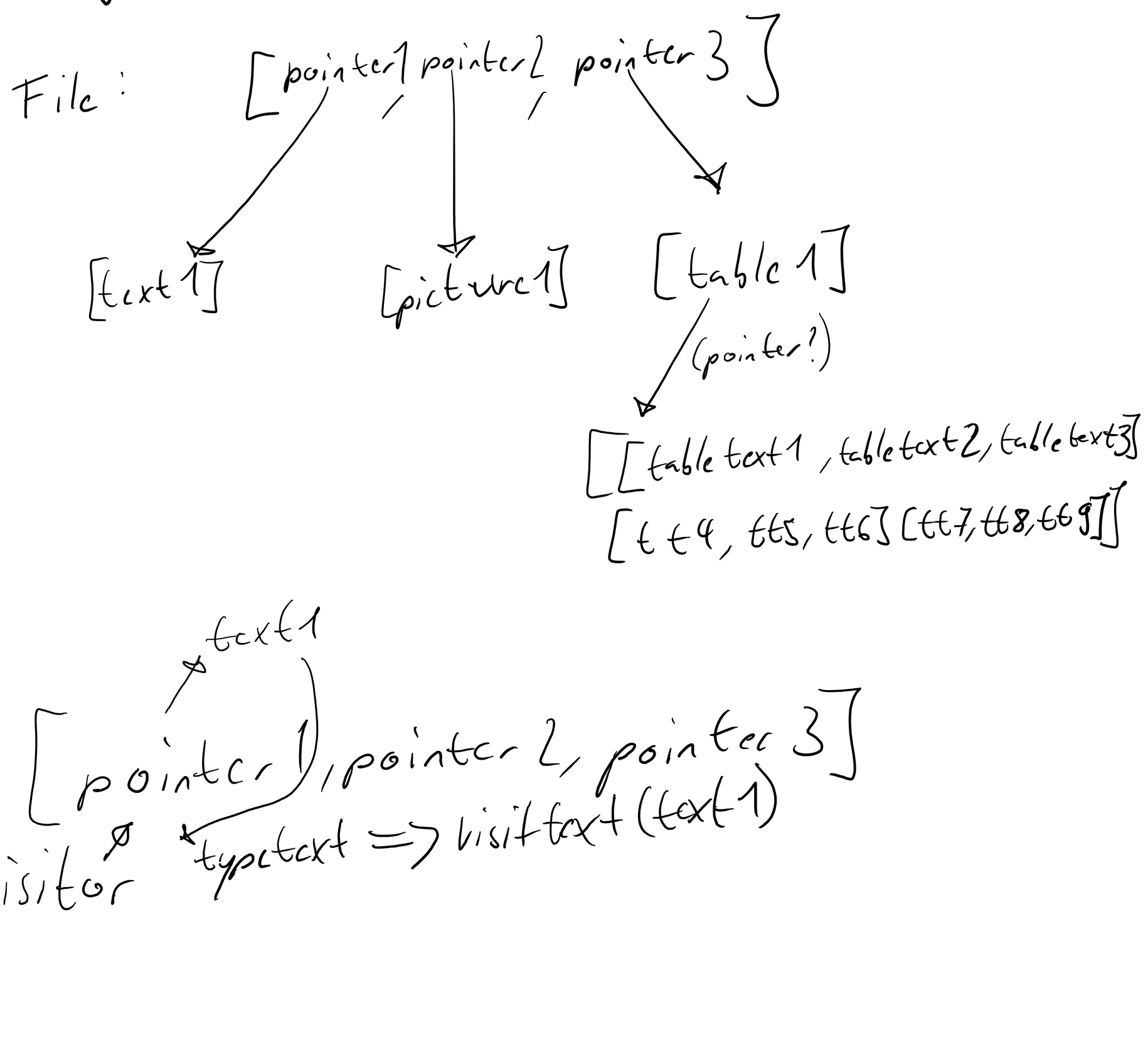
So you can change them around.

Tables have to be implemented by using lower level objects, e.g. text. Won't be hard as we use arrays.

How do we safe the objects?

↳ maybe have a function file.append(object) this way it would be easier to implement the tables that way

Picturing:


File:
[pointer1 pointer2 pointer3]
[text1]
[picture1]
[table1]
(pointer?)
[[[table text1, table text2, table text3]
[tt4, tt5, tt6] [tt7, tt8, tt9]]
text1
[pointer1, pointer2, pointer3]
visitor
typetext => visit text (text1)