Visitors and friends

Alexandre Bergel
abergel@dcc.uchile.cl
28/10/2010

Objective

The objective is this lecture is double

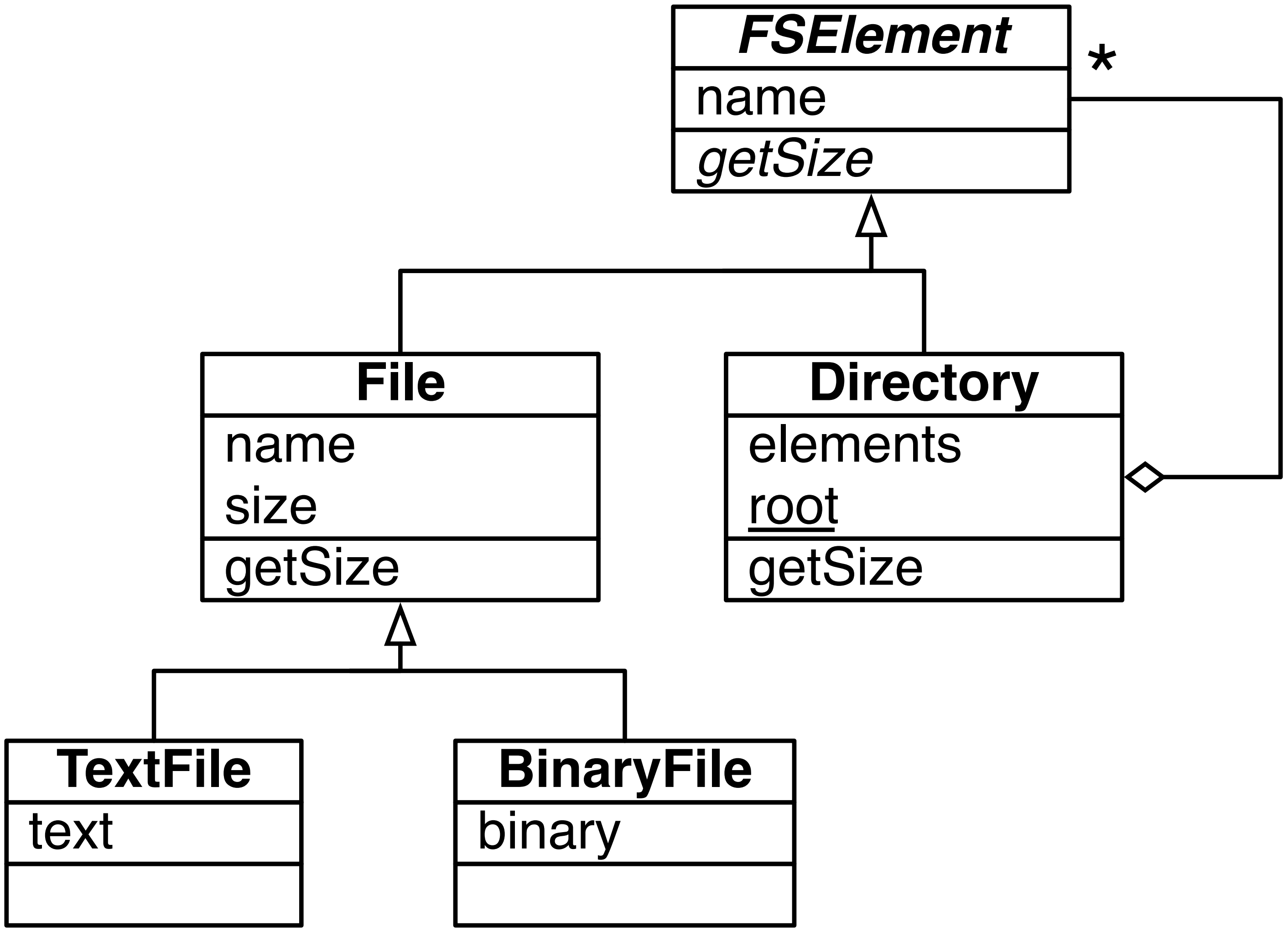
- introduce the visitor pattern, which is comprise many situation we have seen so far
- face the problem addressed by the visitor pattern

Exercise

A "file system" es un componente esencial de mucho sistemas operativos. Por este ejercicio, vamos a considerar los elementos siguientes:

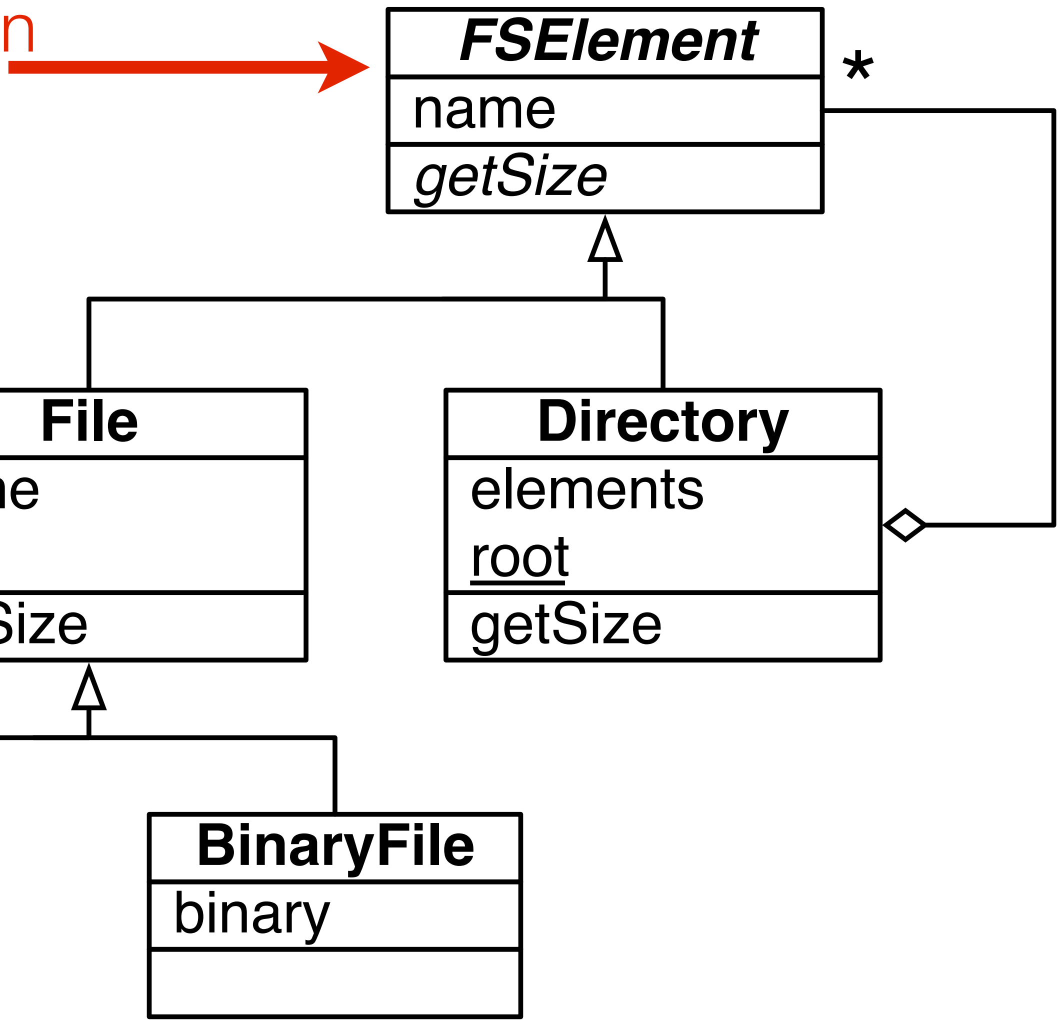
- un file system tiene files y directories
- un file puede ser un textual file o un binary file
- un file system tiene solamente un directory root
- un directory puede contenir textual files, binary files y directories
- cada elemento de un filesystem tiene un tamaño y un nombre

A solution

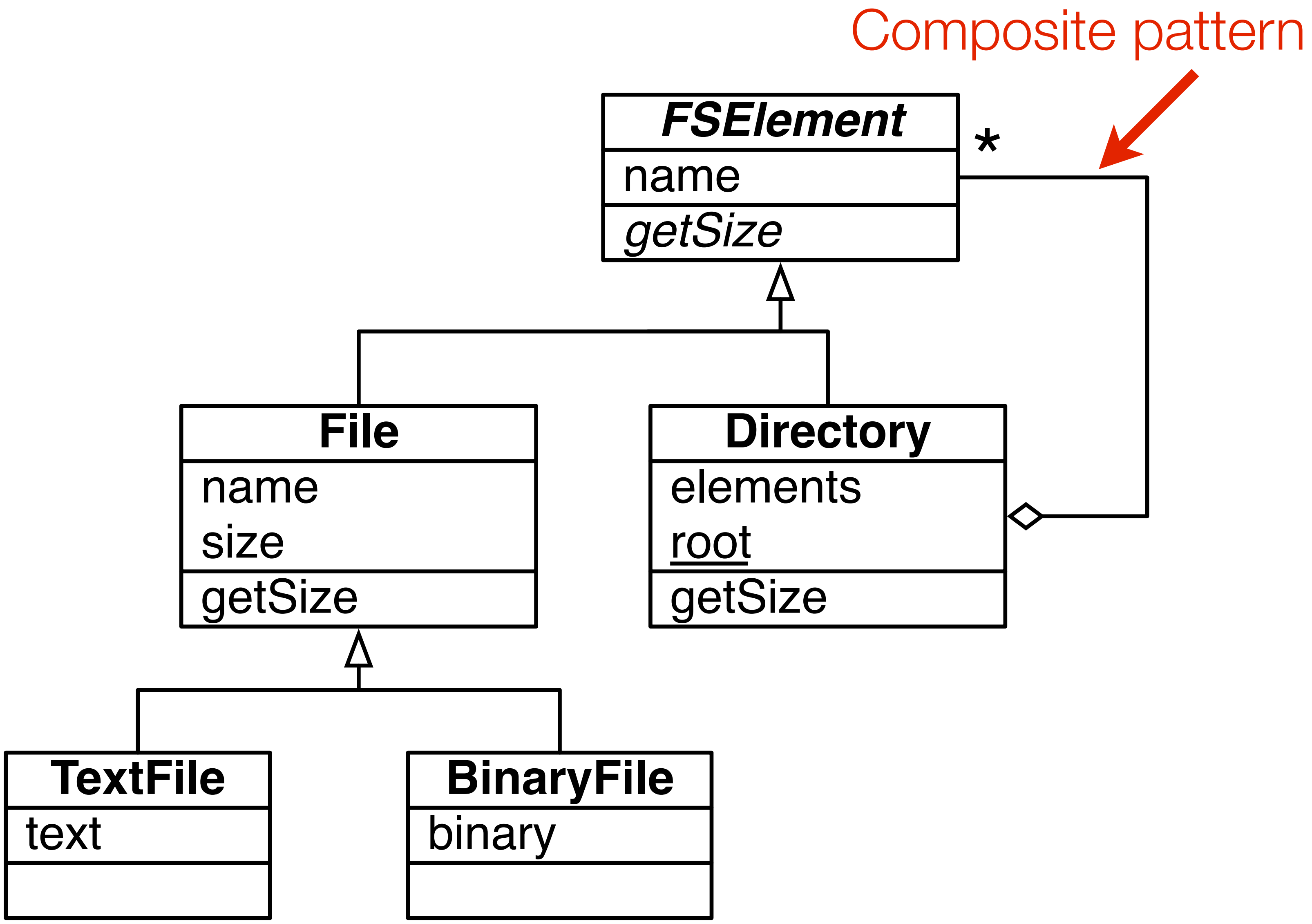

FSElement
name
getSize
*
File
name
size
getSize
Directory
elements
root
getSize
TextFile
text
BinaryFile
binary

A solution

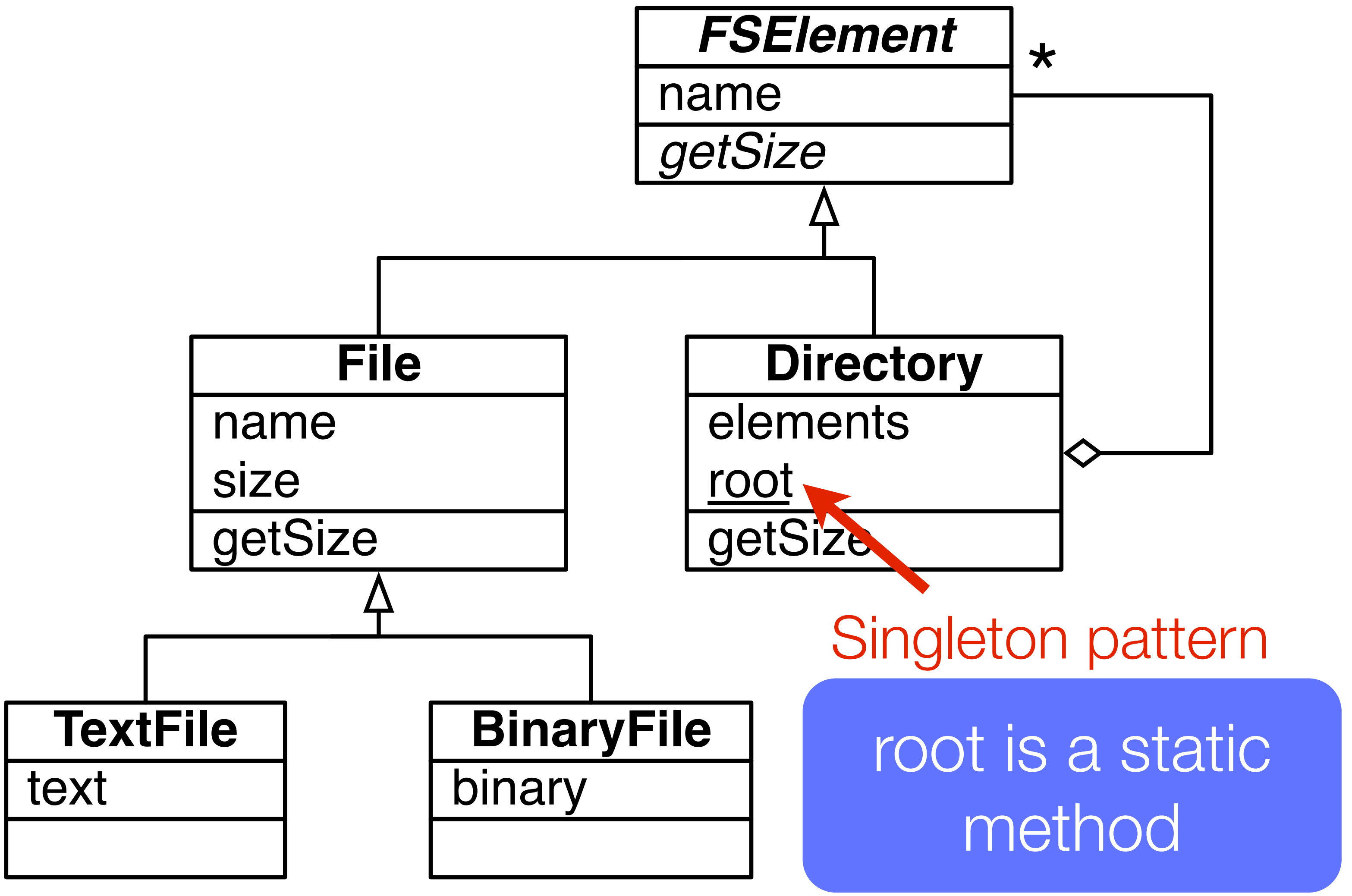
Abstract class (since its name is in *italic*)



A solution

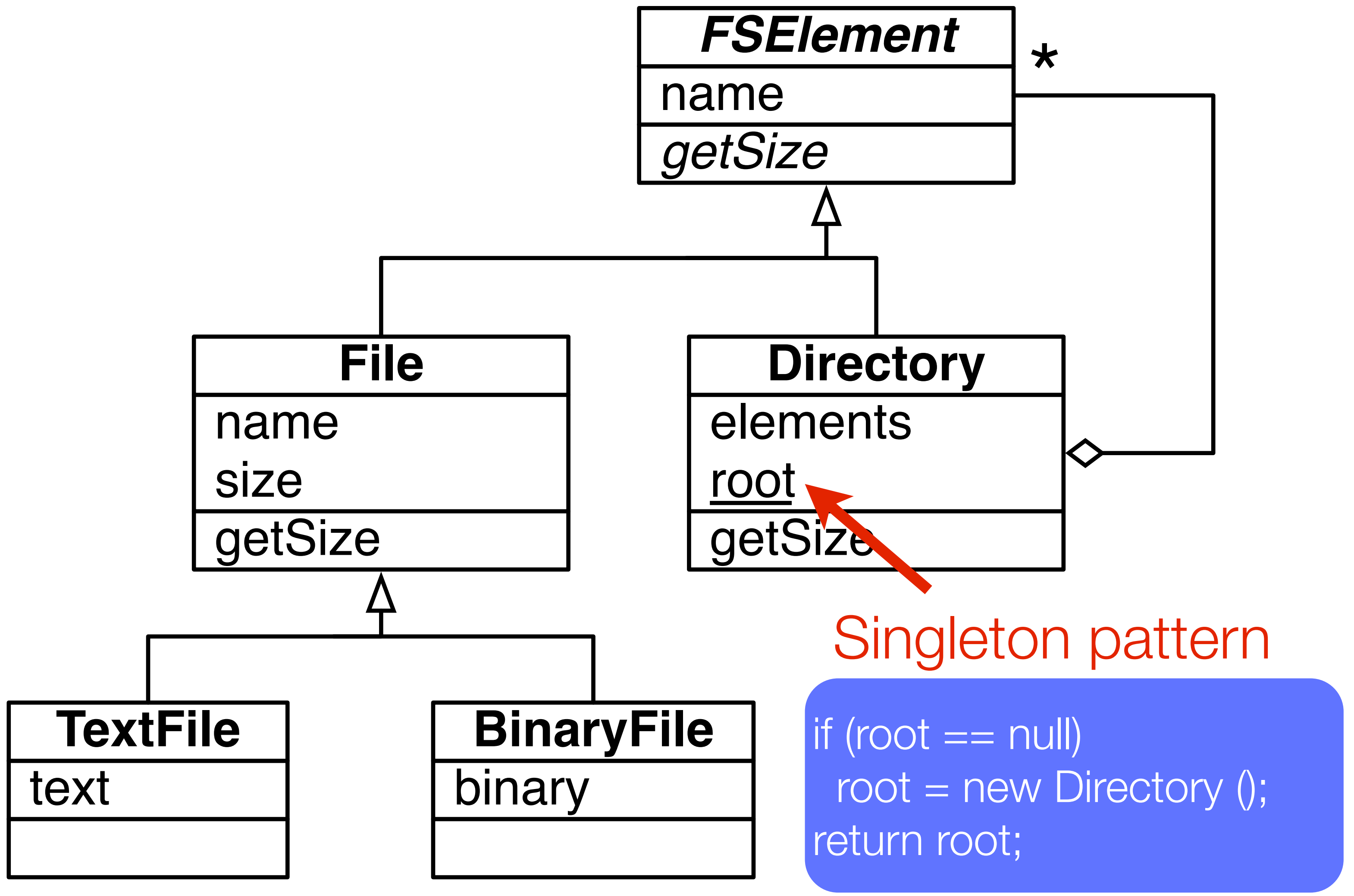

Composite pattern
FSElement
name
getSize
*
File
name
size
getSize
Directory
elements
root
getSize
TextFile
text
BinaryFile
binary

A solution



root is a static method

A solution



```
if (root == null)
  root = new Directory ();
return root;
```

Size method

A version version of getSize() can be

```
class File extends FSElement {
    private int size;
    public int getSize () { return size; }
    ...
}
```

Size method

For the class Directory:

```
class Directory extends FSElement {
  private List<FSElement> elements = new List<FSElement>();
   public int getSize () {
       int size = 0;
       for (Element el : elements) size += el.getSize();
       return size;
   }
   ... }
```

Exercise...

Now, we would like to add some operations

- get the size of a folder
- get the total number of files contained in a directory
- delete a particular element, which may be deeply nested
- ...

Important questions

How would I write the invocation of such operation?

Do I need a class hierarchy for the different recursive operations?

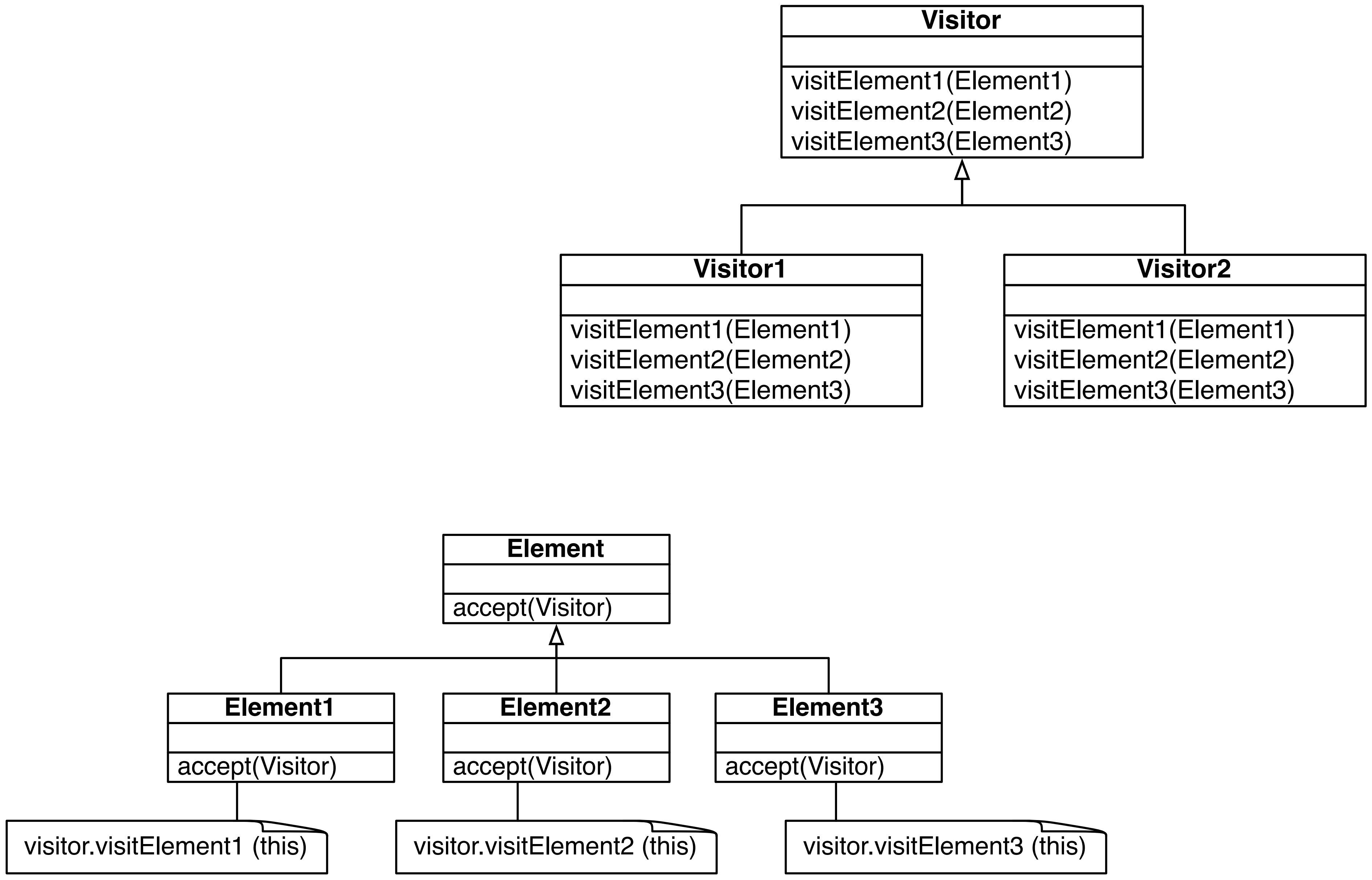
What is the cost of adding a new operation?

Is there any code duplication?

Adding operations

Implementing these operations has a high cost

- the domain (file and directory) has to be modified at each operation
- can be cumbersome if the domain is externally provided
- each of these operations contains duplication, notably the recursion over the structure


Visitor
visitElement1(Element1)
visitElement2(Element2)
visitElement3(Element3)
Visitor1
visitElement1(Element1)
visitElement2(Element2)
visitElement3(Element3)
Visitor2
visitElement1(Element1)
visitElement2(Element2)
visitElement3(Element3)
Element
accept(Visitor)
Element1
accept(Visitor)
Element2
accept(Visitor)
Element3
accept(Visitor)
visitor.visitElement1 (this)
visitor.visitElement2 (this)
visitor.visitElement3 (this)

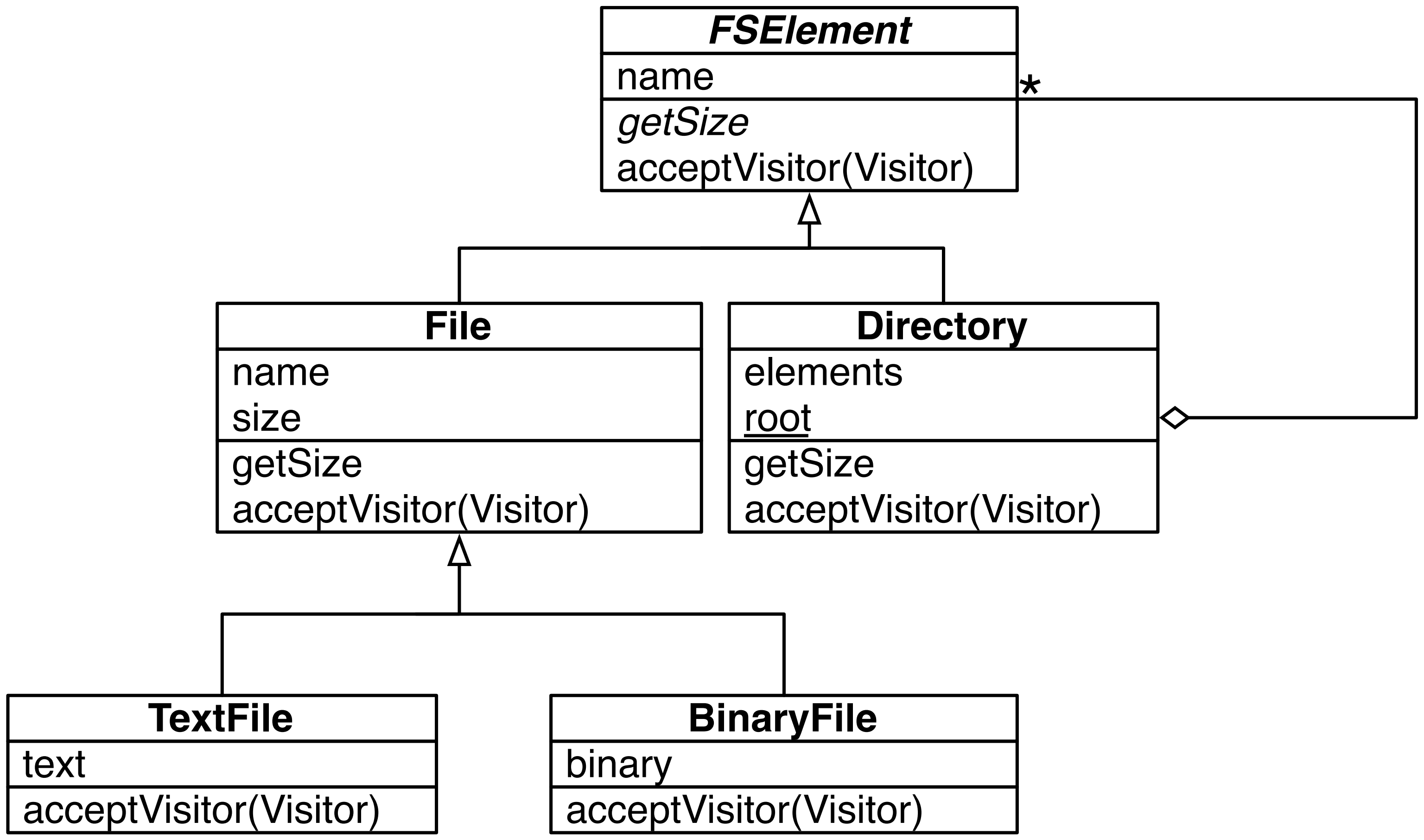
Adding operations

The visitor pattern is a nice solution to add new operations

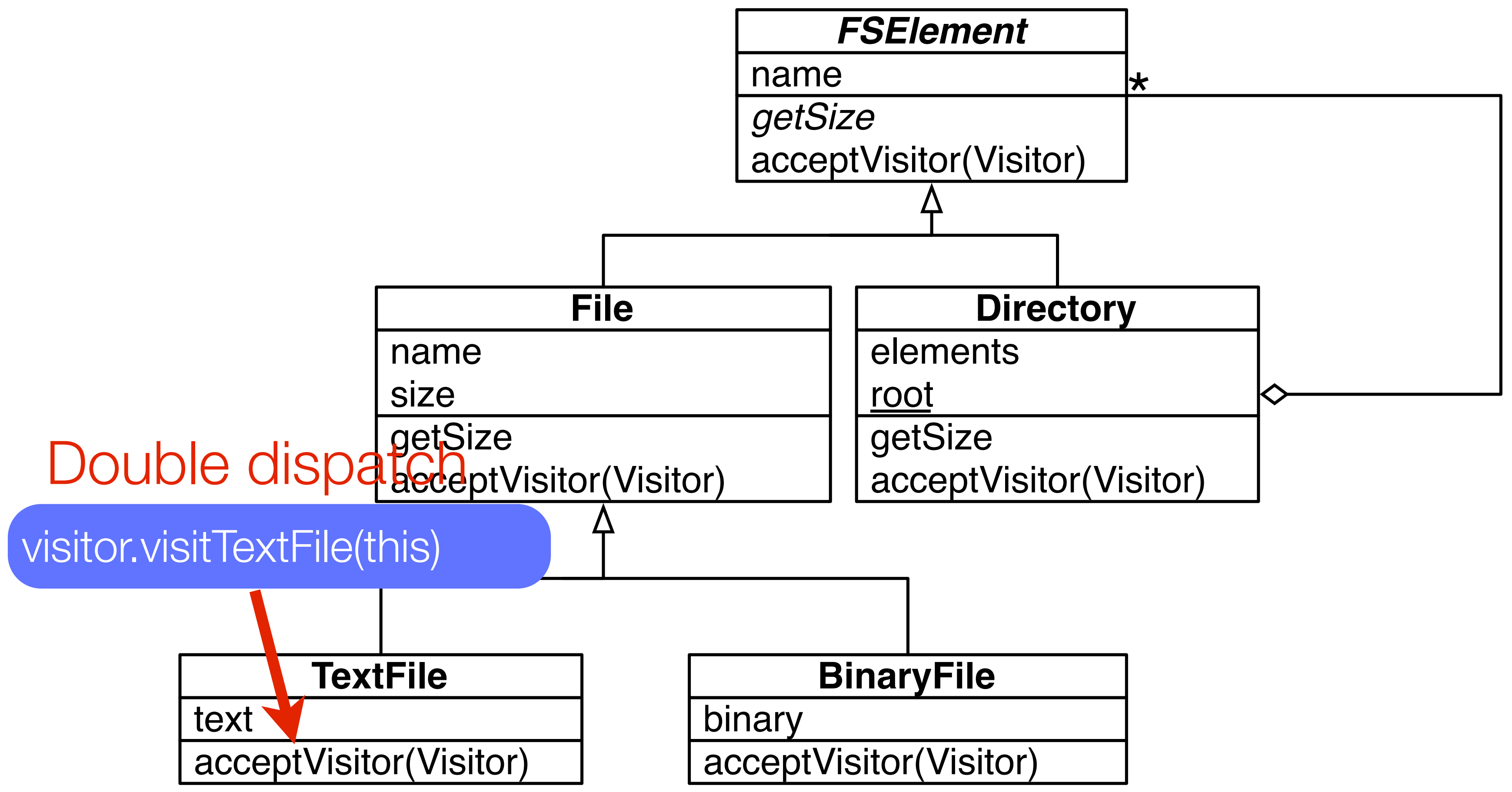
Operation are defined *externally* from the domain, by subclassing *Visitor*

The drawback is that it usually enforces the state of the objects to be accessible from outside

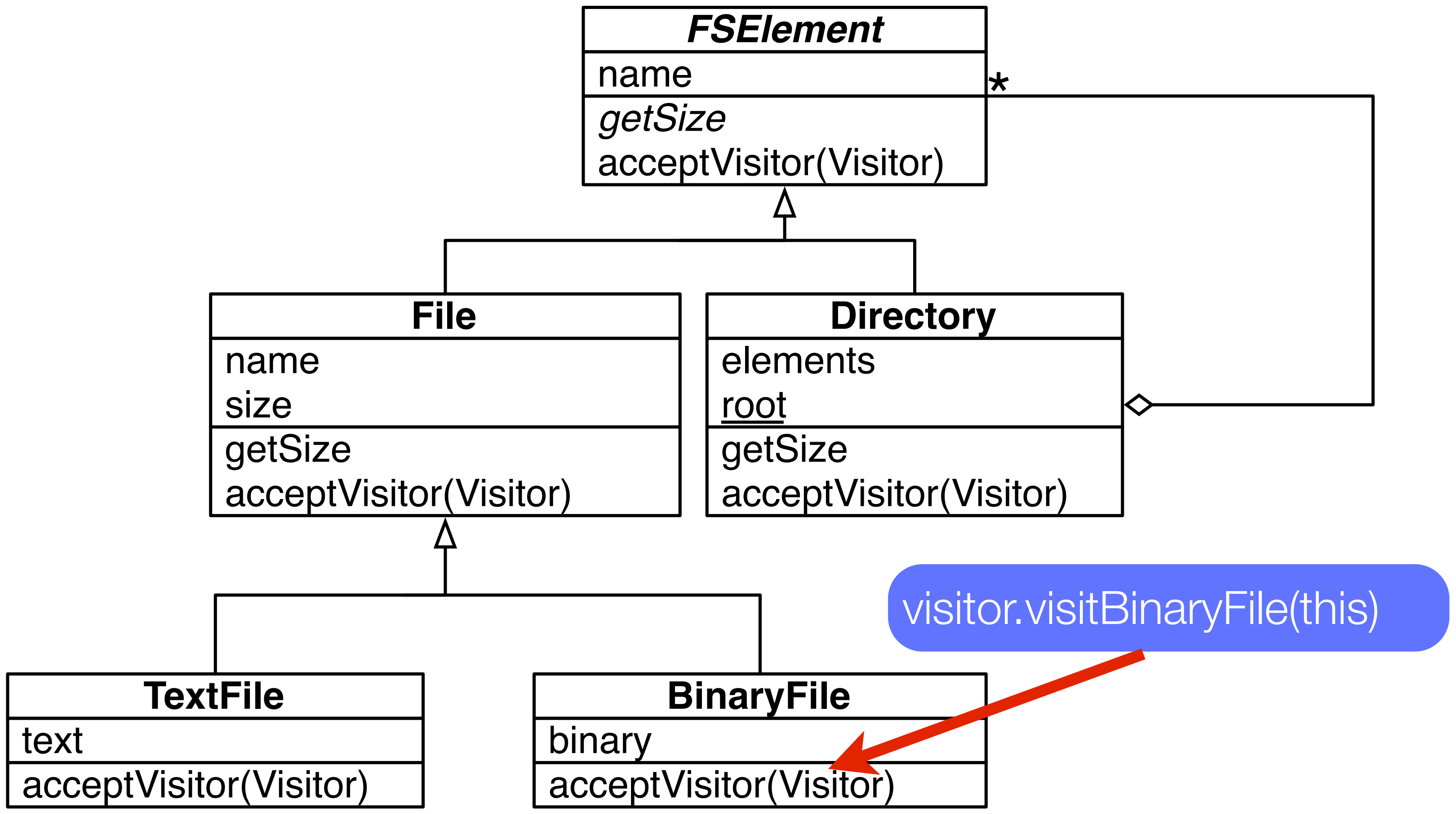
New version of our file system



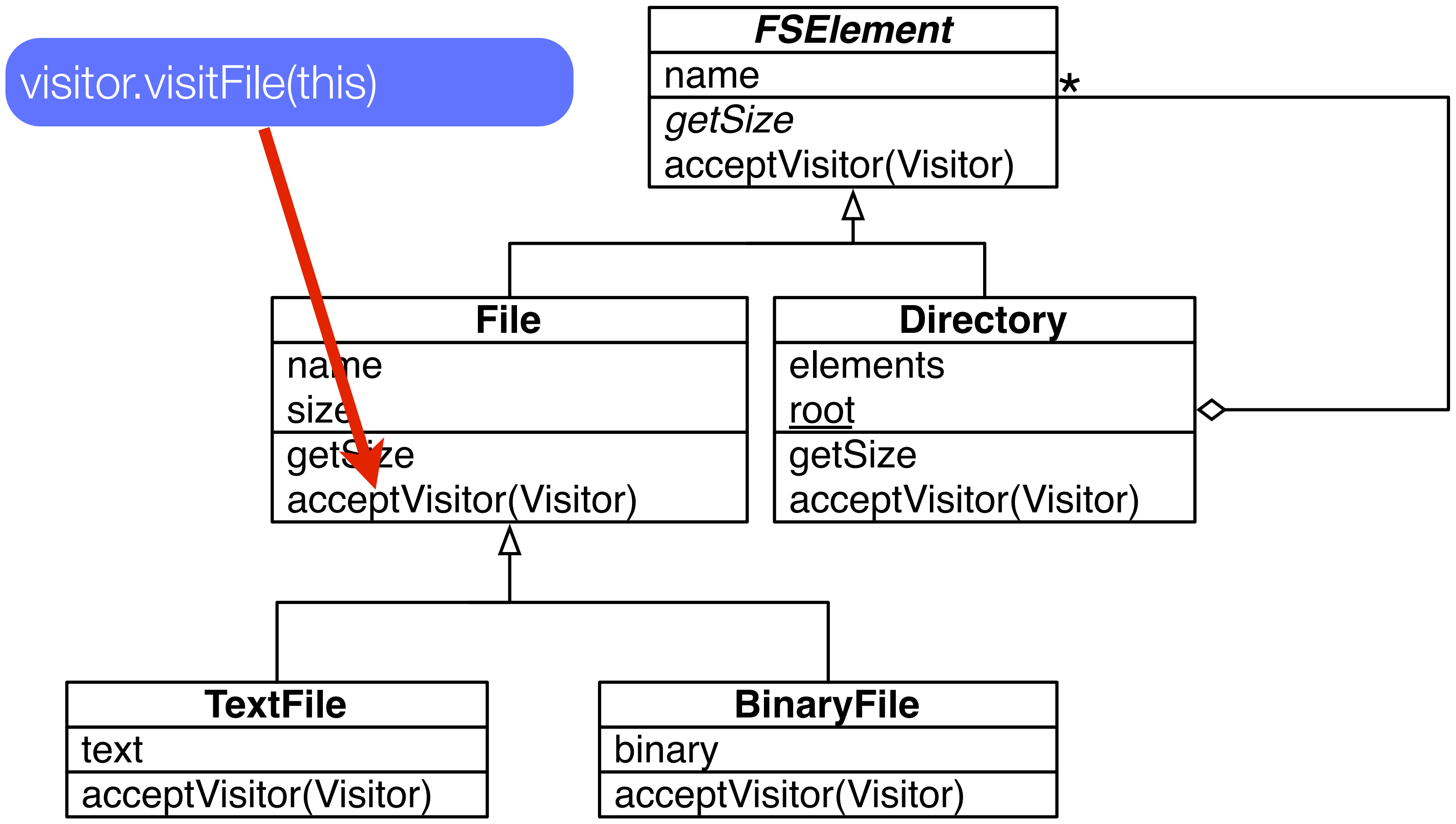
New version of our file system



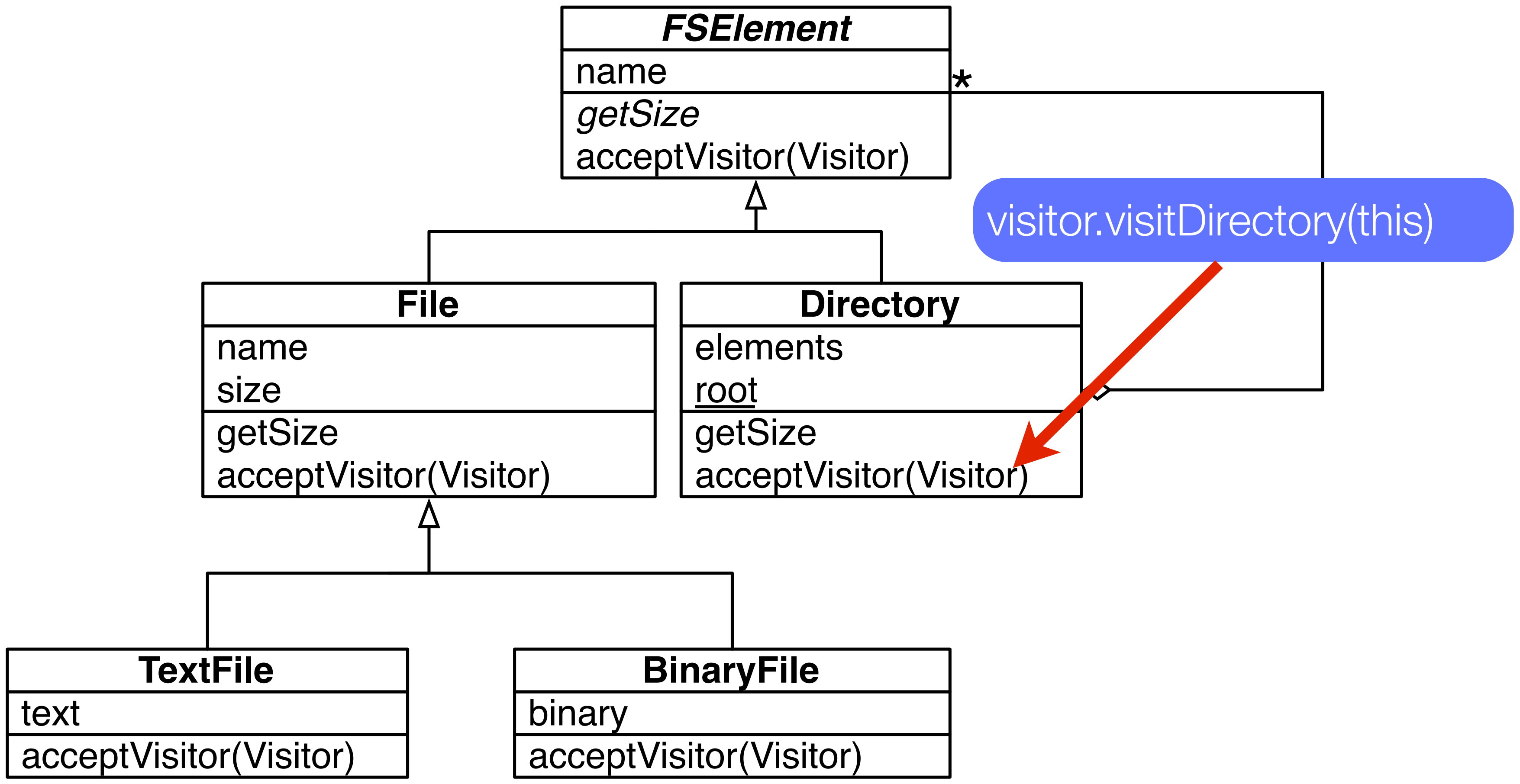
New version of our file system

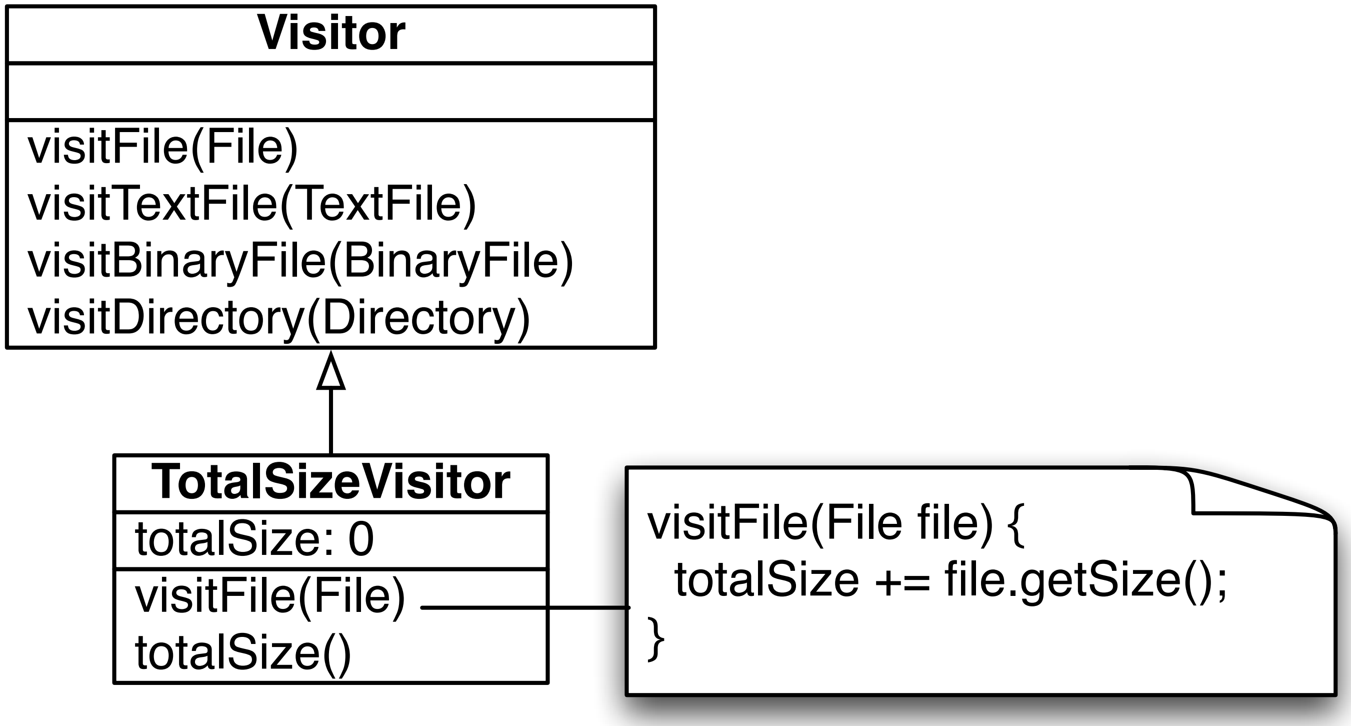


New version of our file system



New version of our file system





```
visitFile(File file) {
  totalSize += file.getSize();
}
```

The case for Dictionary

The recursion over the structure at runtime can be achieved in the Visitor

```
class Visitor {
    public void visitDirectory(Directory d) {
        for (Element e : d.getElements() )
            e.acceptVisitor(this);
    }
   public void visitFile(File d) { }
    ...
}
```