Group 11, Exercise 1

Mahsa.poorjafari, mpourjaf@rhrk.uni-kl.de, Rama Klaho, klaho@rhrk.uni-kl.de

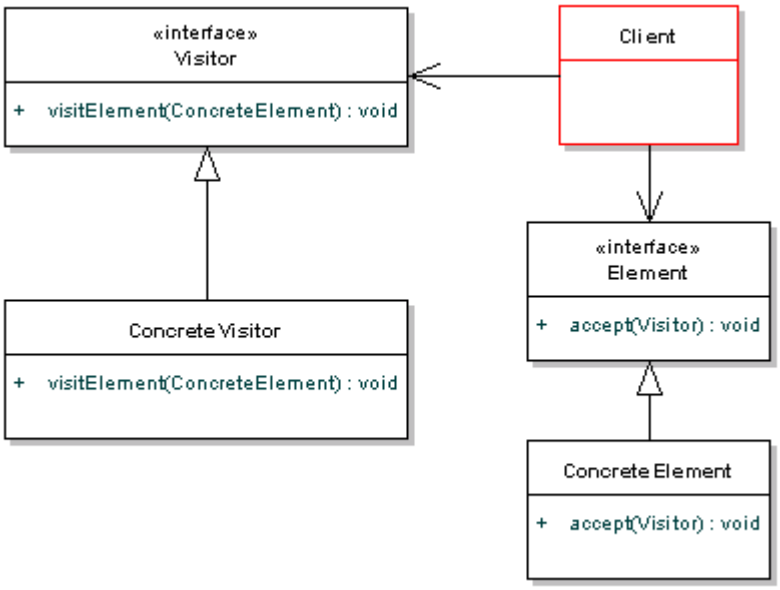
May 2017

1 Binding strength

Binding strength or as it called Precedence will reduce the ambiguous of grammar. By write the grammar rules for all binary and unary operators, we still have very ambiguous grammar with many parsing conflicts, with regards to disambiguating our grammar rules we specified precedence for all the operators and the associativity of the binary operators in the following form:
/* Precedences */
precedence left PLUS, MINUS;
precedence left TIMES, DIV;
precedence left UMINUS;
Then we describe it that UMINUS has the most priority.
[1]More experienced grammar authors may encouter situations, in which it is helpful to fine-tune precedences of specific production alternatives. For these cases, CUP permitted to annotate grammar alternatives with a
"

2 The Visitor Pattern

[2]The Visitor is known as a behavioural pattern,as it's used to manage algorithms, relationships and responsibilities between objects. The definition of Visitor provided in the original Gang of Four book on Design Patterns states:
"Allows for one or more operation to be applied to a set of objects at runtime, decoupling the operations from the object structure. "
What the Visitor pattern actually does is create an external class that uses data in the other classes. If you need to perform operations across a dispate set of objects, Visitor might be the pattern for you. The GoF book says that the Visitor pattern can provide additional functionality to a class without changing it. Let's see how that can work, first by taking a look at the classic diagram definition of the Visitor pattern:



The core of this pattern is the **Visitor** interface (in our case **ExprVisitor**) interface. This interface defines a visit operation for each type of **ConcreteElement**(in our case element in AST folder) in the object structure. Meanwhile, the **ConcreteVisitor** (in our case **ExprPrintVisitor** for Parser Tests and **ExprCalVisitor** for interpreter Tests) implements the operations defined in the Visitor interface. The concrete visitor will store local state, typically as it traverses the set of elements. The element interface simply defines an **accept** method to allow the visitor to run some action over that element - the ConcreteElement will implement this accept method. We have two accept method in our AST elements, one with *String* type to pass Parser Tests and one with *int* type to pass interpreter tests.

References

[1] cup group. Cup2 user manual.

[2] James Sugrue. Visitor pattern tutorial with java examples.