Interpreter Design Pattern

The interpreter pattern is a design pattern that specifies how to evaluate sentences in a language

Basic idea is to have a class for each symbol (terminal or nonterminal) in a specialized computer language

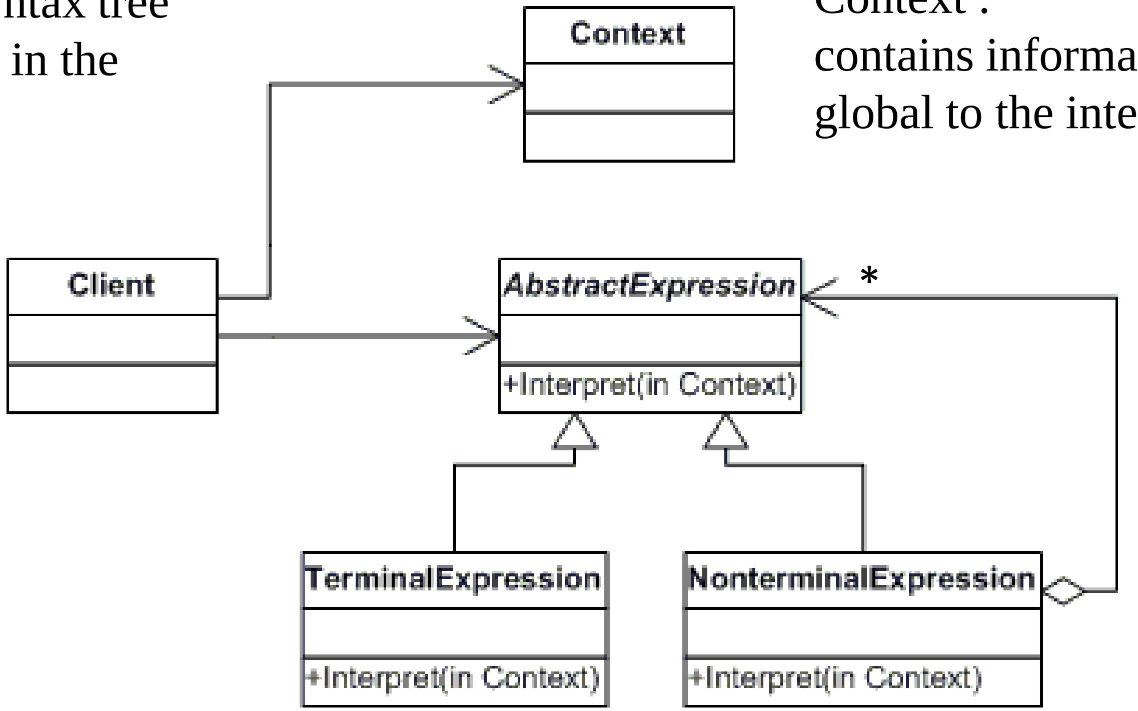
The syntax tree of a sentence in the language is an instance of the composite pattern

The syntax tree is traversed to evaluate (interpret) the sentence

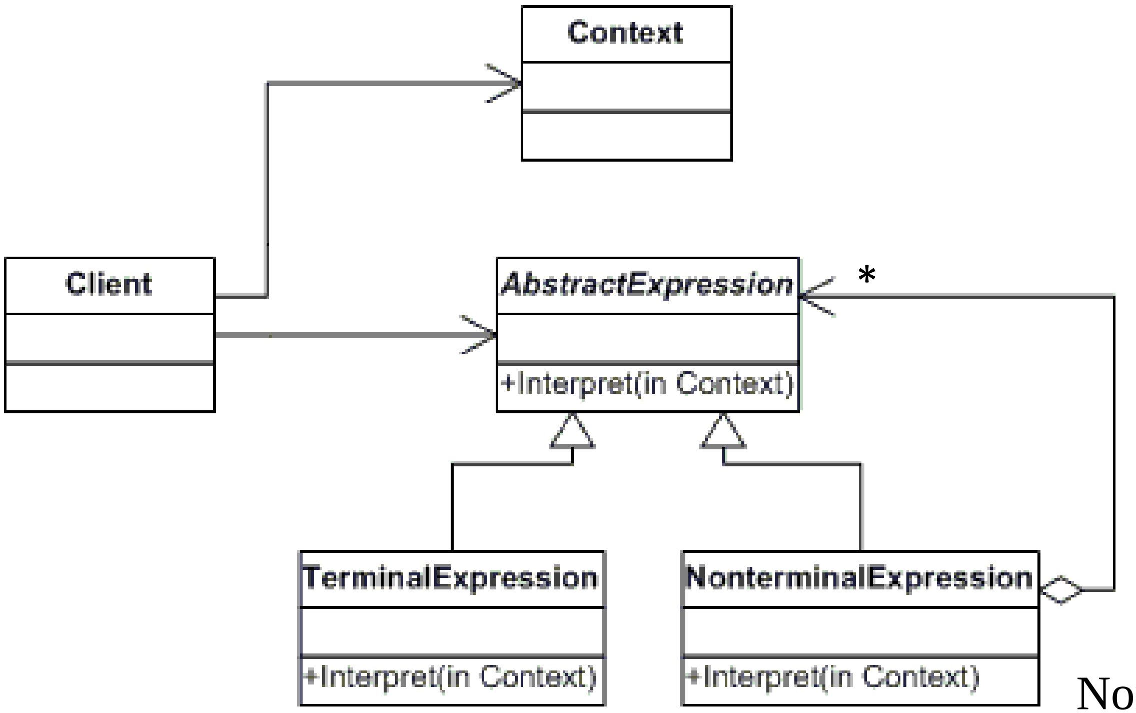
Interpreter Design Pattern

Client :
builds (or is given) a syntax tree representing a sentence in the language

Context :
contains information that is global to the interpreter



Interpreter Design Pattern



AbstractExpression :

declares an interface for executing an operation

NonterminalExpression :

Implements interpret() for nonterminal symbols in the grammar.

interpret() typically calls itself recursively

TerminalExpression:

implements interpret() for terminal symbols in the grammar.

Interpreter Design Pattern

In wikipedia see:

http://en.wikipedia.org/wiki/Interpreter_pattern
Java code for the reverse polish example

http://en.wikipedia.org/wiki/Backus%E2%80%93Naur_Form
BNF examples

http://en.wikipedia.org/wiki/Syntax_diagram
BNF as a syntax diagram

Last few pages of
http://www.standardpascal.org/The_Programming_Language_Pascal_1973.pdf
Pascal described in diagrams

Interpreter Design Pattern

The grammar

```
expression ::= plus | minus | variable | number
plus       ::= expression expression '+'
minus      ::= expression expression '-'
variable   ::= 'a' | 'b' | 'c' | ... | 'z'
digit      ::= '0' | '1' | ... '9'
number     ::= digit | digit number
```

See wikipedia

The above defines
An *expression* to be one of : a *plus*, a *minus*, a *variable*, or a *number*.
A *plus* is an *expression* followed by another *expression* which in turn is followed by a *plus sign*.
A *number* is a *digit*, or a *digit* followed by a *number*.
etc

Interpreter Design Pattern

Examples of sentences in the grammar are:

5 10 +
a b c + -
5 10 + 2 4 - -

A sentence must be evaluated.
How do we evaluate the above?

Interpreter pattern requires one class per grammar rule
See web page for code

Interpreter Design Pattern

```
interface Expression {
   public int interpret(Map<String,Expression> variables);
}
```

Interpreter Design Pattern

```
 class Plus implements Expression {
    Expression leftOperand;
    Expression rightOperand;
    public Plus(Expression left, Expression right) {
        leftOperand   = left;
        rightOperand = right;
    }

    public int interpret(Map<String,Expression> variables)
{
        return leftOperand.interpret(variables) +
            rightOperand.interpret(variables);
    }
}
```

………. Can see more at en.wikipedia.org/wiki/Interpreter_pattern

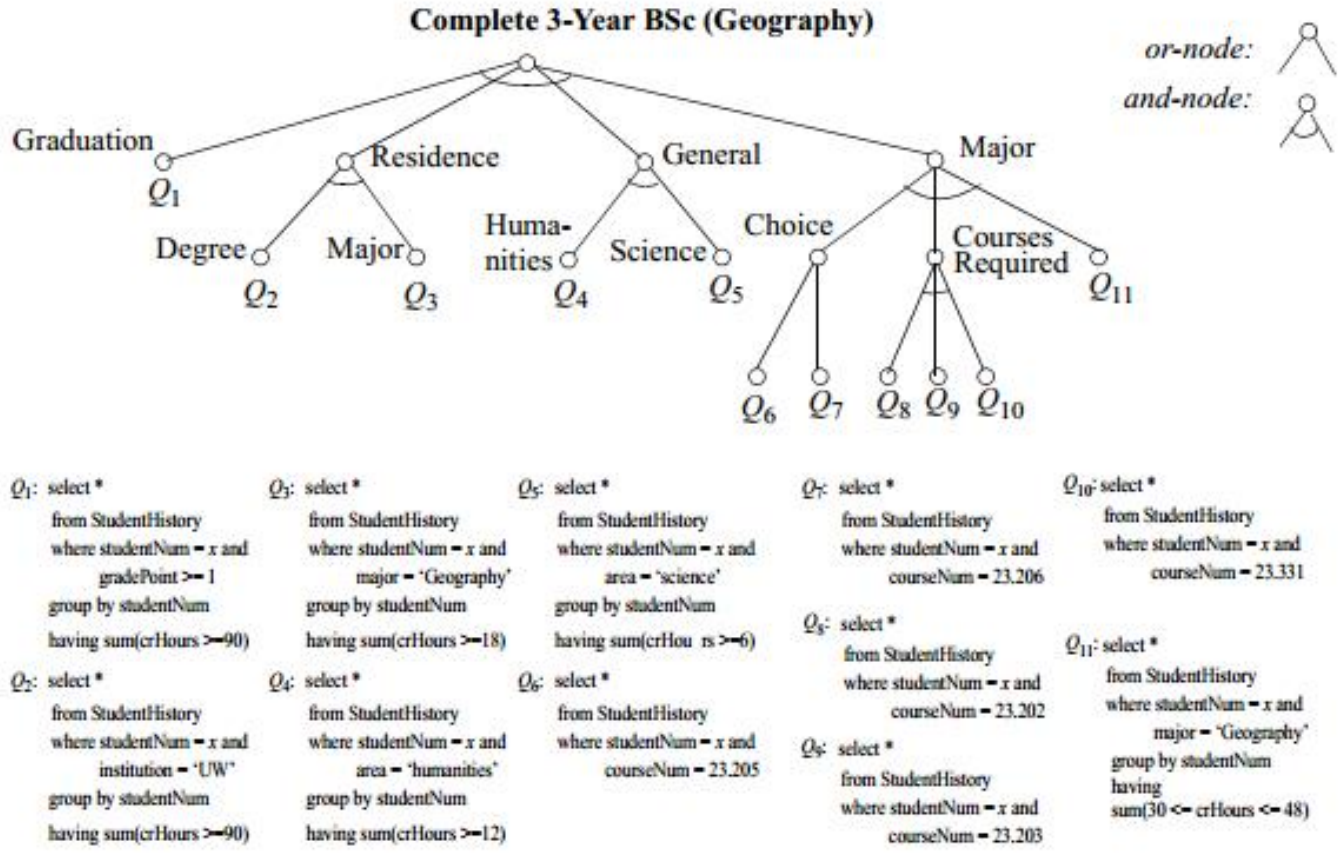
Interpreter Design Pattern

Example 2

Some previous work (with Yangjun Chen) involved a Synthesized Query Tree representing graduation requirements for a major

Based on a student’s academic record and declared major …

Does the student satisfy requirements to graduate?



```
Q1: select *
    from StudentHistory
    where studentNum = x and
          gradePoint >= 1
    group by studentNum
    having sum(crHours >=90)

Q2: select *
    from StudentHistory
    where studentNum = x and
          institution = 'UW'
    group by studentNum
    having sum(crHours >=90)

Q3: select *
    from StudentHistory
    where studentNum = x and
          major = 'Geography'
    group by studentNum
    having sum(crHours >=18)

Q4: select *
    from StudentHistory
    where studentNum = x and
          area = 'humanities'
    group by studentNum
    having sum(crHours >=12)

Q5: select *
    from StudentHistory
    where studentNum = x and
          area = 'science'
    group by studentNum
    having sum(crHou rs >=6)

Q6: select *
    from StudentHistory
    where studentNum = x and
          courseNum = 23.205

Q7: select *
    from StudentHistory
    where studentNum = x and
          courseNum = 23.206

Q8: select *
    from StudentHistory
    where studentNum = x and
          courseNum = 23.202

Q9: select *
    from StudentHistory
    where studentNum = x and
          courseNum = 23.203

Q10: select *
    from StudentHistory
    where studentNum = x and
          courseNum = 23.331

Q11: select *
    from StudentHistory
    where studentNum = x and
          major = 'Geography'
    group by studentNum
    having
    sum(30 <= crHours <= 48)
```

Interpreter Design Pattern

A simpler situation would be evaluating a student's academic record to see if the student meets a pre-requisite requirement to enroll in a course

A requirement is a course that must be taken. Suppose a requirement is met if the student received a C or better in the course.

Interpreter Design Pattern

Example

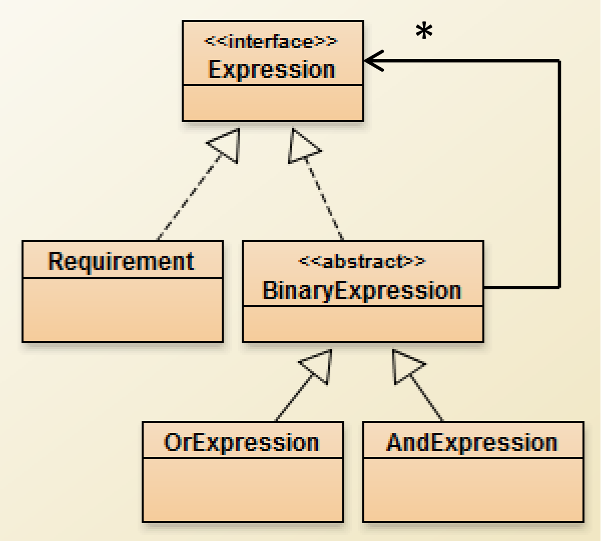
Suppose we need to evaluate pre-requisite expressions for the UW

Assume pre-requisite expressions are defined as:

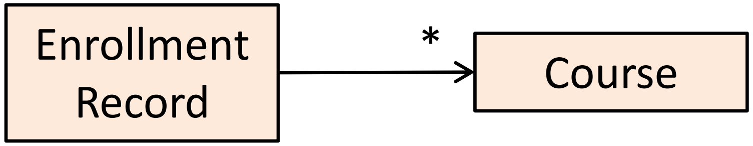
Expression ← Requirement | BinaryExpression
BinaryExpression ← OrExpression | AndExpression
OrExpression ← expression OR expression
AndExpression ← expression AND expression

Interpreter Design Pattern

Expression ← Requirement | BinaryExpression
BinaryExpression ← OrExpression | AndExpression
OrExpression ← expression ‘OR’ expression
AndExpression ← expression ‘AND’ expression



To evaluate a requirement we check a student’s enrollment record (the context) to see if the student has taken the course and received a C or better.


Enrollment Record
*
Course

To evaluate a binary expression …

Interpreter Design Pattern

Code for example:

- Driver
- Course
- EnrollmentRecord
- Expression
- Requirement
- BinaryExpression
- OrExpression
- AndExpression

Object diagram?
Sequence diagram?