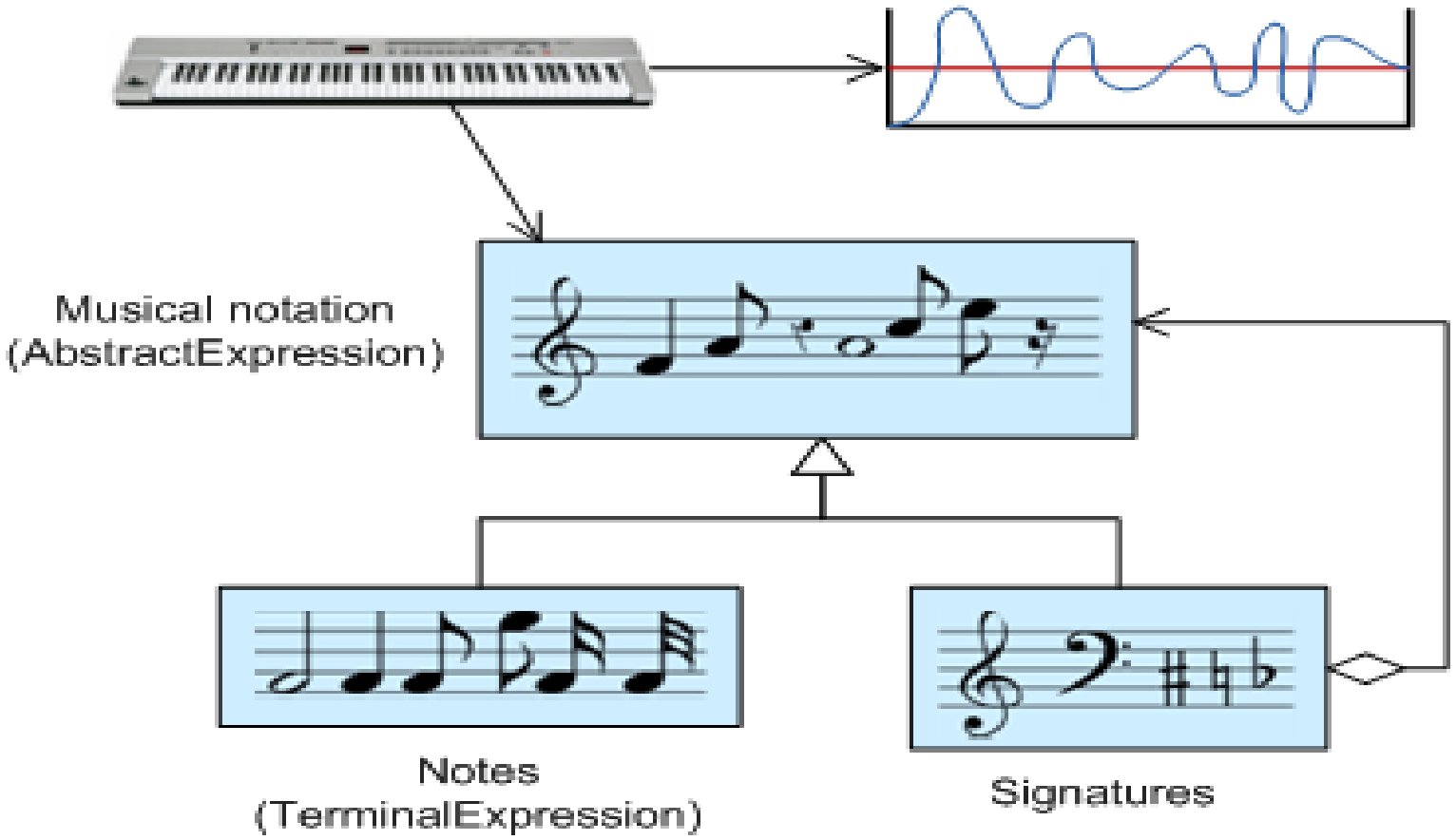
CS5551 Advanced Software Engineering
Design Patterns: Interpreter Pattern

By
Muktevi, Vamsi Krishna
Tummala, Vijay Kumar
Panja, Gopal
Murakonda, Sravani

Interpreter Pattern

- The main intent of Interpreter pattern is to map a domain to a language, the language to a grammar, and the grammar to a hierarchical object oriented design.
  - Each given expression is interpreted and some conditions are validated and a Boolean output is produced for each interpreted expressions.
  - The client provides the String (Context) that is to be interpreted.
  - Some validations are performed by the interpreter on the given context and a Boolean output is generated by the interpreter.

Interpreter Pattern

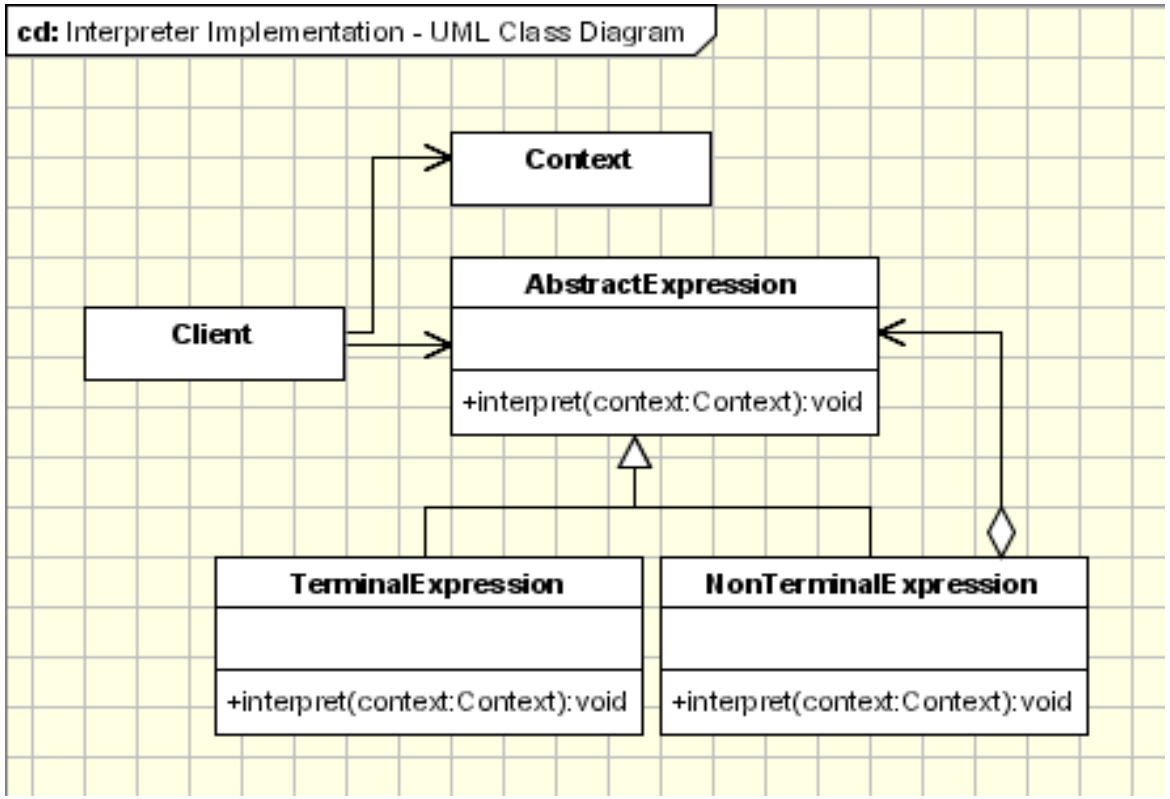


Interpreter Pattern

- The Interpreter pattern has a limited area where it can be applied.
- It can be only discussed in terms of formal grammar but in this area there are better solutions.
- This is the reason why this pattern is not so frequently used.
- This pattern can be applied for parsing light expressions defined in simple grammars.

Interpreter Pattern

- The implementation of Interpreter Patten is just the use of composite pattern to represent grammar.
- The interpreter defines the behavior while composite defines the structure.



http://www.oodesign.com/images/behavioral/interpreter-pattern.gif

Interpreter Pattern

- Expression.java
- TerminalExpression.java
- OrExpression.java
- AndExpression.java
- Client.java

Interpreter Pattern

Expression.java

```
public abstract class Expression {

    abstract public boolean interpret(String context);

}
```

Interpreter Pattern

TerminalExpression.java

```
public class TerminalExpression extends Expression {

    private String literal=null;

    public TerminalExpression(String context){
        literal = context;
    }

    @Override
    public boolean interpret(String context) {
        StringTokenizer st = new StringTokenizer(context);
        while (st.hasMoreTokens()){
            String token = st.nextToken();
            if (token.equals(literal)){
                return true;
            }
        }
        return false;
    }

}
```

Interpreter Pattern

AndExpression.java

```
public class AndExpression extends Expression {

    private Expression exp1 = null;
    private Expression exp2 = null;

    public AndExpression(Expression expression1,Expression expression2) {
        exp1=expression1;
        exp2=expression2;
    }

    @Override
    public boolean interpret(String context) {

        return exp1.interpret(context) && exp2.interpret(context);
    }

}
```

Interpreter pattern

OrExpression.java

```
public class OrExpression  extends Expression{

    private Expression exp1 = null;
    private Expression exp2 = null;

    public OrExpression(Expression expression1,Expression expression2) {
        exp1=expression1;
        exp2=expression2;
    }

    @Override
    public boolean interpret(String context) {
        return exp1.interpret(context) || exp2.interpret(context);
    }

}
```

Interpreter pattern

Client.java

```
public class Client {

    public static void main(String[] args) {
        Expression terminal1 = new TerminalExpression("John");
        Expression terminal2 = new TerminalExpression("Henry");
        Expression terminal3 = new TerminalExpression("Mary");
        Expression terminal4 = new TerminalExpression("Owen");

      // Henry or Mary
        Expression alternation1 = new OrExpression(terminal2, terminal3);

         // John or (Henry or Mary)
        Expression alternation2 = new OrExpression(terminal1, alternation1);

         // Owen and (John or (Henry or Mary))
        Expression define = new  AndExpression(terminal4, alternation2);

        String context = "Owen Marry Henry";
        System.out.println("Input String: "+ context);
        System.out.println(context+ "is :" + define.interpret(context));
    }
}
```

Interpreter Pattern

Input & output

```
<terminated> Client [Java Application] C:\Program Files\Java\jdk1.7.0_02\jre\bin\javaw.exe (Apr 18, 2016, 5:39:47 AM)
Input String: Owen Marry Henry
Owen Marry Henryis :true
```

References

- http://www.oodesign.com/interpreter-pattern.html
- http://www.tutorialspoint.com/design_pattern/interpreter_pattern.htm
- https://en.wikipedia.org/wiki/Interpreter_pattern
- https://sourcemaking.com/files/sm/images/patterns/Interpreter_example1.png