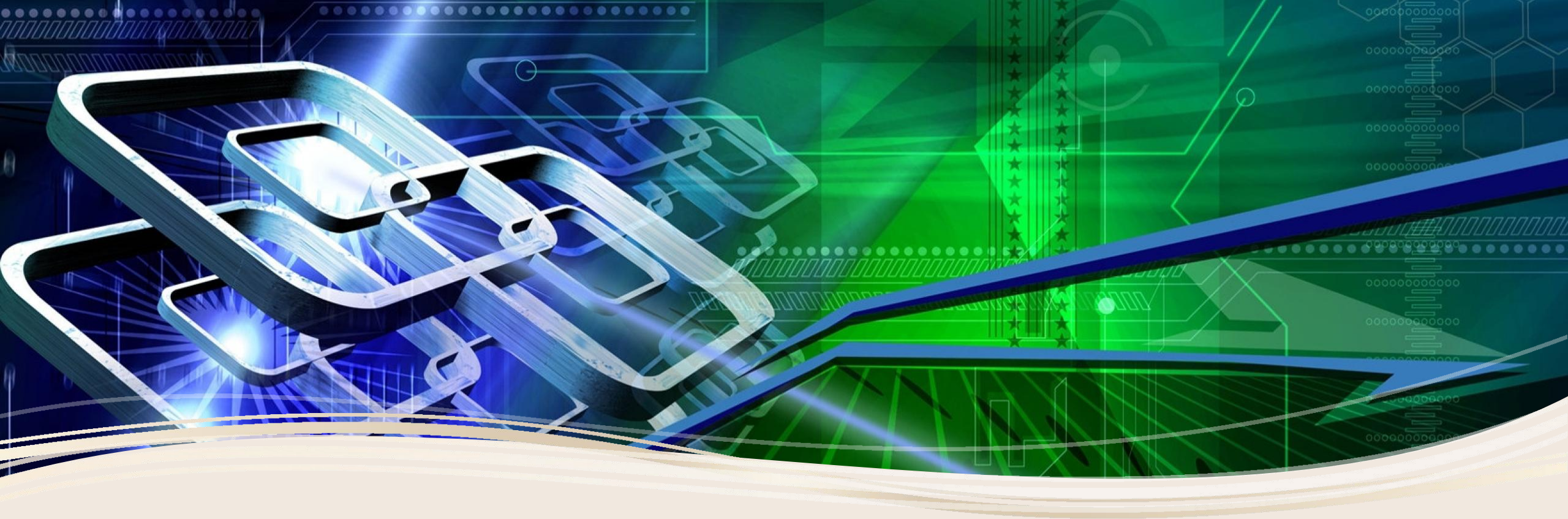




UNIVERSITAS
KRISTEN
MARANATHA



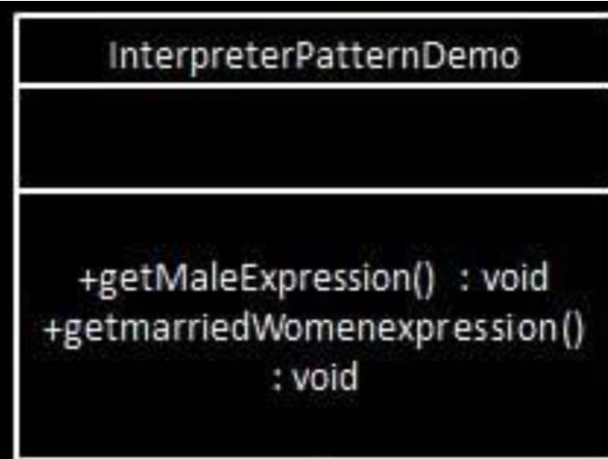
Interpreter Pattern

Pola Desain Perangkat Lunak
Tjatur Kandaga G.
Fakultas IT - UK Maranatha

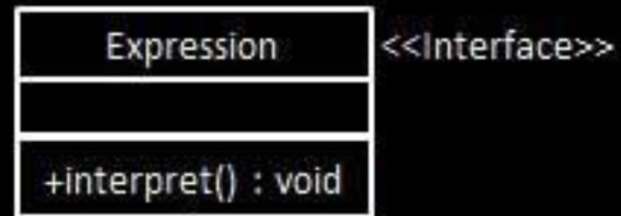
Mengenal Interpreter Pattern

- Interpreter pattern menyediakan sarana untuk mengevaluasi tata bahasa atau ekspresi (language grammar or expression).
- Interpreter pattern termasuk behavioural pattern.
- Pattern ini meliputi implementasi sebuah interface Expression yang bertugas menginterpretasi sebuah konteks tertentu.
- Interpreter pattern digunakan dalam SQL parsing, mesin pemroses simbol dll.

Contoh Interpreter Pattern

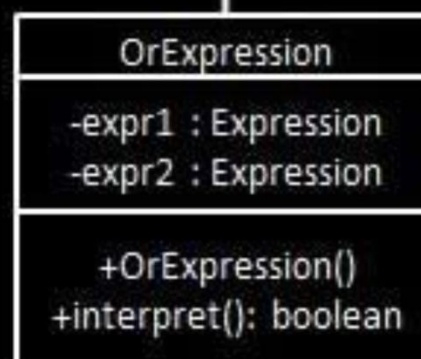
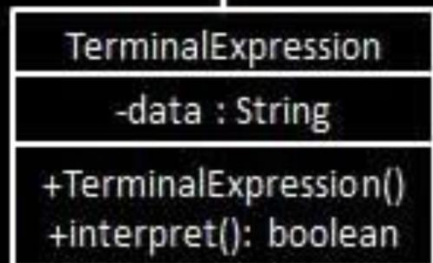


uses



implements

implement



Contoh Interpreter Pattern

- Kita akan membuat interface Expression beserta kelas-kelas yang meng-implementnya.
- Kelas TerminalExpression berlaku sebagai interpreter utama untuk konteks yang sedang diproses.
- Kelas OrExpression dan AndExpression digunakan untuk membentuk kombinasi ekspresi.
- Kelas InterpreterPatternDemo menggunakan kelas Expression untuk membuat rule kemudian menggunakannya untuk menginterpret konteks permasalahan.

Contoh Interpreter Pattern

```
1 package edu.maranatha.pdpl;  
2  
3 public interface Expression {  
4     public boolean interpret(String context);  
5 }
```

Contoh Interpreter Pattern

```
3 public class TerminalExpression implements Expression {
4     private String data;
5
6     public TerminalExpression(String data) {
7         this.data = data;
8     }
9
10    @Override
11    public boolean interpret(String context) {
12        if (context.contains(data)) {
13            return true;
14        }
15        return false;
16    }
17 }
```



Contoh Interpreter Pattern

```
3 public class OrExpression implements Expression {
4     private Expression expr1 = null;
5     private Expression expr2 = null;
6
7     public OrExpression(Expression expr1, Expression expr2) {
8         this.expr1 = expr1;
9         this.expr2 = expr2;
10    }
11
12    @Override
13    public boolean interpret(String context) {
14        return expr1.interpret(context) || expr2.interpret(context);
15    }
16 }
```


Contoh Interpreter Pattern

```
3 public class AndExpression implements Expression {
4     private Expression expr1 = null;
5     private Expression expr2 = null;
6
7     public AndExpression(Expression expr1, Expression expr2) {
8         this.expr1 = expr1;
9         this.expr2 = expr2;
10    }
11
12    @Override
13    public boolean interpret(String context) {
14        return expr1.interpret(context) && expr2.interpret(context);
15    }
16 }
```

```

3 public class InterpreterPatternDemo {
4     //Rule: Robert and John are male
5     public static Expression getMaleExpression() {
6         Expression robert = new TerminalExpression("Robert");
7         Expression john = new TerminalExpression("John");
8         return new OrExpression(robert, john);
9     }
10
11    //Rule: Julie is a married women
12    public static Expression getMarriedWomanExpression() {
13        Expression julie = new TerminalExpression("Julie");
14        Expression married = new TerminalExpression("Married");
15        return new AndExpression(julie, married);
16    }
17
18    public static void main(String[] args) {
19        Expression isMale = getMaleExpression();
20        Expression isMarriedWoman = getMarriedWomanExpression();
21        System.out.println("John is male? " + isMale.interpret("John"));
22        System.out.println("Julie is a married woman? "
23            + isMarriedWoman.interpret("Married Julie"));
24    }
25 }

```

Contoh Interpreter Pattern

Contoh Interpreter Pattern

```
John is male? true  
Julie is a married woman? true
```

Contoh Interpreter Pattern (C#)

```
7 namespace InterpreterCS
8 {
9     22 references
10     public interface Expression
11     {
12         9 references
13         bool interpret(String context);
14     }
15 }
```


Contoh Interpreter Pattern (C#)

```
7 namespace InterpreterCS
8 {
9     5 references
10    public class TerminalExpression : Expression
11    {
12
13        4 references
14        public TerminalExpression(String data)
15        {
16            this.data = data;
17        }
18
19        9 references
20        bool Expression.interpret(String context)
21        {
22            if (context.Contains(data))
23            {
24                return true;
25            }
26            return false;
27        }
28    }
29 }
```

Contoh Interpreter Pattern (C#)

```
7 namespace InterpreterCS
8 {
9     2 references
    public class AndExpression : Expression
10     {
11         private Expression expr1 = null;
12         private Expression expr2 = null;
13
14         1 reference
    public AndExpression(Expression expr1, Expression expr2)
15     {
16         this.expr1 = expr1;
17         this.expr2 = expr2;
18     }
19     2
20     9 references
    bool Expression.interpret(String context)
21     {
22         return expr1.interpret(context) && expr2.interpret(context);
23     }
24 }
25 }
```



Contoh Interpreter Pattern (C#)

```
7 namespace InterpreterCS
8 {
9     2 references
    public class OrExpression : Expression
10     {
11         private Expression expr1 = null;
12         private Expression expr2 = null;
13
14         1 reference
    public OrExpression(Expression expr1, Expression expr2)
15     {
16         this.expr1 = expr1;
17         this.expr2 = expr2;
18     }
19     2
20     9 references
    bool Expression.interpret(String context)
21     {
22         return expr1.interpret(context) || expr2.interpret(context);
23     }
24 }
25 }
```

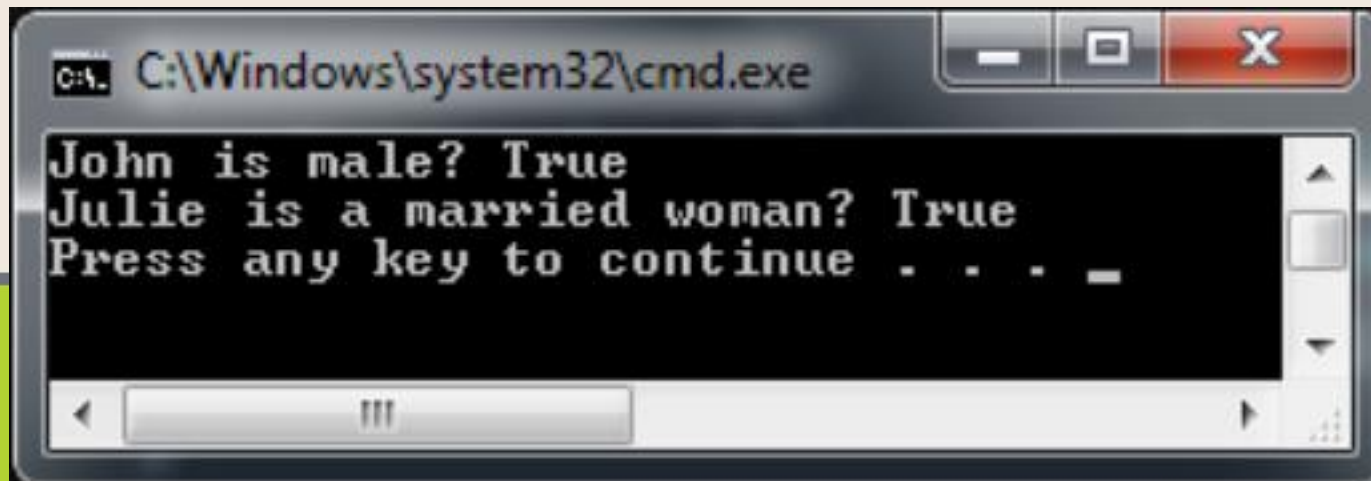


Contoh Interpreter Pattern (C#)

```
7 namespace InterpreterCS
8 {
9     0 references
10    class Program
11    {
12        //Rule: Robert and John are male
13        1 reference
14        public static Expression getMaleExpression()
15        {
16            Expression robert = new TerminalExpression("Robert");
17            Expression john = new TerminalExpression("John");
18            return new OrExpression(robert, john);
19        }
20
21        //Rule: Julie is a married women
22        1 reference
23        public static Expression getMarriedWomanExpression()
24        {
25            Expression julie = new TerminalExpression("Julie");
26            Expression married = new TerminalExpression("Married");
27            return new AndExpression(julie, married);
28        }
29    }
30 }
```


Contoh Interpreter Pattern (C#)

```
27 public static void Main(String[] args)
28 {
29     Expression isMale = getMaleExpression();
30     Expression isMarriedWoman = getMarriedWomanExpression();
31     Console.WriteLine("John is male? " + isMale.interpret("John"));
32     Console.WriteLine("Julie is a married woman? "
33         + isMarriedWoman.interpret("Married Julie"));
34 }
35 }
36 }
```



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The output of the program is displayed as follows:

```
John is male? True
Julie is a married woman? True
Press any key to continue . . . _
```

Thank You!

Any Question?

Any doubt?

Just ask don't interpret it by yourself.