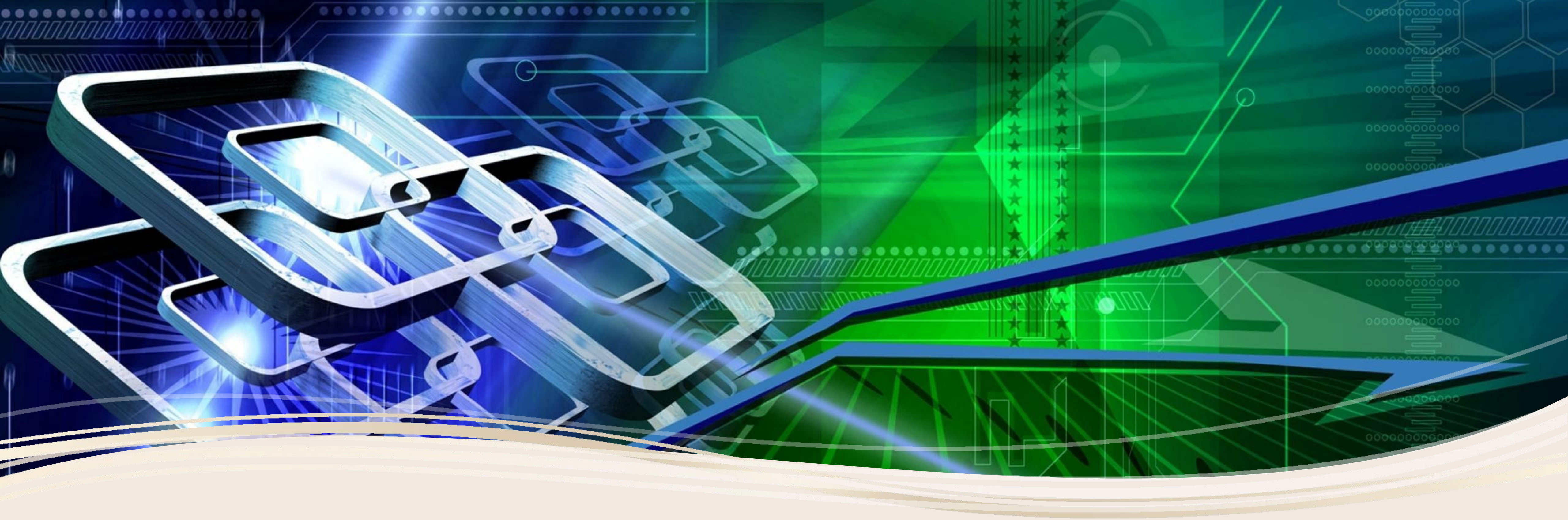
UNIVERSITAS
KRISTEN
MARANATHA



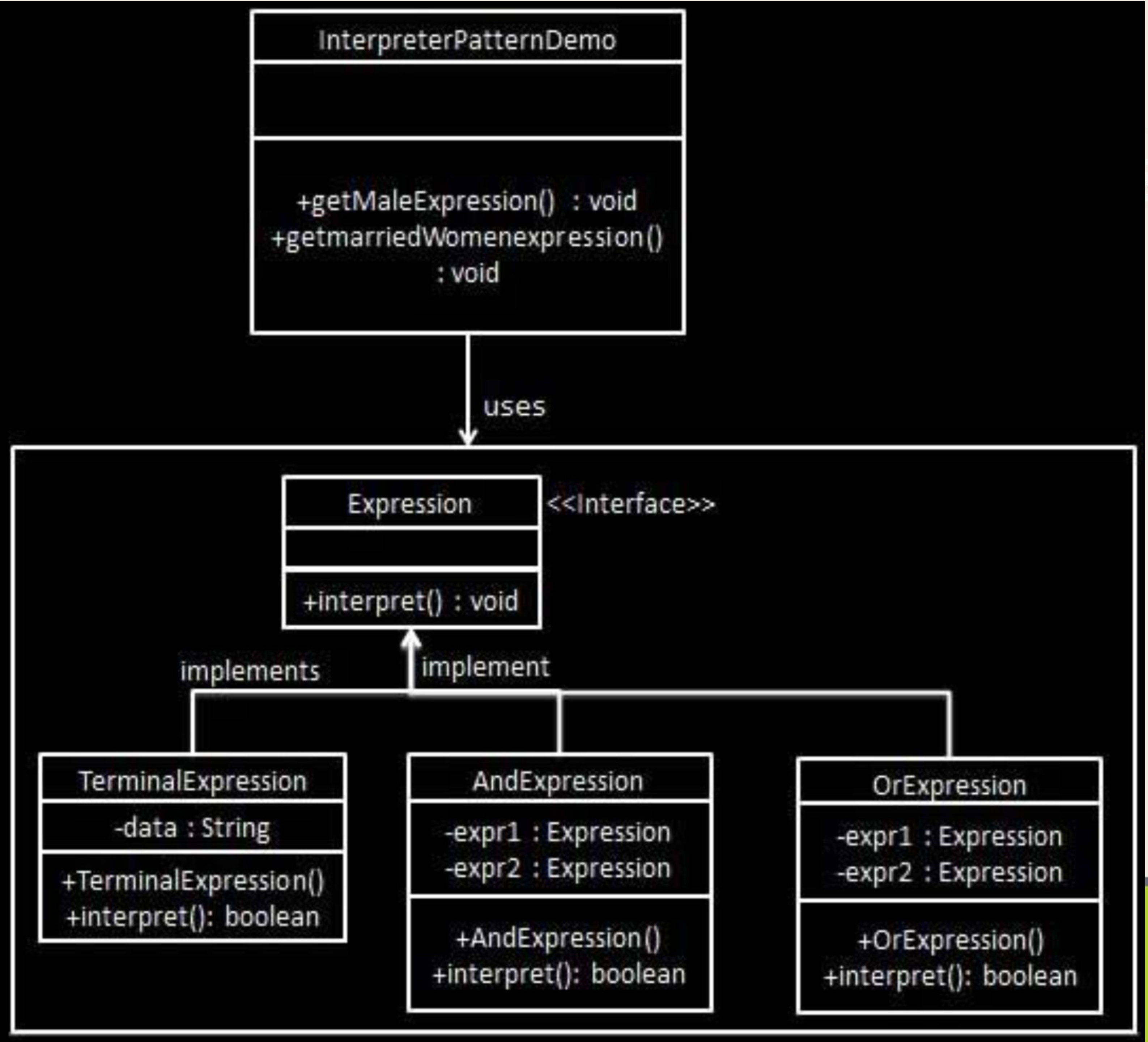
Interpreter Pattern

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.

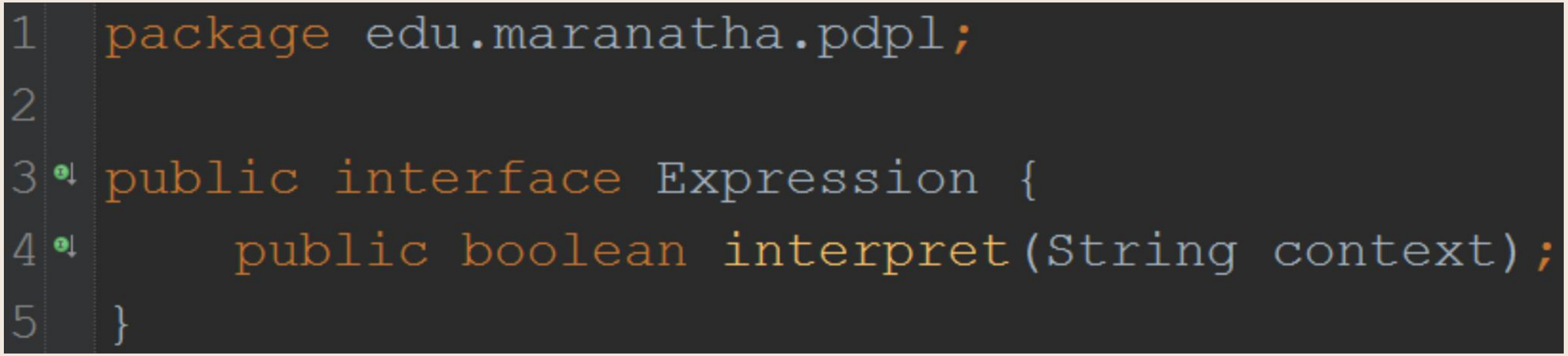
Contoh Interpreter Pattern



Contoh Interpreter Pattern

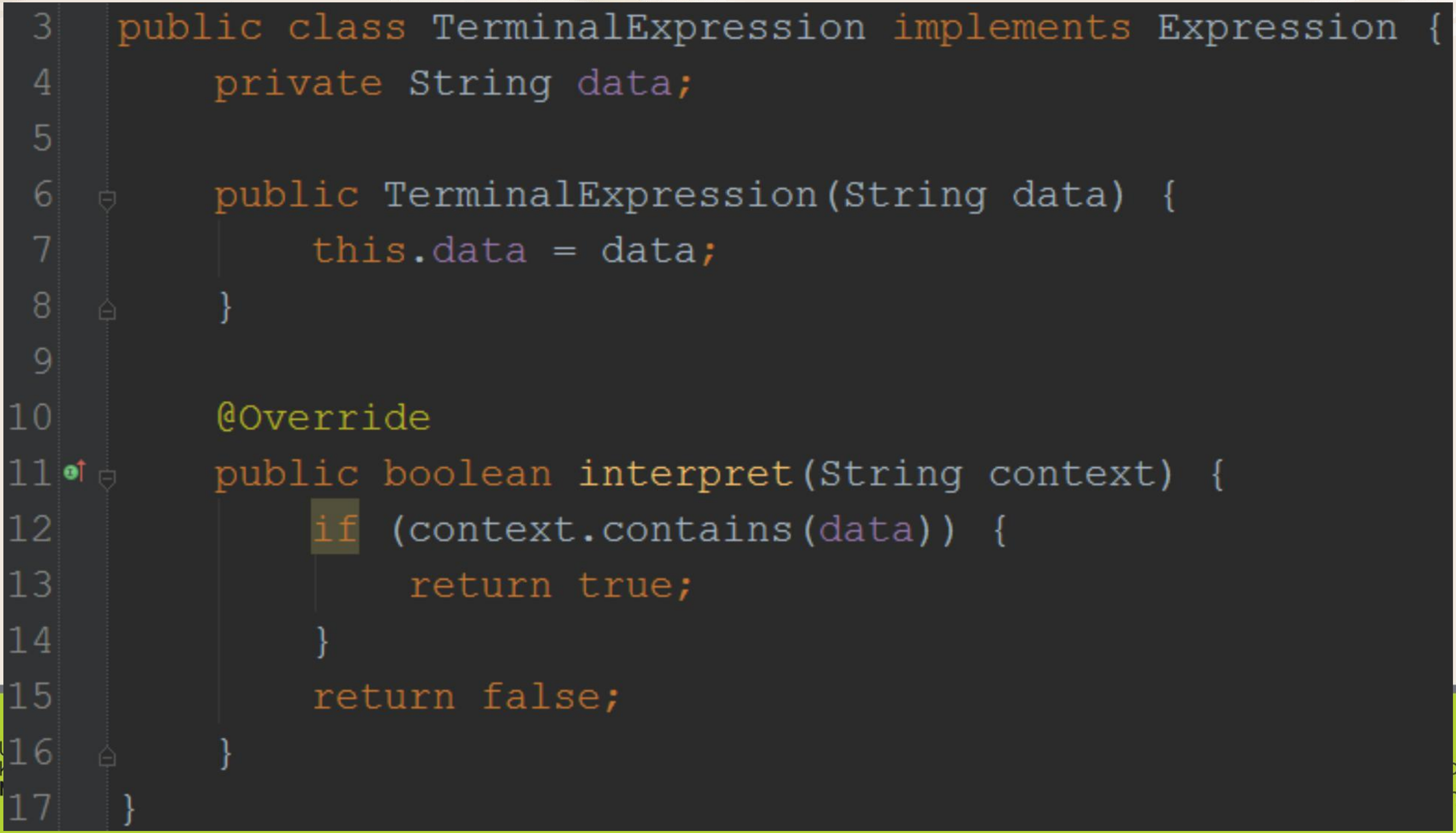
- Kita akan membuat interface Expression beserta kelas-kelas yang meng-implementnya.
- Kelas TerminalExpression berlaku sebagai interpreter utama untuk konteks yang sedang diproses.
- Kelas OrExpression dan AndExpression digunakan untuk membentuk kombinasi ekspresi.
- Kelas InterpreterPatternDemo menggunakan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

Contoh Interpreter Pattern



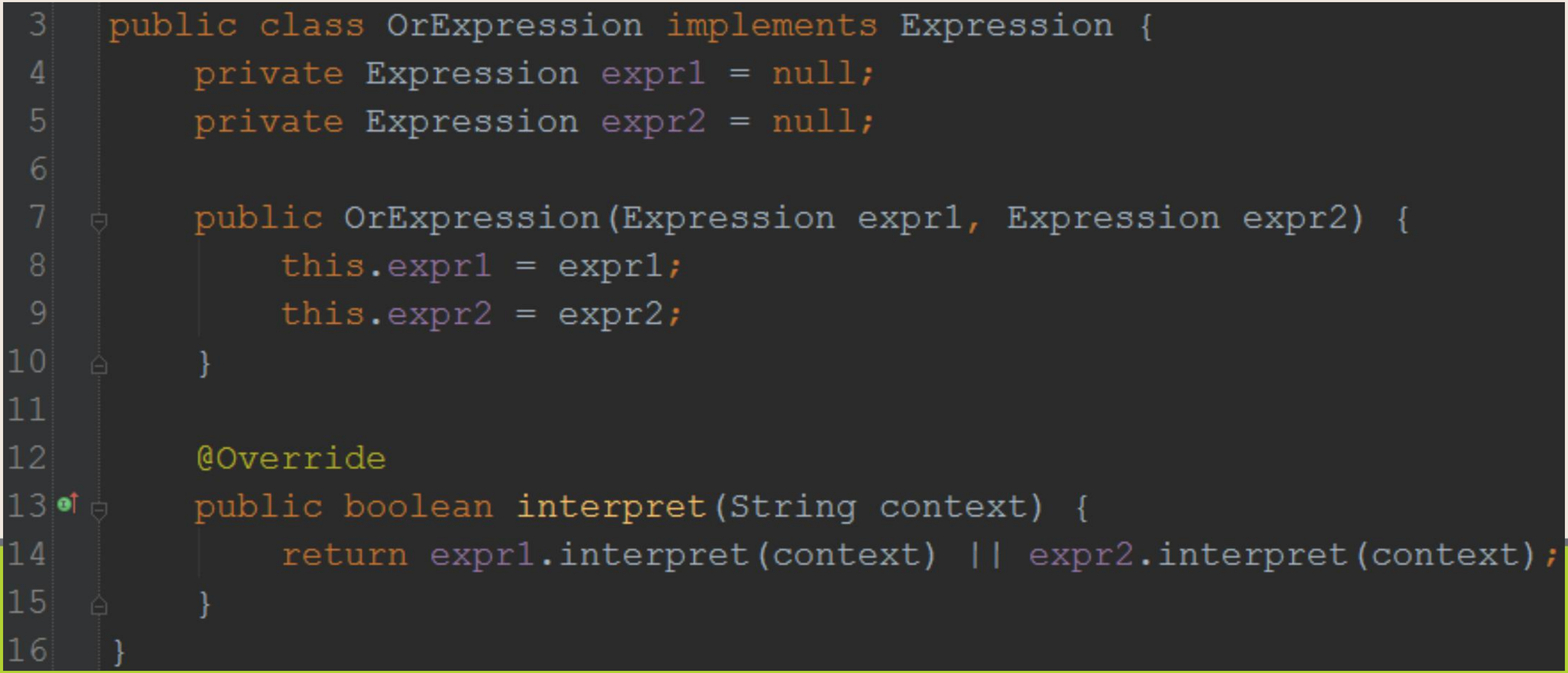
```
package edu.maranatha.pdpl;

public interface Expression {
    public boolean interpret(String context);
}
```

Contoh Interpreter Pattern



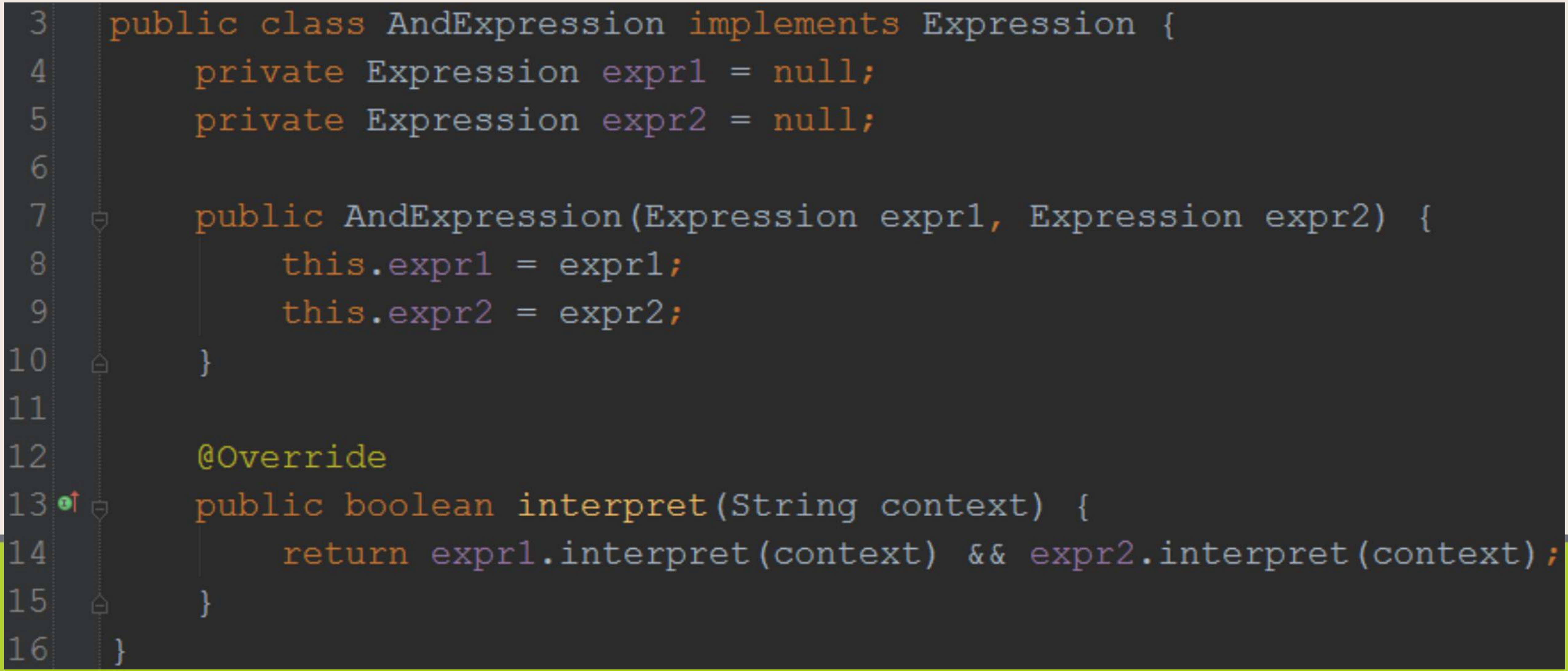
```java
public class TerminalExpression implements Expression {
    private String data;

    public TerminalExpression(String data) {
        this.data = data;
    }

    @Override
    public boolean interpret(String context) {
        if (context.contains(data)) {
            return true;
        }
        return false;
    }
}
```

Contoh Interpreter Pattern

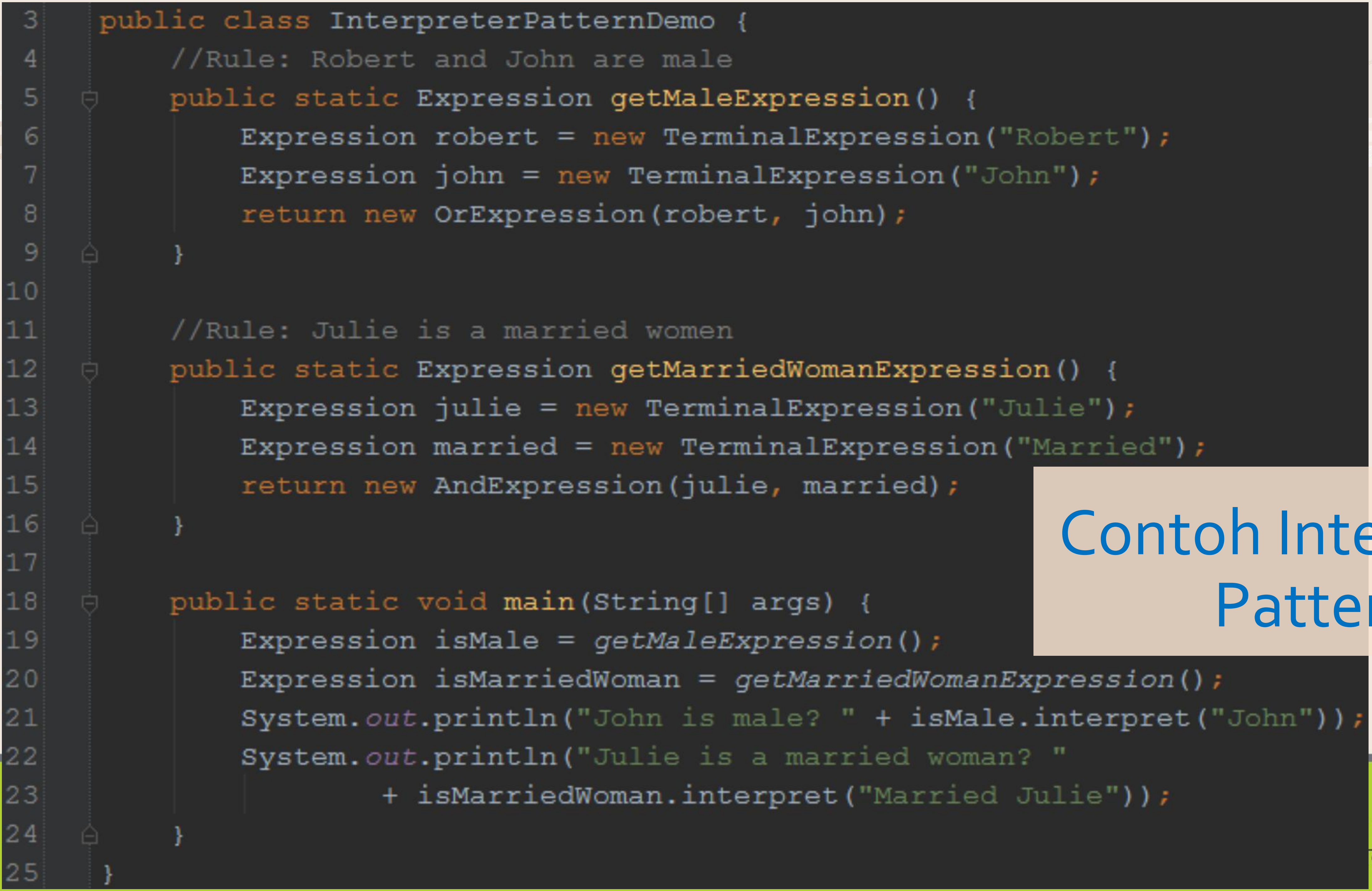


```java
public class OrExpression implements Expression {
    private Expression expr1 = null;
    private Expression expr2 = null;

    public OrExpression(Expression expr1, Expression expr2) {
        this.expr1 = expr1;
        this.expr2 = expr2;
    }

    @Override
    public boolean interpret(String context) {
        return expr1.interpret(context) || expr2.interpret(context);
    }
}
```

Contoh Interpreter Pattern



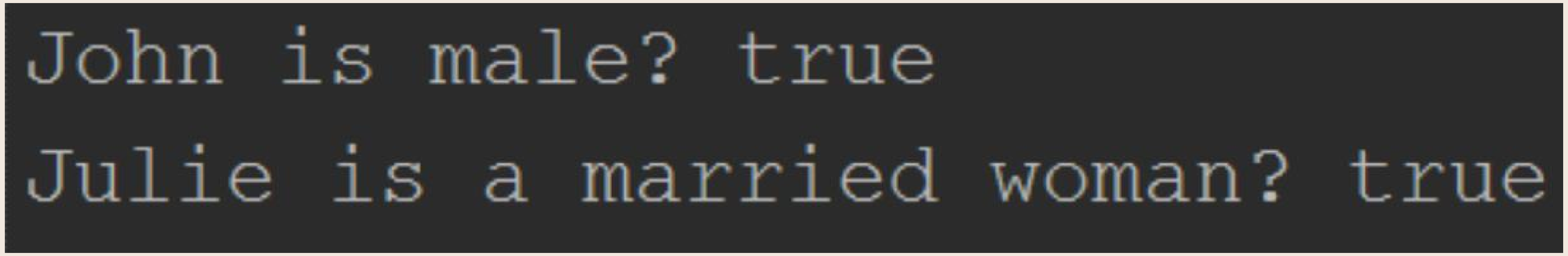
```java
public class AndExpression implements Expression {
    private Expression expr1 = null;
    private Expression expr2 = null;

    public AndExpression(Expression expr1, Expression expr2) {
        this.expr1 = expr1;
        this.expr2 = expr2;
    }

    @Override
    public boolean interpret(String context) {
        return expr1.interpret(context) && expr2.interpret(context);
    }
}
```



```java
public class InterpreterPatternDemo {
    //Rule: Robert and John are male
    public static Expression getMaleExpression() {
        Expression robert = new TerminalExpression("Robert");
        Expression john = new TerminalExpression("John");
        return new OrExpression(robert, john);
    }

    //Rule: Julie is a married women
    public static Expression getMarriedWomanExpression() {
        Expression julie = new TerminalExpression("Julie");
        Expression married = new TerminalExpression("Married");
        return new AndExpression(julie, married);
    }

    public static void main(String[] args) {
        Expression isMale = getMaleExpression();
        Expression isMarriedWoman = getMarriedWomanExpression();
        System.out.println("John is male? " + isMale.interpret("John"));
        System.out.println("Julie is a married woman? "
                + isMarriedWoman.interpret("Married Julie"));
    }
}
```

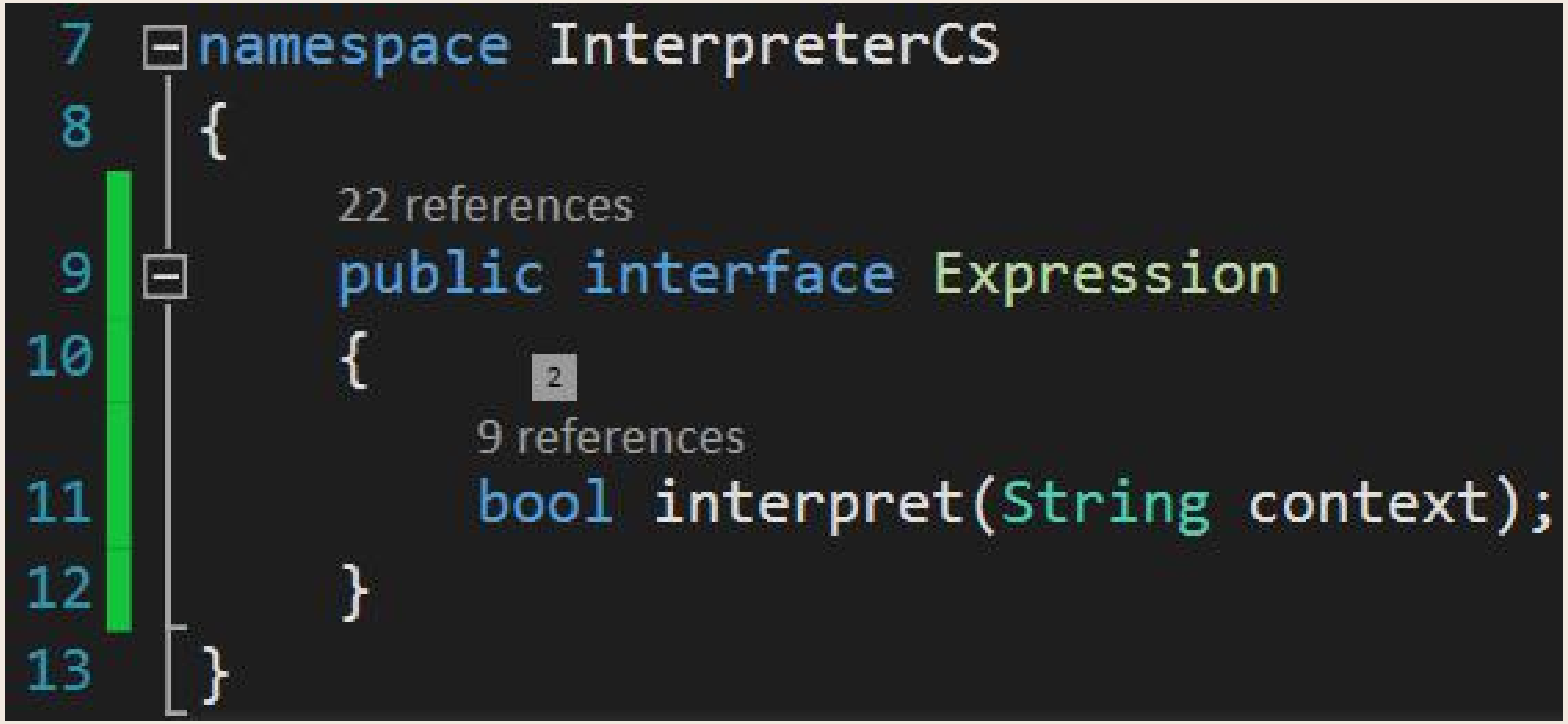
Contoh Interpreter Pattern

Contoh Interpreter Pattern



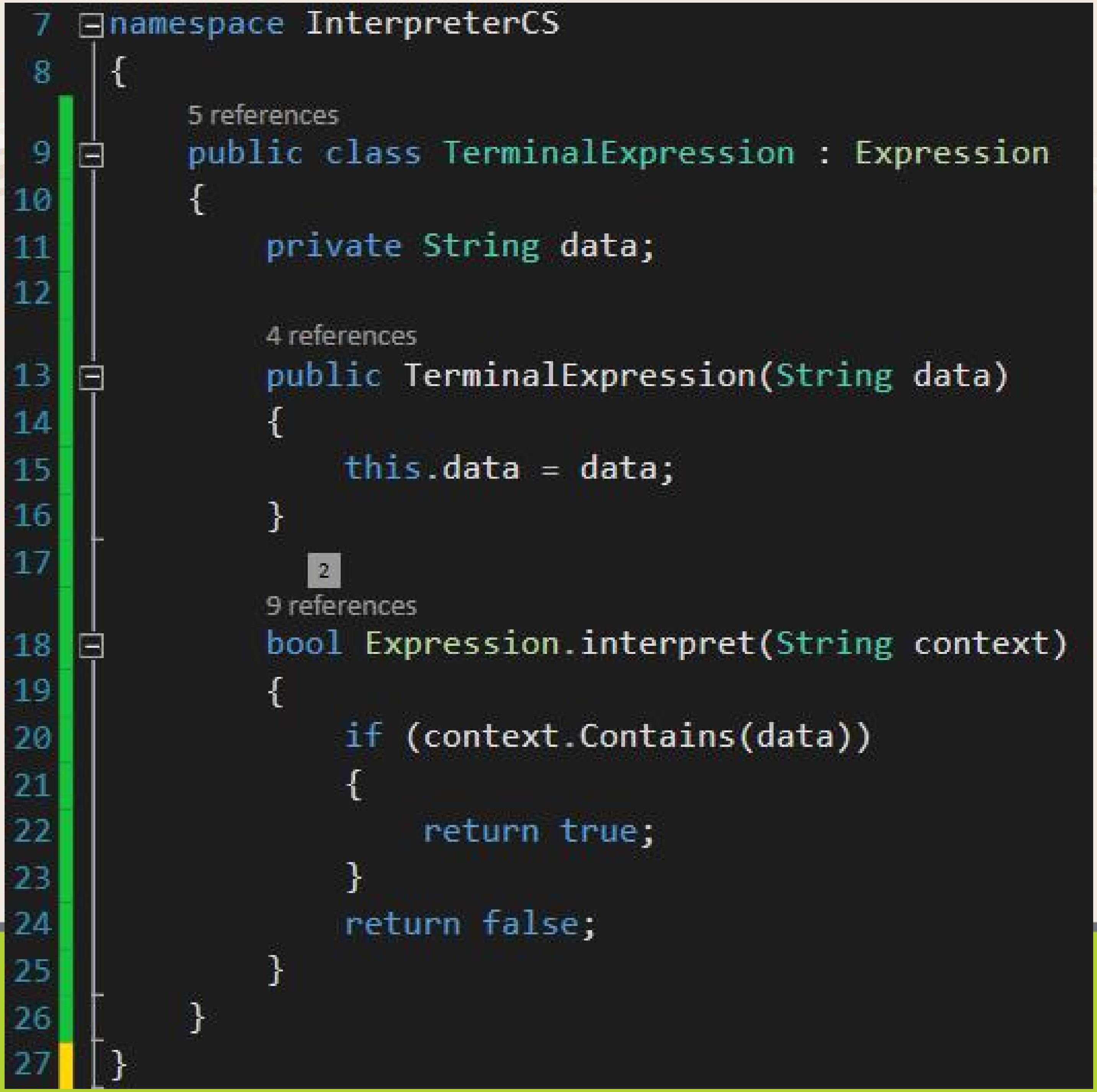
```
John is male? true
Julie is a married woman? true
```

Contoh Interpreter Pattern (C#)



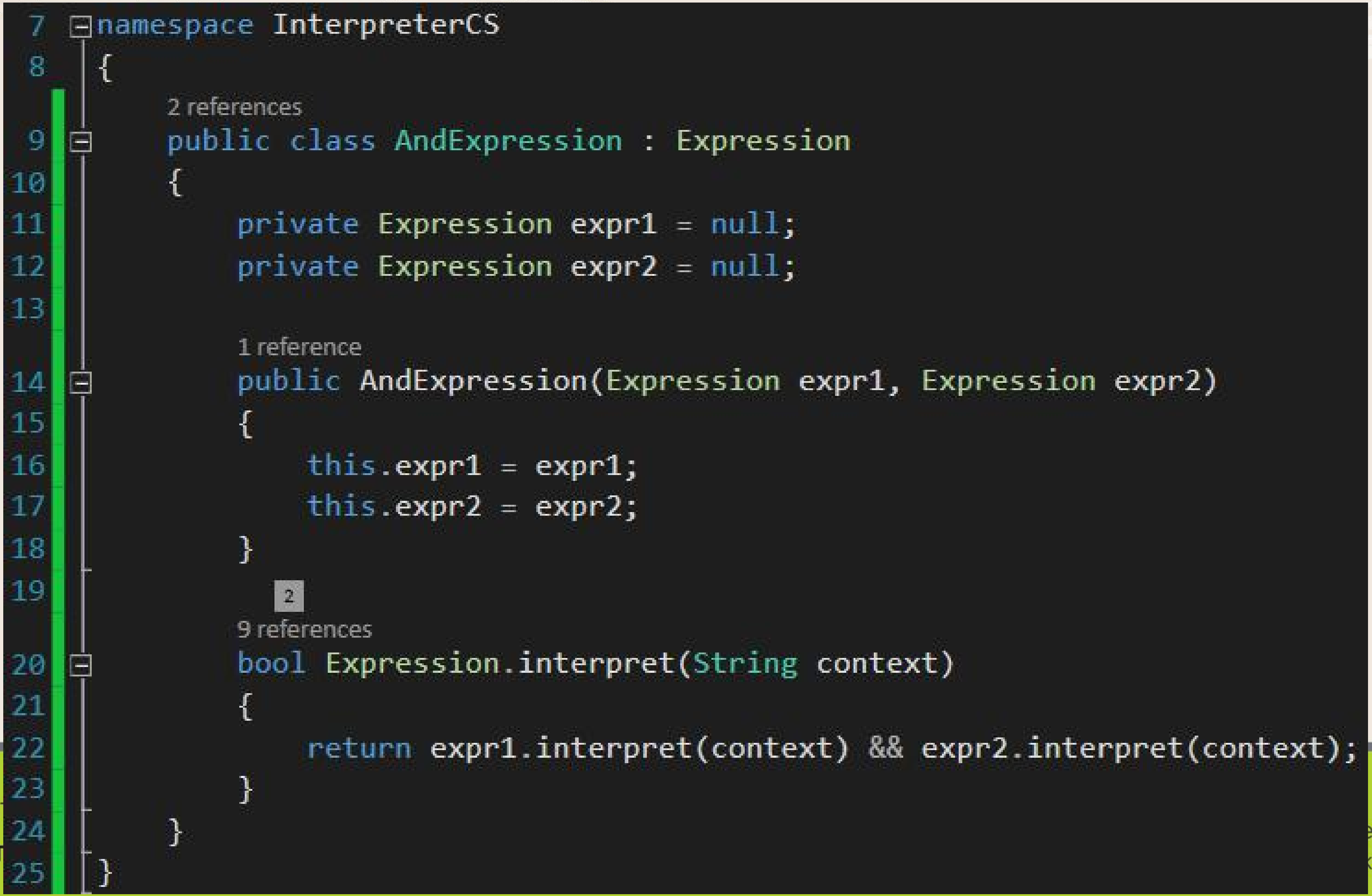
```
namespace InterpreterCS
{
        22 references
        public interface Expression
        {        2
            9 references
            bool interpret(String context);
        }
}
```

Contoh Interpreter Pattern (C#)



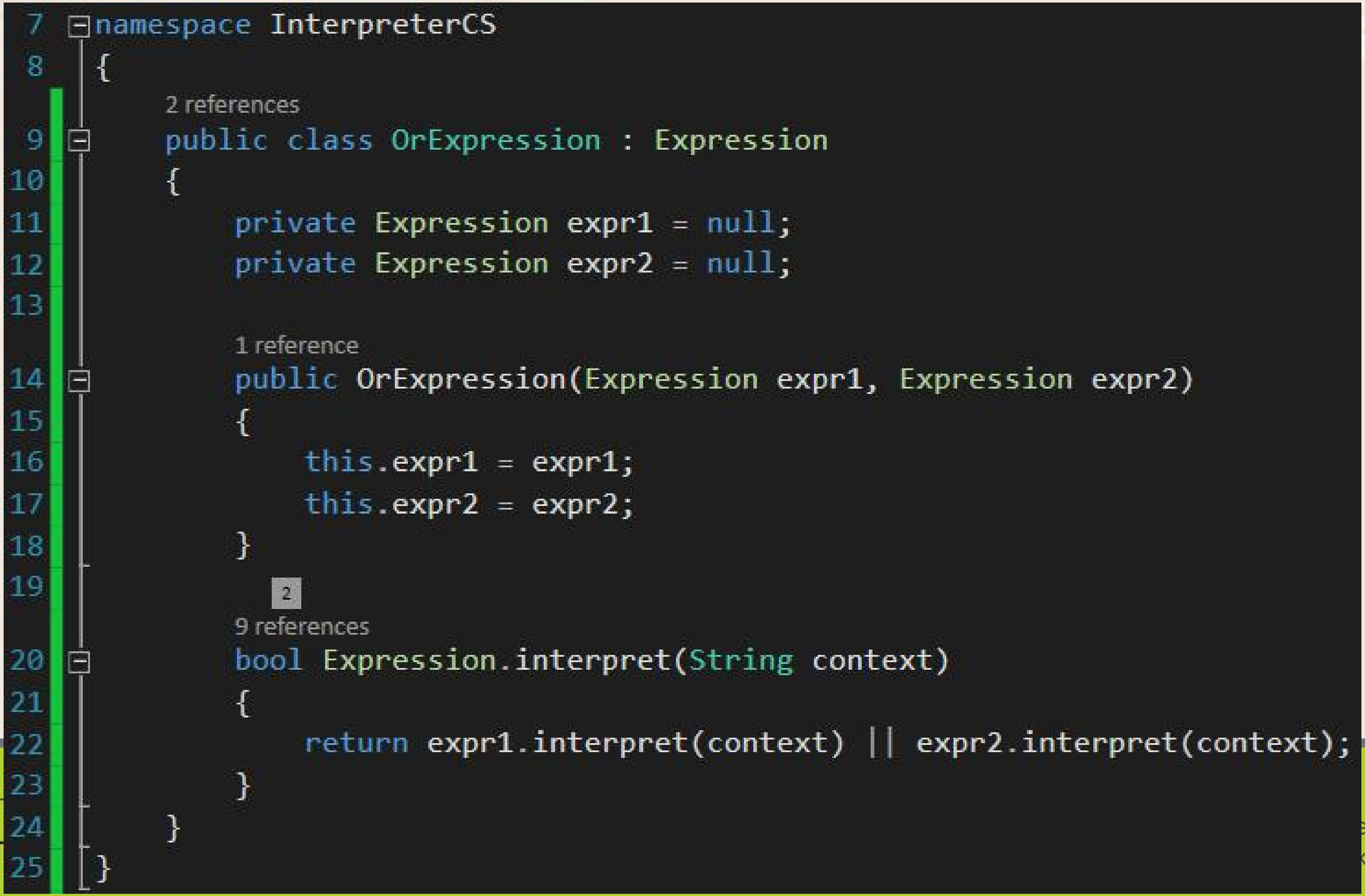
```
namespace InterpreterCS
{
        5 references
        public class TerminalExpression : Expression
        {
            private String data;

            4 references
            public TerminalExpression(String data)
            {
                this.data = data;
            }

            2
            9 references
            bool Expression.interpret(String context)
            {
                if (context.Contains(data))
                {
                    return true;
                }
                return false;
            }
        }
}
```

Contoh Interpreter Pattern (C#)



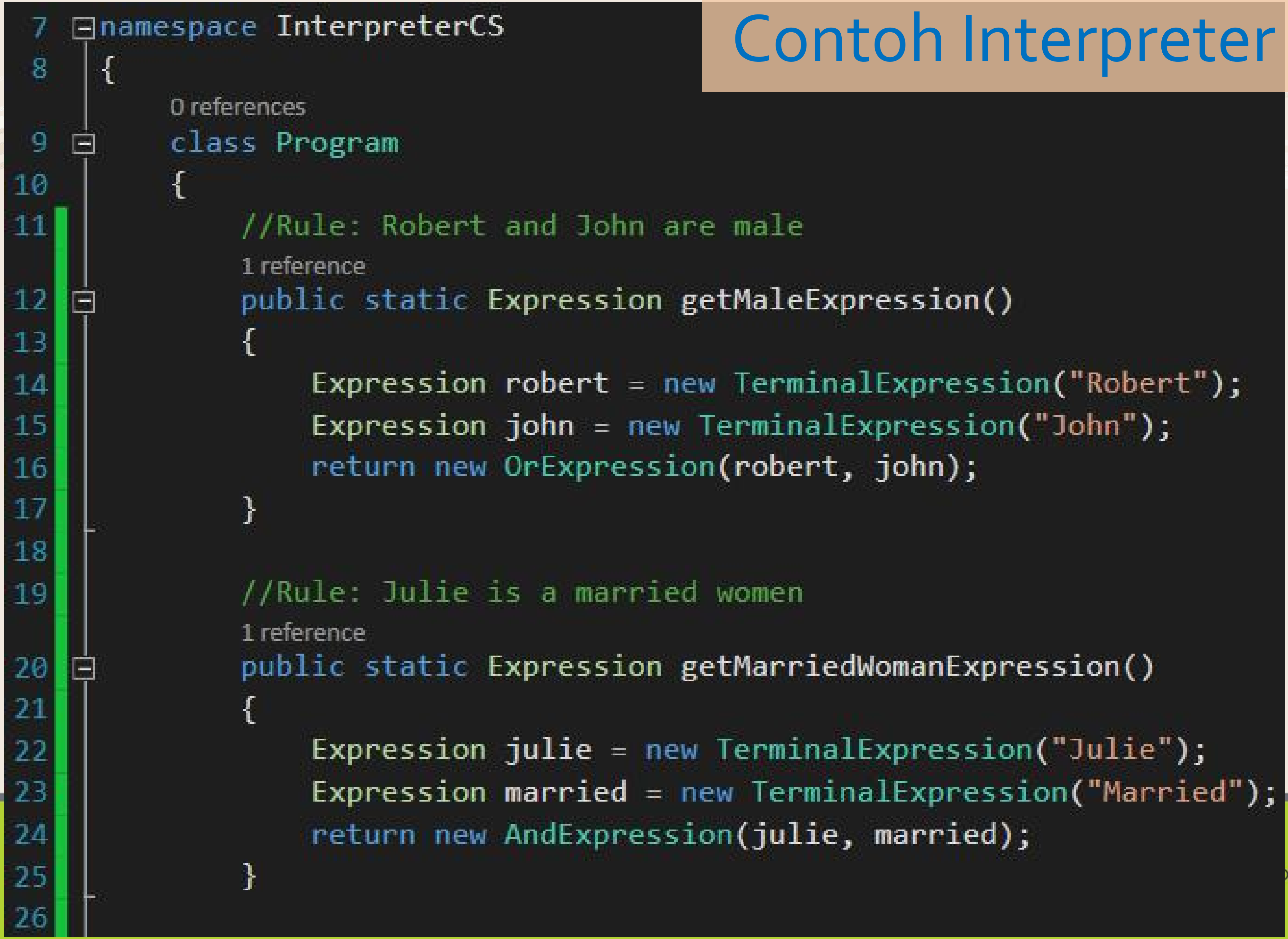
```
namespace InterpreterCS
{
    2 references
    public class AndExpression : Expression
    {
        private Expression expr1 = null;
        private Expression expr2 = null;

        1 reference
        public AndExpression(Expression expr1, Expression expr2)
        {
            this.expr1 = expr1;
            this.expr2 = expr2;
        }

          2
        9 references
        bool Expression.interpret(String context)
        {
            return expr1.interpret(context) && expr2.interpret(context);
        }
    }
}
```

Contoh Interpreter Pattern (C#)



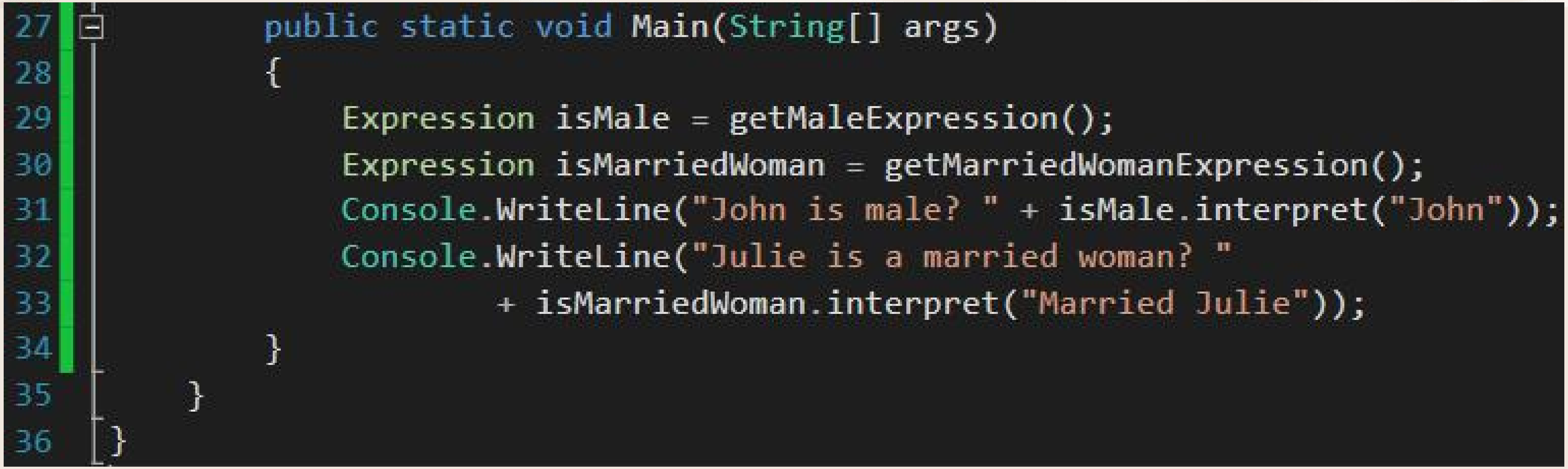
```
namespace InterpreterCS
{
        2 references
        public class OrExpression : Expression
        {
            private Expression expr1 = null;
            private Expression expr2 = null;

            1 reference
            public OrExpression(Expression expr1, Expression expr2)
            {
                this.expr1 = expr1;
                this.expr2 = expr2;
            }

              2
            9 references
            bool Expression.interpret(String context)
            {
                return expr1.interpret(context) || expr2.interpret(context);
            }
        }
}
```

Contoh Interpreter Pattern (C#)

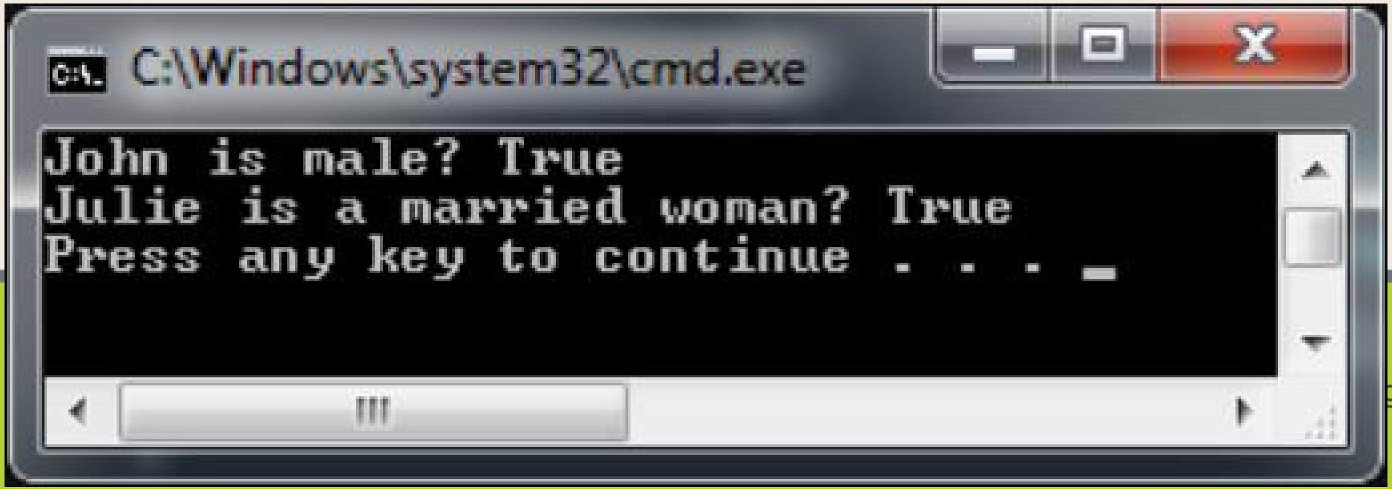


```
namespace InterpreterCS
{
        0 references
        class Program
        {
            //Rule: Robert and John are male
            1 reference
            public static Expression getMaleExpression()
            {
                Expression robert = new TerminalExpression("Robert");
                Expression john = new TerminalExpression("John");
                return new OrExpression(robert, john);
            }

            //Rule: Julie is a married women
            1 reference
            public static Expression getMarriedWomanExpression()
            {
                Expression julie = new TerminalExpression("Julie");
                Expression married = new TerminalExpression("Married");
                return new AndExpression(julie, married);
            }

```

Contoh Interpreter Pattern (C#)



```csharp
        public static void Main(String[] args)
        {
            Expression isMale = getMaleExpression();
            Expression isMarriedWoman = getMarriedWomanExpression();
            Console.WriteLine("John is male? " + isMale.interpret("John"));
            Console.WriteLine("Julie is a married woman? "
                    + isMarriedWoman.interpret("Married Julie"));
        }
    }
}
```



C:\Windows\system32\cmd.exe

```
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.