Interpreter Design Pattern in Java

APRIL 2, 2018 BY PANKAJ — 7 COMMENTS

Interpreter design pattern is one of the **behavioral design pattern**. Interpreter pattern is used to defines a grammatical representation for a language and provides an interpreter to deal with this grammar.

Interpreter Design Pattern

The best example of interpreter design pattern is java compiler that interprets the java source code into byte code that is understandable by JVM. Google Translator is also an example of interpreter pattern where the input can be in any language and we can get the output interpreted in another language.

Interpreter Pattern Example

To implement interpreter pattern, we need to create Interpreter context engine that will do the interpretation work.

Then we need to create different Expression implementations that will consume the functionalities provided by the interpreter context.

Finally we need to create the client that will take the input from user and decide which Expression to use and then generate output for the user.

Let's understand this with an example where the user input will be of two forms – "`<Number> in Binary`" or "`<Number> in Hexadecimal`." Our interpreter client should return it in format "`<Number> in Binary= <Number_Binary_String>`" and "`<Number> in Hexadecimal= <Number_Binary_String>`" respectively.

Our first step will be to write the Interpreter context class that will do the actual interpretation.

```
package com.journaldev.design.interpreter;

public class InterpreterContext {

        public String getBinaryFormat(int i){
                return Integer.toBinaryString(i);
        }

        public String getHexadecimalFormat(int i){
                return Integer.toHexString(i);
        }
}
```

Now we need to create different types of Expressions that will consume the interpreter context class.

```
package com.journaldev.design.interpreter;

public interface Expression {

        String interpret(InterpreterContext ic);
}
```

We will have two expression implementations, one to convert int to binary and other to convert int to hexadecimal format.

```
package com.journaldev.design.interpreter;

public class IntToBinaryExpression implements Expression {

        private int i;

        public IntToBinaryExpression(int c){
                this.i=c;
        }
        @Override
        public String interpret(InterpreterContext ic) {
                return ic.getBinaryFormat(this.i);
        }

}
```

```
package com.journaldev.design.interpreter;

public class IntToHexExpression implements Expression {

        private int i;

        public IntToHexExpression(int c){
                this.i=c;
        }

        @Override
        public String interpret(InterpreterContext ic) {
                return ic.getHexadecimalFormat(i);
        }

}
```

Now we can create our client application that will have the logic to parse the user input and pass it to correct expression and then use the output to generate the user response.

```
package com.journaldev.design.interpreter;

public class InterpreterClient {

        public InterpreterContext ic;

        public InterpreterClient(InterpreterContext i){
                this.ic=i;
        }

        public String interpret(String str){
                Expression exp = null;
                //create rules for expressions
                if(str.contains("Hexadecimal")){
                        exp=new
IntToHexExpression(Integer.parseInt(str.substring(0,str.indexOf(" "))));
```

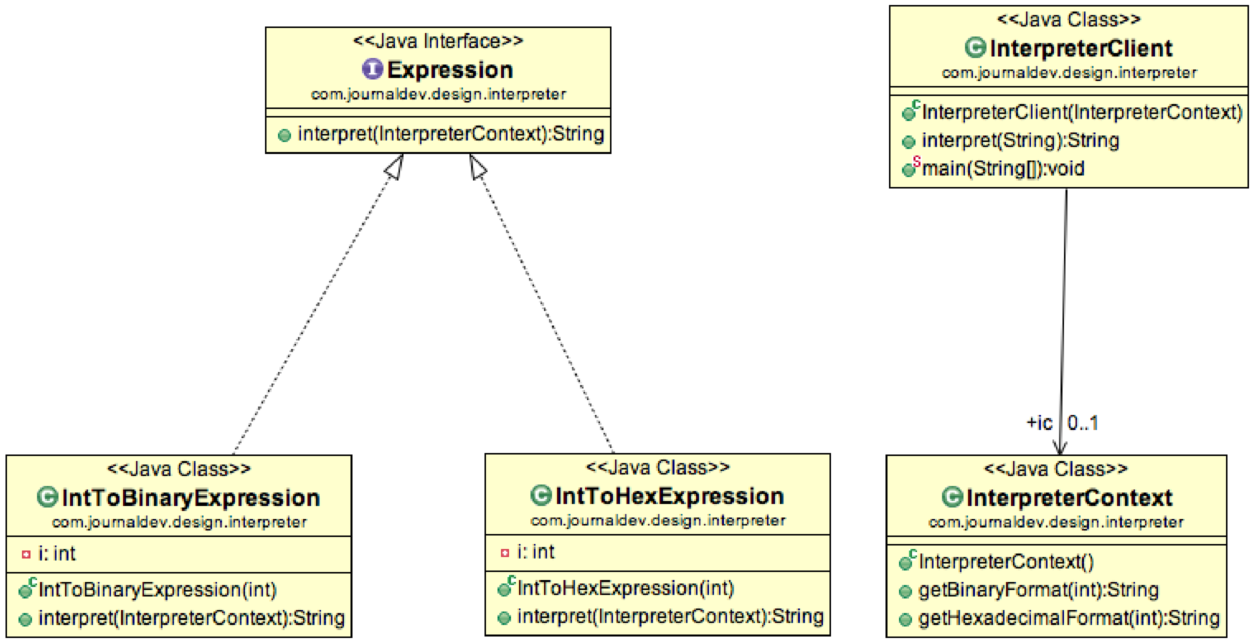
```
                }else if(str.contains("Binary")){
                        exp=new
IntToBinaryExpression(Integer.parseInt(str.substring(0,str.indexOf(" "))));
                }else return str;

                return exp.interpret(ic);
```

The client also has a main method for testing purpose, when we run above we get following output:

```
28 in Binary= 11100
28 in Hexadecimal= 1c
```

Interpreter Design Pattern Example Class Diagram



<<Java Interface>>
Expression
com.journaldev.design.interpreter
interpret(InterpreterContext):String
<<Java Class>>
InterpreterClient
com.journaldev.design.interpreter
InterpreterClient(InterpreterContext)
interpret(String):String
main(String[]):void
+ic 0..1
<<Java Class>>
IntToBinaryExpression
com.journaldev.design.interpreter
i: int
IntToBinaryExpression(int)
interpret(InterpreterContext):String
<<Java Class>>
IntToHexExpression
com.journaldev.design.interpreter
i: int
IntToHexExpression(int)
interpret(InterpreterContext):String
<<Java Class>>
InterpreterContext
com.journaldev.design.interpreter
InterpreterContext()
getBinaryFormat(int):String
getHexadecimalFormat(int):String

Important Points about Interpreter pattern

1. Interpreter pattern can be used when we can create a syntax tree for the grammar we have.
2. Interpreter design pattern requires a lot of error checking and a lot of expressions and code to evaluate them. It gets complicated when the grammar becomes more complicated and hence hard to maintain and provide efficiency.
3. `java.util.Pattern` and subclasses of `java.text.Format` are some of the examples of interpreter pattern used in JDK.

« PREVIOUS
Command Design Pattern

NEXT »
Iterator Design Pattern in Java

About Pankaj
If you have come this far, it means that you liked what you are reading. Why not reach little more and connect with me directly on **Google Plus**, **Facebook** or **Twitter**. I would love to hear your thoughts and opinions on my articles directly.

Recently I started creating video tutorials too, so do check out my videos on **Youtube**.

FILED UNDER: DESIGN PATTERNS

Comments

Nirav says
FEBRUARY 24, 2018 AT 8:19 AM

Why the InterpreterContext directly available to the client? Because of that Client can context without required to expression, isn't it?
Reply

Sajgoniarz says
AUGUST 25, 2017 AT 1:44 AM
It's not a Interpreter pattern,
Reply

eduardo milpas says
OCTOBER 24, 2016 AT 11:35 AM
What's the porpouse of having a context which stores the whole busssiness logic?
If the client calls directly the Expression(with the implemented logic inside) is even easier to use it as simple inheritance.
Honestly I don't see the porpuose of this pattern
Reply

Ajay says
MAY 24, 2016 AT 3:24 AM
Based on the the value of the input type, we are calling either Binary/Hexadecimal. Why can't this eg. belong to a factory design pattern ?
Reply

karteek says
DECEMBER 18, 2014 AT 9:44 AM
Why do we need a InterpreterContext in the client program ??
Reply

Martin says
MARCH 1, 2014 AT 9:26 PM
What you describe is the Strategy pattern Not the Interpreter pattern.
Reply

Pankaj says

MARCH 2, 2014 AT 3:32 AM

Strategy pattern allows client application to set the strategy but here it's done internally.

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

☐

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

Search for tutorials...

DOWNLOAD ANDROID APP

GET IT ON Google Play

- Java Generics
- Exception Handling in Java
- Java Reflection
- Java Design Patterns
- JDBC Tutorial

Java EE Tutorials

- Servlet JSP Tutorial
- Struts2 Tutorial
- Spring Tutorial
- Hibernate Tutorial
- Primefaces Tutorial
- Apache Axis 2
- JAX-RS
- Memcached Tutorial