**Interpreter Design Pattern**

Interpreter [design pattern](https://www.journaldev.com/1827/java-design-patterns-example-tutorial) 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](https://www.journaldev.com/546/difference-jdk-vs-jre-vs-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.

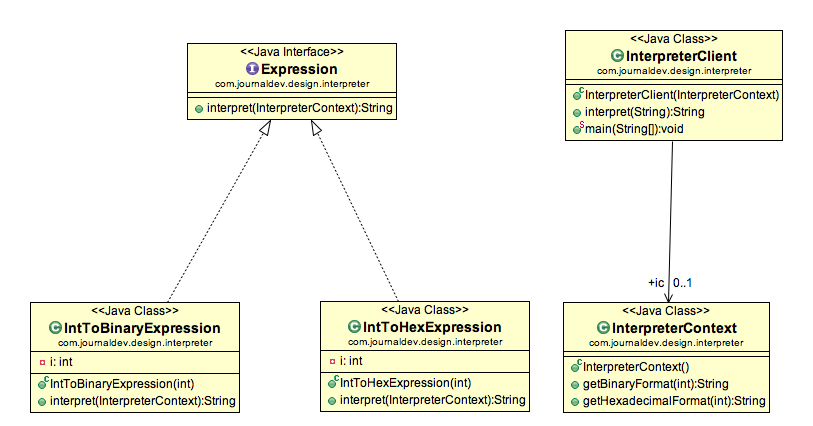
Our first step will be to write the Interpreter context class that will do the actual interpretation

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

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

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.

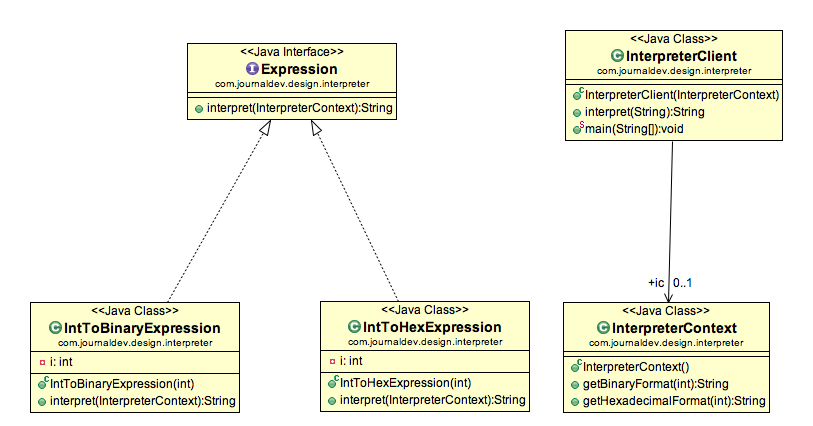
### Interpreter Design Pattern Example Class Diagram



### 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

### Interpreter Design Pattern Example Class Diagram



### 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.