# Package
# lib6005.parser

Parser library that creates a recursive-descent parser automatically from a grammar.

# lib6005.parser
# Class GrammarCompiler

```
java.lang.Object
   |
   +-lib6005.parser.GrammarCompiler
```

public class **GrammarCompiler**
extends java.lang.Object

Convert a grammar into a Parser.

The grammar of a grammar is as follows.

```
    @skip whitespaceAndComments {
     grammar ::= ( production | skipBlock )+
     production ::= nonterminal '::=' union ';'?
     skipBlock ::= '@skip' nonterminal '{' production* '}'
     union :: = concatenation ('|' concatenation)*
     concatenation ::= repetition+
     repetition ::= unit repeatOperator
     repeatOperator ::= [*+?]
     unit ::= nonterminal | terminal | '(' union ')'
   }
   nonterminal ::= [a-zA-Z_][a-zA-Z_0-9]*
   terminal ::= quotedString | characterSet | anyChar | characterClass
   quotedString ::= "'" ([^'\r\n\\] | '\\' . )+ "'"   // e.g. 'hello', '\'',  '\r\n\t'
                  | '"' ([^"\r\n\\] | '\\' . )+ '"'   // e.g. "world", "\"", "\r\n\t"
   characterSet ::= '[' ([^\]\r\n\\] | '\\' . )+ ']'   // e.g. [abc], [a-z], [^a-z], [\]],
[\r\n\t]
   anyChar ::= '.'
   characterClass ::= '\\' [dsw]     // e.g. \d, \s, \w
   whitespaceAndComments ::= (whitespace | oneLineComment | blockComment)*
   whitespace ::= [ \t\r\n]*
   oneLineComment ::= '//' [^\r\n]* [\r\n]+
   blockComment ::= '/*' [^*]* '*' ([^/]* '*')* '/'
```

## Constructor Summary

| | |
|---|---|
| public | GrammarCompiler() |

## Method Summary

| | |
|---|---|
| static Parser | compile(java.io.File in, java.lang.Enum rootNonterminal)<br>Method takes a file with a grammar description and produces a Parser for that grammar. |
| static Parser | compile(java.io.InputStream in, java.lang.Enum rootNonterminal)<br>Method takes an `java.io.InputStream` with a grammar description and produces a Parser for that grammar. |

| | |
|---|---|
| static Parser | compile(java.io.Reader in, java.lang.Enum rootNonterminal)<br>Method takes an java.io.InputStream with a grammar description and produces a Parser for that grammar. |
| static Parser | compile(java.lang.String grammar, java.lang.Enum rootNonterminal)<br>Method takes a String with a grammar description and produces a Parser for that grammar. |

| Methods inherited from class java.lang.Object |
|---|
| equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

# Constructors

## GrammarCompiler

```
public GrammarCompiler()
```

# Methods

## compile

```
public static Parser compile(java.lang.String grammar,
                             java.lang.Enum rootNonterminal)
  throws UnableToParseException
```

Method takes a String with a grammar description and produces a Parser for that grammar.

**Parameters:**

grammar - should be String containing a valid grammar.
rootNonterminal - This is a value in the Enum that corresponds to the root symbol in the grammar.

**Returns:**

A parser for the given grammar.

**Throws:**

UnableToParseException - Thrown if the grammar file does not contain a valid grammar.

## compile

```
public static Parser compile(java.io.Reader in,
                             java.lang.Enum rootNonterminal)
  throws UnableToParseException,
         java.io.IOException
```

Method takes an java.io.InputStream with a grammar description and produces a Parser for that grammar.

**Parameters:**

in - should be an java.io.InputStream containing a valid grammar.
rootNonterminal - This is a value in the Enum that corresponds to the root symbol in the grammar.

**Returns:**

A parser for the given grammar.

**Throws:**

UnableToParseException - if the grammar file does not contain a valid grammar.

(continued on next page)

(continued from last page)

`IOException` - if the `java.io.InputStream` throws an exception.

---

## compile

```
public static Parser compile(java.io.File in,
                             java.lang.Enum rootNonterminal)
  throws UnableToParseException,
         java.io.IOException
```

Method takes a file with a grammar description and produces a Parser for that grammar.

**Parameters:**

`in` - should be File containing a valid grammar.
`rootNonterminal` - This is a value in the Enum that corresponds to the root symbol in the grammar.

**Returns:**

A parser for the given grammar.

**Throws:**

`UnableToParseException` - Thrown if the grammar file does not contain a valid grammar.
`IOException` - Thrown if the grammar file cannot be opened.

---

## compile

```
public static Parser compile(java.io.InputStream in,
                             java.lang.Enum rootNonterminal)
  throws UnableToParseException,
         java.io.IOException
```

Method takes an `java.io.InputStream` with a grammar description and produces a Parser for that grammar.

**Parameters:**

`in` - should be an `java.io.InputStream` containing a valid grammar.
`rootNonterminal` - This is a value in the Enum that corresponds to the root symbol in the grammar.

**Returns:**

A parser for the given grammar.

**Throws:**

`UnableToParseException` - Thrown if the grammar file does not contain a valid grammar.
`IOException` - Thrown if the `java.io.InputStream` throws an exception.

# lib6005.parser
# Class Parser

```
java.lang.Object
   |
   +-lib6005.parser.Parser
```

public class **Parser**
extends java.lang.Object

Parser has an internal representation of a grammar and contains a parse method that takes in a String (or an InputStream or a Reader) and produces a ParseTree representing that string.

## Field Summary

| | |
|---|---|
| public static final | VERSION<br>Value: **1.2** |

## Constructor Summary

| | |
|---|---|
| public | Parser(java.util.Map grammar, java.lang.Enum start) |
| public | Parser(<any> gc, java.lang.Enum start) |

## Method Summary

| | |
|---|---|
| ParseTree | parse(java.io.File f)<br>Parses a string based on the grammar internally represented by the parser. |
| ParseTree | parse(java.io.InputStream stream)<br>Parses a string based on the grammar internally represented by the parser. |
| ParseTree | parse(java.io.Reader in)<br>Parses a string based on the grammar internally represented by the parser. |
| ParseTree | parse(java.lang.String textToParse)<br>Parses a string based on the grammar internally represented by the parser. |
| java.lang.String | toString() |

**Methods inherited from class** `java.lang.Object`

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

## Fields

### VERSION

```
public static final java.lang.String VERSION
```

(continued from last page)

Constant value: **`1.2`**

# Constructors

## Parser

```
public Parser(java.util.Map grammar,
              java.lang.Enum start)
```

---

## Parser

```
public Parser(<any> gc,
              java.lang.Enum start)
```

# Methods

## parse

```
public ParseTree parse(java.io.InputStream stream)
  throws UnableToParseException,
         java.io.IOException
```

Parses a string based on the grammar internally represented by the parser.

**Parameters:**

`stream` - InputStream from which to extract the text to be parsed.

**Returns:**

`ParseTree` representing the string.

**Throws:**

`UnableToParseException` - If the string cannot be parsed, this class will throw an UnableToParseException that describes approximately where the parsing error occurred.
`IOException` - if the `java.io.InputStream` throws an exception.

---

## parse

```
public ParseTree parse(java.io.Reader in)
  throws UnableToParseException,
         java.io.IOException
```

Parses a string based on the grammar internally represented by the parser.

**Parameters:**

`in` - Reader from which to extract the text to be parsed.

**Returns:**

`ParseTree` representing the string.

**Throws:**

`UnableToParseException` - If the string cannot be parsed, this class will throw an UnableToParseException that describes approximately where the parsing error occurred.
`IOException` - if the `java.io.Reader` throws an exception.

---

(continued from last page)

## parse

```
public ParseTree parse(java.io.File f)
  throws UnableToParseException,
         java.io.IOException
```

Parses a string based on the grammar internally represented by the parser.

**Parameters:**

`f` - File containing the text to be parsed.

**Returns:**

`ParseTree` representing the string.

**Throws:**

`UnableToParseException` - If the string cannot be parsed, this class will throw an UnableToParseException that describes approximately where the parsing error occurred.
`IOException` - if the `java.io.File` throws an exception.

---

## parse

```
public ParseTree parse(java.lang.String textToParse)
  throws UnableToParseException
```

Parses a string based on the grammar internally represented by the parser.

**Parameters:**

`textToParse` - String to be parsed

**Returns:**

`ParseTree` representing the string.

**Throws:**

`UnableToParseException` - If the string cannot be parsed, this class will throw an UnableToParseException that describes approximately where the parsing error occurred.

---

## toString

```
public java.lang.String toString()
```

# lib6005.parser
# Class ParseTree

```
java.lang.Object
   |
   +-lib6005.parser.ParseTree
```

**All Implemented Interfaces:**
java.lang.Iterable

---

public class **ParseTree**
extends java.lang.Object
implements java.lang.Iterable

This data structure represents a parse tree in a way that allows for easy traversal through all its different nodes. You will generally want to translate this data-structure right away into a data-structure that is tailored to your application.
**Parameters:**
`Symbols` - Is an Enum type of all the non-terminals in the parse tree.

---

## Constructor Summary

| | |
|---|---|
| public | `ParseTree(java.lang.String contents, java.util.Map children, java.util.List inOrderChildren)` |
| public | `ParseTree(java.lang.Enum nonterminal, java.lang.String contents, java.util.Map children, java.util.List inOrderChildren)` |
| public | `ParseTree(java.lang.String contents)` |

## Method Summary

| | |
|---|---|
| boolean | `assignedNT()` |
| java.util.List | `children()`<br>Ordered list of all the children nodes of this ParseTree node. |
| java.util.List | `childrenByName(java.lang.Enum name)`<br>Get all the children of this PareseTree node corresponding to a particular production rule |
| static ParseTree | `concat(ParseTree p1, ParseTree p2)` |
| void | `display()`<br>This method attempts to show you a visualization of the tree in your bowser. |
| void | `displayToFile(java.lang.String filename)`<br>Generate an HTML visualization of the parse tree and write it to a file named 'filename' |
| ParseTree | `filter(java.lang.Enum toExclude)`<br>Filter away the symbol toExclude from the parseTree. |
| java.lang.String | `getContents()`<br>Returns the substring of the original string that corresponds to this parse tree. |

| | |
|---|---|
| java.lang.Enum | getName()<br>Get the symbol for the terminal or non-terminal corresponding to this parse tree. |
| boolean | isTerminal()<br>Tells you whether a node corresponds to a terminal or a non-terminal. |
| java.util.Iterator | iterator() |
| void | setName(java.lang.Enum nonterminal)<br>Set the name of the parse tree. |
| java.lang.String | toString() |

| Methods inherited from class java.lang.Object |
|---|
| equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

| Methods inherited from interface java.lang.Iterable |
|---|
| forEach, iterator, spliterator |

# Constructors

## ParseTree

```
public ParseTree(java.lang.String contents,
                 java.util.Map children,
                 java.util.List inOrderChildren)
```

## ParseTree

```
public ParseTree(java.lang.Enum nonterminal,
                 java.lang.String contents,
                 java.util.Map children,
                 java.util.List inOrderChildren)
```

## ParseTree

```
public ParseTree(java.lang.String contents)
```

# Methods

## filter

```
public ParseTree filter(java.lang.Enum toExclude)
```

Filter away the symbol toExclude from the parseTree. Returns a brand new parseTree.

**Parameters:**

`toExclude` - should not be equal to the name of the current tree.

(continued from last page)

**Returns:**
new ParseTree that does not include the symbol to exclude.

## assignedNT

```
public boolean assignedNT()
```

## concat

```
public static ParseTree concat(ParseTree p1,
         ParseTree p2)
```

## isTerminal

```
public boolean isTerminal()
```

Tells you whether a node corresponds to a terminal or a non-terminal. If it is terminal, it won't have any children.

**Returns:**
true if it is a terminal value.

## children

```
public java.util.List children()
```

Ordered list of all the children nodes of this ParseTree node.

**Returns:**
a List of all children of this ParseTree node, ordered by position in input

## iterator

```
public java.util.Iterator iterator()
```

## childrenByName

```
public java.util.List childrenByName(java.lang.Enum name)
```

Get all the children of this PareseTree node corresponding to a particular production rule

**Parameters:**
`name` - Name of the non-terminal corresponding to the desired production rule.

**Returns:**
List of children ParseTree objects that match that name.

## getContents

```
public java.lang.String getContents()
```

Returns the substring of the original string that corresponds to this parse tree.

(continued from last page)

**Returns:**

String containing the contents of this parse tree.

## toString

```
public java.lang.String toString()
```

## displayToFile

```
public void displayToFile(java.lang.String filename)
  throws java.io.IOException
```

Generate an HTML visualization of the parse tree and write it to a file named 'filename'

**Parameters:**

`filename` - name of the output file. It is advisable that this be an html file.

**Throws:**

`IOException` - If it is not able to write to 'filename'.

## display

```
public void display()
```

This method attempts to show you a visualization of the tree in your bowser. If you are not connected to the internet, or if the method cannot launch the browser, it will print a very long URL to your console which you can then copy to your browser to see the visualization.

## getName

```
public java.lang.Enum getName()
```

Get the symbol for the terminal or non-terminal corresponding to this parse tree.

**Returns:**

T will generally be an Enum representing the different symbols in the grammar, so the return value will be one of those.

## setName

```
public void setName(java.lang.Enum nonterminal)
```

Set the name of the parse tree. You can only set the name if the name was previously unassigned. If it has already been set, you cannot change it.

**Parameters:**

`nonterminal` - nonterminal that matches this subtree of the parse tree

# lib6005.parser
# Class UnableToParseException

```
java.lang.Object
   |
   +-java.lang.Throwable
      |
      +-java.lang.Exception
         |
         +-lib6005.parser.UnableToParseException
```

**All Implemented Interfaces:**
java.io.Serializable

---

public class **UnableToParseException**
extends java.lang.Exception

---

## Constructor Summary

| | |
|---|---|
| public | UnableToParseException(java.lang.String message) |

**Methods inherited from class** `java.lang.Throwable`

`addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

**Methods inherited from class** `java.lang.Object`

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

---

## Constructors

### UnableToParseException

`public` **`UnableToParseException`**`(java.lang.String message)`