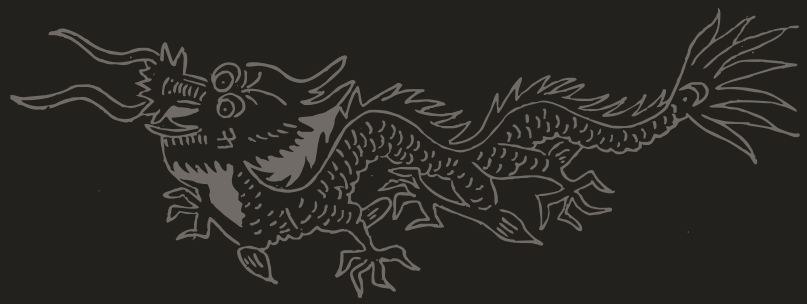
JavaCC in practice

Krzysztof Kaszkowiak

February 20, 2014



Agenda

- Introduction
  - What is JavaCC?
  - Why JavaCC?
  - Lexer & parser
- Case study
  - Problem description
  - Possible solution
  - Better approach
- JavaCC syntax - .jj files

Introduction

What is JavaCC?

What is JavaCC?

Java Compiler Compiler - The Java Parser Generator

What is JavaCC?

Java Compiler Compiler - The Java Parser Generator

- reads a grammar specification and converts it to a Java code that can recognize the matches to the grammar

What is JavaCC?

Java Compiler Compiler - The Java Parser Generator

- reads a grammar specification and converts it to a Java code that can recognize the matches to the grammar
- most popular parser for use with Java applications

What is JavaCC?

Java Compiler Compiler - The Java Parser Generator

- reads a grammar specification and converts it to a Java code that can recognize the matches to the grammar
- most popular parser for use with Java applications
- C++ code can be generated too!

Why JavaCC?

Java Compiler Compiler - what you get

Java Compiler Compiler - what you get

- Define a grammar in easy way...

Java Compiler Compiler - what you get

- Define a grammar in easy way…
- …and do with it whatever you want…

Java Compiler Compiler - what you get

- Define a grammar in easy way…
- …and do with it whatever you want…
- …generated code is certified 100% pure Java.

Java Compiler Compiler - what you get

- Define a grammar in easy way…
- …and do with it whatever you want…
- …generated code is certified 100% pure Java.
- Generated application have excelent error reporting.

Java Compiler Compiler - what you get

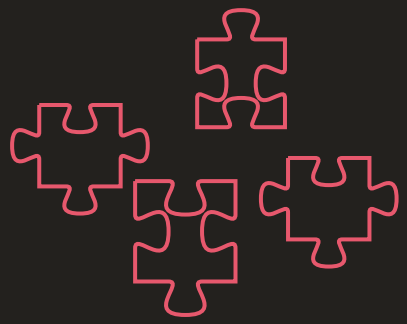
- Define a grammar in easy way…
- …and do with it whatever you want…
- …generated code is certified 100% pure Java.
- Generated application have excelent error reporting.
- Good performance for the most of grammars.

Lexer & parser

TEXT -->

TEXT --> LEXER

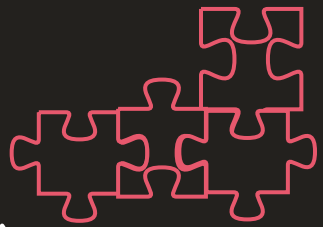
Lexer & parser



TEXT --> LEXER -- (TOKENS)

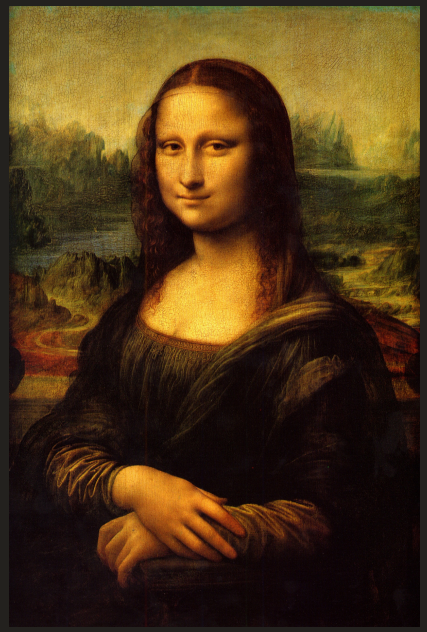
Lexer & parser

TEXT --> LEXER -- (TOKENS) --> PARSER

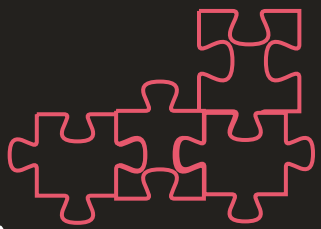


Lexer & parser

Lexer & parser



GRAMMAR



TEXT --> LEXER -- (TOKENS) --> PARSER

Case study

Problem description - DFM file

```
object Form2: TForm1
  Width = 696
  Height = 480
  Caption = 'Notatnik+'
  object Memo1: TMemo
    Left = 0
    Top = 0
    Font.Name = 'Comic Sans MS'
    Font.Style = [fsBold]
    ParentFont = False
    OnChange = Memo1Change
  end
  object StatusBar1: TStatusBar
    Left = 0
    Top = 407
    Panels = <>
  end
  object MainMenu1: TMainMenu
    Left = 32
    Top = 72
    object Plik1: TMenuItem
      Caption = '&Plik'
      object Nowy1: TMenuItem
        Caption = '&Nowy'
        OnClick = Nowy1Click
      end
      object Otwrz1: TMenuItem
        Caption = '&Otw'#243'rz...'
        OnClick = Otwrz1Click
        DesignSize = (
          562
          493)
      end
    end
    object Ustawienia1: TMenuItem
      Caption = '&Ustawienia'
      object Czcionka1: TMenuItem
        Caption = 'Czcionka...'
        OnClick = Czcionka1Click
      end
    end
end
```

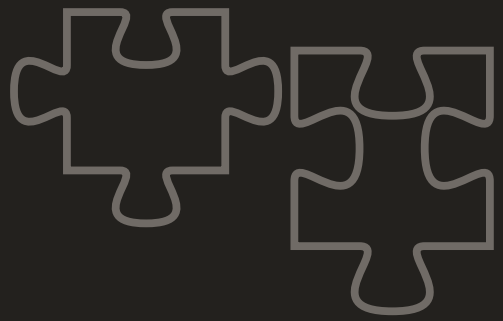
Lexer

```
MORE : { < "'" >  : IN_STRING_SQ }

<IN_STRING_SQ> MORE : { < "\\" ["'", "\\"] >  }
<IN_STRING_SQ> MORE : { < ~["'"] >  }
<IN_STRING_SQ> TOKEN : { < STRING_SQ : "'" >  : DEFAULT }

MORE : { < "(" >  : IN_BRACKETS }
<IN_BRACKETS> MORE : { < ~[")"] >  }
<IN_BRACKETS> TOKEN : { < BRACKETS : ")" >  : DEFAULT }

MORE : { < "[" >  : IN_KW_BRACKETS }
<IN_KW_BRACKETS> MORE : { < ~["]"] >  }
<IN_KW_BRACKETS> TOKEN : { < KW_BRACKETS : "]" >  : DEFAULT }

SKIP :
{
    <WHITESPACE :  ([" ", "\t"])+>
}

TOKEN:
{
        <OBJECT: "object">
    |   <END: "end">
    |   <COLON: ":">
    |   <ENDLINE: ["\r", "\n"])+>
    |   <EQUAL: "=">
}

TOKEN:
{
    <IDENTIFIER : ["A"-"Z", "0"-"9", "a"-"z", "_", ".", "-", "#"])+ >
}
```

Parser

```
void parse() throws ParseException: { }
{
    dfmObject()
    ( <ENDLINE> )*
    <EOF>
}

void string() throws ParseException: { }
{
  <STRING_SQ>
}

void dfmObject() throws ParseException: { }
{
    <OBJECT>
        identifier() <COLON> type() <ENDLINE>
        (     (dfmObject() <ENDLINE>)
            | property()
        )+
    <END>
}

void identifier() throws ParseException: { }
{
   <IDENTIFIER>
}

void type() throws ParseException: { }
{
   <IDENTIFIER>
}
```

```
void property() throws ParseException: { }
{
    identifier() <EQUAL>
    (  string()
     | propValue()
    )+
    <ENDLINE>
}

void propValue() throws ParseException: { }
{
     identifier()  |  <BRACKETS> | <KW_BRACKETS>
}
```



Parser

```
void parse() throws ParseException: { }
{
    dfmObject()
    ( <ENDLINE> )*
    <EOF>
}
```

Parser

```
void parse() throws ParseException: { }
{
    dfmObject()
    ( <ENDLINE> )*
    <EOF>
}


DfmObject parse() throws ParseException:
{
    DfmObject res;
}
{
    res=dfmObject()
    ( <ENDLINE> )*
    <EOF>
    { return res; }
}
```

Parser

```
void dfmObject() throws ParseException: { }
{
    <OBJECT>
        identifier() <COLON> type() <ENDLINE>
        (     (dfmObject() <ENDLINE>)
            | property()
        )+
    <END>
}


private DfmObject dfmObject() throws ParseException:
{
    DfmObject res = new DfmObject();
    DfmProperty prop;
    DfmObject obj;
    String name;
    String type;
}
{
    <OBJECT>
        name=identifier() { res.setName( name ); }
        <COLON>
        type=type() { res.setType( type ); }
    <ENDLINE>
        (     (obj=dfmObject()   { res.addChild(obj);     } <ENDLINE>)
            | (( prop=property() { res.addProperty(prop); }) )
        )+
    <END>
    { return res; }
}
```

JavaCC syntax

JavaCC syntax - .jj files

```
options {
  JAVA_UNICODE_ESCAPE = true;
  ERROR_REPORTING = true;
  STATIC = false;
  JDK_VERSION = "1.5";
  BUILD_PARSER = true;
  IGNORE_CASE = true;
}

PARSER_BEGIN(DfmParser)

package eu.kaszkowiak.jdfm.parser;

import java.io.Reader;
import java.io.StringReader;
import eu.kaszkowiak.jdfm.model.*;

public class DfmParser {

     public DfmParser(String source) {
        this((Reader)(new StringReader(source)));
    }
}
PARSER_END(DfmParser)


 ( lexer )

 ( parser )
```

<- - - - OPTIONAL

Thank you! / Questions?