Music Xml to Braille Compiler

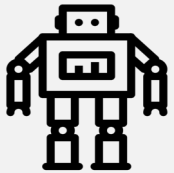
<Scanning & Parsing by JAVA CC>

Ahn Hyung Geun & Seo Han Seoung

INDEX

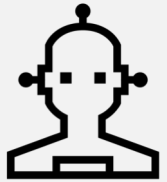
1

Jsoup
Token & Parse



2

JAVA CC
Output Braille



3

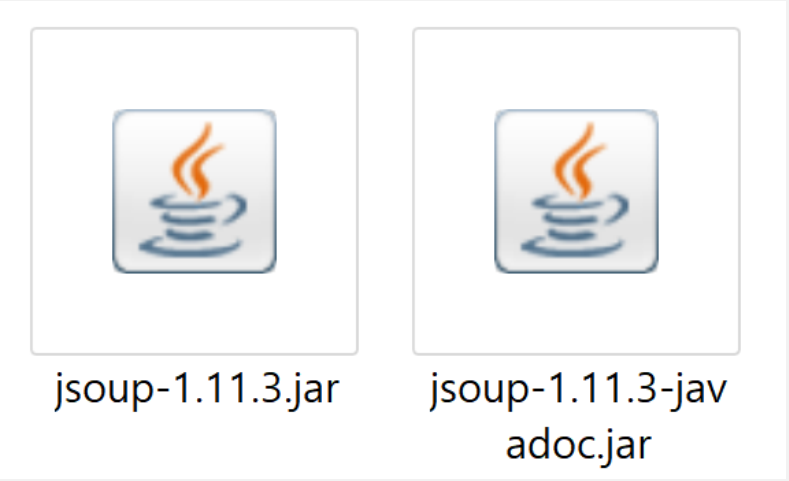
Compiler
Trial and Error

4

Learning in
This Course

Jsoup Library – contain Parser & Token

✓ **Jsoup.jar**



Open Source "Jsoup" – XML Controler

Finally, we found the Open source related with XML in JAVA

But, some MXML tags contains attributes, name of tags are changed

After tokenizing & parsing, deal with be used to j-soup library

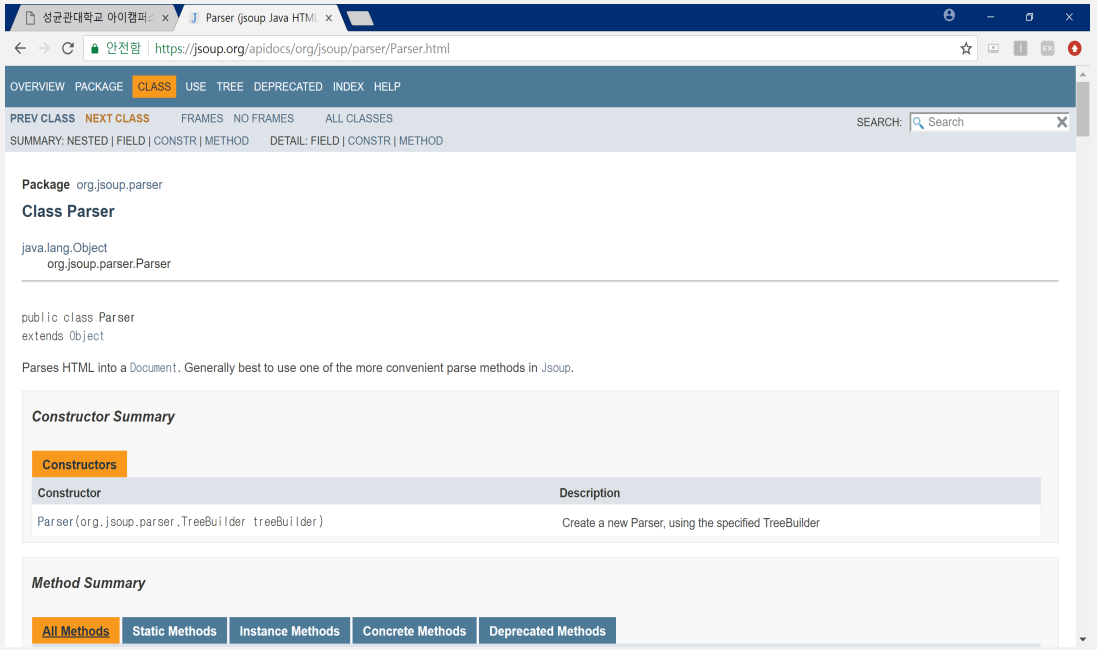
Jsoup Library – contain Parser & Token

✓ Input XML file in JAVA

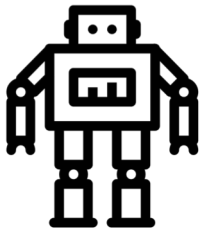
```java
package Jsoup;

import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import java.io.*;

public class Test{
    public static void main(String[] args) throws Exception{
        File input = new File(
                "C:/Users/user/Desktop/다운로드/Debussy_-_Deux_arabesques/lg-20474636
        Document doc = Jsoup.parse(input, "UTF-8");

        System.out.println(doc);
    }
}
```

Make JAVA can read XML(UTF-8) File by upper code

✓ Jsoup function – Detail Info

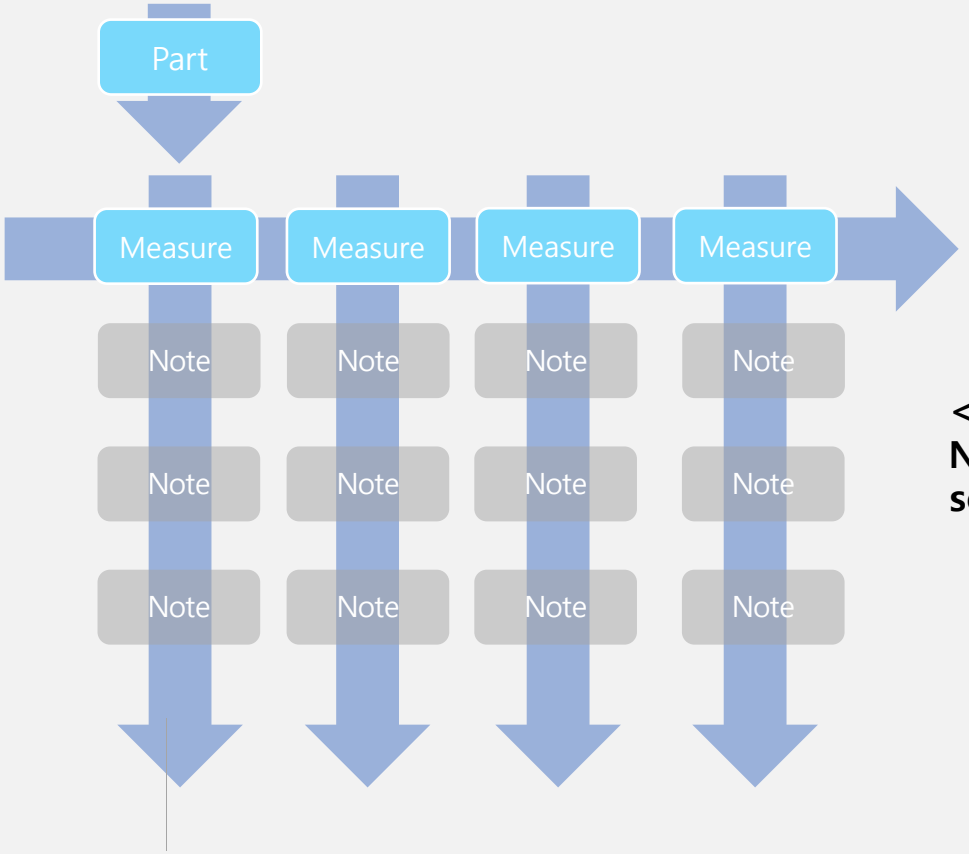


https://jsoup.org/apidocs/org/jsoup/parser/Parser.html



Simple Output of Braille Music Code - Text

✓ **Tree Structure**

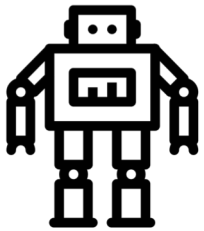


<-
Note contains some value
->

✓ **Can Recognize Measure**

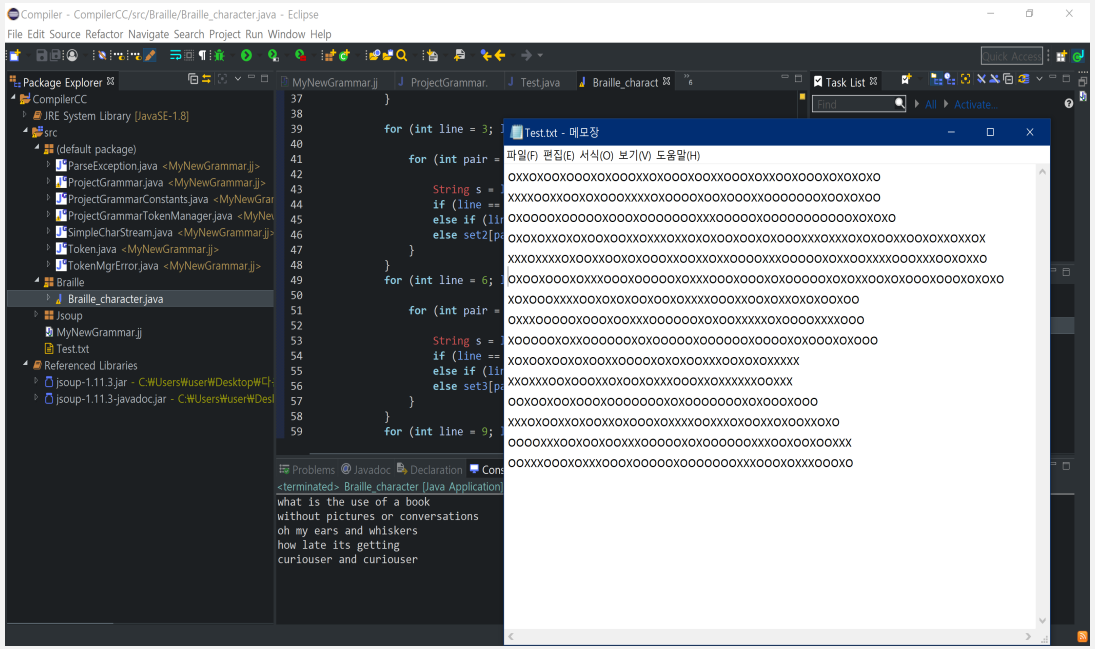
```
                                        <octave>
                                            4
                                        </octave>
                                </pitch>
                                <duration>
                                        2
                                </duration>
                                <type>
                                        eighth
                                </type>
                        </note>
                </measure>
</part>
measure 1
        note 1
                step: D
                octave: 3
                duration: 12
                type: half
measure 2
        note 1
                step: C
                octave: 4
                duration: 6
                type: quarter
        note 2
                step: A
                octave: 4
                duration: 2
                type: eighth

end
```

Output of our parser (Input: XML file)



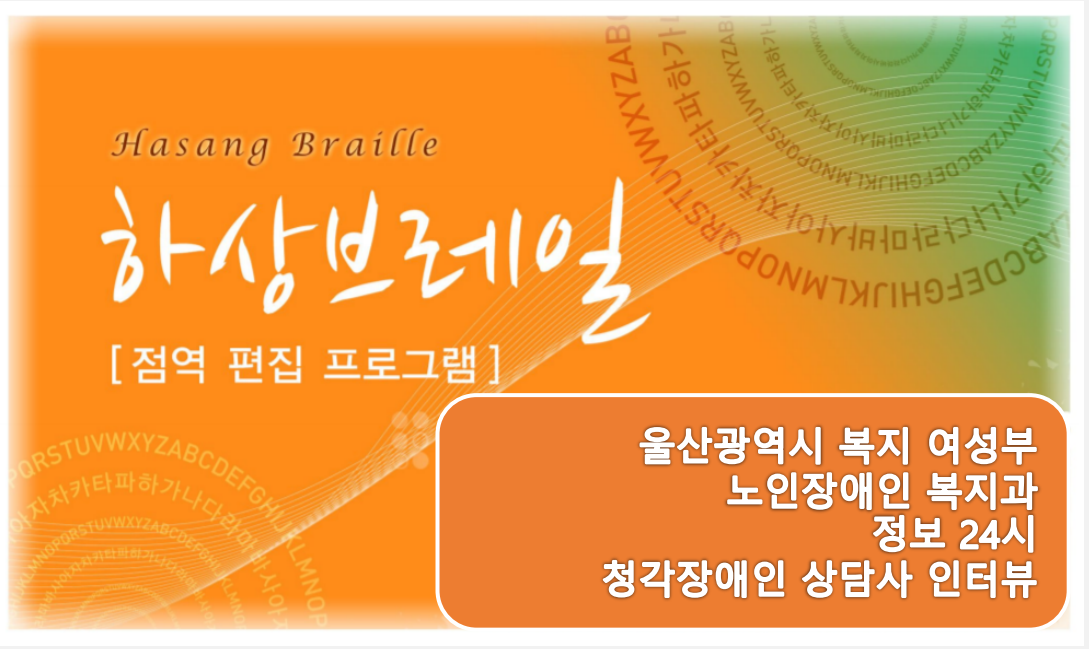
Simple Output of Braille Music Code - Text

✓ **Output Braille code in JAVA**

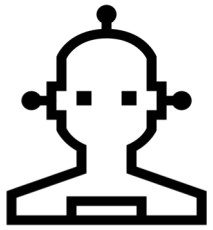


Input Character -> Output Braille Code

✓ **Interview with Consultant**

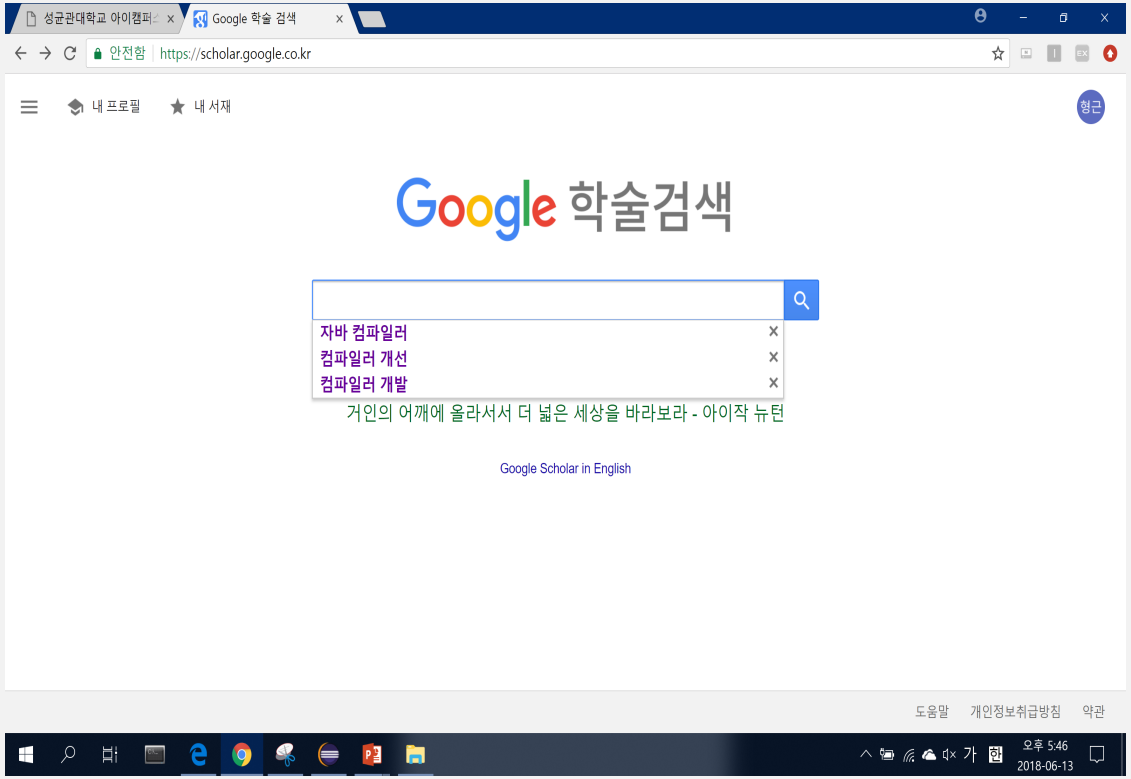


Interview Hearing impairment Consultant



The reference we read for Compiler project

✓ Google Scholar for Compiler

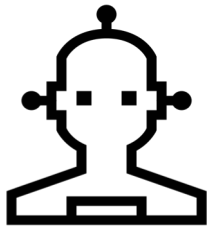


✓ Interest paper and so on

개방 시스템 응용을 위한 **개선**된 ASN. 1 **컴파일러** 설계 및 구현 (The Design and Implementation of an Enhanced ASN. 1 Compiler for Open System Application)
김홍렬 , 임제탁 - 전자공학회논문지-A, 1996 - dbpia.co.kr
Abstract Abstract Syntax Notation One (ASN. 1), defined by ITU-T and ISO), is a formal abstract specification language which has been widely used in international standards specification to inteconnect distributed open systems. It is necessary to have well defined …

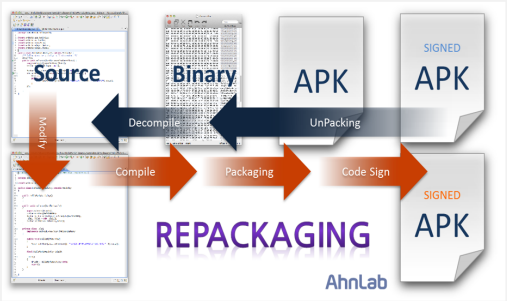
VHDL **컴파일러**-**컴파일러** 시스템에 관한 연구 (A study on VHDL compiler-compiler system)
공진흥 , 박기영 - 대한전자공학회 학술대회, 1994 - dbpia.co.kr
… 표준화된 VHDL 언어는 주기적으로 갱신되는 데, 이에 따른 VHDL 지원 시스템의 신속하고도 효과적인 **개선** 필요성을 **컴파일러** … 실험은 VHDL **컴파일러**의 전반부와 후반부를 나누어서 실행하였다. IW.1 VHDL, **컴파일**레 전반부 실험 생성된 VHDL **컴파일러** 전반부는 VHDL LRM"에 …
☆ 99 1회 인용 관련 학술자료 전체 4개의 버전

C-언어를 사용한 CMOS 셀 **컴파일러**의 개발 (Development of CMOS Cell Compiler Using C-Language)
김용주 , 오창준 , 이철동 - 대한전자공학회 학술대회, 1985 - dbpia.co.kr
… 둘째, 각 실들의 고정된 사양으로 인해 필요에 따라 그 사양을 변경하거나 **개선**하는대에 많은 … 2. CMOS 셀 **컴파일러** 설계의 기본개념 및 과정 (1) 기본기념 셀 **컴파일러**는 컴퓨터 언어로 … 종래의 표준 셈은 셀 라이브러리의 대이다. 베이스가 고정되어 있는 반면 셀 **컴파일**버는 컴퓨터 …



Our team project started with 4 IDEATE

✓ **Decompiler For Hack**



Extracting Source Code from .exe or Android App

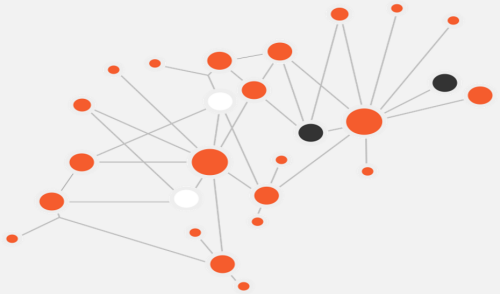
8

Obfuscating Code by add trash token at tokenizing, and trash node at Parsing level.

Make lots of tokenizing, parsing and address instruction algorithm.

3 tokenizing * 4 parsing * 3 address instruction = 36 way to decompiling

✓ **SourceCode = Variables Interact**



SourceCode seems like Society of many Variables.

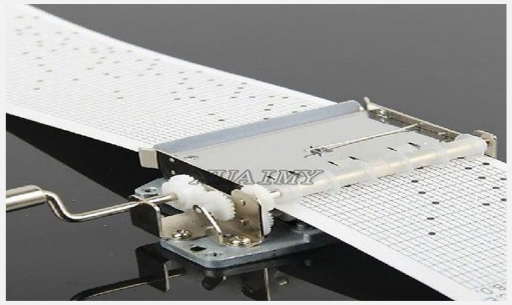
9

Point of programing is Variables. Method and function are just interactive.

Tokenizing string as Operand and Operator. Parsing by Tree Structure.

Starting Point = Design Variable by Class(object)

✓ **Make PunchCard For Orgel**



At Fortran, PunchCard is used for SourceCode.

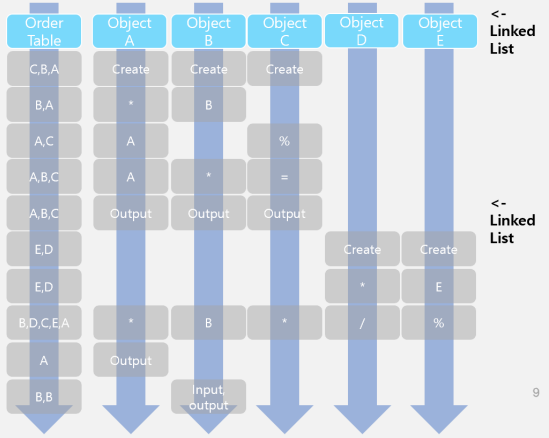
9

Punch card is used for source code + Some Orgels use punch card as sheet music

Then, lets compiling source code(sheet music) to punch card for orgel.

Lexical = note. Parsing = HashCode for implementing Harmony.

✓ **Tree Structure**

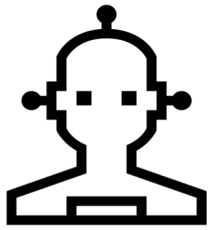


8

✓ **Source Code**

```
Public static main(string arg){
1.      object A, B, C= new object (string );

2.      A.*=(B);
3.      C.%=(A);
4.      C.=(B.*(A));

5.      A.output(); B.output(); C.output();

6.      object D, E = new object (string );

7.      D.*=(E);
8.      A.*=(E.%(C.*(D./(B))));

9.      A.output();
10.     B.output(B.input());

}
```

9

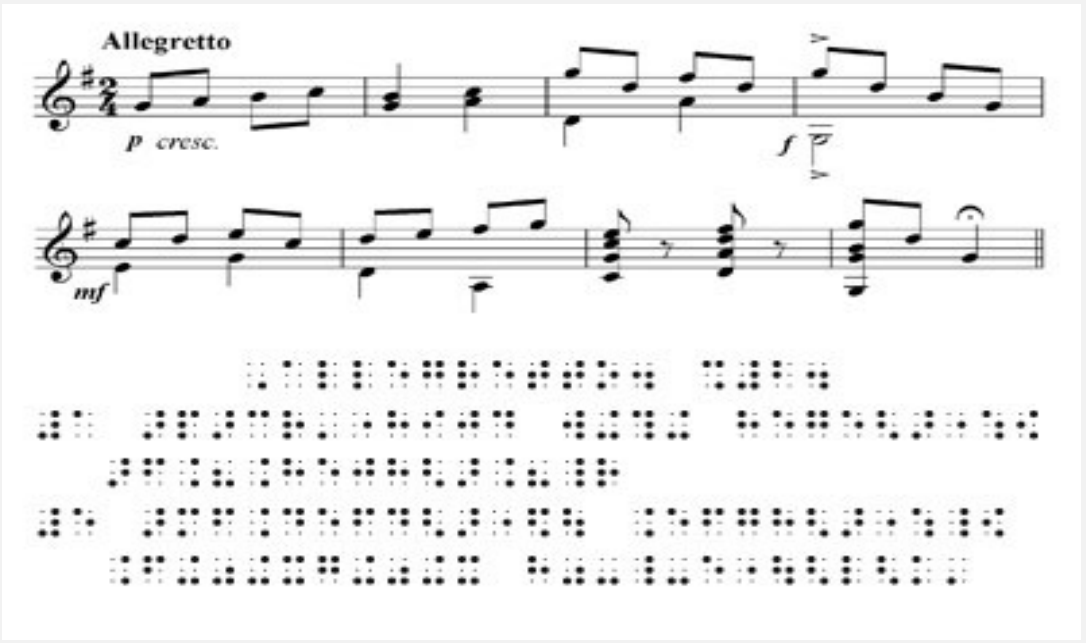


Our team project started with 4 IDEATE

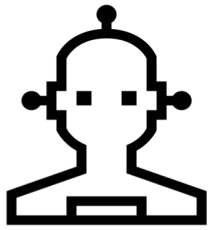
✓ **Just For Translate One Melody**

Not for Translate Harmony Music.

✓ **Helpful for Coversation**



Down to Upper code or Upper to Down code.



Draft Tokenizing by java cc & Revision of Tokenizing

✓ Attributes in Token

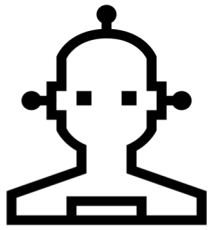
```
- <note default-y="-180.00" default-x="119.00">
    + <pitch>
      <duration>2</duration>
      <voice>1</voice>
      <type>eighth</type>
    - <time-modification>
          <actual-notes>3</actual-notes>
          <normal-notes>2</normal-notes>
      </time-modification>
      <stem>up</stem>
      <staff>2</staff>
      <beam number="1">begin</beam>
    - <notations>
          <tuplet type="start" show-number="none" bracket="no"/>
          <slur type="start" number="1"/>
      </notations>
  </note>
```

In token note - default-x, default-y is existence

✓ Grouping Should be deleted

```
TOKEN :
{
    < _MEASURE : < PRINT >< ATTRIBUTES >< NOTE > >
|   < _ATTRIBUTES : < DIVISIONS >< KEY >< TIME >< STAVES >< CLEF > >
|   < _TIME : < BEATS >< BEATTYPE > >
|   < _CLEF : < SIGN >< LINE > >
}

TOKEN :
{
    < PART : "part" >
|   < MEASURE : "measure" >
|   < PRINT : "print" >
|   < ATTRIBUTES : "attributes" >
|   < DIVISIONS : "divisions" >
|   < KEY : "key" >
|   < FIFTHS : "fifths" >
|   < TIME : "time" >
|   < BEATS : "beats" >
|   < BEATTYPE : "beat-type" >
|   < STAVES : "staves" >
|   < CLEF : "clef" >
|   < SIGN : "sign" >
|   < LINE : "line" >
|   < NOTE : "note" >
|   < PITCH : "pitch" >
|   < STEP : "step" >
|   < OCTAVE : "octave" >
|   < DURATION : "duration" >
|   < TYPE : "type" >
}

TOKEN :
{
    < NUMBER : "number=\"[1-3]\"" >
|   < NEWLINE : "new-system=\"yes\"" >
}
```

Grouping evoke some Error



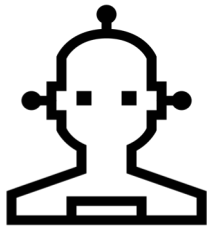
Draft Parsing by java cc

✓ **Source Code of Parser.jj**

```
101 }
102 {
103    "<"< PRINT >">"
104    "</"< PRINT >">"
105      { }
106 }
107
108 void parseAttributes() :
109 {
110 }
111 {
112    "<"< ATTRIBUTES >">"
113 //    ( parseDivisions() | parseKey() | parseTime() | parseStaves() | parseClef(
114    "</"< ATTRIBUTES >">"
115      { }
116 }
117
118 void parseNote() :
119 {
120 }
121 {
122      "<"< NOTE >">"
123 //    ( parsePitch() | parseDuration() | parseType() )+
      "</"< NOTE >">"
```

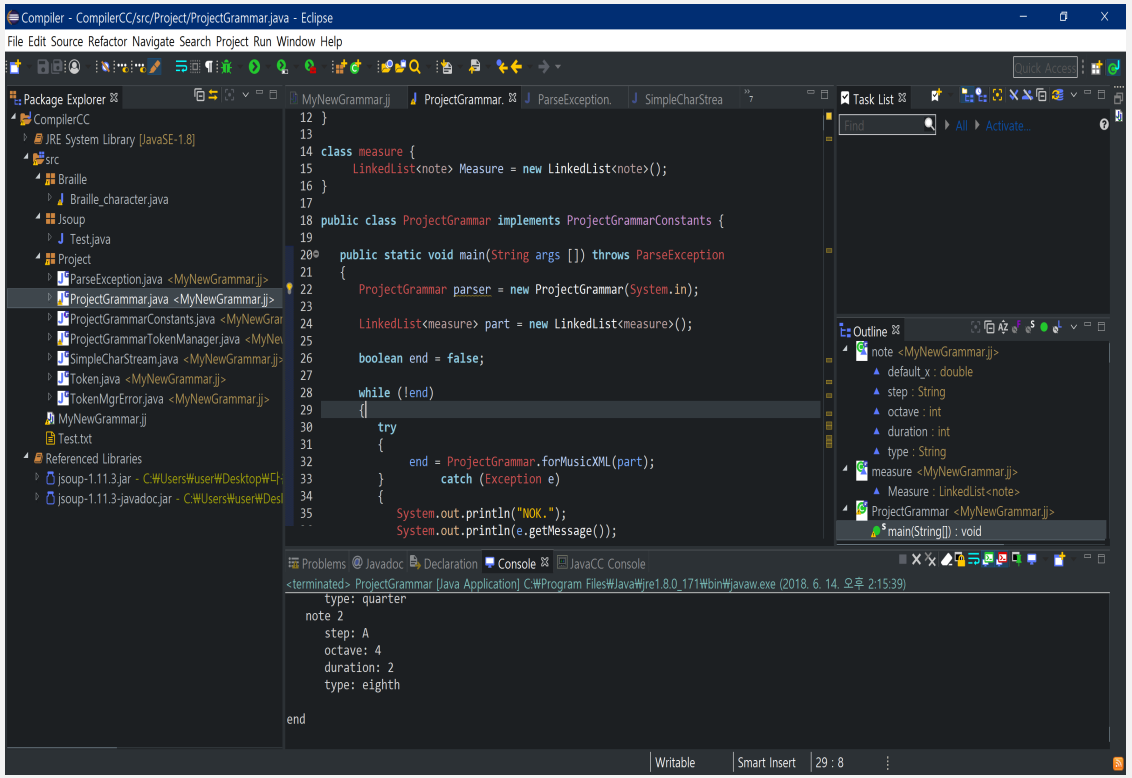
✓ **Result of our parser**

```
<part>
        <measure>
                <attributes>
                </attributes>
        </measure>
        <measure>
                <print>
                </print>
                <print>
                </print>
                <note>
                </note>
        </measure>
</part>
damn!!
```



Updated Parsing by java cc

✓ Source Code of Updated Parser.jj



✓ Can Recognize Measure

```
                                        <octave>
                                            4
                                        </octave>
                                    </pitch>
                                    <duration>
                                        2
                                    </duration>
                                    <type>
                                        eighth
                                    </type>
                                </note>
                            </measure>
                        </part>
measure 1
            note 1
                        step: D
                        octave: 3
                        duration: 12
                        type: half
measure 2
            note 1
                        step: C
                        octave: 4
                        duration: 6
                        type: quarter
            note 2
                        step: A
                        octave: 4
                        duration: 2
                        type: eighth

end
```

Output of our parser (Input: XML file)

Learning through Compiler Course & Project

✓ **Ideate Power Project Design**

✓ **Project Planning Ability**

Study = Ideate (By reference paper, history, and present)

Project = Planning (Study + Design) + Programming (Work)

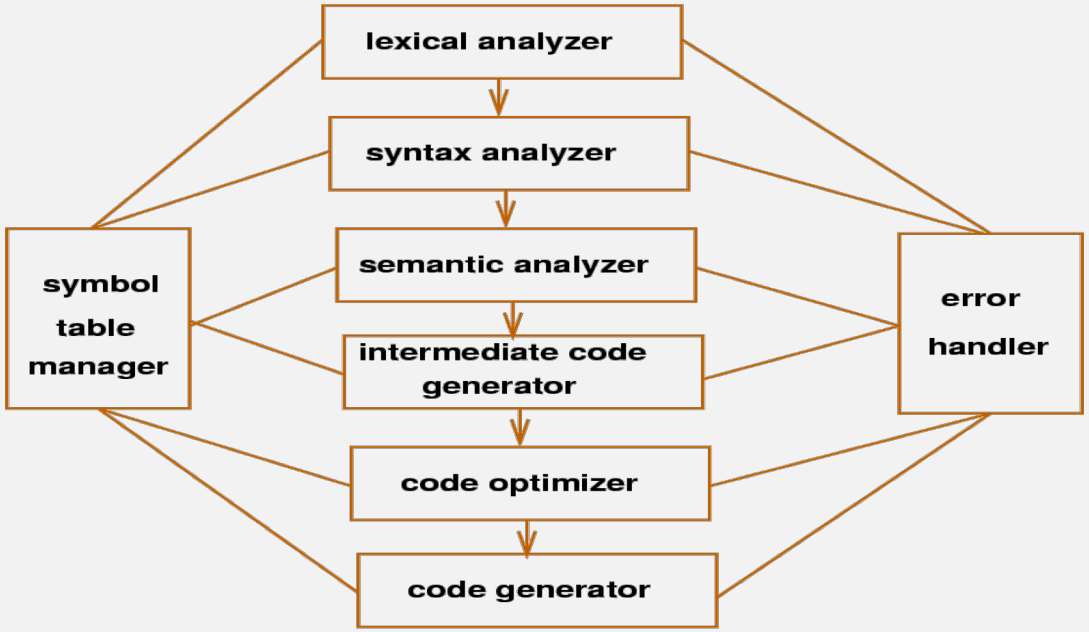
Learning through Compiler Course & Project

✓ Adaptability on Library

Adaptability

Not only Jsoup for XML. Git hub, Bit bucket, program creek...

✓ Conversation with Computer



Concept of translation between code and machine language

Thank you