DATABASE
PROJECT 1 INTRODUCTION

Term Project 1

- SQL의 기본적인 기능들을 수행할 수 있는 간단한 DBMS 구현
  - create table, insert, delete, select, …
- 총 세 단계(Project 1-1 ~ 1-3)에 걸쳐 구현하게 될 것임

- Project 1-1: SQL Parser
  - JavaCC를 이용하여 SQL문을 파싱할 수 있는 SQL 파서를 구현
- Project 1-2: Implementing DDL
  - 1-1에 기반하여 스키마를 저장하고 관리할 수 있는 기능을 구현 (create table, drop table, …)
- Project 1-3: Implementing DML
  - 1-1과 1-2에 기반하여 직접 데이터를 저장/삭제/질의할 수 있는 기능을 구현 (insert, delete, select)

What is Parser?

- 어떤 문장이나 코드를 읽어 들여 주어진 문법을 이용하여 그 구조를 알아내는 구문 분석(parsing)을 행하는 프로그램

- int a = 1;
  - int: data type
  - a: identifier
  - =: assignment operator
  - 1: numeric value
  - ;: end of statement

JavaCC

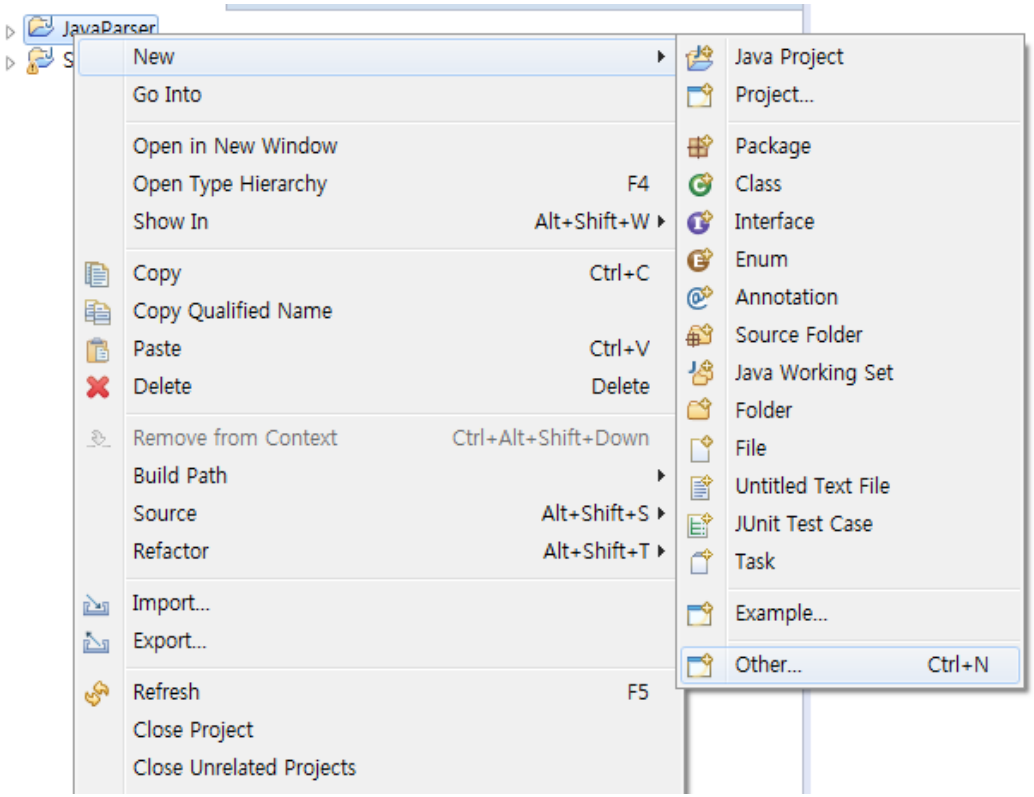
- Java Compiler Compiler
- Parsing Programming Language
- Java 기반
- 정규식(Regular expression)을 사용
- *.jj 파일에 파싱 룰(Grammar)을 정의해주면 JavaCC가 그 문법에 맞는 파서를 만들어줍니다
- We are going to use JavaCC via Eclipse.

How to install JavaCC Compiler

- Install Eclipse Plug-in for JavaCC
  - http://bluexmas.tistory.com/229
- Create new Java Project

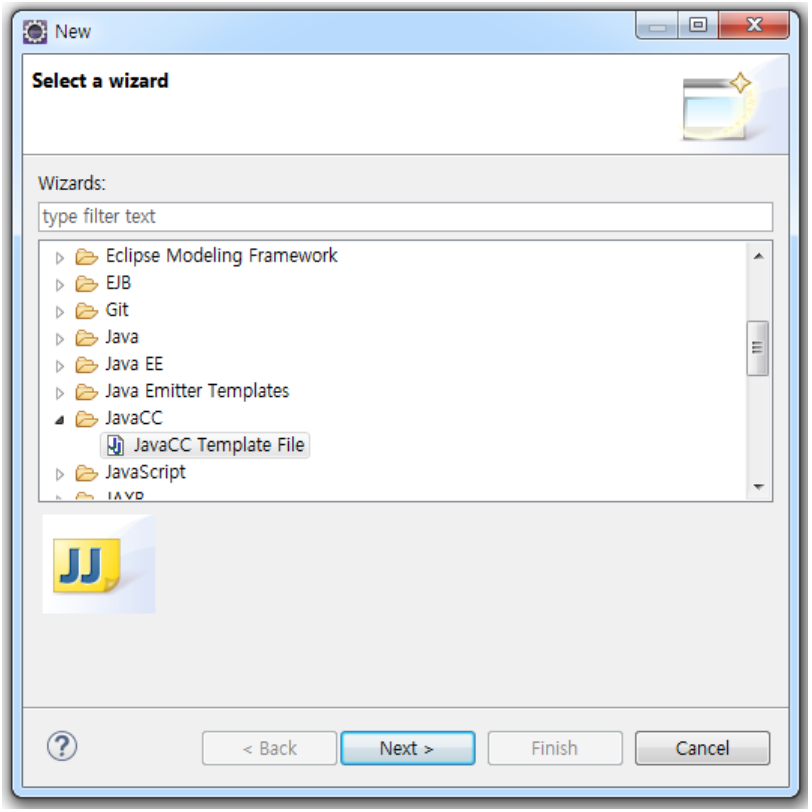
How to install JavaCC Compiler

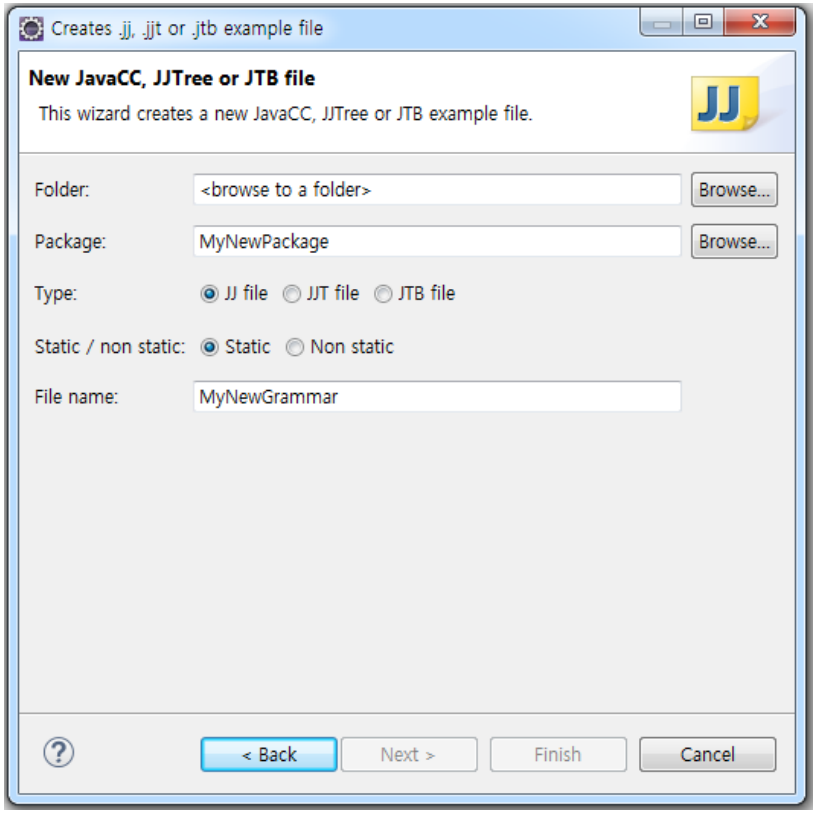
- Right click the Java project you created, then go to New -> Other



How to install JavaCC Compiler

- Select JavaCC/JavaCC Template File and input Folder, Package, File name



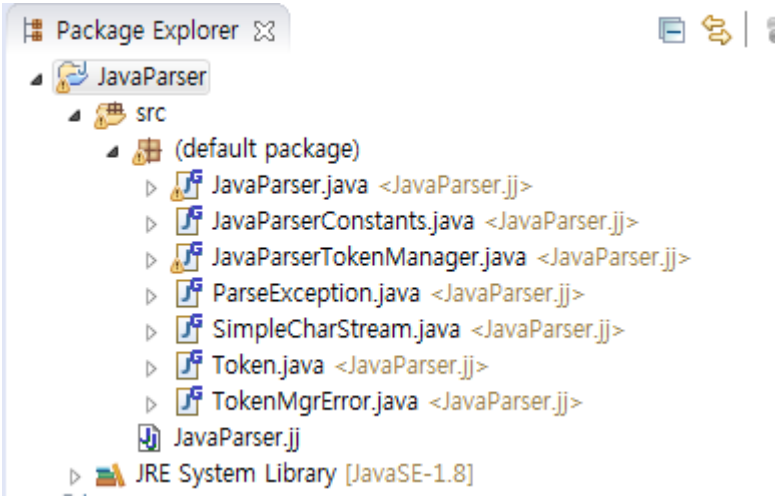


How to install JavaCC Compiler

- Replace **<?parser_name?>** with your own parser name and save. Parser will be auto-generated.

*JavaParser.jj

```
1  /**
2   * JavaCC template file created by SF JavaCC plugin 1.5.
3   */
4  options
5  {
6    static = true;
7  }
8
9  PARSER_BEGIN(<?parser_name?>)
10
11 public class <?parser_name?>
12 {
13   public static void main(String args []) throws ParseEx
14   {
15     <?parser_name?> parser = new <?parser_name?>(System.
16     while (true)
17     {
18       System.out.println("Reading from standard input...
19       System.out.print("Enter an expression like \"1+(2+
20       try
21       {
22         switch (<?parser_name?>.one_line())
23         {
24           case 0 :
25           System.out.println("OK.");
26           break;
27           case 1 :
28           System.out.println("Goodbye.");
29           break;
30           default :
31           break;
32         }
```



Segments of *.jj files

- Options
  - static, DEBUG_PARSER, …
- PARSER_BEGIN(Name) ~ PARSER_END(Name)
  - 파싱 시작 코드
- Token Definition
  - Regular Expression
- Function Definition
  - Regular Expression

옵션들

https://www.informatik.uni-kiel.de//~java/doc/javacc/examples/javaccgrm.html

```
options {
    STATIC=false
}

PARSER_BEGIN(Adder)

//import things

public class Adder
{
  int previous_result = 0;

  public static void main(String[] args) throws ParseException, TokenMgrError
  {
    Adder parser = new Adder(System.in);
    parser.Start();
  }
}

PARSER_END(Adder)
```

시작 자바 코드

```
SKIP : {" "|"\t"}
```
없는 토크 취급

```
TOKEN :
{
  < EOL:"\n"|"\r"|"\r\n" >
}

TOKEN :
{
<PLUS:"+">
| < MINUS:"-" >
| < EQUAL:"=" >
| < NUMBER:(["0"-"9"])+ >
| < SEMICOLON:";" >
}
```
Regular Expression

parser.Start()

```
void Start():
{}
{
        1+2       =       \n
    (
        Expression() < EQUAL > < EOL >
    )*
    < SEMICOLON > ;
}

int Expression():
{}
{     1
  < NUMBER >
  (
    < PLUS >< NUMBER > | < MINUS >< NUMBER >
  )*     +       2
}
```

- 1+2=
- ;

```
void Start():
{}
{
    (
        previous_result = Expression()
        < EQUAL >
        < EOL >
        {   System.out.println(previous_result);    }
    )*
    < SEMICOLON >
}

int Expression():
{
    Token t;
    Token s;
    int result;
}
{
  t = < NUMBER >
  { result = Integer.parseInt(t.image); }
  (
    < PLUS >
    s = < NUMBER >
    {   result += Integer.parseInt(s.image);    }
    |
    < MINUS >
    s = < NUMBER >
    {   result -= Integer.parseInt(s.image);    }
  )*
  { return result;  }

}
```

- 1+2=

3

- ;

String, Double, …

Example: “create table” query

```
public static void main(String args[]) throws ParseException
 {
   SimpleDBMSParser parser = new SimpleDBMSParser(System.in);

   while (true)
   {
     try
     {
       System.out.print(“DB_2019-12345> ");
       command();
     }
     catch (Exception e)
     {
       printMessage(PRINT_SYNTAX_ERROR);
       SimpleDBMSParser.ReInit(System.in);
     }
   }
 }
```

Example: “create table” query

```
SKIP : { " " | "\r" | "\t" | "\n" }
TOKEN :
{
  < SEMICOLON : ";" >
| < LEFT_PAREN : "(" >
| < RIGHT_PAREN : ")" >
| < COMMA : "," >
| < UNDERSCORE : "_" >
| < SIGN : "+" | "-" >
| < DIGIT : [ "0"-"9" ] >
| < ALPHABET : [ "A"-"Z", "a"-"z" ] >
}
```

Example: “create table” query

<Grammar Definition>

<COMMAND> ::= <QUERY LIST>

| **exit** <SEMICOLON>

<QUERY LIST> ::= (<QUERY> <SEMICOLON>)+

<JavaCC Implementation>

```
void command() :
{}
{
  queryList()
| (
    < EXIT >
    < SEMICOLON >
    {
      System.exit(0);
    }
  )
}

void queryList() :
{
  int q;
}
{
  (
    q = query()
    < SEMICOLON >
    {
      System.out.print("DB_2019-12345> ");
      printMessage(q);
    }
  )+
}
```

Example: “create table” query

<Grammar Definition>

<PRIMARY KEY CONSTRAINT> ::=
primary key <COLUMN NAME LIST>

<REFERENTIAL CONSTRAINT> ::=
foreign key <COLUMN NAME LIST>
references <TABLE NAME> <COLUMN NAME LIST>

<JavaCC Implementation>

```
void primaryKeyConstraint() :
{}
{
  < PRIMARY_KEY >
  columnNameList()
}

void referentialConstraint() :
{}
{
  < FOREIGN_KEY >
  columnNameList()
  < REFERENCES >
  tableName()
  columnNameList()
}
```

Notice

- **제출 기한: 2019/4/1 (Mon), 11:59 PM**
  - **10% penalty for ~24hours late**
  - **20% penalty for 24~48hours late**
  - **no credit after 48 hours**

- “create table”문에 대한 레퍼런스 코드(.jj 파일)가 주어짐
  - 조교의 레퍼런스 코드 위에 구현해도 되고
  - 처음부터 직접 구현해도 됩니다

- 자세한 프로젝트 설명과 Grammar 정의는 강의 홈페이지를 참고하세요

- 프로젝트에 대한 질문은 이메일을 이용할 것
  - lecture@europa.snu.ac.kr