APL Assignment 3 – Java Parser

Name: Aishwarya Bhavsar

CSULB ID : 029371509

➢ SOURCE CODE: CECS524TestJavaParser.java

```
/*
* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
*/
package cecs524testjavaparser;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import org.antlr.runtime.ANTLRFileStream;
import org.antlr.runtime.CommonTokenStream;
import org.antlr.runtime.RecognitionException;

public class CECS524TestJavaParser {

    /**
     * @param args the command line arguments
     * @throws java.io.IOException
     * @throws org.antlr.runtime.RecognitionException
     */
    public static void main(String[] args) throws IOException, RecognitionException {
```

```
    String filePath =System.getProperty("user.dir") + File.separator + "src" + File.separator + args[0];

    ANTLRFileStream input = new ANTLRFileStream(filePath);

    JavaLexer lexer = new JavaLexer(input);

    CommonTokenStream tokens = new CommonTokenStream(lexer);

    JavaParser parser = new JavaParser(tokens);


    parser.compilationUnit();

  }


}
```

MyClass.java

```
public class MyClass extends Base implements Class1 {

  private int x,y;

  public MyClass(){}

  public void hello(int ix, int iy){

  x=ix;

  y=iy;

  }

 /* public static void main(String[] args) {

    System.out.println("Hello");

  }*/

}
```

Base.java

```
/*

* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
```

```
*/

/**
*
* @author aishu
*/
class Base {

}
```

Class1.java

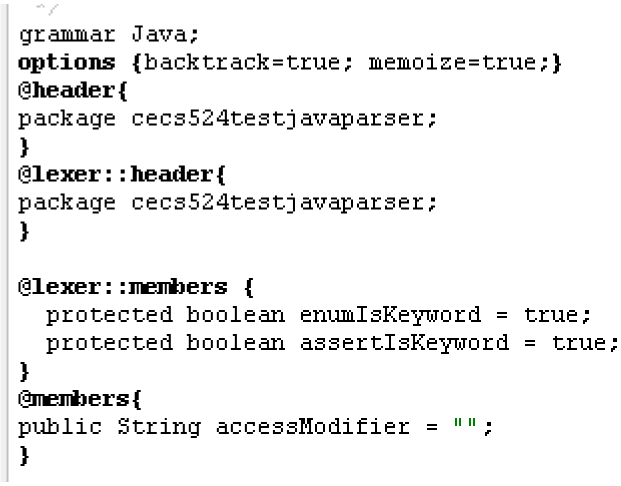
```
/*
* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Interface.java to edit this template
*/

/**
*
* @author aishu
*/
interface Class1 {

}
```

➢ Changes Made in The Grammar File:

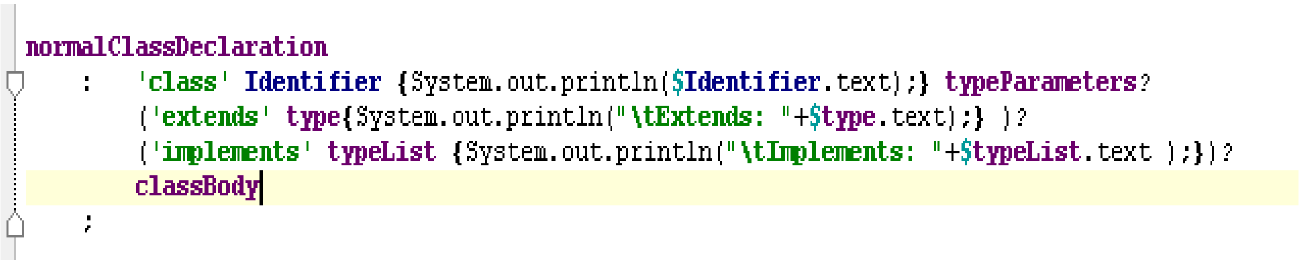
- We define a header and package name which will be placed at the start of the generated parser class. This will allow us to designate a package to import in our java code.
- We also define @members class.



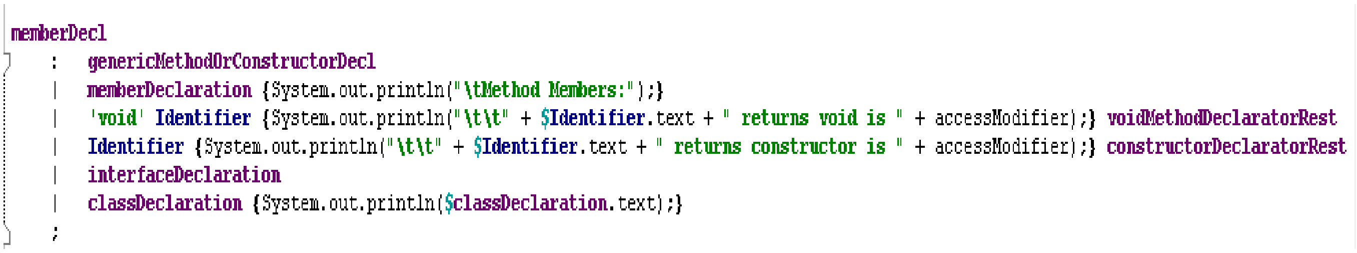
```
*/
grammar Java;
options {backtrack=true; memoize=true;}
@header{
package cecs524testjavaparser;
}
@lexer::header{
package cecs524testjavaparser;
}

@lexer::members {
  protected boolean enumIsKeyword = true;
  protected boolean assertIsKeyword = true;
}
@members{
public String accessModifier = "";
}
```

- We embed actions within grammars.
- We are using the attribute text which is a string, it includes the text for all tokens.
- *$Identifier.text* returns the complete text matched by a preceding rule invocation of rule *normalClassDeclaration Identifier.*



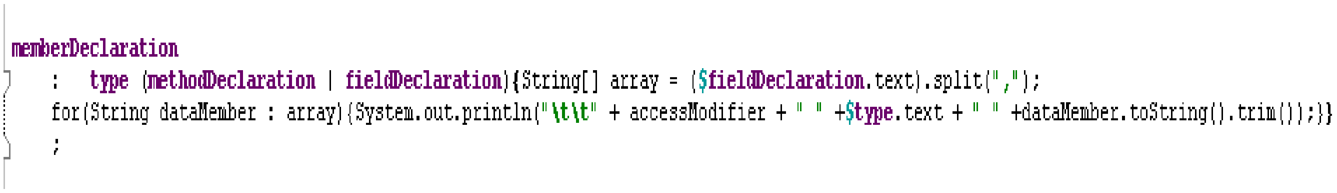
```
normalClassDeclaration
    :   'class' Identifier {System.out.println($Identifier.text);} typeParameters?
        ('extends' type{System.out.println("\tExtends: "+$type.text);} )?
        ('implements' typeList {System.out.println("\tImplements: "+$typeList.text );})?
        classBody
    ;
```



```
memberDecl
    :   genericMethodOrConstructorDecl
    |   memberDeclaration {System.out.println("\tMethod Members:");}
    |   'void' Identifier {System.out.println("\t\t" + $Identifier.text + " returns void is " + accessModifier);} voidMethodDeclaratorRest
    |   Identifier {System.out.println("\t\t" + $Identifier.text + " returns constructor is " + accessModifier);} constructorDeclaratorRest
    |   interfaceDeclaration
    |   classDeclaration {System.out.println($classDeclaration.text);}
    ;
```

- We created an array of Strings, applied for loop.

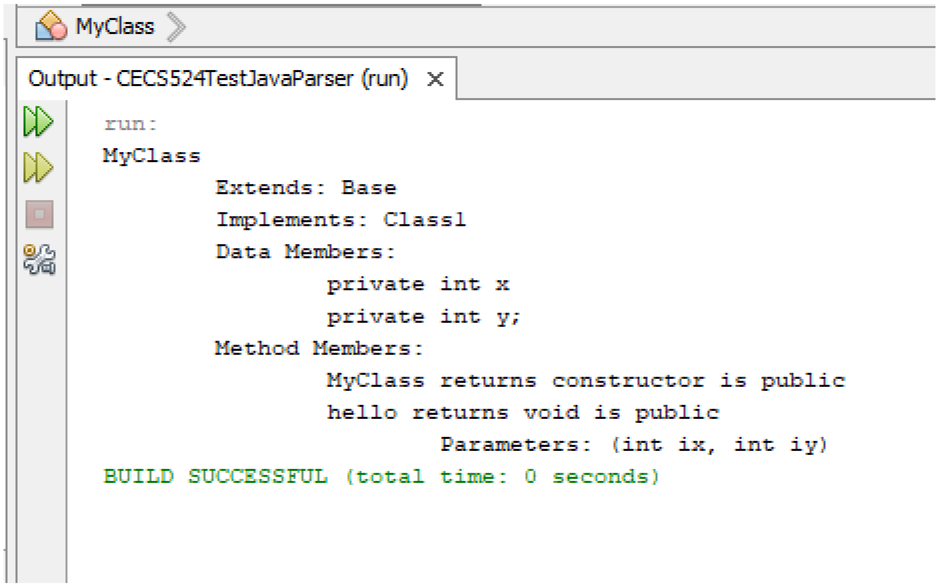
- ○ **The field declaration x, y is printed.**
- ○ **We used the split () and trim () string methods to break the string and return a char array and to remove in trailing or leading spaces.**



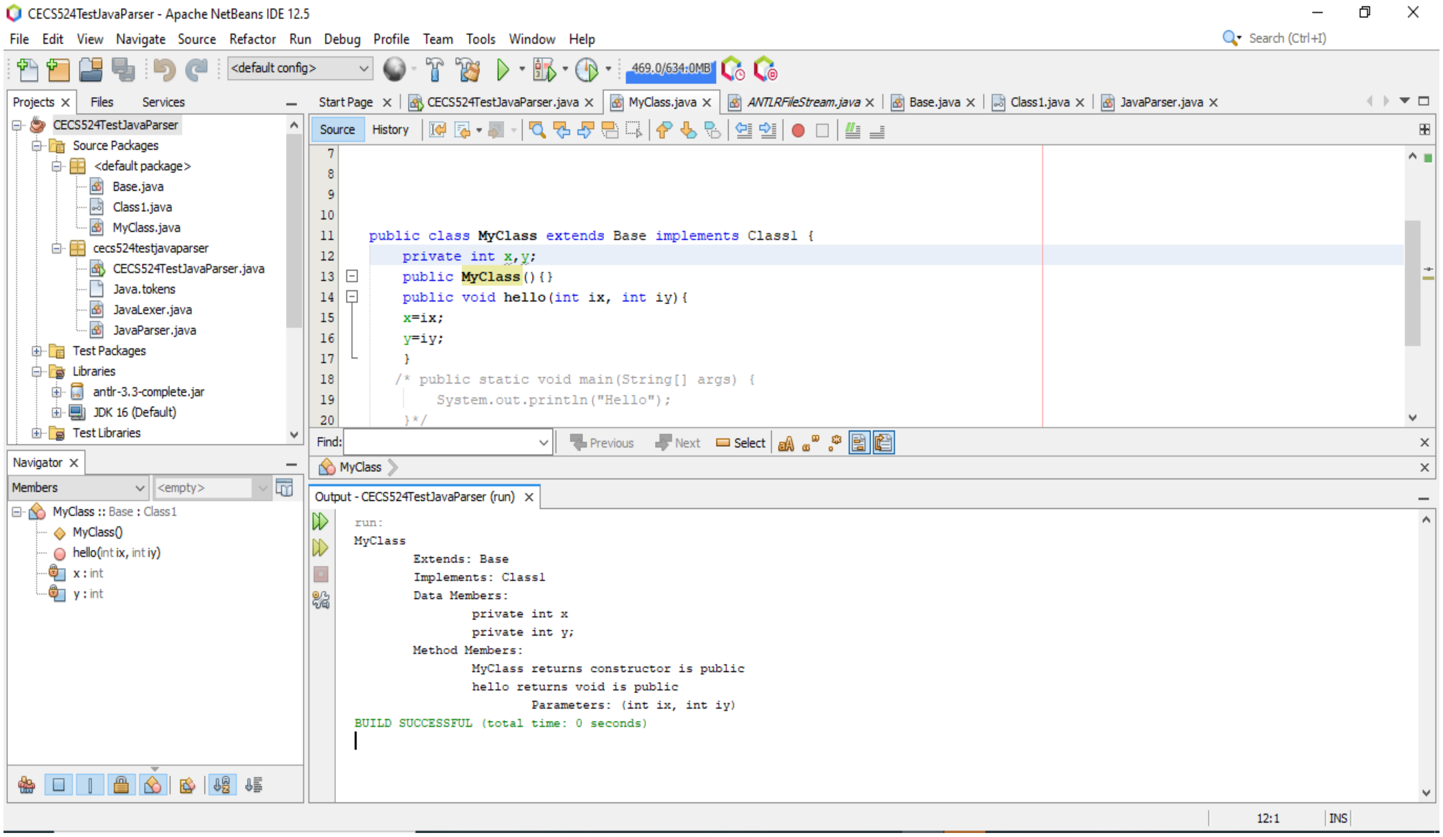
```
memberDeclaration
    :   type (methodDeclaration | fieldDeclaration){String[] array = ($fieldDeclaration.text).split(",");
    for(String dataMember : array){System.out.println("\t\t" + accessModifier + " " +$type.text + " " +dataMember.toString().trim());}}
    ;
```

➢ <u>OUTPUT:</u>

- ○ Here, the method will be parsed first and then the parameters, hence the parameters are getting printed at the last.



```
run:
MyClass
        Extends: Base
        Implements: Class1
        Data Members:
                private int x
                private int y;
        Method Members:
                MyClass returns constructor is public
                hello returns void is public
                        Parameters: (int ix, int iy)
BUILD SUCCESSFUL (total time: 0 seconds)
```



```java



public class MyClass extends Base implements Class1 {
    private int x,y;
    public MyClass(){}
    public void hello(int ix, int iy){
    x=ix;
    y=iy;
    }
  /* public static void main(String[] args) {
       System.out.println("Hello");
    }*/
```

```
run:
MyClass
        Extends: Base
        Implements: Class1
        Data Members:
                private int x
                private int y;
        Method Members:
                MyClass returns constructor is public
                hello returns void is public
                        Parameters: (int ix, int iy)
BUILD SUCCESSFUL (total time: 0 seconds)
```