

SPL-1 Project Report, 2019

Java Parsing

SE 305: Software Project Lab-1

Submitted by

Dip Saha

BSSE Roll No: 1001

BSSE Session: 2017-18

Supervised by

Abdus Sattar

Designation: *Lecturer*

Institute of Information Technology



Institute of Information Technology

University of Dhaka

Date: 29-05-2019

Table of contents

1. Introduction.....	1
1.1 Background Study.....	1-2
1.2 Challenges.....	2-3
2. Project Overview.....	3
3. User Manual.....	3-6
4. Conclusion.....	6
5. Appendix.....	6
6. Reference.....	6

Index of Figure

Fig 1: Structure of a java code.....	1
Fig 2: Sample input file.....	4
Fig 3: Input a file.....	4
Fig 4: Sample Output.....	5
Fig 5: Sample Output.....	6

1. Introduction

A **parser** is a software component that takes input data (frequently text) and builds a **data structure** – often some kind of **parse tree**, **abstract syntax tree** or other hierarchical structure, giving a structural representation of the input while checking for correct syntax.

A “**Java Parser**” is used to parsing a java file. That is, it separates the variables, method signatures, class information from a java class and analyzes them.

1.1 Background Study

Before building a java parser, I gathered knowledge about structure of a java code. A java code is nothing but a set of data members (variables or attributes), user defined functions which contains block of statements defined inside a class.

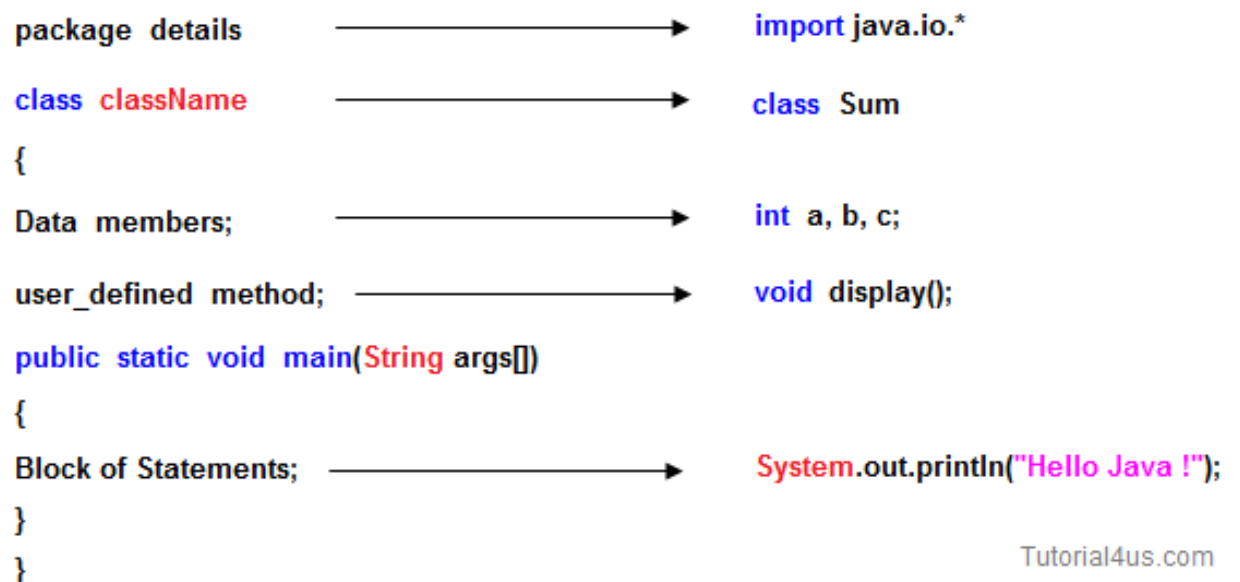


Fig 1: Structure of a java code

Finite state machine

A finite state machine (sometimes called a finite state automaton) is a computation model that can be implemented with hardware or software and can be used to simulate sequential logic and some computer programs.

Finite state machine is used to recognize patterns. Finite automata machine takes the string of symbol as input and changes its state accordingly. In the input, when a desired symbol is found then the transition occurs. While transition, the automata can either move to the next state or stay in the same state.

FA has two states: accept state or reject state. When the input string is successfully processed and the automata reached its final state then it will accept.

Pushdown Automata

Pushdown automata is a finite automata with extra memory called stacks which helps pushdown automata to recognize context free languages.

Tokenization

In computer science, tokenization is the process of converting a sequence of characters into a sequence of tokens (identifiers, keyword, separator, operator, comment, literal). A program that performs lexical analysis may be termed as lexer or tokenizer. A lexer is generally combined with a parser, which together analyze the syntax of programming language, web pages and so forth.

1.2 Challenges

To develop a software project, one may face a lot of challenges. The process can be overwhelming, confusing and lengthy. During implementing this project, I have faced some challenges. For example-

1. Handling large code for the first time
2. Developing an algorithm for the project
3. Process non-primitive data type.
4. Delete comment from the code

5. Tokenization

2. Project Overview

A **Java Parser** is used to parsing a java file. That is, it separates the variables, method signatures, class information from a java class.

It then analyzes the variables and method signatures. For example : it finds out the return type, access modifier, method name, parameter list of a method; data type, access modifier and other modifiers of a variable. So,

Input

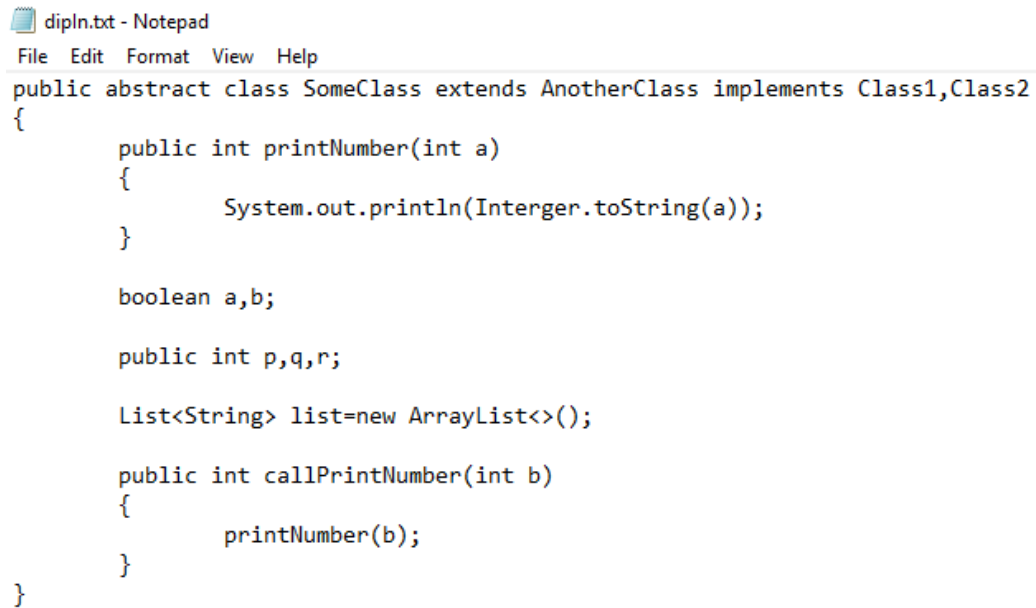
A java source code without any error (more specifically, a java class)

Output

1. Separates the method signature, variable declaration, class name of the class.
2. Finds out the access modifier, return type, method name, parameter list of a method.
3. List data type, access modifier, variable name of a variable.
4. Finds out the super class of the input class and also checks whether this input class is an abstract class or not and interface or not.
5. Prints line number of a method declaration and where this method is called in several points in the code.
6. Prints all abstract methods and static methods in the class.

3. User Manual

At first, you should input a .txt file which contains a java code. For example-



```
File Edit Format View Help
public abstract class SomeClass extends AnotherClass implements Class1,Class2
{
    public int printNumber(int a)
    {
        System.out.println(Integer.toString(a));
    }

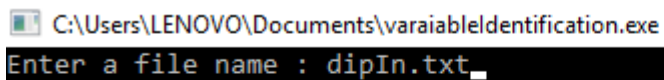
    boolean a,b;

    public int p,q,r;

    List<String> list=new ArrayList<>();

    public int callPrintNumber(int b)
    {
        printNumber(b);
    }
}
```

Fig 2: Sample input file



```
C:\Users\LENOVO\Documents\variableIdentification.exe
Enter a file name : dipIn.txt_
```

Fig 3: Input a file

The output looks like-

C:\Users\LENOVO\Documents\variableIdentification.exe

Enter a file name : dipIn.txt

Class Name : SomeClass
Is it Interface? : No
Is it abstract? : Yes
SuperClass : AnotherClass
Implements : Class1 Class2

all methods

method name : printNumber
return type : int
access modifier : public
parameter list : int a
declared in line : 3
called in lines : 16

method name : callPrintNumber
return type : int
access modifier : public
parameter list : int b
declared in line : 14
called in lines :

total method : 2

abstract methods

method name : printNumber
return type : int
access modifier : public
parameter list : int a
declared in line : 3

static methods

no static method found

attribute name	data type	access modifier
-----	-----	-----

Fig 4: Sample Output


```
C:\Users\LENOVO\Documents\variableIdentification.exe
no static method found

attribute name  data type      access modifier
-----
a              boolean      public
b              boolean      public
p              int          public
q              int          public
r              int          public
list           List         public

total attribute : 6

Process returned 0 (0x0)   execution time : 228.844 s
Press any key to continue.
```

Fig 5: Sample Output

4. Conclusion

During implementation, I have learned how to handle a large code for the first time. It also develops my coding skill as well. I hope, it will help me to deal with difficulties in future. This project was quiet challenging and I have gathered a lot of experiences from it. I want to thank my supervisor for guiding me a lot during this project.

5. Appendix

I have just worked with simple java code and one java class as input. In future, I will work for more than one class simultaneously and more complex java code. And, I will also try to give input a .java file rather than .txt file.

6. Reference

1. <https://en.m.wikipedia.org/wiki/Parsing> (25-01-2019)
2. Fig 1 image source: www.sitesbay.com
3. <https://www.javatpoint.com/finite-state-machine> (30-01-2019)