# Lab 02

CSC2006: Programming Language Theory

## Lab 02) Implementation of a Parser for Language S

- Parser Implementation for Language S (Java)

(1) Complete the implementation of a parser for Language S based on its Abstract Syntax Tree (AST)

- Parse expressions and statements to construct and return the AST.
- Add support for comparison and logical operations.
- Include support for \`while\`, \`read\`, and \`print\` statements.

(2) Display the AST nodes in a tree structure

Implement a \`display\` method that uses indentation levels to print each AST node in a hierarchical tree format.

## Lab 02) Implementation of a Parser for Language S

- Parser Implementation for Language S (Java)

  Syntax of language S (EBNF)

```
<program> → {<command>}
<command> → <decl> | <stmt>
<decl> → <type> id [=<expr>];
<stmt> → id = <expr>;
   | ‘{‘ <stmts> ‘}’
   | if (<expr>) then <stmt> [else <stmt>]
   | while (<expr>) <stmt>
   | read id;
   | print <expr>;
   | let <decls> in <stmts> end;
<stmts> → {<stmt>}
<decls> → {<decl>}
<type> → int | bool | string
```

# Lab 02) Implementation of a Parser for Language S

- Parser Implementation for Language S (Java)

  Expression syntax in language S (EBNF)

  <expr> → **<bexp> {& <bexp> | '|' <bexp>}** | !<expr> | true | false

  <bexp> → **<aexp> [<relop> <aexp>]**

  <relop> → == | != | < | > | <= | >=

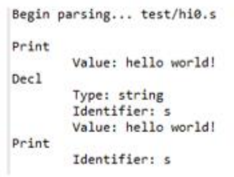
  <aexp> → <term> {+ <term> | - <term>}

  <term> → <factor> {* <factor> | / <factor>}

  <factor> → [-] ( <number> | (<aexp>) | id ) | strliteral

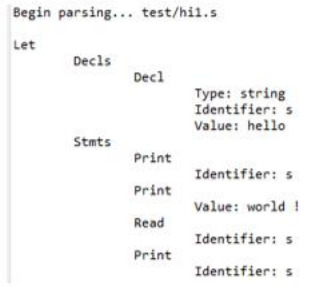
# Lab 02) Implementation of a Parser for Language S

hi0.s



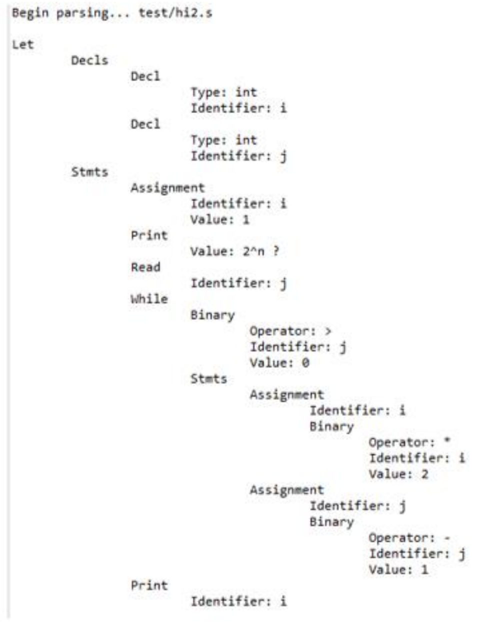
```
Begin parsing... test/hi0.s

Print
        Value: hello world!
Decl
        Type: string
        Identifier: s
        Value: hello world!
Print
        Identifier: s
```

hi1.s



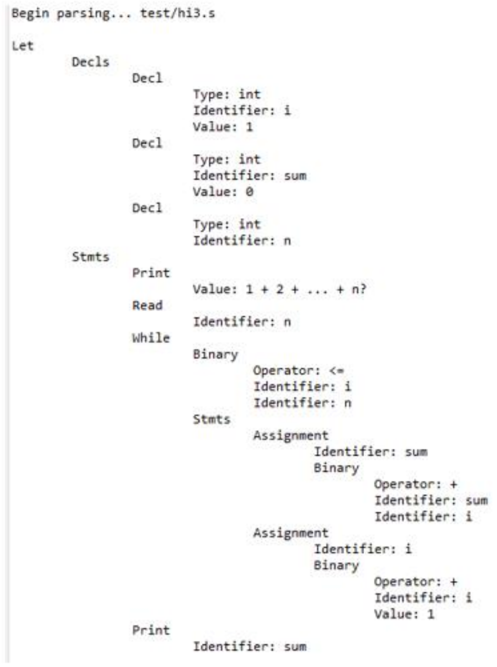
```
Begin parsing... test/hi1.s

Let
        Decls
                Decl
                        Type: string
                        Identifier: s
                        Value: hello
        Stmts
                Print
                        Identifier: s
                Print
                        Value: world !
                Read
                        Identifier: s
                Print
                        Identifier: s
```

hi2.s



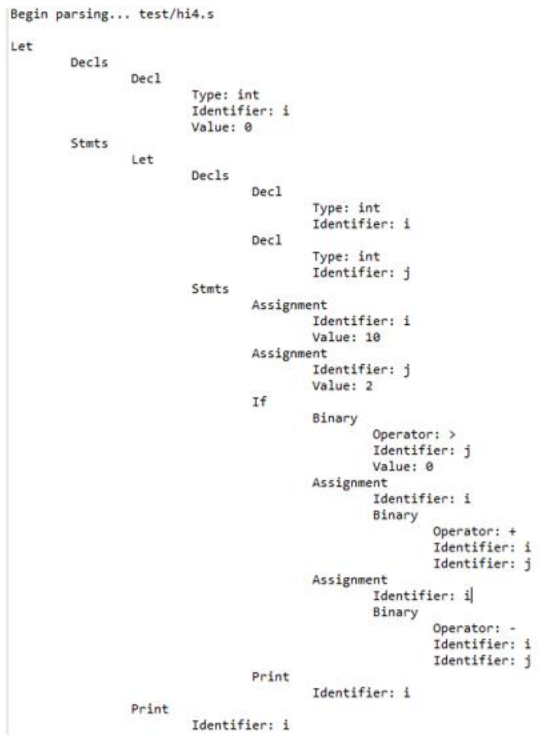
```
Begin parsing... test/hi2.s

Let
        Decls
                Decl
                        Type: int
                        Identifier: i
                Decl
                        Type: int
                        Identifier: j
        Stmts
                Assignment
                        Identifier: i
                        Value: 1
                Print
                        Value: 2^n ?
                Read
                        Identifier: j
                While
                        Binary
                                Operator: >
                                Identifier: j
                                Value: 0
                        Stmts
                                Assignment
                                        Identifier: i
                                        Binary
                                                Operator: *
                                                Identifier: i
                                                Value: 2
                                Assignment
                                        Identifier: j
                                        Binary
                                                Operator: -
                                                Identifier: j
                                                Value: 1
                Print
                        Identifier: i
```

hi3.s



```
Begin parsing... test/hi3.s

Let
        Decls
                Decl
                        Type: int
                        Identifier: i
                        Value: 1
                Decl
                        Type: int
                        Identifier: sum
                        Value: 0
                Decl
                        Type: int
                        Identifier: n
        Stmts
                Print
                        Value: 1 + 2 + ... + n?
                Read
                        Identifier: n
                While
                        Binary
                                Operator: <=
                                Identifier: i
                                Identifier: n
                        Stmts
                                Assignment
                                        Identifier: sum
                                        Binary
                                                Operator: +
                                                Identifier: sum
                                                Identifier: i
                                Assignment
                                        Identifier: i
                                        Binary
                                                Operator: +
                                                Identifier: i
                                                Value: 1
                Print
                        Identifier: sum
```

hi4.s



```
Begin parsing... test/hi4.s

Let
        Decls
                Decl
                        Type: int
                        Identifier: i
                        Value: 0
        Stmts
                Let
                        Decls
                                Decl
                                        Type: int
                                        Identifier: i
                                Decl
                                        Type: int
                                        Identifier: j
                        Stmts
                                Assignment
                                        Identifier: i
                                        Value: 10
                                Assignment
                                        Identifier: j
                                        Value: 2
                                If
                                        Binary
                                                Operator: >
                                                Identifier: j
                                                Value: 0
                                        Assignment
                                                Identifier: i
                                                Binary
                                                        Operator: +
                                                        Identifier: i
                                                        Identifier: j
                                        Assignment
                                                Identifier: i
                                                Binary
                                                        Operator: -
                                                        Identifier: i
                                                        Identifier: j
                                Print
                                        Identifier: i
                Print
                        Identifier: i
```

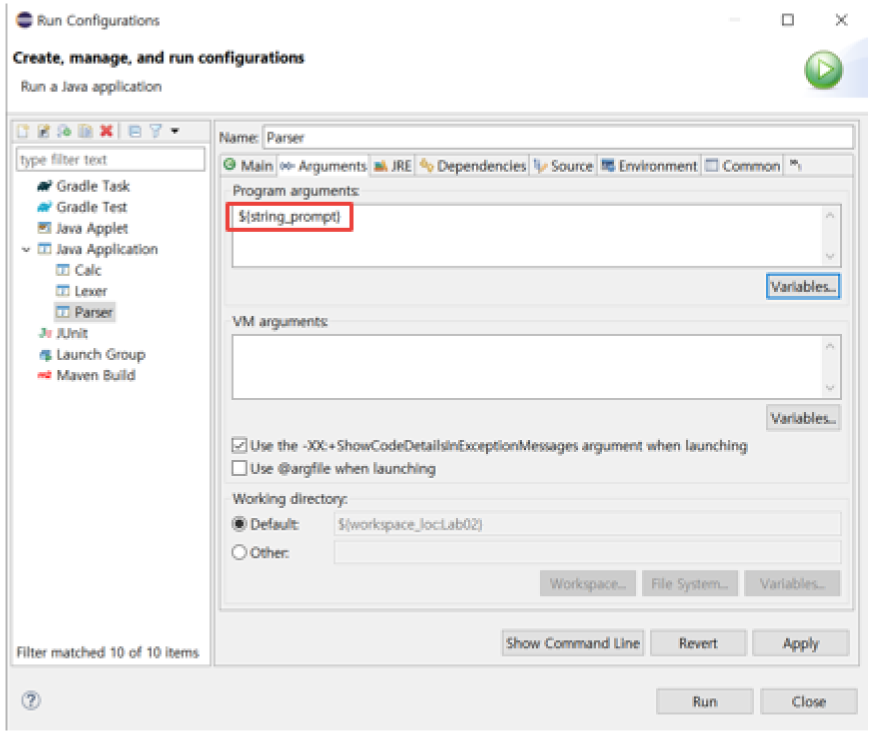
# Lab 02) Implementation of a Parser for Language S

- Parser Implementation for Language S (Java)

  Example and results

  Example codes in test folder

  ① hi0.s
  ② hi1.s
  ③ hi2.s
  ④ hi3.s
  ⑤ hi4.s
  ⑥ hi5.s
  ⑦ hi6.s
  ⑧ hi7.s



Name: Parser

Program arguments: ${string_prompt}

VM arguments:

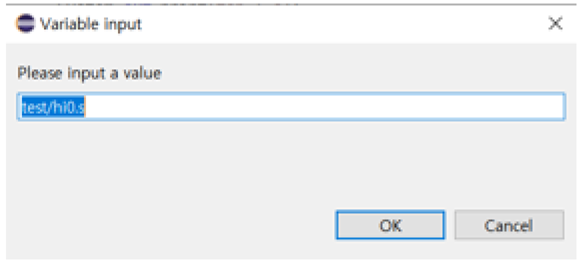
[✓] Use the -XX:+ShowCodeDetailsInExceptionMessages argument when launching

[ ] Use @argfile when launching

Working directory:

(●) Default: ${workspace_loc:Lab02}

( ) Other:

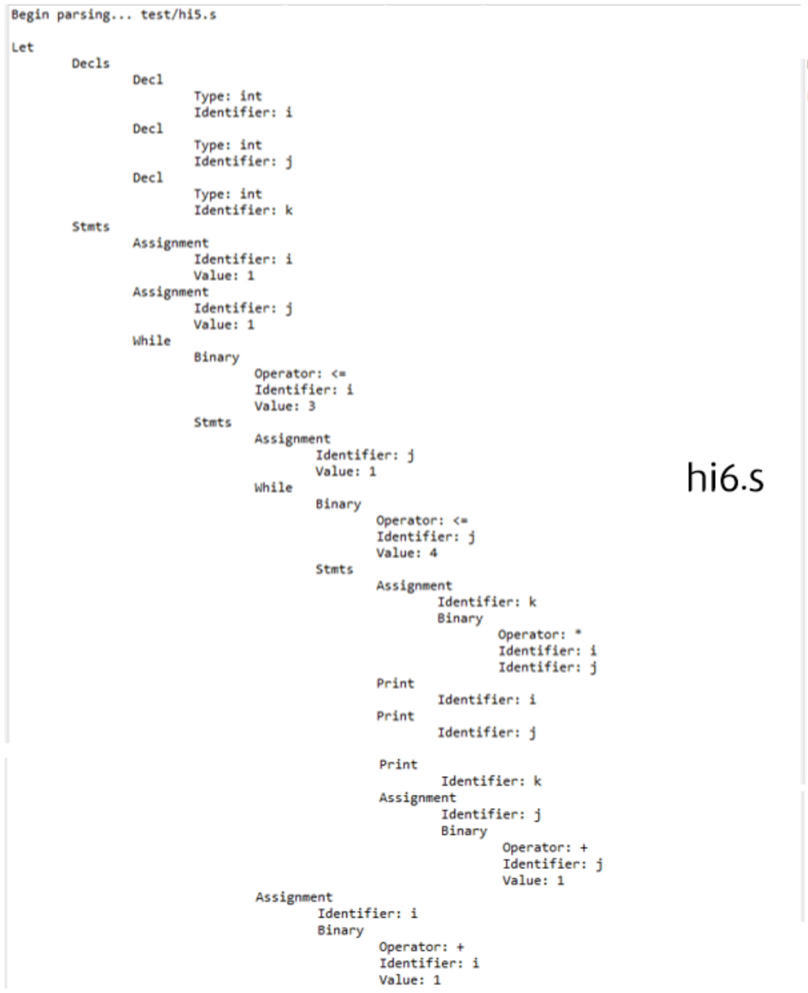


Please input a value

test/hi0.s

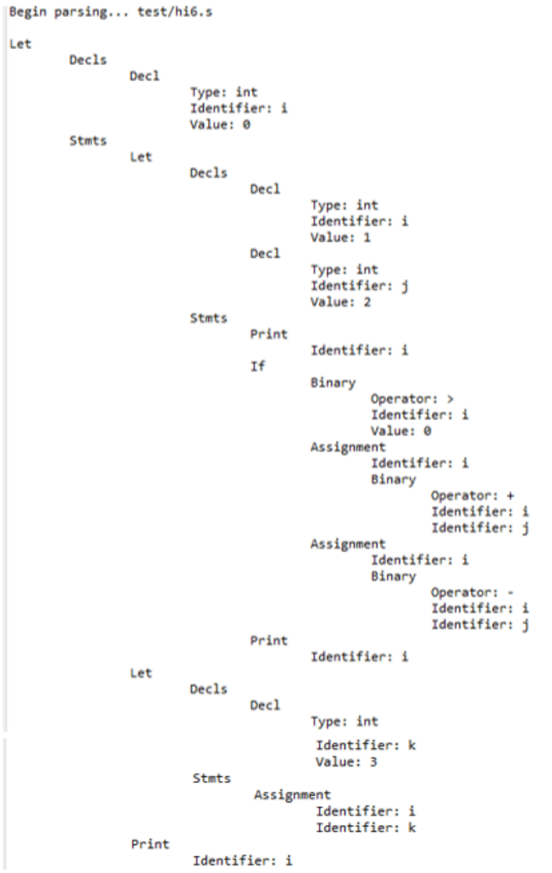
# Lab 02) Implementation of a Parser for Language S

hi5.s



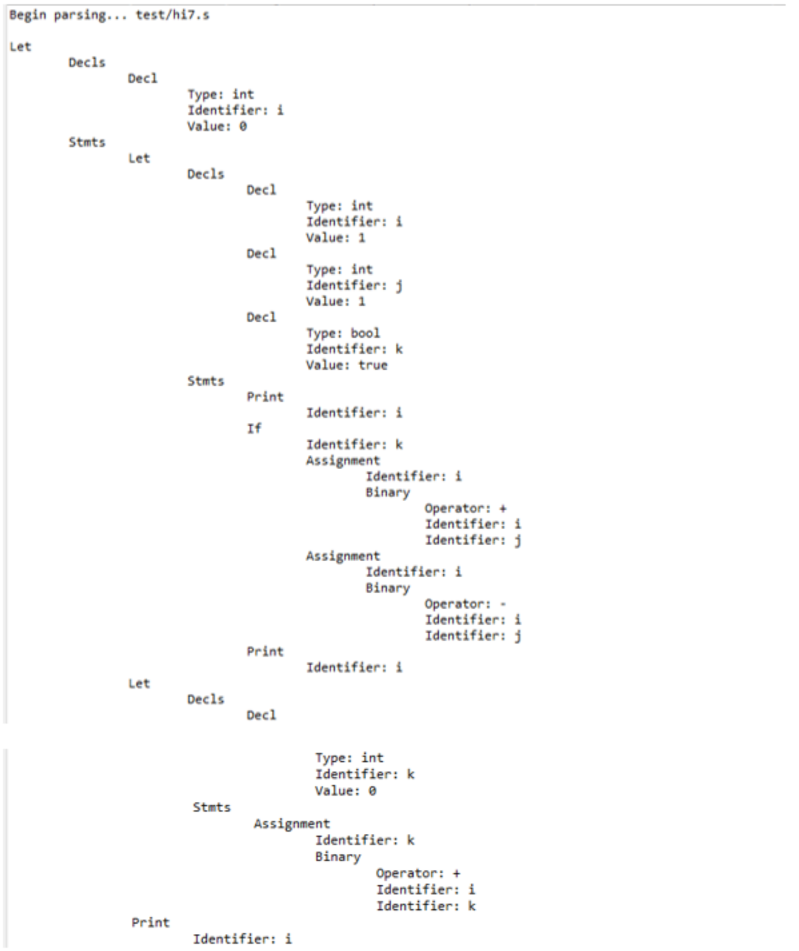
```
Begin parsing... test/hi5.s

Let
        Decls
                Decl
                        Type: int
                        Identifier: i
                Decl
                        Type: int
                        Identifier: j
                Decl
                        Type: int
                        Identifier: k
        Stmts
                Assignment
                        Identifier: i
                        Value: 1
                Assignment
                        Identifier: j
                        Value: 1
                While
                        Binary
                                Operator: <=
                                Identifier: i
                                Value: 3
                        Stmts
                                Assignment
                                        Identifier: j
                                        Value: 1
                                While
                                        Binary
                                                Operator: <=
                                                Identifier: j
                                                Value: 4
                                        Stmts
                                                Assignment
                                                        Identifier: k
                                                        Binary
                                                                Operator: *
                                                                Identifier: i
                                                                Identifier: j
                                                Print
                                                        Identifier: i
                                                Print
                                                        Identifier: j

                                                Print
                                                        Identifier: k
                                                Assignment
                                                        Identifier: j
                                                        Binary
                                                                Operator: +
                                                                Identifier: j
                                                                Value: 1
                                Assignment
                                        Identifier: i
                                        Binary
                                                Operator: +
                                                Identifier: i
                                                Value: 1
```

hi6.s



```
Begin parsing... test/hi6.s

Let
        Decls
                Decl
                        Type: int
                        Identifier: i
                        Value: 0
        Stmts
                Let
                        Decls
                                Decl
                                        Type: int
                                        Identifier: i
                                        Value: 1
                                Decl
                                        Type: int
                                        Identifier: j
                                        Value: 2
                        Stmts
                                Print
                                        Identifier: i
                                If
                                        Binary
                                                Operator: >
                                                Identifier: i
                                                Value: 0
                                        Assignment
                                                Identifier: i
                                                Binary
                                                        Operator: +
                                                        Identifier: i
                                                        Identifier: j
                                        Assignment
                                                Identifier: i
                                                Binary
                                                        Operator: -
                                                        Identifier: i
                                                        Identifier: j
                                Print
                                        Identifier: i
                Let
                        Decls
                                Decl
                                        Type: int
                                        Identifier: k
                                        Value: 3
                        Stmts
                                Assignment
                                        Identifier: i
                                        Identifier: k
                Print
                        Identifier: i
```

hi7.s

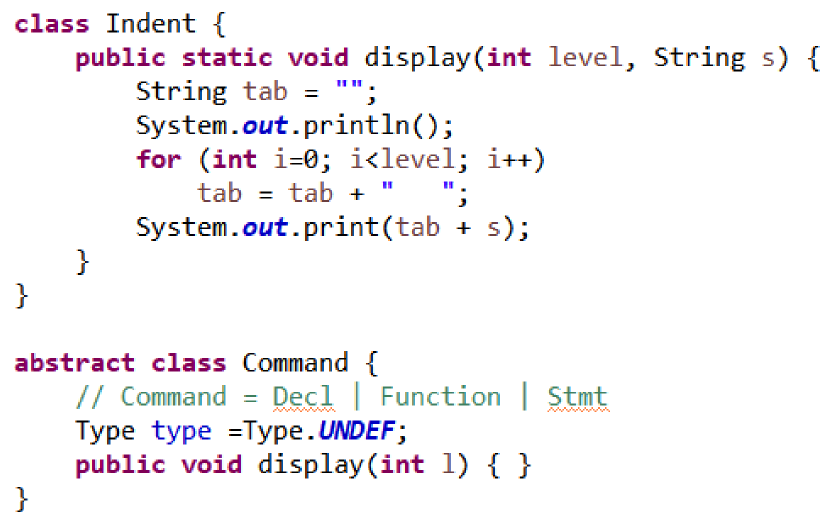


```
Begin parsing... test/hi7.s

Let
        Decls
                Decl
                        Type: int
                        Identifier: i
                        Value: 0
        Stmts
                Let
                        Decls
                                Decl
                                        Type: int
                                        Identifier: i
                                        Value: 1
                                Decl
                                        Type: int
                                        Identifier: j
                                        Value: 1
                                Decl
                                        Type: bool
                                        Identifier: k
                                        Value: true
                        Stmts
                                Print
                                        Identifier: i
                                If
                                        Identifier: k
                                        Assignment
                                                Identifier: i
                                                Binary
                                                        Operator: +
                                                        Identifier: i
                                                        Identifier: j
                                        Assignment
                                                Identifier: i
                                                Binary
                                                        Operator: -
                                                        Identifier: i
                                                        Identifier: j
                                Print
                                        Identifier: i
                Let
                        Decls
                                Decl

                                        Type: int
                                        Identifier: k
                                        Value: 0
                        Stmts
                                Assignment
                                        Identifier: k
                                        Binary
                                                Operator: +
                                                Identifier: i
                                                Identifier: k
                Print
                        Identifier: i
```

# Lab 02) Implementation of a Parser for Language S

- Parser Implementation for Language S (Java)

  - Tip (AST.java)

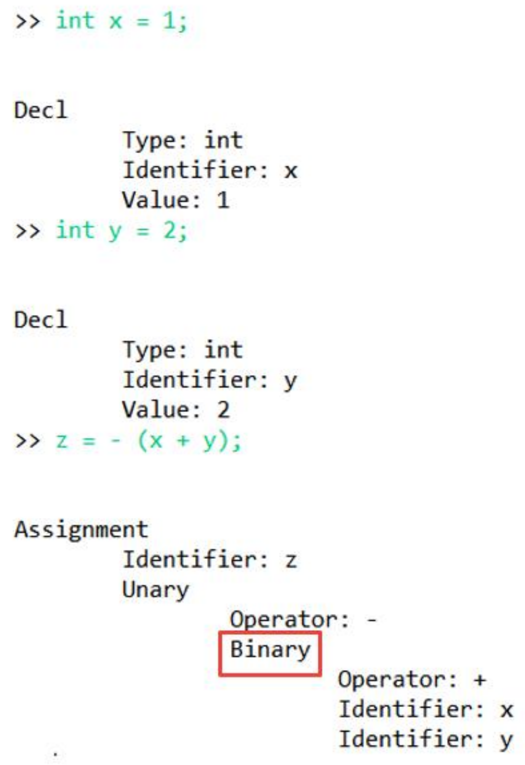


```java
class Indent {
    public static void display(int level, String s) {
        String tab = "";
        System.out.println();
        for (int i=0; i<level; i++)
            tab = tab + "    ";
        System.out.print(tab + s);
    }
}

abstract class Command {
    // Command = Decl | Function | Stmt
    Type type =Type.UNDEF;
    public void display(int l) { }
}
```

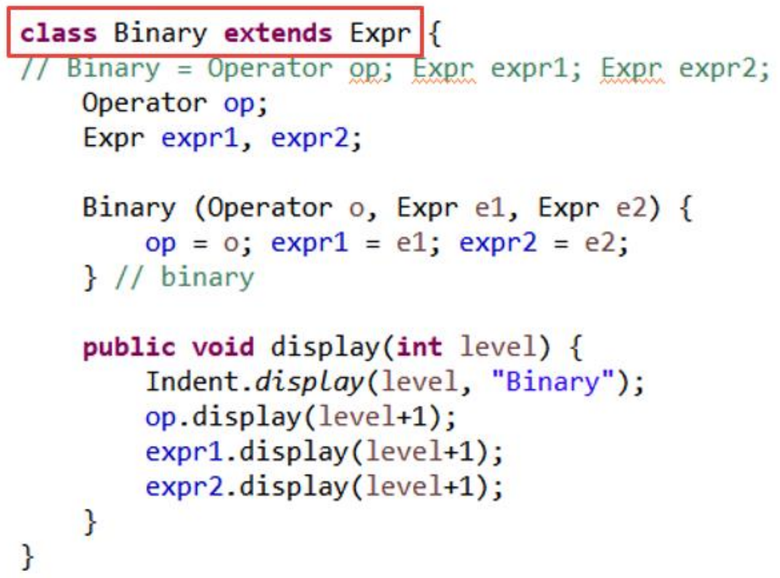
# Lab 02) Implementation of a Parser for Language S

- Parser Implementation for Language S (Java)

  - Tip (AST.java – AST of Expression)



```
>> int x = 1;


Decl
        Type: int
        Identifier: x
        Value: 1
>> int y = 2;


Decl
        Type: int
        Identifier: y
        Value: 2
>> z = - (x + y);


Assignment
        Identifier: z
        Unary
                Operator: -
                Binary
                        Operator: +
                        Identifier: x
                        Identifier: y
```

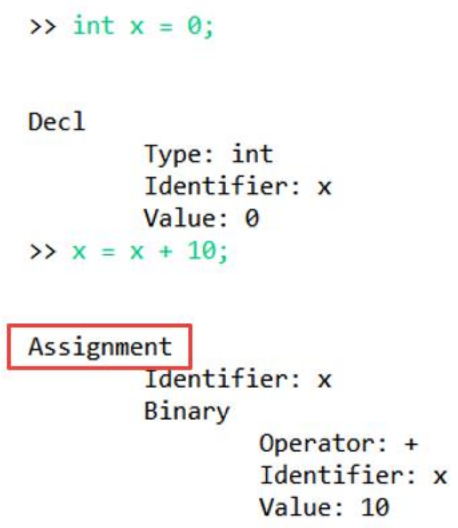


```
class Binary extends Expr {
// Binary = Operator op; Expr expr1; Expr expr2;
    Operator op;
    Expr expr1, expr2;

    Binary (Operator o, Expr e1, Expr e2) {
        op = o; expr1 = e1; expr2 = e2;
    } // binary

    public void display(int level) {
        Indent.display(level, "Binary");
        op.display(level+1);
        expr1.display(level+1);
        expr2.display(level+1);
    }
}
```

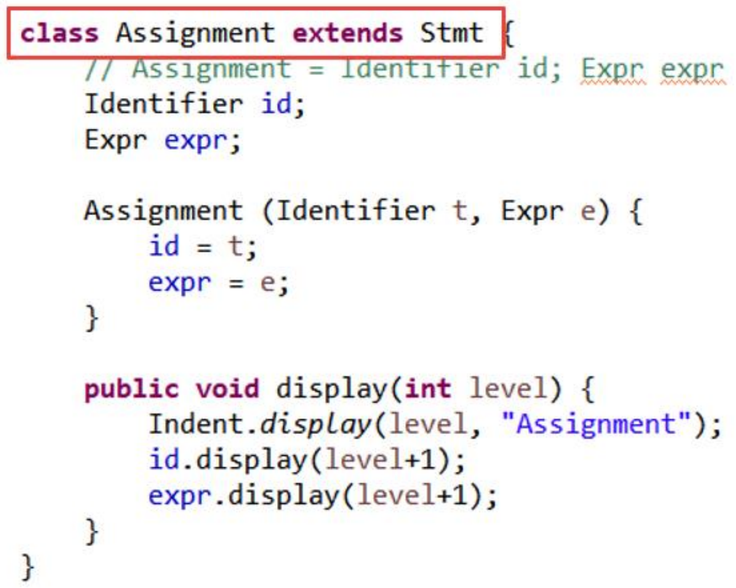
# Lab 02) Implementation of a Parser for Language S

- Parser Implementation for Language S (Java)

  - Tip (AST.java – AST of Statement)



```
>> int x = 0;


Decl
        Type: int
        Identifier: x
        Value: 0
>> x = x + 10;


Assignment
        Identifier: x
        Binary
                Operator: +
                Identifier: x
                Value: 10
```

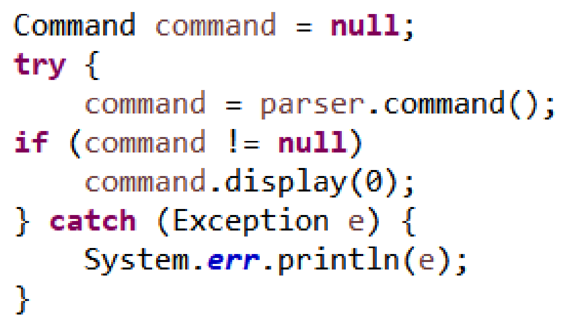


```java
class Assignment extends Stmt {
    // Assignment = Identifier id; Expr expr
    Identifier id;
    Expr expr;

    Assignment (Identifier t, Expr e) {
        id = t;
        expr = e;
    }

    public void display(int level) {
        Indent.display(level, "Assignment");
        id.display(level+1);
        expr.display(level+1);
    }
}
```

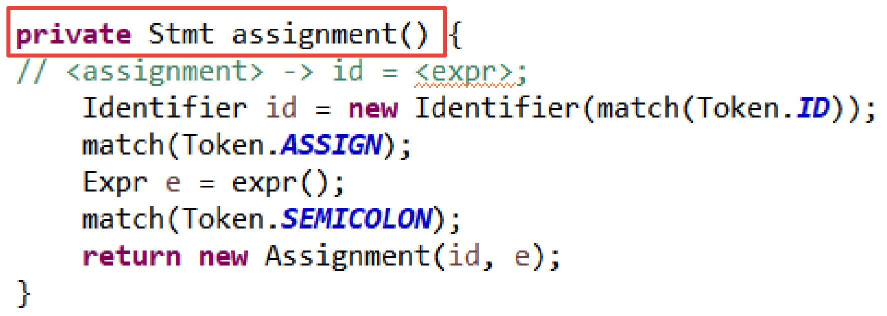
# Lab 02) Implementation of a Parser for Language S

- Parser Implementation for Language S (Java)

  - Tip (Parser.java)



```
Command command = null;
try {
    command = parser.command();
if (command != null)
    command.display(0);
} catch (Exception e) {
    System.err.println(e);
}
```



```
private Stmt assignment() {
// <assignment> -> id = <expr>;
    Identifier id = new Identifier(match(Token.ID));
    match(Token.ASSIGN);
    Expr e = expr();
    match(Token.SEMICOLON);
    return new Assignment(id, e);
}
```