Lab 05

CSC2006: Programming Language Theory

Lab 05) Implementing Parser and Interpreter extensions

- Additional implementation of function-related features of language S (Java)

  - Function-related grammar of language S (EBNF)

```
<command> → <decl> | <stmt> | <function>
<stmt> → …
   | return <expr>;
   | id(<expr> {, <expr>});
<function> → fun <type> id( <params> ) <stmt>
<params> → <type> id {, <type> id}
<type> → int | bool | string | void
<factor> → …
   | id( <expr> {, <expr>});
```

Lab 05) Implementing Parser and Interpreter extensions

- Additional implementation of function-related features of language S (Java)

  (1) Function parsing and AST implementation

  Function definition : <function> → fun <type> id(<params>) <stmt>

  <params> → <type> id {, <type> id}

  Function call : <stmt> → id(<expr> {, <expr>});

  Return statement : <stmt> → return <expr>;

  (2) Function implementation in interpreter

  Function definition

  Function call (if there is a return value)

  Function return

Lab 05) Implementing Parser and Interpreter extensions

Examples and Results

① hi8.s

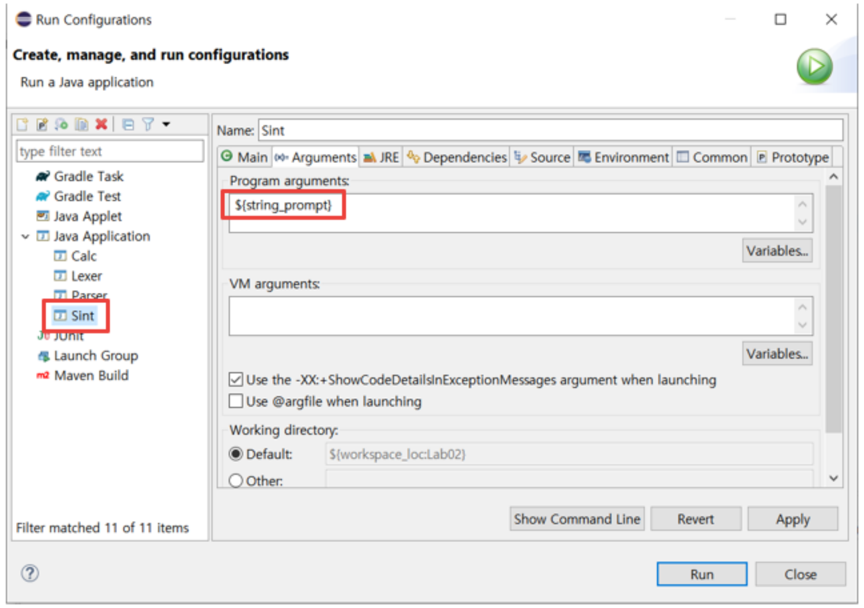
② hi9.s

③ hi10.s

④ hi11.s

⑤ hi12.s

⑥ hi13.s



Run Configurations

Create, manage, and run configurations

Run a Java application

Name: Sint

Program arguments:

```
${string_prompt}
```

VM arguments:

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

[ ] Use @argfile when launching

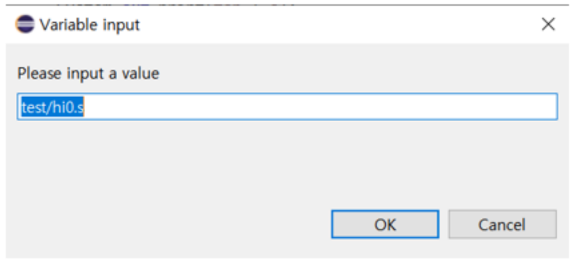
Working directory:

(●) Default: ${workspace_loc:Lab02}

( ) Other:

Filter matched 11 of 11 items

↓



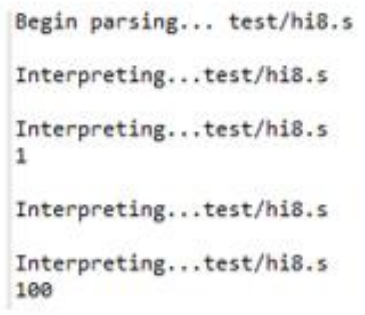
Variable input

Please input a value

```
test/hi0.s
```

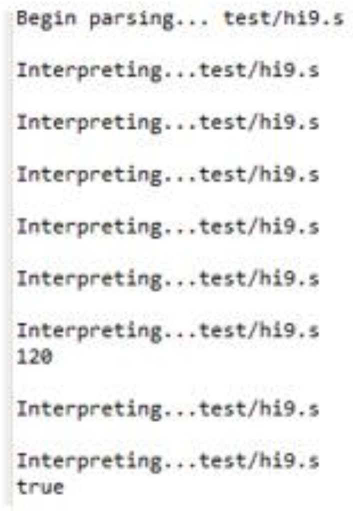
Lab 05) Implementing Parser and Interpreter extensions

hi8.s



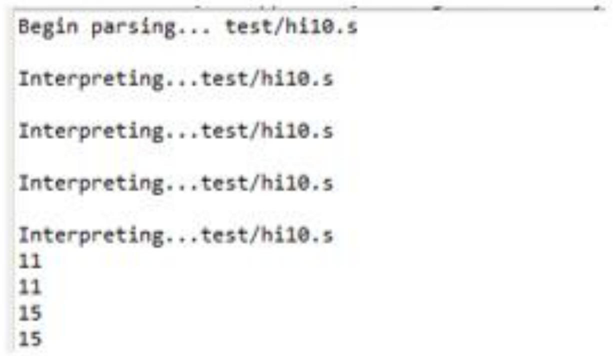
```
Begin parsing... test/hi8.s

Interpreting...test/hi8.s

Interpreting...test/hi8.s
1

Interpreting...test/hi8.s

Interpreting...test/hi8.s
100
```

hi9.s



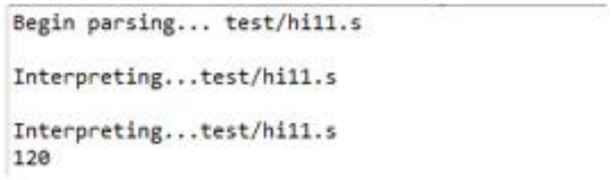
```
Begin parsing... test/hi9.s

Interpreting...test/hi9.s

Interpreting...test/hi9.s

Interpreting...test/hi9.s

Interpreting...test/hi9.s

Interpreting...test/hi9.s

Interpreting...test/hi9.s
120

Interpreting...test/hi9.s

Interpreting...test/hi9.s
true
```

hi10.s



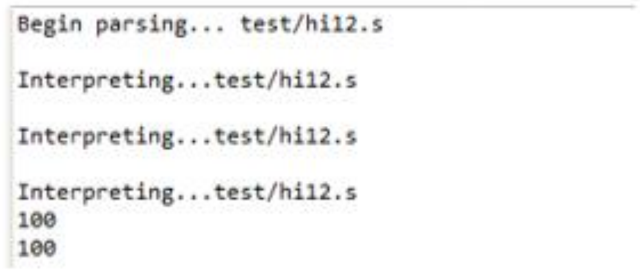
```
Begin parsing... test/hi10.s

Interpreting...test/hi10.s

Interpreting...test/hi10.s

Interpreting...test/hi10.s

Interpreting...test/hi10.s
11
11
15
15
```

hi11.s



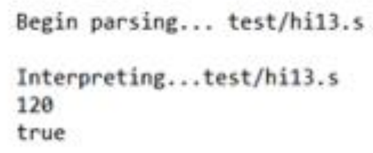
```
Begin parsing... test/hi11.s

Interpreting...test/hi11.s

Interpreting...test/hi11.s
120
```

hi12.s



```
Begin parsing... test/hi12.s

Interpreting...test/hi12.s

Interpreting...test/hi12.s

Interpreting...test/hi12.s
100
100
```

hi13.s

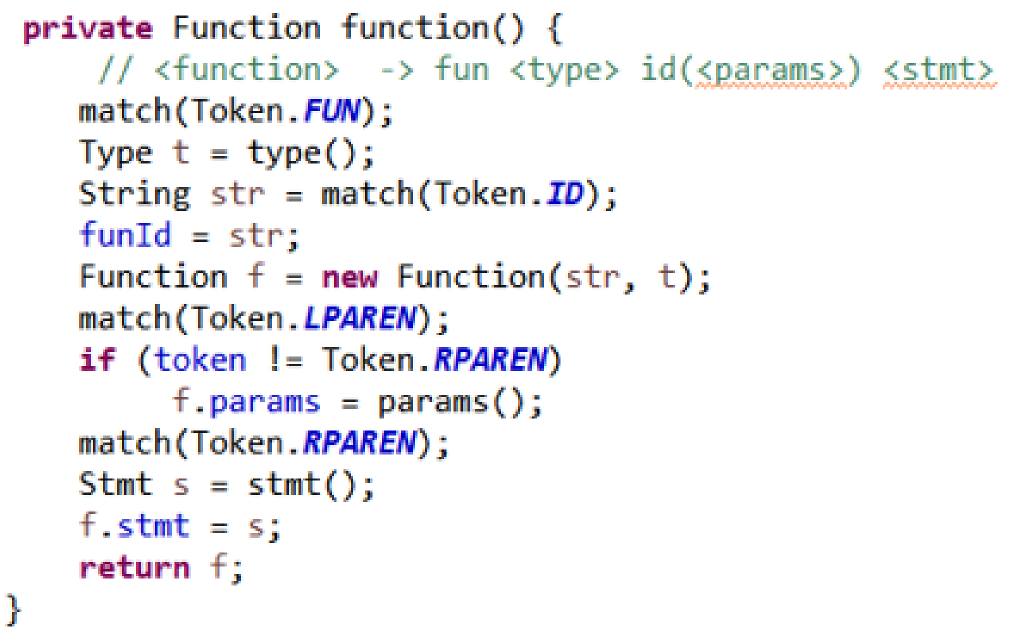


```
Begin parsing... test/hi13.s

Interpreting...test/hi13.s
120
true
```

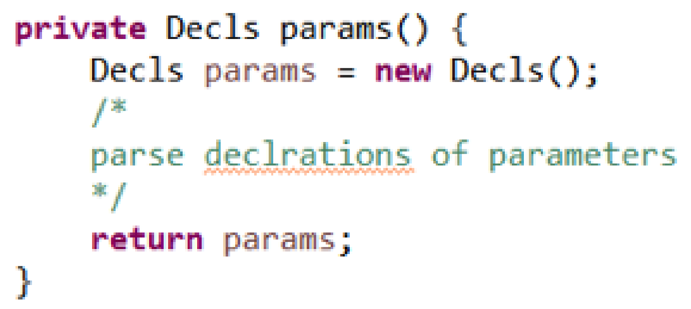
Lab 05) Implementing Parser and Interpreter extensions

- Additional implementation of function-related features of language S (Java)
  - Parser.java

Function Definition



```
private Function function() {
     // <function>  -> fun <type> id(<params>) <stmt>
    match(Token.FUN);
    Type t = type();
    String str = match(Token.ID);
    funId = str;
    Function f = new Function(str, t);
    match(Token.LPAREN);
    if (token != Token.RPAREN)
         f.params = params();
    match(Token.RPAREN);
    Stmt s = stmt();
    f.stmt = s;
    return f;
}
```

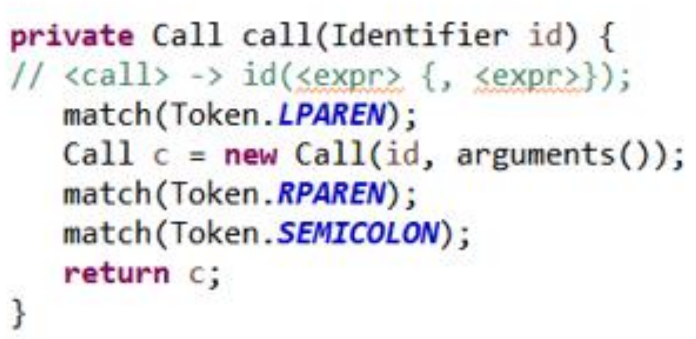


```
private Decls params() {
     Decls params = new Decls();
     /*
     parse declrations of parameters
     */
     return params;
}
```

Lab 05) Implementing Parser and Interpreter extensions

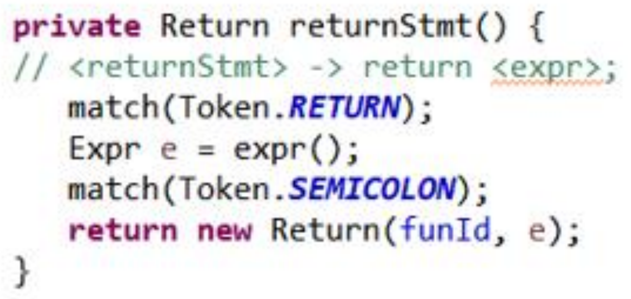
- Additional implementation of function-related features of language S (Java)
  - Parser.java

Function Call



```
private Call call(Identifier id) {
// <call> -> id(<expr> {, <expr>});
    match(Token.LPAREN);
    Call c = new Call(id, arguments());
    match(Token.RPAREN);
    match(Token.SEMICOLON);
    return c;
}
```

Return Statement



```
private Return returnStmt() {
// <returnStmt> -> return <expr>;
    match(Token.RETURN);
    Expr e = expr();
    match(Token.SEMICOLON);
    return new Return(funId, e);
}
```

Lab 05) Implementing Parser and Interpreter extensions

- Additional implementation of function-related features of language S (Java)
  - AST.java

Function Definition

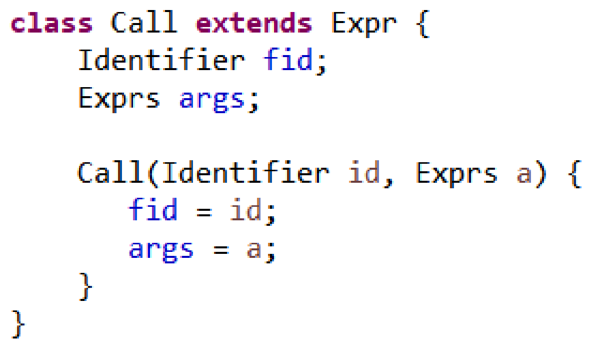
```java
class Function extends Command  {
    // Function = Type type; Identifier id; Decls params; Stmt stmt
    Identifier id;
    Decls params;
    Stmt stmt;

    Function(String s, Type t) {
        id = new Identifier(s); type = t; params = null; stmt = null;
    }

    public String toString ( ) {
       return id.toString()+params.toString();
    }
}
```

```java
class Value extends Expr {
    // Value = int | bool | string | array | function
    protected boolean undef = true;
    Object value = null; // Type type;

    Value(Type t) {
        type = t;
        if (type == Type.INT) value = new ~~Integer~~(0);
        if (type == Type.BOOL) value = new ~~Boolean~~(false);
        if (type == Type.STRING) value = "";
        undef = false;
    }

    Value(Object v) {
        if (v instanceof Function) type = Type.FUN;
        value = v; undef = false;
    }

    Function funValue ( ) {
        if (value instanceof Function)
            return (Function) value;
        else return null;
    }
```

Lab 05) Implementing Parser and Interpreter extensions

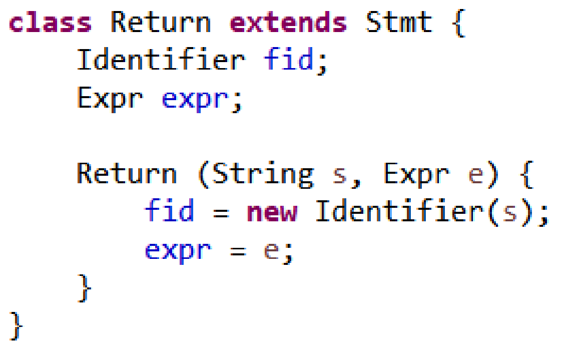
- Additional implementation of function-related features of language S (Java)
  - AST.java

Function Call



```
class Call extends Expr {
    Identifier fid;
    Exprs args;

    Call(Identifier id, Exprs a) {
       fid = id;
       args = a;
    }
}
```

Return Statement

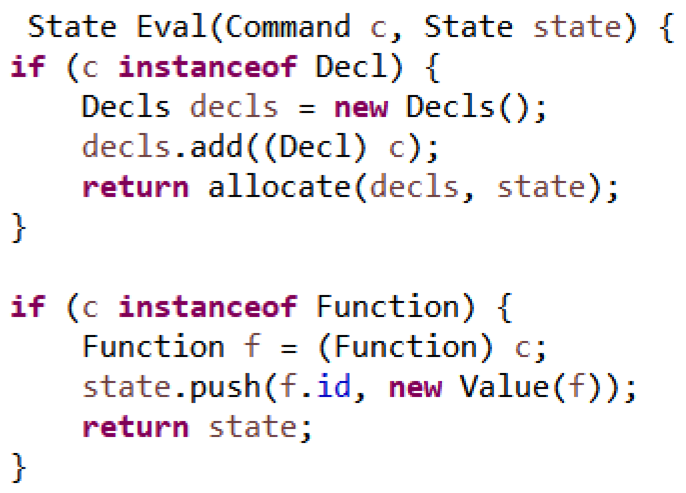


```
class Return extends Stmt {
    Identifier fid;
    Expr expr;

    Return (String s, Expr e) {
        fid = new Identifier(s);
        expr = e;
    }
}
```

Lab 05) Implementing Parser and Interpreter extensions

- Additional implementation of function-related features of language S (Java)
  - Sint.java

Function Definition



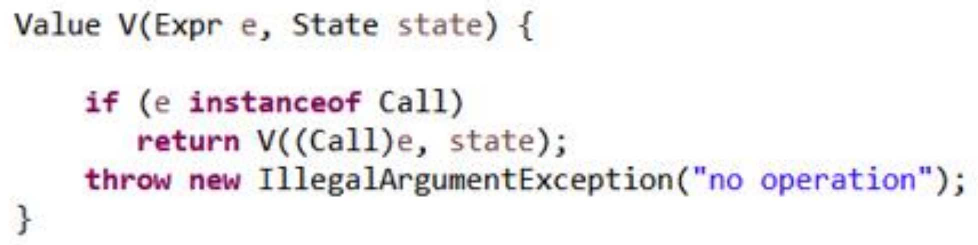
```
 State Eval(Command c, State state) {
if (c instanceof Decl) {
    Decls decls = new Decls();
    decls.add((Decl) c);
    return allocate(decls, state);
}

if (c instanceof Function) {
    Function f = (Function) c;
    state.push(f.id, new Value(f));
    return state;
}
```

Lab 05) Implementing Parser and Interpreter extensions

- Additional implementation of function-related features of language S (Java)
  - Sint.java

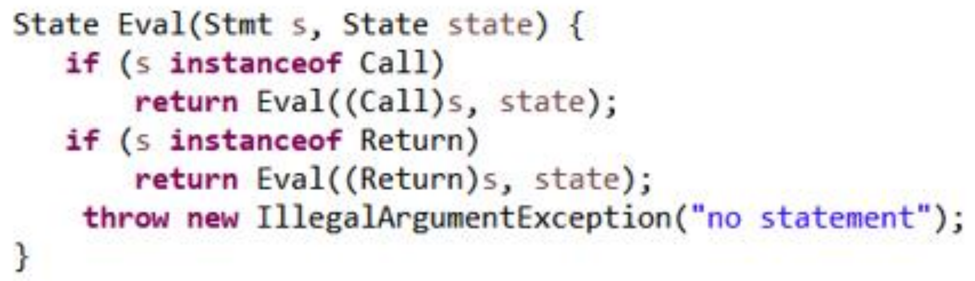
Function Call

Calling a function with a return value

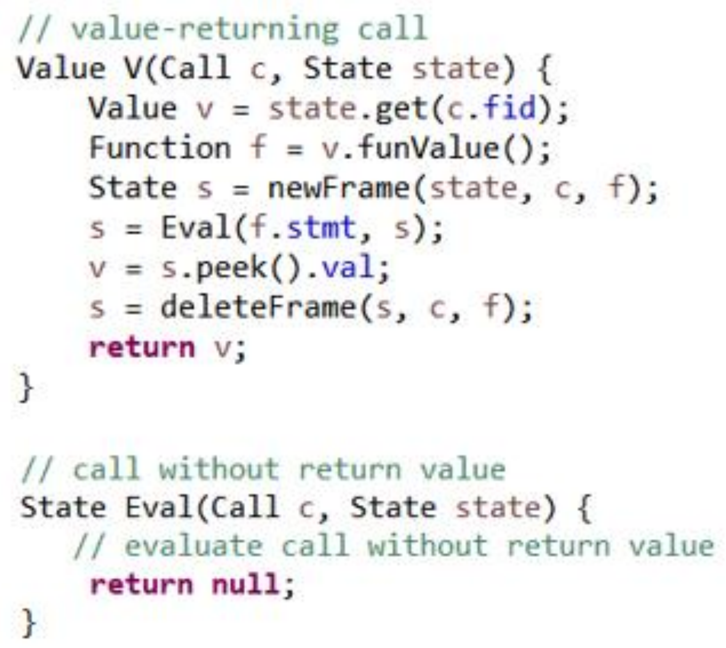


```
Value V(Expr e, State state) {

    if (e instanceof Call)
       return V((Call)e, state);
    throw new IllegalArgumentException("no operation");
}
```

Calling a function without a return value



```
State Eval(Stmt s, State state) {
   if (s instanceof Call)
       return Eval((Call)s, state);
   if (s instanceof Return)
       return Eval((Return)s, state);
    throw new IllegalArgumentException("no statement");
}
```



```
// value-returning call
Value V(Call c, State state) {
     Value v = state.get(c.fid);
     Function f = v.funValue();
     State s = newFrame(state, c, f);
     s = Eval(f.stmt, s);
     v = s.peek().val;
     s = deleteFrame(s, c, f);
     return v;
}

// call without return value
State Eval(Call c, State state) {
    // evaluate call without return value
     return null;
}
```

Lab 05) Implementing Parser and Interpreter extensions

- Additional implementation of function-related features of language S (Java)
  - Sint.java

Frame construction and parameter passing

1. Compute argument values
2. Allocate memory for formal parameters
3. Copy argument values to formal parameters
4. Add a return value entry to the frame just above the parameters

```java
State newFrame (State state, Call c, Function f) {
   if (c.args.size() == 0)
        return state;
   Value val[] = new Value[f.params.size()];
   int i = 0;
   // 인자 값을 계산하여 그 값을 val[]에 저장
   for (Expr e: c.args)
      val[i++] = V(e, state);
   // 현재 상태에 매개변수 기억공간 할당(allocate 사용)
   // 인자의 값을 매개변수에 전달
   // 프레임에 반환 값을 위한 엔트리 추가
   // 상태 반환
   return null;
}

State deleteFrame (State state, Call c, Function f) {
   // 프레임에서 반환 값 엔트리 제거
   // 프레임에서 매개변수를 위한 기억공간 제거(free 사용)
   return null;
}
```

Return Statement

```java
State Eval(Return r, State state) {
    Value v = V(r.expr, state);
    return state.set(new Identifier("return"), v);
}
```