Lab 05

CSE2024: Programming Language Concept

과제 05) 언어 S의 파서, 인터프리터 확장 구현

- 언어 S의 함수 관련 기능 추가 구현 (Java)
  - 언어 S의 함수 관련 문법 (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>});

과제 05) 언어 S의 파서, 인터프리터 확장 구현

- 언어 S의 함수 관련 기능 추가 구현 (Java)

(1) 함수 파싱과 AST 구현

함수 정의 : <function> → fun <type> id(<params>) <stmt>

<params> → <type> id {, <type> id}

함수 호출 : <stmt> → id(<expr> {, <expr>});

리턴문 : <stmt> → return <expr>;

(2) 인터프리터에 함수 구현

함수 정의

함수 호출 (반환값이 있는 경우)

함수 반환

과제 05) 언어 S의 파서, 인터프리터 확장 구현

- 언어 S에 배열 선언 및 사용기능 추가 (Java)
  - 예제 및 결과

    test 폴더에 있는 예제 파일

    ① hi8.s

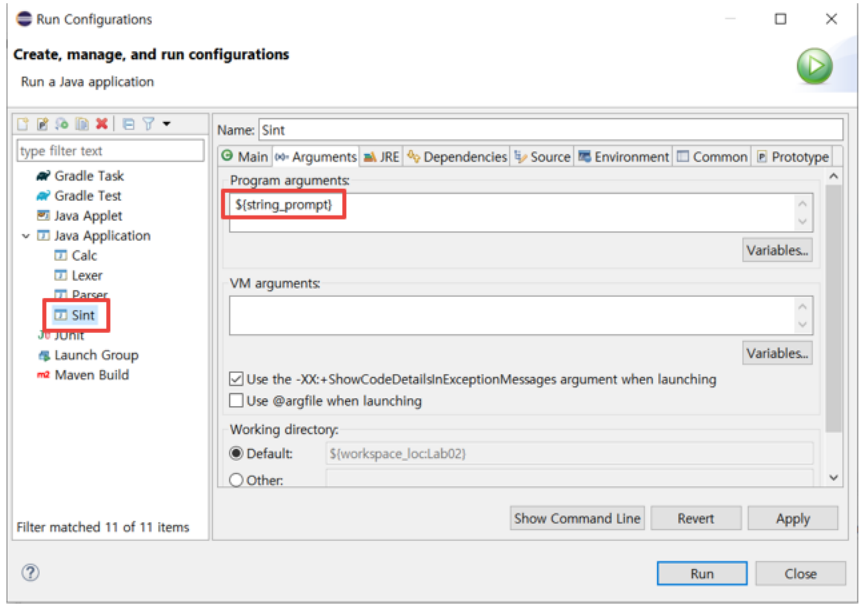
    ② hi9.s

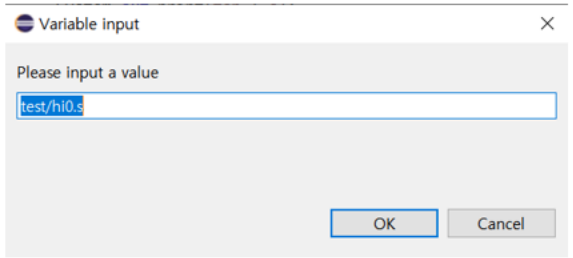
    ③ hi10.s

    ④ hi11.s

    ⑤ hi12.s

    ⑥ hi13.s





과제 05) 언어 S의 파서, 인터프리터 확장 구현

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
```

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
```

hi12.s

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

Interpreting...test/hi12.s

Interpreting...test/hi12.s

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

hi11.s

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

Interpreting...test/hi11.s

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

hi13.s

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

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

과제 05) 언어 S의 파서, 인터프리터 확장 구현

- 언어 S에 함수 관련 기능 추가 구현 (Java)
  - Parser.java

함수 정의

```
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;
}
```

과제 05) 언어 S의 파서, 인터프리터 확장 구현

- 언어 S에 함수 관련 기능 추가 구현 (Java)
  - Parser.java

함수 호출

```
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;
}
```

리턴문

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

과제 05) 언어 S의 파서, 인터프리터 확장 구현

- 언어 S에 함수 관련 기능 추가 구현 (Java)
  - AST.java

함수 정의

```
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();
    }
}
```

```
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;
    }
```

과제 05) 언어 S의 파서, 인터프리터 확장 구현

- 언어 S에 함수 관련 기능 추가 구현 (Java)
  - AST.java

함수 호출

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

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

리턴문

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

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

과제 05) 언어 S의 파서, 인터프리터 확장 구현

- 언어 S에 함수 관련 기능 추가 구현 (Java)
  - Sint.java

함수 정의

```
 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;
}
```

과제 05) 언어 S의 파서, 인터프리터 확장 구현

- 언어 S에 함수 관련 기능 추가 구현 (Java)
  - Sint.java

함수 호출

반환값이 있는 함수 호출

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

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

반환값이 없는 함수 호출

```
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;
}
```

과제 05) 언어 S의 파서, 인터프리터 확장 구현

- 언어 S에 함수 관련 기능 추가 구현 (Java)
  - Sint.java

프레임 구성과 매개변수 전달

1. 인자 값 계산
2. 형식 매개변수들을 위한 기억공간 할당
3. 인자 값들을 형식 매개변수에 복사
4. 프레임에 반환 값 엔트리를 매개변수 바로 위에 추가

함수 반환

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

```
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;
}
```