Lab 03

CSC2006: Programming Language Theory

Lab 03) Implementing an Interpreter for Language S

- Interpreter Implementation for Language S (Java)

  (1) Implement `allocate` and `free` functions for `Let` statement support.

  - Add entries for declared variables to the state (allocate).
  - Remove entries for declared variables from the state (free).

  (2) Implement functionality to perform relational and logical operations according to the syntax of Language S.

  - Extend `binaryOperation()` to support relational operations for integers and strings, as well as logical operations for boolean values.

  (3) Add `do-while` and `for` statements to Language S and implement an interpreter to evaluate these statements.

  <stmt> → ...

  | do <stmt> while (<expr>);

  | for (<type> id = <expr>; <expr>; id = <expr>) <stmt>

Lab 03) Implementing an Interpreter for Language S

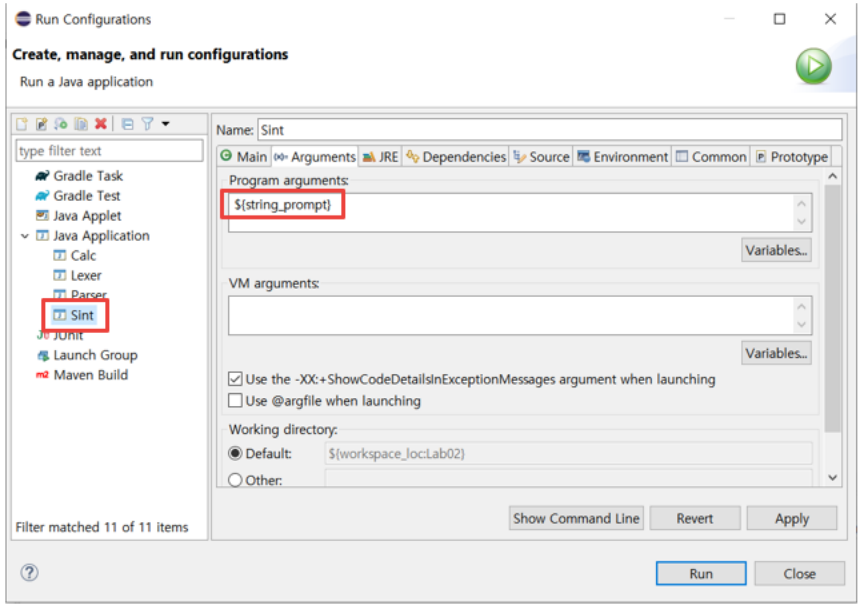
- Interpreter Implementation for Language S (Java)
  - Syntax of Language S (EBNF)

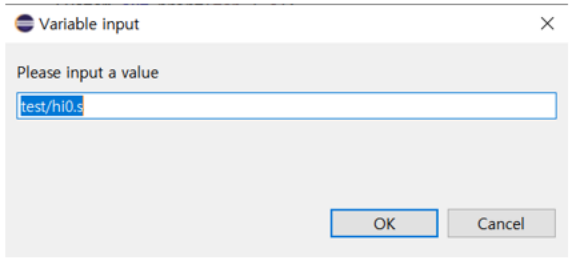
<program> → { <decl> | <stmt> | <function> }
<decl> → <type> id [=<expr>];
<stmt> → id = <expr>;
| '{' <stmts> '}'
| if (<expr>) then <stmt> [else <stmt>]
| while (<expr>) <stmt>
| read id;
| print <expr>;
| let <decls> in <stmts> end;
<stmts> → {<stmt>}
<decls> → {<decl>}
<type> → int | bool | string

Lab 03) Implementing an Interpreter for Language S

- Interpreter Implementation for Language S (Java)
  - Examples and Results
    - Example files in the test folder

① hi0.s
② hi2.s
③ hi3.s
④ hi4.s
⑤ hi5.s
⑥ hi6.s
⑦ hi7.s
+ ⑧, ⑨ 2 tests for string relational operations
+ ⑩, ⑪ 2 tests for logical operations
+ ⑫ for ⑬ do-while 1 test for each





Lab 03) Implementing an Interpreter for Language S

hi0.s

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

Print
        Value: hello world!
Interpreting...test/hi0.s
hello world!

Decl
        Type: string
        Identifier: s
        Value: hello world!
Interpreting...test/hi0.s

Print
        Identifier: s
Interpreting...test/hi0.s
hello world!
```

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
Interpreting...test/hi2.s
2^n ?
1
```

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
Interpreting...test/hi3.s
1 + 2 + ... + n?
0
```

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
Interpreting...test/hi4.s
12
0
```

Lab 03) Implementing an Interpreter 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
Interpreting...test/hi5.s
1
1
1
1
2
2
1
3
3
1
4
4
2
1
2
2
2
4
2
3
6
2
4
8
3
1
3
3
2
6
3
3
9
3
4
12
```

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
Interpreting...test/hi6.s
1
3
3
```

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
Interpreting...test/hi7.s
1
2
0
```

Lab 03) Implementing an Interpreter for Language S

Relational Operation Test

- Examples and Results

stringrelop1.s

stringrelop1.s

```
1  string i = "apple";
2  string j = "banana";
3  if (i == j)
4      then print "strings are equal";
5  else
6      print "strings are not equal";
```

```
Begin parsing... test/stringrelop1.s

Decl
        Type: string
        Identifier: i
        Value: apple
Interpreting...test/stringrelop1.s

Decl
        Type: string
        Identifier: j
        Value: banana
Interpreting...test/stringrelop1.s

If
        Binary
                Operator: ==
                Identifier: i
                Identifier: j
        Print
                Value: strings are equal
        Print
                Value: strings are not equal
Interpreting...test/stringrelop1.s
strings are not equal
```

stringrelop2.s

stringrelop2.s

```
1  string i = "apple";
2  string j = "banana";
3  if (i < j)
4      then print "banana is located behind the dictionary";
5  else
6      print "apple is located behind the dictionary";
```

```
Begin parsing... test/stringrelop2.s

Decl
        Type: string
        Identifier: i
        Value: apple
Interpreting...test/stringrelop2.s

Decl
        Type: string
        Identifier: j
        Value: banana
Interpreting...test/stringrelop2.s

If
        Binary
                Operator: <
                Identifier: i
                Identifier: j
        Print
                Value: banana is located behind the dictionary
        Print
                Value: apple is located behind the dictionary
Interpreting...test/stringrelop2.s
banana is located behind the dictionary
```

Lab 03) Implementing an Interpreter for Language S

Logical Operation Test

logicalop1.s

logicalop1.s

```
1   bool i = true;
2   bool j = true;
3   if (i & j)
4       then print "both are true";
5   else
6       print "one or both are false";
```

```
Begin parsing... test/logicalop1.s

Decl
        Type: bool
        Identifier: i
        Value: true
Interpreting...test/logicalop1.s

Decl
        Type: bool
        Identifier: j
        Value: true
Interpreting...test/logicalop1.s

If
        Binary
                Operator: &
                Identifier: i
                Identifier: j
        Print
                Value: both are true
        Print
                Value: one or both are false
Interpreting...test/logicalop1.s
both are true
```

logicalop2.s

logicalop2.s

```
1   bool i = true;
2   bool j = false;
3   if (i | j)
4       then print "one or both are true";
5   else
6       print "both are false";
```

```
Begin parsing... test/logicalop2.s

Decl
        Type: bool
        Identifier: i
        Value: true
Interpreting...test/logicalop2.s

Decl
        Type: bool
        Identifier: j
        Value: false
Interpreting...test/logicalop2.s

If
        Binary
                Operator: |
                Identifier: i
                Identifier: j
        Print
                Value: one or both are true
        Print
                Value: both are false
Interpreting...test/logicalop2.s
one or both are true
```

Lab 03) Implementing an Interpreter for Language S

`for` Loop and `do-while` Loop Test

for.s

```
for (int i=0; i<10; i = i+1) print i;
```

```
Begin parsing... test/for.s

Let
        Decls
                Decl
                        Type: int
                        Identifier: i
                        Value: 0
        Stmts
                While
                        Binary
                                Operator: <
                                Identifier: i
                                Value: 10
                        Stmts
                                Print
                                        Identifier: i
                                Assignment
                                        Identifier: i
                                        Binary
                                                Operator: +
                                                Identifier: i
                                                Value: 1

Interpreting...test/for.s
0
1
2
3
4
5
6
7
8
9
```

dowhile.s

```
let
    int i = 5;
in
    do {
        print i;
        i = i - 1;
    }
    while (i > 0);
end;
```

```
Begin parsing... test/dowhile.s

Let
        Decls
                Decl
                        Type: int
                        Identifier: i
                        Value: 5
        Stmts
                Stmts
                        Stmts
                                Print
                                        Identifier: i
                                Assignment
                                        Identifier: i
                                        Binary
                                                Operator: -
                                                Identifier: i
                                                Value: 1
                        While
                                Binary
                                        Operator: >
                                        Identifier: i
                                        Value: 0
                                Stmts
                                        Print
                                                Identifier: i
                                        Assignment
                                                Identifier: i
                                                Binary
                                                        Operator: -
                                                        Identifier: i
                                                        Value: 1

Interpreting...test/dowhile.s
5
4
3
2
1
```

Lab 03) Implementing an Interpreter for Language S

- Interpreter Implementation for Language S (Java)
  - Tip (Sint.java – Let)

```
State Eval(Let l, State state) {
    State s = allocate(l.decls, state);
    s = Eval(l.stmts,s);
    return free(l.decls, s);
}

State allocate (Decls ds, State state) {
    if (ds != null) {
        // add entries for declared variables on the state
    }
    return null;
}

State free (Decls ds, State state) {
    if (ds != null) {
        // free the entries for declared variables from the state
     }
     return null;
}
```

Lab 03) Implementing an Interpreter for Language S

- Interpreter Implementation for Language S (Java)
  - Tip (Sint.java – binaryOperation)

```
Value binaryOperation(Operator op, Value v1, Value v2) {
    check(!v1.undef && !v2.undef,"reference to undef value");
    switch (op.val) {
    case "+":
        return new Value(v1.intValue() + v2.intValue());
    case "-":
        return new Value(v1.intValue() - v2.intValue());
    case "*":
        return new Value(v1.intValue() * v2.intValue());
    case "/":
        return new Value(v1.intValue() / v2.intValue());

    // relational operations

    // logical operations and or not

    default:
        throw new IllegalArgumentException("no operation");
    }
}
```

Lab 03) Implementing an Interpreter for Language S

- Interpreter Implementation for Language S (Java)
  - Tip (Sint.java – dowhile, for)

dowhile : Repeat at least once

do <stmt> while (<expr>);

=

{
<stmt>
  while (<expr>)
        <stmt>
}

cf. while : Repeat zero or more times

for

for (<type> id = <expr>; <expr>; id=<expr>) <stmt>

=

let
  <type> id = <expr>
in
  while (<expr>)
        <stmt>
end