Lab 04

CSC2006: Programming Language Theory

Lab 04) Extension of a Parser and Interpreter for Language S

- **Add array declaration and usage functionality to Language S (Java)**

Array: A data structure composed of contiguous variables of the same type
Array Element: An individual variable that forms part of the array, represented by the name and an index

(1) Array Declaration

<decl> → …
| <type> id[n];

(2) Array Element Assignment

<stmt> → …
| id[<expr>] = <expr>;

(3) Array Element Usage

<factor> → …
| id[<expr>]

Lab 04) Extension of a Parser and Interpreter for Language S

- Add array declaration and usage functionality to Language S (Java)

Examples and Results

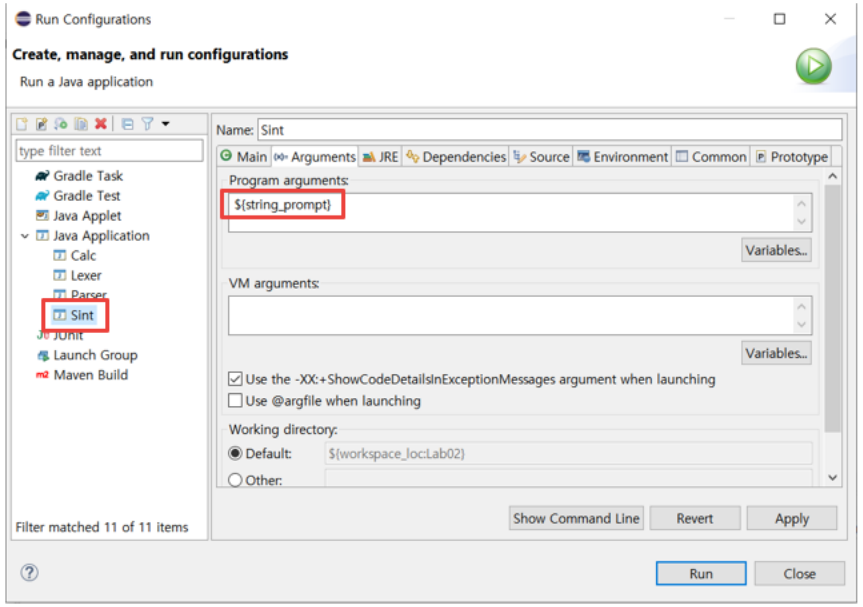
① q1.s

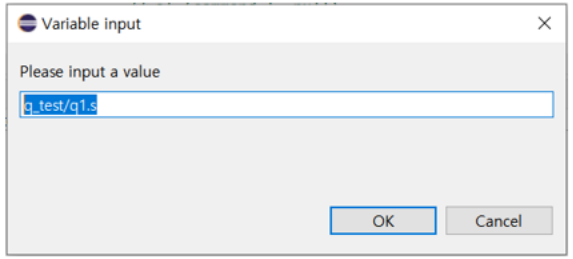
② q2.s

③ q3.s

④ q4.s

⑤ q5.s





Lab 04) Extension of a Parser and Interpreter for Language S

q1.s

```
Begin parsing... test/q1.s

Interpreting...test/q1.s

Interpreting...test/q1.s

Interpreting...test/q1.s

Interpreting...test/q1.s

Interpreting...test/q1.s
7
```

q2.s

```
Begin parsing... test/q2.s

Interpreting...test/q2.s
Sum
10
```

q3.s

```
Begin parsing... test/q3.s

Interpreting...test/q3.s
Max
6
```

q4.s

```
Begin parsing... test/q4.s

Interpreting...test/q4.s
Sum of positive numbers
22
```

q5.s

```
Begin parsing... test/q5.s

Interpreting...test/q5.s
A
U
F
G
A
B
E
true
```

Lab 04) Extension of a Parser and Interpreter for Language S

- **Add array declaration and usage functionality to Language S (Java)**
  - Parser.java

```
private Decl decl() {
// <decl> -> <type> id[n];
// <decl> -> <type> id [=<expr>];
   return null;
}


private Stmt assignment() {
// <assignment> -> id[<expr>] = <expr>;
// <assignment> -> id = <expr>;
  return null;
}


private Expr factor() {
    // <factor> -> [-](id | id'['<expr>']' | <call> | literal | '('<aexp>')')
    return null;
}
```

Lab 04) Extension of a Parser and Interpreter for Language S

- **Add array declaration and usage functionality to Language S (Java)**
  - AST.java – base code (provided)

```
class Decl extends Command {
    // Decl = Type type; Identifier id
    Type type;
    Identifier id;
    Expr expr = null;
    int arraysize = 0;

    Decl (String s, Type t) {
        id = new Identifier(s); type = t;
    } // declaration

    Decl (String s, Type t, int n) {
        id = new Identifier(s); type = t; arraysize = n;
    } // array declaration

    Decl (String s, Type t, Expr e) {
        id = new Identifier(s); type = t; expr = e;
    } // declaration

    public void display (int level) {
        Indent.display(level, "Decl");
        type.display(level+1);
        id.display(level+1);
        if (expr != null)
            expr.display(level+1);
        // arraysize
    }
}
```

<type> id[n];

```
class Array extends Expr {
    // Array = Identifier id; Expr expr
    Identifier id;
    Expr expr = null;

    Array(Identifier s, Expr e) {id = s; expr = e;}

    public String toString( ) { return id.toString(); }

    public boolean equals (Object obj) {
        String s = ((Array) obj).id.toString();
        return id.equals(s);
    }

    public void display(int level) {
        Indent.display(level, "Array");
        System.out.print(": " + id);
        // expr.display(level+1);
    }
}
```

id[<expr>]

```
class Assignment extends Stmt {
    // Assignment = Identifier id; Expr expr
    Identifier id;
    Array ar = null;
    Expr expr;

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

    Assignment (Array a, Expr e) {
        ar = a;
        expr = e;
    }

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

id[<expr>] = <expr>;

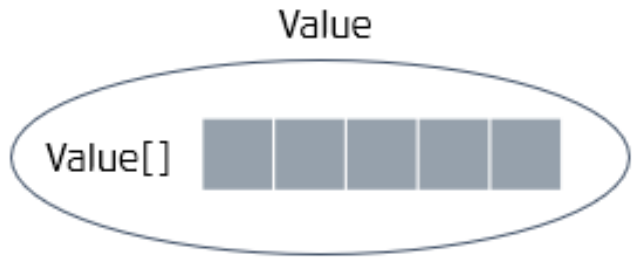
Lab 04) Extension of a Parser and Interpreter for Language S

- **Add array declaration and usage functionality to Language S (Java)**
  - AST.java – base code (provided)

```
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 Integer) type = Type.INT;
        if (v instanceof Boolean) type = Type.BOOL;
        if (v instanceof String) type = Type.STRING;
        if (v instanceof Function) type = Type.FUN;
        if (v instanceof Value[]) type = Type.ARRAY;
        value = v; undef = false;
    }

    Object value() { return value; }

    int intValue( ) {
        if (value instanceof Integer)
            return ((Integer) value).intValue();
        return 0;
    }
```

```
    boolean boolValue( ) {
        if (value instanceof Boolean)
            return ((Boolean) value).booleanValue();
        return false;
    }

    String stringValue ( ) {
        if (value instanceof String)
            return (String) value;
        return null;
    }

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

    Value[] arrValue ( ) {
        if (value instanceof Value[])
            return (Value[]) value;
        else return null;
    }
```

```
    Type type ( ) { return type; }

    public String toString( ) {
        if (type == Type.INT) return "" + intValue();
        if (type == Type.BOOL) return "" + boolValue();
        if (type == Type.STRING) return "" + stringValu
        if (type == Type.FUN) return "" + funValue();
        if (type == Type.ARRAY) return "" + arrValue();
        return "undef";
    }

    public void display(int level) {
        Indent.display(level, "Value");
        System.out.print(": " + value);
    }
}
```

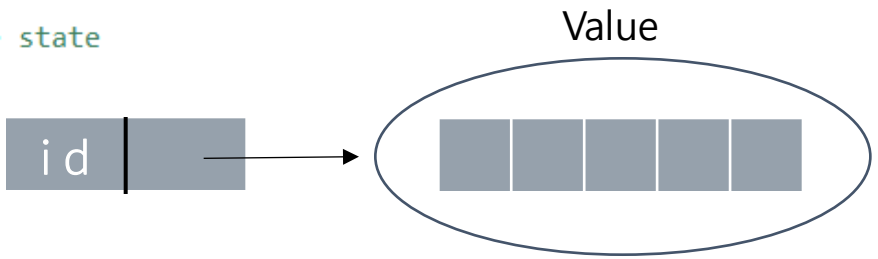


Lab 04) Extension of a Parser and Interpreter for Language S

- **Add array declaration and usage functionality to Language S (Java)**
  - Sint.java

<type> id[n];

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



id[<expr>]

```
Value V(Expr e, State state) {
    if (e instanceof Array) {
       // id[<expr>]
    }
```

id[<expr>] = <expr>;

```
State Eval(Assignment a, State state) {
    // replace array element in array represented by array name
    return state;
}
```