Lab 01

CSC2006: Programming Language Theory

Lab 01) Calculator (Calc.java)

- Recursive-descent parser / Calculator extension implementation (Java)

  - Grammar (EBNF)

    <expr> → <bexp> {& <bexp> | '|' <bexp>} | !<expr> | true | false
    <bexp> → <aexp> [<relop> <aexp>]
    <relop> → == | != | < | > | <= | >=
    <aexp> → <term> {+ <term> | - <term>}
    <term> → <factor> {* <factor> | / <factor>}
    <factor> → [-] (<number> | (<aexp>))
    <number> → <digit> {<digit>}

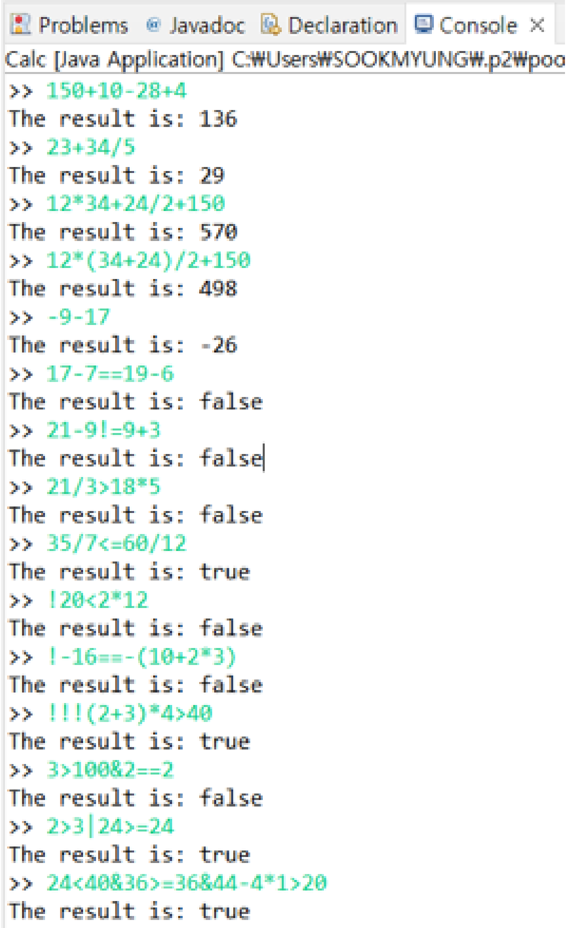
  - Extension

    ① Add subtraction (-), division (/)
    ② Add comparison operations (==, !=, >, <, !)
    ③ Add logical operations (&, |, !)

Lab 01) Calculator (Calc.java)

- Recursive-descent parser / Calculator extension implementation (Java)
  - Example & results

① 150+10−28+4
② 23+34/5
③ 12*34+24/2+150
④ 12*(34+24)/2+150
⑤ −9−17
⑥ 17−7==19−6
⑦ 21−9!=9+3
⑧ 21/3>18*5
⑨ 35/7<=60/12
⑩ !20<2*12
⑪ !−16==−(10+2*3)
⑫ !!!(2+3)*4>40
⑬ 3>100&2==2
⑭ 2>3|24>=24
⑮ 24<40&36>=36&44−4*1>20

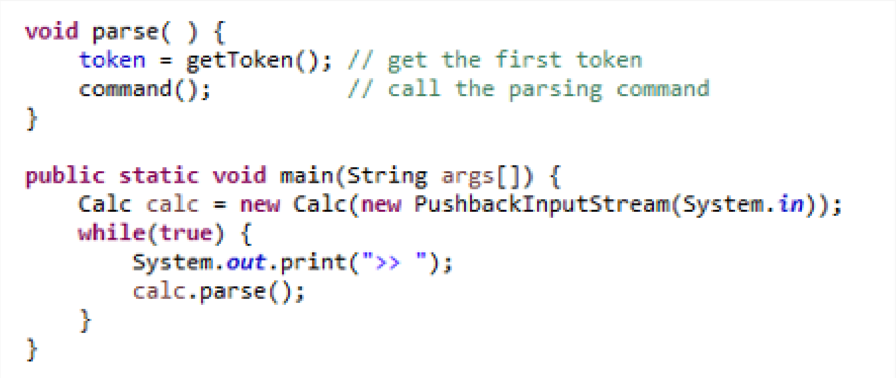


Input →
Output →

```
Calc [Java Application] C:₩Users₩SOOKMYUNG₩.p2₩poo
>> 150+10-28+4
The result is: 136
>> 23+34/5
The result is: 29
>> 12*34+24/2+150
The result is: 570
>> 12*(34+24)/2+150
The result is: 498
>> -9-17
The result is: -26
>> 17-7==19-6
The result is: false
>> 21-9!=9+3
The result is: false
>> 21/3>18*5
The result is: false
>> 35/7<=60/12
The result is: true
>> !20<2*12
The result is: false
>> !-16==-(10+2*3)
The result is: false
>> !!!(2+3)*4>40
The result is: true
>> 3>100&2==2
The result is: false
>> 2>3|24>=24
The result is: true
>> 24<40&36>=36&44-4*1>20
The result is: true
```

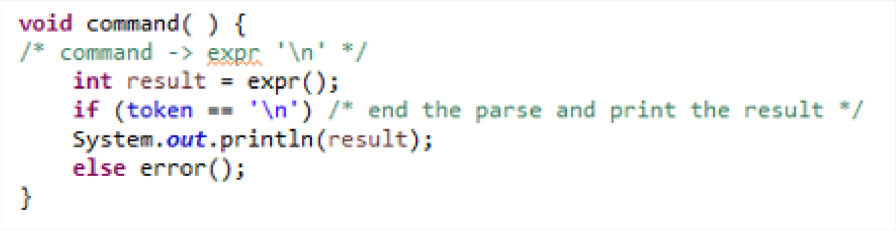
Lab 01) Calculator (Calc.java)

- Recursive-descent parser / Calculator extension implementation (Java)
  - Tips

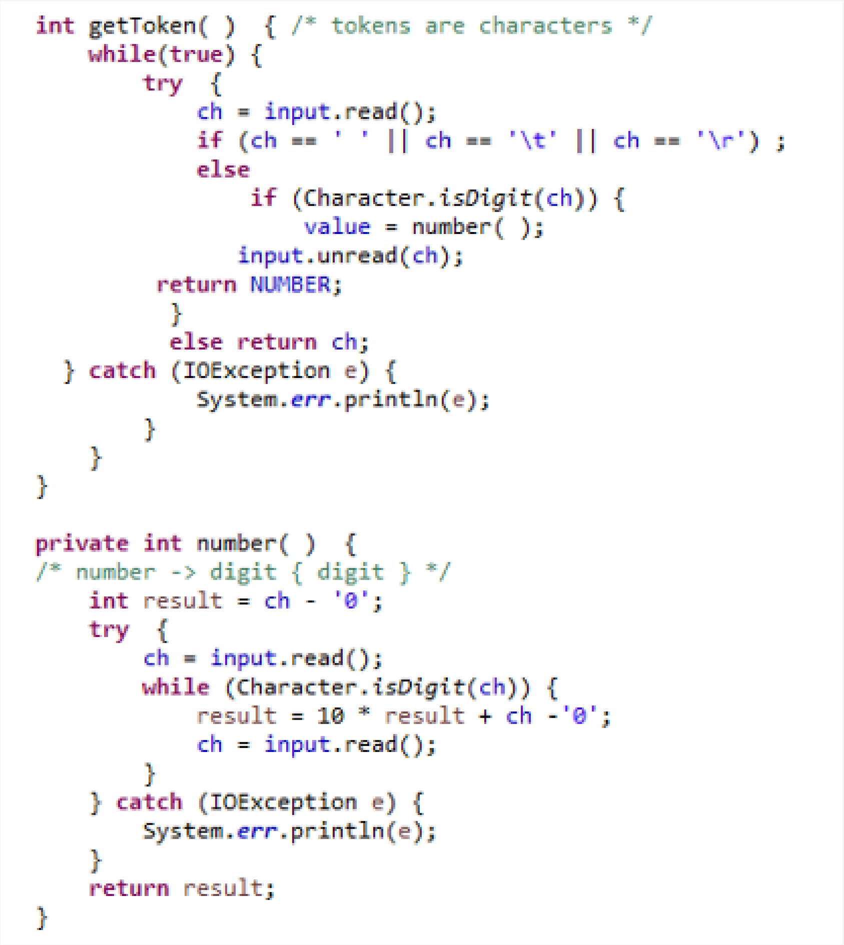


```
void parse( ) {
    token = getToken(); // get the first token
    command();          // call the parsing command
}

public static void main(String args[]) {
    Calc calc = new Calc(new PushbackInputStream(System.in));
    while(true) {
        System.out.print(">> ");
        calc.parse();
    }
}
```



```
void command( ) {
/* command -> expr '\n' */
    int result = expr();
    if (token == '\n') /* end the parse and print the result */
    System.out.println(result);
    else error();
}
```



```
int getToken( )  { /* tokens are characters */
    while(true) {
        try  {
            ch = input.read();
            if (ch == ' ' || ch == '\t' || ch == '\r') ;
            else
                if (Character.isDigit(ch)) {
                    value = number( );
               input.unread(ch);
         return NUMBER;
          }
          else return ch;
  } catch (IOException e) {
            System.err.println(e);
        }
    }
}

private int number( )  {
/* number -> digit { digit } */
    int result = ch - '0';
    try  {
        ch = input.read();
        while (Character.isDigit(ch)) {
            result = 10 * result + ch -'0';
            ch = input.read();
        }
    } catch (IOException e) {
        System.err.println(e);
    }
    return result;
}
```

Lab 01) Calculator (Calc.java)

- Recursive-descent parser / Calculator extension implementation (Java)
    - Tips

<expr> → <bexp> {& <bexp> | ‘|’ <bexp>} | !<expr> | true | false

<bexp> → <aexp> [<relop> <aexp>]

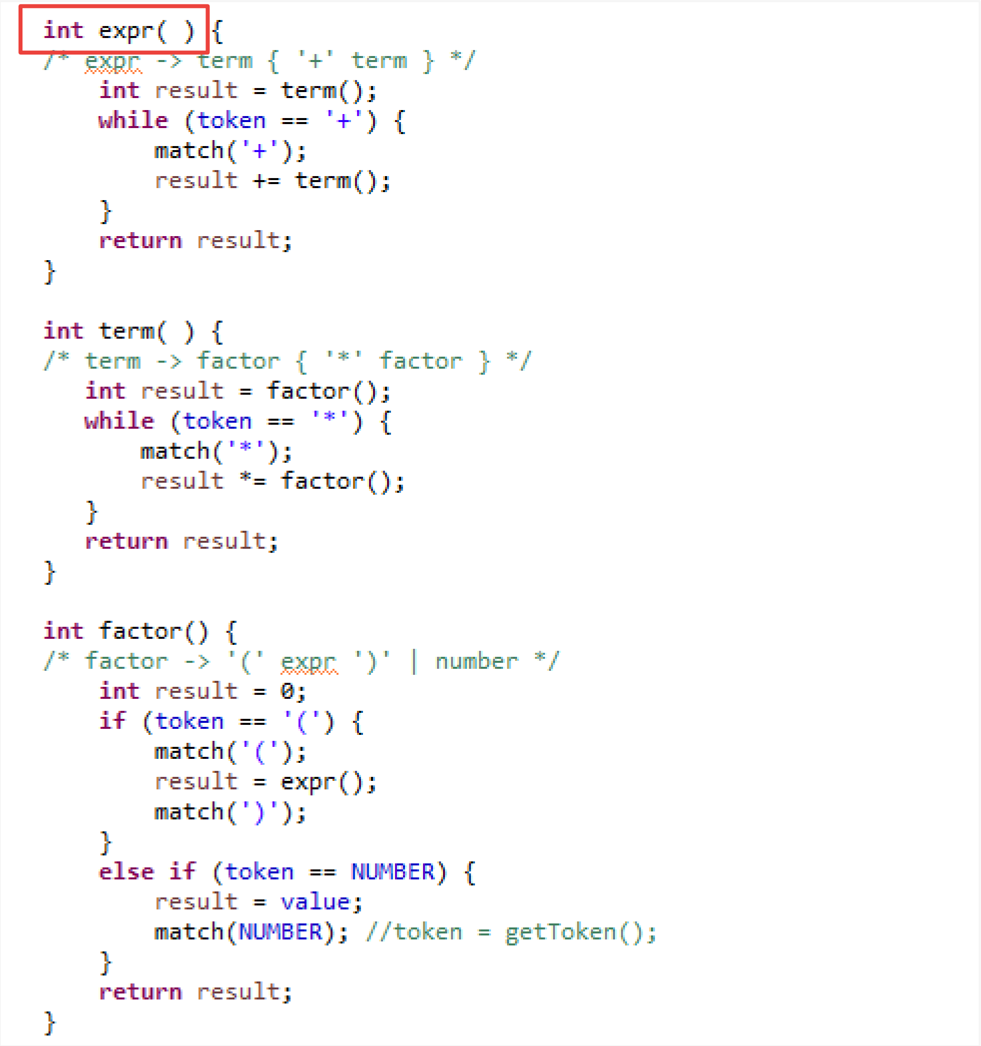
<relop> → == | != | < | > | <= | >=

<aexp> → <term> {+ <term> | - <term>}

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

<factor> → [-] (<number> | (<aexp>))

<number> → <digit> {<digit>}



```
int expr( ) {
/* expr -> term { '+' term } */
    int result = term();
    while (token == '+') {
        match('+');
        result += term();
    }
    return result;
}

int term( ) {
/* term -> factor { '*' factor } */
   int result = factor();
   while (token == '*') {
       match('*');
       result *= factor();
   }
   return result;
}

int factor() {
/* factor -> '(' expr ')' | number */
    int result = 0;
    if (token == '(') {
        match('(');
        result = expr();
        match(')');
    }
    else if (token == NUMBER) {
        result = value;
        match(NUMBER); //token = getToken();
    }
    return result;
}
```

Lab 01) Calculator (Calc.java)

- Recursive-descent parser / Calculator extension implementation (Java)
    - Tips

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

<bexp> → <aexp> [<relop> <aexp>]

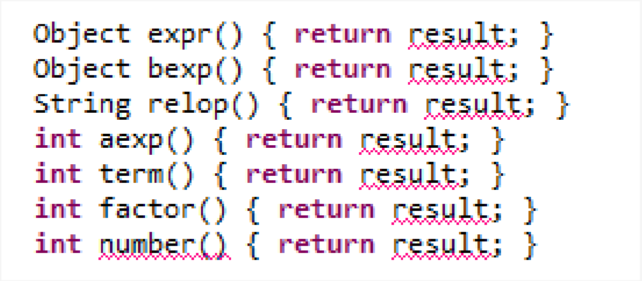
<relop> → == | != | < | > | <= | >=

<aexp> → <term> {+ <term> | - <term>}

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

<factor> → [-] (<number> | (<aexp>))

<number> → <digit> {<digit>}



```
Object expr() { return result; }
Object bexp() { return result; }
String relop() { return result; }
int aexp() { return result; }
int term() { return result; }
int factor() { return result; }
int number() { return result; }
```

Lab 01) Calculator (Calc.java)

- Recursive-descent parser / Calculator extension implementation (Java)

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

<bexp> → <aexp> [<relop> <aexp>]

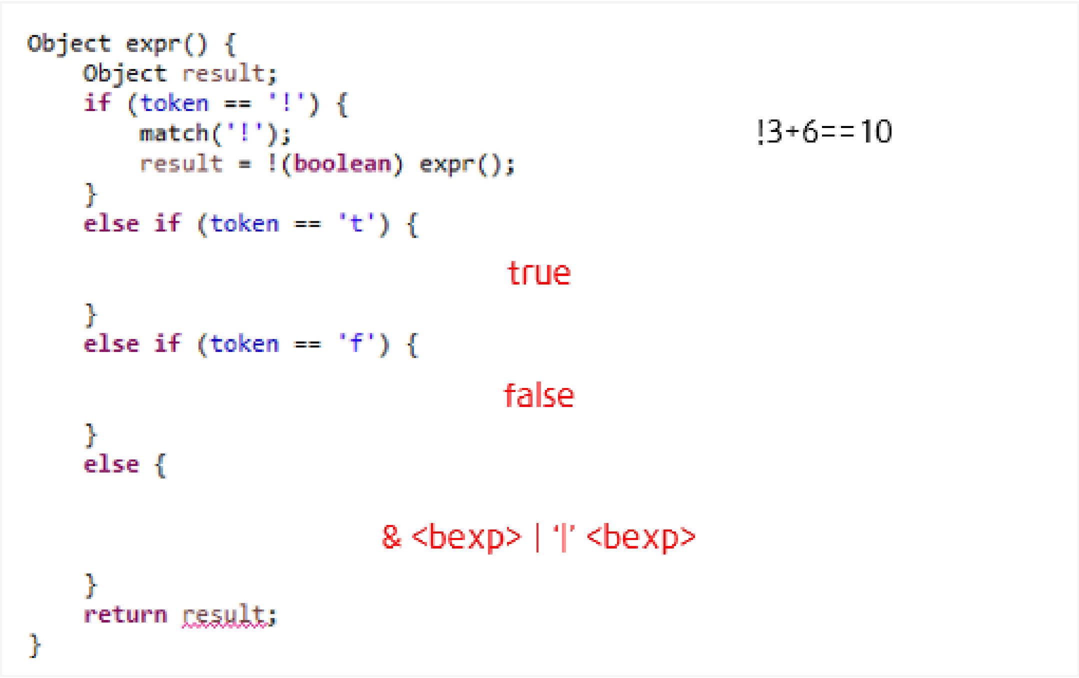
<relop> → == | != | < | > | <= | >=

<aexp> → <term> {+ <term> | - <term>}

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

<factor> → [-] (<number> | (<aexp>))

<number> → <digit> {<digit>}



```
Object expr() {
    Object result;
    if (token == '!') {
        match('!');
        result = !(boolean) expr();
    }
    else if (token == 't') {
                              true
    }
    else if (token == 'f') {
                              false
    }
    else {
                        & <bexp> | '|' <bexp>
    }
    return result;
}
```

!3+6==10