Advanced Object-Oriented Programming, Spring 2021

Homework Assignment #4

Due midnight Wednesday, May 19, 2021

Instructions

1. If any question is unclear, please ask for a clarification.
2. You may try to reuse as much of the source code supplemented as possible.
3. Unless stated otherwise, all the line numbers for the program listings are for reference only.
4. You are required to do all the homework assignments on Linux and use g++ version 10.2.0 or later.
5. You are required to give your TA a demo of your program. Make sure that your program can compile and run on the server machine, which will be used for the demo.
6. For the program that you write, you are required to include a Makefile. Otherwise, your homework will not be graded—meaning that you will receive zero marks.
7. Unless stated otherwise, you are required to work on the homework assignment individually.
8. No late homework will be accepted.

Programming Project

This assignment requires that you implement a scanner using inheritance in C++ for the language defined in Appendix A of the dragon book, second edition,[1] the source and class hierarchy of which are given in Listing 1 to Listing 8.

Listing 1: Main.java

```
1 package main;
2
3 import java.io.*;
4 import lexer.*;
5
6 public class Main {
7     public static void main(String[] args) throws IOException
8     {
```

用在宣告方法時，表示該方法可能要丟擲異常

[1] Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman, *Compilers: Principles, Techniques, and Tools, Second Edition*, Addison-Wesley, 2007. Fortunately, you don't have to understand the material as far as the homework assignment is concerned. Instead, all you have to do is figure out how the Java version works and follow the class hierarchy for the C++ implementation.

```
        Lexer lexer = new Lexer();
        try {
            while (true) {
                Token t = lexer.scan();
                System.out.println(t.toString());
            }
        }
        catch (IOException e) {
            System.out.println(e.getMessage());  判斷是否到文件末端
        }
    }
}
```

Listing 2: Lexer.java 詞法分析器

```
package lexer;

import java.io.*;
import java.util.*;

public class Lexer {
    public static int line = 1;
    char peek = ' ';
    Hashtable words = new Hashtable();
    void reserve(Word w) 保存
    {
        words.put(w.lexeme, w);
    }
    public Lexer() constructor
    {
        reserve(new Word("if",          Tag.IF)         );
        reserve(new Word("else",        Tag.ELSE)       );
        reserve(new Word("while",       Tag.WHILE)      );
        reserve(new Word("do",          Tag.DO)         );
        reserve(new Word("break",       Tag.BREAK)      );
        reserve(Word.True);
        reserve(Word.False);
        reserve(Type.Int);
        reserve(Type.Char);
        reserve(Type.Bool);
        reserve(Type.Float);
    }
    void readch() throws IOException 讀取字元1   用在宣告方法時，表示該方法可能要丟擲異常
    {
        int i = System.in.read();
        if (i != -1)
            peek = (char) i;
        else
            throw new IOException("End of file reached");  判斷是否到文件末端
    }
    boolean readch(char c) throws IOException  讀取字元2
    {
        readch();
        if (peek != c)
            return false;
        peek = ' ';
        return true;
    }
    public Token scan() throws IOException  處理字符
    {
        for ( ; ; readch()) { 處理無意義的字元
            if (peek == ' ' || peek == '\t')
                continue;
            else if (peek == '\n')
```

```
50              line = line + 1;
51          else
52              break;
53      }
54      switch (peek) { 處理特殊符號
55      case '&':
56          if (readch('&'))
57              return Word.and;
58          else
59              return new Token('&');
60      case '|':
61          if (readch('|'))
62              return Word.or;
63          else
64              return new Token('|');
65      case '=':
66          if (readch('='))
67              return Word.eq;
68          else
69              return new Token('=');
70      case '!':
71          if (readch('='))
72              return Word.ne;
73          else
74              return new Token('!');
75      case '<':
76          if (readch('='))
77              return Word.le;
78          else
79              return new Token('<');
80      case '>':
81          if (readch('='))
82              return Word.ge;
83          else
84              return new Token('>');
85      }
86
87      if (Character.isDigit(peek)) {  處理數字
88          int v = 0;
89          do {
90              v = 10 * v + Character.digit(peek, 10);  十進位
91              readch();
92          } while (Character.isDigit(peek));
93          if (peek != '.')
94              return new Num(v); 如果為整數（沒有.）則回傳v
95          float x = v;
96          float d = 10;
97          for (;;) {
98              readch();
99              if (!Character.isDigit(peek)) 如果不是數字，迴圈就停止
100                 break;
101             x = x + Character.digit(peek, 10) / d;
102             d = d * 10;
103         }
104         return new Real(x); 回傳real number
105     }
106
107     if (Character.isLetter(peek)){ 處理字母
108         StringBuffer b = new StringBuffer();
109         do {
110             b.append(peek);
111             readch();
112         } while (Character.isLetterOrDigit(peek));
113         String s = b.toString();
```

```
114             Word w = (Word) words.get(s);
115             if (w != null)
116                 return w;
117             w = new Word(s, Tag.ID);
118             words.put(s, w);
119             return w;
120         }
121 
122         Token tok = new Token(peek);
123         peek = ' ';
124         return tok;
125     }
126 }
```

Listing 3: Tag.java

```
1  package lexer;
2  
3  public class Tag {
4      public final static int
5          AND     = 256,
6          BASIC   = 257,
7          BREAK   = 258,
8          DO      = 259,
9          ELSE    = 260,
10         EQ      = 261,
11         FALSE   = 262,
12         GE      = 263,
13         ID      = 264,
14         IF      = 265,
15         INDEX   = 266,
16         LE      = 267,
17         MINUS   = 268,
18         NE      = 269,
19         NUM     = 270,
20         OR      = 271,
21         REAL    = 272,
22         TEMP    = 273,
23         TRUE    = 274,
24         WHILE   = 275;
25 }
```

Listing 4: Token.java

```
1  package lexer;
2  
3  public class Token {
4      public final int tag;
5      public Token(int t)
6      {
7          tag = t;
8      }
9      public String toString()
10     {
11         return "" + (char) tag;
12     }
13 }
```

Listing 5: Word.java

```
1  package lexer;
2  
3  public class Word extends Token {  繼承
```

```
4     public String lexeme = "";
5     public Word(String s, int tag)    constructor
6     {
7         super(tag); //呼叫父類別中的建構子
8         lexeme = s;
9     }
10    public String toString()
11    {
12        return lexeme;
13    }
14    public static final Word
15        and     = new Word("&&",        Tag.AND),
16        or      = new Word("||",        Tag.OR),
17        eq      = new Word("==",        Tag.EQ),
18        ne      = new Word("!=",        Tag.NE),
19        le      = new Word("<=",        Tag.LE),
20        ge      = new Word(">=",        Tag.GE),
21        minus   = new Word("minus",     Tag.MINUS),
22        True    = new Word("true",      Tag.TRUE),
23        False   = new Word("false",     Tag.FALSE),
24        temp    = new Word("t",         Tag.TEMP);
25 }
```

Java: final

1. 類別：當宣告在類別上時，該類別就無法被繼承！
2. 函數：當一個函數被宣告為final時，則繼承他的子類別無法覆寫
3. 變數：當一個變數被宣告為final時，意思是他是一個常數，是無法被修改的。

Listing 6: Type.java

```
1  package lexer;
2
3  public class Type extends Word {
4      public int width = 0;
5      public Type(String s, int tag, int w)  constructor
6      {
7          super(s, tag);
8          width = w;
9      }
10     public static final Type
11         Int     = new Type("int",       Tag.BASIC, 4),
12         Float   = new Type("float",     Tag.BASIC, 8),
13         Char    = new Type("char",      Tag.BASIC, 1),
14         Bool    = new Type("bool",      Tag.BASIC, 1);
15     public static boolean numeric(Type p)
16     {
17         if (p == Type.Char || p == Type.Int || p == Type.Float)
18             return true;
19         else
20             return false;
21     }
22     public static Type max(Type p1, Type p2)
23     {
24         if (!numeric(p1) || !numeric(p2))
25             return null;
26         else if (p1 == Type.Float || p2 == Type.Float)
27             return Type.Float;
28         else if (p1 == Type.Int || p2 == Type.Int)
29             return Type.Int;
30         else
31             return Type.Char;
32     }
33 }
```

Listing 7: Num.java

```
1  package lexer;
2
```

```
3  public class Num extends Token {
4      public final int value;
5      public Num(int v)
6      {
7          super(Tag.NUM);
8          value = v;
9      }
10     public String toString()
11     {
12         return "" + value;
13     }
14 }
```

Listing 8: Real.java

```
1  package lexer;
2
3  public class Real extends Token {
4      public final float value;
5      public Real(float v)
6      {
7          super(Tag.REAL);
8          value = v;
9      }
10     public String toString()
11     {
12         return "" + value;
13     }
14 }
```

To make it easier for you to test your implementation, also given are the test programs (Listings 9 and 11) and the output of the scanner in C++ (Listings 10 and 12). Note that the output of the scanner in C++ is different from that in Java.

Listing 9: Test program 1

```
1{ 2 int i; 3 int j; 4 float v; 5 float x; 6 float[100] a; 7 while (true) { 8 do i = i+1; while (a[i] < v); 9 do j
= j-1; while (a[j] > v); 10 if (i >= j) break; 11 x = a[i]; 12 a[i] = a[j]; 13 a[j] = x; 14 } 15 }
1 {
2      int i;
3      int j;
4      float v;
5      float x;
6      float[100] a;
7      while (true) {
8          do i = i+1; while (a[i] < v);
9          do j = j-1; while (a[j] > v);
10         if (i >= j) break;
11         x = a[i];
12         a[i] = a[j];
13         a[j] = x;
14     }
15 }
```

Listing 10: Output of the test program 1

```
1  Token: {        ({)
2  Token: int      (BASIC)
3  Token: i        (ID)
4  Token: ;        (;)
5  Token: int      (BASIC)
6  Token: j        (ID)
7  Token: ;        (;)
8  Token: float    (BASIC)
```

```
Token: v        (ID)
Token: ;        (;)
Token: float    (BASIC)
Token: x        (ID)
Token: ;        (;)
Token: float    (BASIC)
Token: [        ([)
Token: 100      (NUM)
Token: ]        (])
Token: a        (ID)
Token: ;        (;)
Token: while    (WHILE)
Token: (        (()
Token: true     (TRUE)
Token: )        ())
Token: {        ({)
Token: do       (DO)
Token: i        (ID)
Token: =        (=)
Token: i        (ID)
Token: +        (+)
Token: 1        (NUM)
Token: ;        (;)
Token: while    (WHILE)
Token: (        (()
Token: a        (ID)
Token: [        ([)
Token: i        (ID)
Token: ]        (])
Token: <        (<)
Token: v        (ID)
Token: )        ())
Token: ;        (;)
Token: do       (DO)
Token: j        (ID)
Token: =        (=)
Token: j        (ID)
Token: -        (-)
Token: 1        (NUM)
Token: ;        (;)
Token: while    (WHILE)
Token: (        (()
Token: a        (ID)
Token: [        ([)
Token: j        (ID)
Token: ]        (])
Token: >        (>)
Token: v        (ID)
Token: )        ())
Token: ;        (;)
Token: if       (IF)
Token: (        (()
Token: i        (ID)
Token: >=       (GE)
Token: j        (ID)
Token: )        ())
Token: break    (BREAK)
Token: ;        (;)
Token: x        (ID)
Token: =        (=)
Token: a        (ID)
Token: [        ([)
Token: i        (ID)
Token: ]        (])
```

```
Token: ;        (;)
Token: a        (ID)
Token: [        ([)
Token: i        (ID)
Token: ]        (])
Token: =        (=)
Token: a        (ID)
Token: [        ([)
Token: j        (ID)
Token: ]        (])
Token: ;        (;)
Token: a        (ID)
Token: [        ([)
Token: j        (ID)
Token: ]        (])
Token: =        (=)
Token: x        (ID)
Token: ;        (;)
Token: }        (})
Token: }        (})
End of file reached
```

Listing 11: Test program 2

```
{
    int i;
    int j;
    float[10][10] a;
    i = 0;
    while ( i < 10 ) {
        j = 0;
        while ( j < 10 ) {
            a[i][j] = 0;
            j = j+1;
        }
        i = i+1;
    }
    i = 0;
    while ( i < 10 ) {
        a[i][j] = 1;
        i = i+1;
    }
}
```

Listing 12: Output of the test program 2

```
Token: {        ({)
Token: int      (BASIC)
Token: i        (ID)
Token: ;        (;)
Token: int      (BASIC)
Token: j        (ID)
Token: ;        (;)
Token: float    (BASIC)
Token: [        ([)
Token: 10       (NUM)
Token: ]        (])
Token: [        ([)
Token: 10       (NUM)
Token: ]        (])
Token: a        (ID)
Token: ;        (;)
Token: i        (ID)
```

```
Token: =        (=)
Token: 0        (NUM)
Token: ;        (;)
Token: while    (WHILE)
Token: (        (()
Token: i        (ID)
Token: <        (<)
Token: 10       (NUM)
Token: )        ())
Token: {        ({)
Token: j        (ID)
Token: =        (=)
Token: 0        (NUM)
Token: ;        (;)
Token: while    (WHILE)
Token: (        (()
Token: j        (ID)
Token: <        (<)
Token: 10       (NUM)
Token: )        ())
Token: {        ({)
Token: a        (ID)
Token: [        ([)
Token: i        (ID)
Token: ]        (])
Token: [        ([)
Token: j        (ID)
Token: ]        (])
Token: =        (=)
Token: 0        (NUM)
Token: ;        (;)
Token: j        (ID)
Token: =        (=)
Token: j        (ID)
Token: +        (+)
Token: 1        (NUM)
Token: ;        (;)
Token: }        (})
Token: i        (ID)
Token: =        (=)
Token: i        (ID)
Token: +        (+)
Token: 1        (NUM)
Token: ;        (;)
Token: }        (})
Token: i        (ID)
Token: =        (=)
Token: 0        (NUM)
Token: ;        (;)
Token: while    (WHILE)
Token: (        (()
Token: i        (ID)
Token: <        (<)
Token: 10       (NUM)
Token: )        ())
Token: {        ({)
Token: a        (ID)
Token: [        ([)
Token: i        (ID)
Token: ]        (])
Token: [        ([)
Token: j        (ID)
Token: ]        (])
Token: =        (=)
```

```
Token: 1        (NUM)
Token: ;        (;)
Token: i        (ID)
Token: =        (=)
Token: i        (ID)
Token: +        (+)
Token: 1        (NUM)
Token: ;        (;)
Token: }        (})
Token: }        (})
End of file reached
```

Grading Policy

The grading policy for this assignment is as follows:

- Make sure that a **Makefile**, which contains at least three targets—**all**, **dep**, and **clean**—is provided. Otherwise, the grade for your program will be zero.
- 100 points if your program compiles without errors and warnings and runs correctly.