

```

package lab3;

import java.util.Enumeration;

public class Fsa {

    static int GOTO[][] = {
/*      B      D      S      O      C      S      T      N      S      W      O */
/*      l      e      e      r      o      t      e      o      t      i      t */
/*      a      f      m          m      r      r      n      a      t      h */
/*      n      i      i          m          m          r      h      e */
/*      k      n          a          i          t          r */
/*          e          n          */
/*          a          */
/*          l          */
/*          */
{          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0} /* 0 */
};

    static int ACTION[][] = {
/*      B      D      S      O      C      S      T      N      S      W      O */
/*      l      e      e      r      o      t      e      o      t      i      t */
/*      a      f      m          m      r      r      n      a      t      h */
/*      n      i      i          m          m          r      h      e */
/*      k      n          a          i          t          r */
/*          e          n          */
/*          a          */
/*          l          */
/*          */
{          1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1} /* 0 */
};

    public Fsa(Enumeration e) {
        // Uncomment the line below and each token processed will be
        // echoed.
        // ((TokenEnum)e).setDebug(true);

        SymbolTable symboltable = new SymbolTable();

        int state = 0;

        String
            lhs      = "",
            term     = "",
            nonterm  = "$FINAL$";

        while (e.hasMoreElements()) {
            Token t = (Token)e.nextElement();

            System.out.println("    Read token type " + t.type() + ": " + t);

            int action = ACTION[state][t.type()];
            int newstate = GOTO[state][t.type()];

```

```
        System.out.println("State " + state +
            " Performing action " + action + " and going to " +
newstate);

        switch (action) {
            case 1: /* do nothing */
                break;
        }

        state = newstate;
    }
    if (state != 0) oops("End in bad state: " + state);
}

void oops(String s) {
    System.err.println("Error: " + s);
    System.out.println("ABORT");
    System.exit(-1);
}
}
```

```
package lab3;
import java.util.Enumeration;

public class RLgram {

    public static void main (String args[]) {

        if (args.length != 1) throw new Error("Usage: java rlgram file-or-
URL");

        Enumeration e = new TokenEnum(args[0], false);
        Fsa fsa = new Fsa(e);
    }
}
```

```
package lab3;

public class SymbolNotFoundError extends Error {
    public SymbolNotFoundError(String s) {
        super(s);
    }
}
```

```

package lab3;

import java.util.Hashtable;

public class SymbolTable {
    private static Object Terminal, NonTerminal;
    private Hashtable<String, Object> ht;

    static {
        Terminal = new Object();
        NonTerminal = new Object();
    }

    public SymbolTable() {
        ht = new Hashtable<String, Object>();
    }

    public final boolean inTable(String s) {
        return(ht.get(s.trim()) != null);
    }

    public boolean isTerminal(String s) {
        s = s.trim();
        if (inTable(s)) {
            return(ht.get(s) == Terminal);
        }
        else throw new SymbolNotFoundError(
            "Symbol \"" + s + "\" not previous entered in table"
        );
    }

    public void enterTerminal(String s) {
        ht.put(s.trim(), Terminal);
    }

    public void enterNonTerminal(String s) {
        ht.put(s.trim(), NonTerminal);
    }

    public static void main(String[] args) { // just for checking
        String t = "I am a terminal";
        String nt = "I am a nonterminal";
        SymbolTable st = new SymbolTable();
        st.enterNonTerminal(nt);
        st.enterTerminal(t);
        System.out.println("Nonterminal: " + st.isTerminal(nt+" "));
        System.out.println("Terminal: " + st.isTerminal(t+""));
        System.out.println("Undeclared: " + st.isTerminal("missing"));
    }
}

```

```

package lab3;

public class Token {

    public final static int
        Blank      = 0,
        Define     = 1,
        Semi       = 2,
        Or         = 3,
        Comma      = 4,
        Str        = 5,
        Terminal   = 6,
        Non        = 7,
        Start      = 8,
        With       = 9,
        Other      = 10;

    private static String[] nice;

    static {
        nice = new String[11];
        nice[Blank   ] = "Blank";
        nice[Define  ] = "Define";
        nice[Semi    ] = ";";
        nice[Or      ] = "|";
        nice[Comma   ] = ",";
        nice[Str     ] = "Str";
        nice[Terminal] = "Terminal";
        nice[Non     ] = "Non";
        nice[Start   ] = "Start";
        nice[With    ] = "With";
        nice[Other   ] = "Other";
    }

    int tokentype;
    String info;

    public Token(int tokentype) {
        this(tokentype, "");
    }

    public Token(int tokentype, String info) {
        this.tokentype = tokentype;
        this.info      = info;
    }

    public final int type() {
        return(tokentype);
    }

    public final String strValue() {
        return(info);
    }
}

```

```
public String toString() {  
    return("Token " +  
        "<" + nice[tokenype] + "> " +  
        "\"" + info + "\""  
        );  
    }  
}
```

```
package lab3;

import java.util.Enumeration;
import java.io.DataInputStream;
import common.OpenFile;
import autogen.Scanner;

public class TokenEnum implements Enumeration {
    private Scanner scanner;
    private Token next;
    private boolean debug;
    public TokenEnum(String f, boolean debug) {
        this.debug = debug;
        DataInputStream dis = new OpenFile(f);
        scanner = new Scanner(dis);
        advance();
    }
    public void setDebug(boolean val) { this.debug = val; }
    private void advance() {
        try {
            next = scanner.yylex();
        } catch (java.io.IOException e) {
            next = null;
        }
    }
    public boolean hasMoreElements() {
        return(next != null);
    }
    public Object nextElement() {
        Object ans = next;
        if (debug) System.out.println("TokenEnum returning " + ans);
        advance();
        return(ans);
    }
}
```