Klase koje se koriste u implementaciji tabele simbola:

```
package SymbolTable;

// klasa za predstavljanje simbola
public class SymbolNode {

        public static int TYPE = 0;
        public static int VARIABLE = 1;
        public String name;
        public int kind;
        public Type type;
        public SymbolNode next;

        public SymbolNode( String symbolName,
                           int symbolKind,
                           Type symbolType,
                           SymbolNode nextSymbol )
    {
        name = symbolName;
        kind = symbolKind;
        type = symbolType;
        next = nextSymbol;
    }
}
```

```
package SymbolTable;

// klasa za predstavljanje tipova
public class Type extends SymbolNode {

        public static int INTEGER = 0;
        public static int CHARACTER = 1;
        public static int UNKNOWN = 2;
        public int tkind;

        public Type ( String name,
                      int typeKind,
                      SymbolNode next)
        {
                super( name, SymbolNode.TYPE, null, next );
                this.tkind = typeKind;
                this.type = this;
        }
}
```

```
package SymbolTable;

// klasa za predstavljanje promenljivih
public class Variable extends SymbolNode {

	public int last_def;
	public int last_use;
	public Variable( String name,
				Type type,
				SymbolNode next )
	{
		super(name,SymbolNode.VARIABLE, type, next);
		last_def = -1;
		last_def = -1;
	}
}
```

```
package SymbolTable;

// klasa za predstavljanje tabele simbola
public class SymbolTable {

	/*tabela simbola za "language scope"
	u ovom slucaju tu pripadaju samo tipovi*/
	private SymbolNode types;

	/* tabela simbola za oblast vazenja programa */
	private SymbolNode variables;

	public SymbolTable( )
	{
	  types = new Type("unknown",Type.UNKNOWN,null);
	  types = new Type( "char", Type.CHARACTER,types);
	  types = new Type( "integer", Type.INTEGER, types );
	  variables = null;
	 }

	public boolean addVar( String name, Type type )
	{
		Variable existing = this.getVar( name );
		if ( existing != null )
			return false;
		variables = new Variable( name, type, variables );
		return true;
	}

	public Variable getVar( String name )
	{
		SymbolNode current = variables;
		while ( current != null &&
				current.name.compareTo( name ) != 0 )
			current = current.next;
		return ( Variable ) current;
	}
```

```
    public Type getType(String typeName)
    {
        SymbolNode current = types;
        while ( current != null &&
                current.name.compareTo( typeName ) !=0)
            current = current.next;
        return ( Type ) current;
    }

    public SymbolNode getVariables()
    {
        return variables;
    }
}
```

```
package SymbolTable;

public class Constant {
	public Type type;
	public Object value;
	public Constant( Type constType, Object constValue )
	{
		type = constType;
		value = constValue;
	}
}
```

CUP specifikacija:

```
//import section

import java_cup.runtime.*;
import java.io.*;
import java.util.*;

import SymbolTable.*;

parser code {:

   public int errNo = 0;
   public int warnNo = 0;

   SymbolTable symbolTable;

   public static void main( String[] args )
   {
      try
      {
        FileReader file = new FileReader(args[0]);
        java_cup.runtime.Scanner scanner =
            new MPLexer( file );
        MPParser parser = new MPParser( scanner );
        parser.parse();
        parser.checkWarnings();
        if ( parser.errNo==0 && parser.warnNo==0 )
          System.out.println(
            "Analiza zavrsena. U kodu nema gresaka." );
        else
           System.out.println(
             "Analiza zavrsena. Broj gresaka: "
             + parser.errNo + " Broj upozorenja: " +
             parser.warnNo );
      }
      catch( Exception e )
      {
           System.out.println(e);
      }
   }

   public void checkWarnings()
```

```
    {
       SymbolNode current = symbolTable.getVariables();
       while ( current != null )
       {
          Variable var = ( Variable ) current;
          if ( var.last_def == -1 && var.last_use == -1 )
          {
            System.out.println("Upozorenje: Promenljiva "
              + var.name +
              "je deklarisana, ali se ne koristi.");
              warnNo++;
          }
          else if ( var.last_def > var.last_use )
          {
             System.out.println(
              "Upozorenje: Vrednost dodeljena prom. "
              + var.name + " u liniji " + var.last_def +
              " se nigde ne koristi." );
             warnNo++;
          }
          current = current.next;
      }
    }

    public void syntax_error(Symbol cur_token)
    {

    }

    public int getLine()
    {
        return (( MPLexer) getScanner()).getLine();
    }
:};

init with {:
     symbolTable = new SymbolTable();
:}

//Terminal symbols
terminal PROGRAM, VAR, INTEGER, CHAR, BEGIN, END, READ,
WRITE, IF, THEN, ELSE;
terminal PLUS, MUL, LEFTPAR, RIGHTPAR, DOTDOT, ASSIGN,
SEMI, COMMA, DOT;
```

```
terminal String ID;
terminal Integer INTCONST;
terminal Character CHARCONST;


//Nonterminal symbols
non terminal Program1, Program, DeklProm, Blok, NizDekl,
Deklaracija, NizNar, Naredba;
non terminal Ulaz, Izlaz, Dodela, IfNar;
non terminal Type Tip, Izraz, PIzraz, FIzraz;
non terminal ArrayList NizImena;
non terminal Constant  Konstanta;




//Grammar

Program ::= PROGRAM LEFTPAR NizImena RIGHTPAR SEMI
             DeklProm Blok DOT

           | PROGRAM LEFTPAR NizImena RIGHTPAR SEMI
             DeklProm Blok error:e
             {:
                System.out.println(
                  "Nedostaje '.' na kraju programa" );
                parser.errNo++;
             :}
           | PROGRAM LEFTPAR NizImena RIGHTPAR SEMI
            DeklProm error
             {:
                System.out.println("Ggreska u liniji " +
                  parser.getLine() + ": " +
                  "Telo programa je nekorektno.");
                parser.errNo++;
             :}
           | PROGRAM LEFTPAR NizImena RIGHTPAR SEMI e
            error
             {:
               System.out.println("Ggreska u liniji " +
                 parser.getLine() + ": " +
                 "Nedostaju deklaracije promenljivih.");
               parser.errNo++;
             :}
```

```
          | PROGRAM LEFTPAR NizImena RIGHTPAR error
            {:
              System.out.println("Ggreska u liniji " +
               parser.getLine()+": "+"Nedostaje ';'.");
               parser.errNo++;
            :}
          | PROGRAM LEFTPAR NizImena error
            {:
               System.out.println("Ggreska u liniji " +
                  parser.getLine() + ": " +
                  "Nedostaje ')'.");
               parser.errNo++;
            :}
          | PROGRAM LEFTPAR error
            {:
               System.out.println("Ggreska u liniji " +
                 parser.getLine() + ": " +
                 "Nedostaju argumenti programa.");
               parser.errNo++;
            :}
          | PROGRAM error
            {:
               System.out.println("Ggreska u liniji " +
                 parser.getLine() + ": " +
                 "Nedostaje '('.");
               parser.errNo++;
            :}
          | error
            {:
               System.out.println("Ggreska u liniji " +
                  parser.getLine() + ": " +
                 "Nedostaje kljucna rec 'program'.");
               parser.errNo++;
            :}
          ;

DeklProm ::= VAR NizDekl
           | VAR error
             {:
                 System.out.println("Ggreska u liniji "
                    + parser.getLine() + ": " +
                    "Nedostaju deklaracije prom.");
                 parser.errNo++;
             :}
```

```
            ;

NizDekl ::= NizDekl Deklaracija
          | Deklaracija
          ;

Deklaracija ::= NizImena:niz DOTDOT Tip:t
                {:
                  for ( int i=0; i<niz.size(); i++ )
                  {
                    String ime = (String) niz.get(i);
                    if ( ! parser.symbolTable.addVar(
                              ime, t ) )
                    {
                      System.out.println(
                         "Ggreska u liniji " +
                         parser.getLine() + ": " +
                         "Promenljiva " + ime +
                         " je vec deklarisana." );
                       parser.errNo++;
                    }
                  }
                :}
              | NizImena:niz DOTDOT error
                {:
                  Type t;
                  t=parser.symbolTable.getType("unknown");
                  for ( int i=0; i<niz.size(); i++ )
                  {
                    String ime = (String) niz.get(i);
                    if ( ! parser.symbolTable.addVar(
                          ime, t))
                    {
                       System.out.println(
                              "Ggreska u liniji "
                              + parser.getLine() + ": " +
                              "Promenljiva " + ime +
                              " je vec deklarisana." );
                          parser.errNo++;
                    }
                  }
                  System.out.println( "Ggreska u liniji " +
                     parser.getLine() + ": " +
                     "Nekorektno ime tipa." );
```

```
                    parser.errNo++;
                :}
              | NizImena:niz error
                {:
                  Type t;
                  t=parser.symbolTable.getType("unknown");
                  for ( int i=0; i<niz.size(); i++ )
                  {
                     String ime = ( String ) niz.get(i);
                     if ( ! parser.symbolTable.addVar(
                              ime,t) )
                     {
                        System.out.println(
                              "Ggreska u liniji "
                               + parser.getLine() + ": " +
                              "Promenljiva " + ime +
                              " je vec deklarisana.");
                         parser.errNo++;
                     }
                   }
                   System.out.println( "Ggreska u liniji "
+
                         parser.getLine() + ": " +
                         "Nedostaje simbol ':'." );
                   parser.errNo++;
                :}

              ;

NizImena ::= NizImena:niz COMMA ID:ime
              {:
                 RESULT = niz;
                 RESULT.add( ime );
              :}
            | NizImena:niz COMMA error
              {:
                 System.out.println("Ggreska u liniji " +
                   parser.getLine() + ": " +
                   "Nedostaje ime promenljive." );
                parser.errNo++;
                RESULT = niz;
              :}
            | ID:ime
              {:
```

```
                RESULT = new ArrayList();
                RESULT.add( ime );
              :}
            ;

Tip ::= INTEGER
        {:
          RESULT = parser.symbolTable.getType( "integer" );
        :}
      | CHAR
        {:
          RESULT = parser.symbolTable.getType( "char" );
        :}
      ;

Blok ::= BEGIN NizNar END
       | BEGIN NizNar error
         {:
            System.out.println("Ggreska u liniji " +
              parser.getLine() + ": " +
              Nedostaje kljucna rec 'end'.");
            parser.errNo++;
         :}
       | BEGIN error
         {:
             System.out.println("Ggreska u liniji " +
                 parser.getLine() + ": " +
                 "Telo bloka je nekorektno");
            parser.errNo++;
         :}
       ;

NizNar ::= NizNar SEMI Naredba
         | NizNar SEMI error
           {:
              System.out.println( "Ggreska u liniji " +
                 parser.getLine() + ": " +
                 "Nedostaje naredba nakon simbola ';'." );
              parser.errNo++;
           :}
         | NizNar error Naredba
           {:
              System.out.println( "Ggreska u liniji " +
                 parser.getLine() + ": " +
```

```
                "Nedostaje ';'." );
              parser.errNo++;
            :}
          | Naredba
          ;

Naredba ::= Ulaz
          | Izlaz
          | Dodela
          | Blok
          | IfNar
          ;

Ulaz ::= READ LEFTPAR ID:ime RIGHTPAR
         {:
            Variable var = parser.symbolTable.getVar(ime);
            if ( var == null )
            {
                 System.out.println( "Ggreska u liniji " +
                     parser.getLine() +
                     ": promenljiva " + ime +
                     " nije deklarisana.");
                 parser.errNo++;
            }
            else
                  var.last_def = parser.getLine();
          :}
        | READ LEFTPAR ID:ime error
          {:
            Variable var = parser.symbolTable.getVar(ime);
             if ( var == null )
             {
                 System.out.println( "Ggreska u liniji " +
                     parser.getLine() +
                     ": promenljiva " + ime +
                     " nije deklarisana.");
                 parser.errNo++;
             }
             else
                  var.last_def = parser.getLine();
            System.out.println( "Ggreska u liniji " +
             parser.getLine() + ": " +
             "Nedostaje ')'.");
            parser.errNo++;
```

```
          :}
        | READ LEFTPAR error
          {:
             System.out.println( "Ggreska u liniji " +
              parser.getLine() + ": " +
              "Nedostaje ime promenljive.");
             parser.errNo++;
          :}
        | READ error
          {:
             System.out.println("Ggreska u liniji " +
              parser.getLine() + ": " +
              "Nedostaje '('.");
             parser.errNo++;
          :}
       ;


Izlaz ::= WRITE LEFTPAR Izraz RIGHTPAR
        | WRITE LEFTPAR  Izraz error
          {:
             System.out.println( "Ggreska u liniji " +
                 parser.getLine() + ": " +
                 "Nedostaje ')'.");
             parser.errNo++;
          :}
        | WRITE LEFTPAR error
          {:
             System.out.println( "Ggreska u liniji " +
                 parser.getLine() + ": " +
                 "Nekorektan izraz.");
             parser.errNo++;
          :}
        | WRITE error
          {:
             System.out.println("Ggreska u liniji " +
                 parser.getLine() + ": " +
                 "Nedostaje '('.");
             parser.errNo++;
          :}
        ;


Dodela ::= ID:ime ASSIGN Izraz:i
```

```
  {:
    Variable var = parser.symbolTable.getVar(ime);
     if ( var == null )
     {
         System.out.println( "Ggreska u liniji " +
              parser.getLine() +
              ": promenljiva " + ime +
              " nije deklarisana.");
         parser.errNo++;
     }
     else
     {
       var.last_def = parser.getLine();
       if ( var.type.tkind != i.tkind )
       {
         System.out.println( "Ggreska u liniji "
           + parser.getLine() +
           ": Neslaganje tipa u naredbi dodele.");
          parser.errNo++;
       }
     }
   :}
| ID:ime ASSIGN error
  {:
    Variable var = parser.symbolTable.getVar(ime);
     if ( var == null )
     {
         System.out.println( "Ggreska u liniji " +
              parser.getLine() +
              ": promenljiva " + ime +
              " nije deklarisana.");
         parser.errNo++;
     }
     else
         var.last_def = parser.getLine();

     System.out.println("Ggreska u liniji " +
         parser.getLine() + ": " +
         "Nekorektan izraz.");
     parser.errNo++;
  :}
| ID:ime error
  {:
    Variable var = parser.symbolTable.getVar(ime);
```

```
                if ( var == null )
                {
                    System.out.println( "Ggreska u liniji " +
                        parser.getLine() +
                        ": promenljiva " + ime +
                        " nije deklarisana.");
                    parser.errNo++;
                }
                else
                    var.last_def = parser.getLine();
                System.out.println("Ggreska u liniji " +
                    parser.getLine() + ": " +
                    "Nedostaje ':='.");
                parser.errNo++;
            :}
          ;

Izraz ::= Izraz:i1 PLUS PIzraz:i2
          {:
              if ( i1.tkind != Type.INTEGER ||
                  i2.tkind != Type.INTEGER )
              {
                  System.out.println("Ggreska u liniji " +
                      parser.getLine() + ": " +
                     "Operator + se ne moze primeniti nad
                      operandima tipa " +
                     i1.name + " i " + i2.name );
                  parser.errNo++;
              }
              RESULT=parser.symbolTable.getType("integer");
          :}
        | Izraz:i PLUS error
          {:
              if ( i.tkind != Type.INTEGER )
              {
                  System.out.println("Ggreska u liniji " +
                      parser.getLine() + ": " +
                     "Operator + se ne moze primeniti nad
                      operandom tipa " + i.name );
                  parser.errNo++;
              }
              System.out.println("Ggreska u liniji " +
                  parser.getLine() + ": " +
                  "Nekorektan izraz.");
```

```
                parser.errNo++;
                RESULT=parser.symbolTable.getType("integer");
            :}
          | PIzraz:i
            {:
                RESULT = i;
            :}
          ;

PIzraz ::= PIzraz:i1 MUL FIzraz:i2
            {:
                if ( i1.tkind != Type.INTEGER
                        || i2.tkind != Type.INTEGER )
                {
                   System.out.println("Ggreska u liniji " +
                        parser.getLine() + ": " +
                      "Operator * se ne moze primeniti nad
                        operandima tipa " +
                      i1.name + " i " + i2.name );
                   parser.errNo++;
                }
                RESULT=parser.symbolTable.getType("integer”);
            :}
          | PIzraz:i MUL error
            {:
                if ( i.tkind != Type.INTEGER )
                {
                   System.out.println("Ggreska u liniji " +
                        parser.getLine() + ": " +
                      "Operator * se ne moze primeniti nad
                        operandom tipa " + i.name );
                   parser.errNo++;
                }
                System.out.println("Ggreska u liniji " +
                   parser.getLine() + ": " +
                   "Nekorektan izraz.");
                parser.errNo++;
                RESULT=parser.symbolTable.getType("integer");
            :}
          | FIzraz:i
            {:
                RESULT = i;
            :}
          ;
```

```
FIzraz ::= ID:ime
            {:
             Variable var = parser.symbolTable.getVar(ime);
             if ( var == null )
             {
                  System.out.println( "Ggreska u liniji " +
                      parser.getLine() +
                      ": promenljiva " + ime +
                      " nije deklarisana.");
                  RESULT =
                    parser.symbolTable.getType("unknown" );
                  parser.errNo++;
             }
             else
             {
                  RESULT = var.type;
                  if ( var.last_def == -1 )
                  {
                      System.out.println("Greska u liniji "
                          + parser.getLine() +
                          ": promenljiva " + ime + " nije
                          inicijalizovana.");
                      parser.errNo++;
                  }
                 var.last_use = parser.getLine();
               }
            :}
          | Konstanta:k
            {:
               RESULT = k.type;
            :}
          | LEFTPAR Izraz:i RIGHTPAR
            {:
                 RESULT = i;
            :}
          | LEFTPAR Izraz:i error
            {:
               System.out.println( "Ggreska u liniji " +
                    parser.getLine() + ": " +
                   "Nedostaje ')'.");
               parser.errNo++;
               RESULT = i;
            :}
```

```
          | LEFTPAR error
            {:
               System.out.println("Ggreska u liniji " +
                  parser.getLine() + ": " +
                  "Nekorektan izraz.");
               parser.errNo++;
               RESULT=parser.symbolTable.getType("unknown");
            :}
          ;

Konstanta ::= INTCONST:c
               {:
                  RESULT = new Constant(
                  parser.symbolTable.getType( "integer" ),
                  c );
               :}
             | CHARCONST:c
               {:
                  RESULT = new Constant(
                  parser.symbolTable.getType( "char" ),c );
               :}
             ;

IfNar ::= IF Izraz:i THEN Naredba ELSE Naredba
          {:
              if ( i.tkind != Type.INTEGER )
              {
                 System.out.println("Greska u liniji " +
                     parser.getLine() + ": " +
                    "Uslov ne moze biti tipa " + i.name );
                 parser.errNo++;
              }
          :}
        | IF Izraz:i THEN Naredba ELSE error
          {:
              if ( i.tkind != Type.INTEGER )
              {
                 System.out.println("Greska u liniji " +
                 parser.getLine() + ": " +
                    "Uslov ne moze biti tipa " + i.name );
                 parser.errNo++;
              }
             System.out.println( "Ggreska u liniji " +
                 parser.getLine() + ": " +
```

```
            "Nedostaje naredba u 'else' grani.");
        parser.errNo++;
    :}
  | IF Izraz:i THEN Naredba error
    {:
        if ( i.tkind != Type.INTEGER )
        {
            System.out.println("Greska u liniji " +
               parser.getLine() + ": " +
               "Uslov ne moze biti tipa " + i.name );
            parser.errNo++;
        }
        System.out.println("Ggreska u liniji " +
            parser.getLine() + ": " +
            "Nedostaje kljucna rec 'else'.");
        parser.errNo++;
    :}
  | IF Izraz:i THEN error
    {:
        if ( i.tkind != Type.INTEGER )
        {
            System.out.println("Greska u liniji " +
                parser.getLine() + ": " +
               "Uslov ne moze biti tipa " + i.name );
            parser.errNo++;
        }
        System.out.println("Ggreska u liniji " +
            parser.getLine() + ": " +
            "Nedostaje naredba u 'then' grani.");
        parser.errNo++;
    :}
  | IF Izraz:i error
    {:
        if ( i.tkind != Type.INTEGER )
        {
            System.out.println("Greska u liniji " +
                parser.getLine() + ": " +
               "Uslov ne moze biti tipa " + i.name );
            parser.errNo++;
        }
        System.out.println("Ggreska u liniji " +
            parser.getLine() + ": " +
            "Nedostaje kljucna rec 'then'.");
        parser.errNo++;
```

```
  :}
| IF error
  {:
     System.out.println("Ggreska u liniji " +
         parser.getLine() + ": " +
         "Nedostaje '('.");
     parser.errNo++;
  :}
;
```