پروژه نهایی کامپایلر – فاز دوم (تحلیل‌گر نحوی)

بنیامین رمضانی (۹۸۰۱۲۲۶۸۰۰۰۸)

زهرا صداقت (۹۹۰۱۲۲۶۸۱۰۰۳)

در این فاز بر اساس فایل JFlex نوشته شده در فاز قبل، به کمک CUP فایلی با پسوند cup. برای تحلیل‌گر نحوی زبان شبه جاوای مد نظر صورت پروژه می‌سازیم.

توضیحات فایل CUP:

برای استفاده از دو کتابخانه با پسوند jar. (java-cup-11b و java-cup-11b-runtime) دستور مربوط به اضافه کردن آن‌ها به صورت زیر می‌نویسیم:

```
import java_cup.runtime.*;
```

در بخش parser code، کدهایی نوشته می‌شوند که نیاز به ذکر دقیق و عینی آن‌ها در فایل‌های javaی تولید شده است. در این بخش برای پیاده‌سازی از قابلیت پشتیبانی از مقداردهی‌ها دو Hash Map (یکی برای متغیرهای نوع int و دیگری از نوع double) را مطابق زیر تعریف می‌کنیم:

```
parser code  {:

    Map<String,Integer> var = new HashMap<>(); //CHANGED
    Map<String,Double> vard = new HashMap<>(); //CHANGED
    final ScheduledExecutorService executorService =
Executors.newSingleThreadScheduledExecutor(); //CHANGED
```

پس از اتمام بستن بخش parser code به سراغ تعریف ترمینال‌ها (terminal) و نان‌ترمینال‌ها (non terminal) می‌رویم. در این بخش علاوه بر تعریف تمام کلیدواژه‌ها، عملگرها و کلمات رزرو شده، عملیات و متغیرهای مورد نظر صورت پروژه را نیز اضافه می‌کنیم:

عملگر POW برای توان:

```
terminal SEMICOLON, MULT, POW, COMMA, LBRACE, RBRACE, EQ, LPAREN, RPAREN, COLON;
```

توابع از پیش تعریف شده برای چاپ در کنسول خروجی و پاک کردن کنسول بعد از t ثانیه:

```
terminal PRINTLN, CLEAR;
```

تابع از پیش تعریف شده دیگر برای محاسبه مقدار صحیح ریشه دوم هر عدد:

```
terminal SQRT;
```

تعریف ترمینال‌هایی برای انواع Primitive Type:

```
terminal java.lang.Number INTEGER_LITERAL;
terminal java.lang.Number FLOATING_POINT_LITERAL;
terminal java.lang.Boolean BOOLEAN_LITERAL;
terminal java.lang.Character CHARACTER_LITERAL;
terminal java.lang.String STRING_LITERAL;
terminal java.lang.String IDENTIFIER; // INTEGER name
terminal java.lang.String IDENTIFIERD; // DOUBLE name
terminal NULL_LITERAL;
```

اضافه کردن نان‌ترمینال‌های جدید و جدا از جاوا جهت پشتیبانی زبان از محاسبات ریاضی:

```
non terminal Integer expr, factor1, factor2, factor3; //CHANGED
non terminal Double exprd, factord1, factord2, factord3; //CHANGED
```

پس از تعریف تمام ترمینال‌ها و نان‌ترمینال‌های مورد نیاز، به تعریف و توصیف گرامرها به کمک آن‌ها می‌پردازیم. گرامرهای تعریف شده مطابق با خواسته صورت پروژه عبارتند از:

ساختارهای حلقه:

:for

```
for_statement ::=
            FOR LPAREN for_init_opt SEMICOLON expression_opt SEMICOLON
                  for_update_opt RPAREN statement
      ;
```

:do... while

```
do_statement ::=
            DO statement WHILE LPAREN expression RPAREN SEMICOLON
      ;
```

:while

```
while_statement ::=
            WHILE LPAREN expression RPAREN statement
      ;
while_statement_no_short_if ::=
            WHILE LPAREN expression RPAREN statement_no_short_if
      ;
```

ساختارهای شرطی:

:if…else / else if

```
if_then_statement ::=
            IF LPAREN expression RPAREN statement
      ;
if_then_else_statement ::=
            IF LPAREN expression RPAREN statement_no_short_if
                  ELSE statement
      ;
if_then_else_statement_no_short_if ::=
            IF LPAREN expression RPAREN statement_no_short_if
                  ELSE statement_no_short_if
      ;
```

:switch

```
switch_statement ::=
            SWITCH LPAREN expression RPAREN switch_block
      ;
switch_block ::=
            LBRACE switch_block_statement_groups switch_labels RBRACE
      |     LBRACE switch_block_statement_groups RBRACE
      |     LBRACE switch_labels RBRACE
      |     LBRACE RBRACE
      ;
switch_block_statement_groups ::=
            switch_block_statement_group
      |     switch_block_statement_groups switch_block_statement_group
      ;
switch_block_statement_group ::=
            switch_labels block_statements
      ;
switch_labels ::=
            switch_label
      |     switch_labels switch_label
      ;
switch_label ::=
            CASE constant_expression COLON
      |     DEFAULT COLON
      ;
```

ساختار توابع:

```
method_declaration ::=
            method_header method_body
      ;
method_header ::=
            modifiers_opt type method_declarator throws_opt
      |     modifiers_opt VOID method_declarator throws_opt
      ;
method_declarator ::=
            IDENTIFIER LPAREN formal_parameter_list_opt RPAREN
      |     method_declarator LBRACK RBRACK // deprecated
      // be careful; the above production also allows 'void foo() []'
      ;
```

```
method_invocation ::= //CHANGED
           PRINTLN LPAREN STRING_LITERAL:s RPAREN
           {: System.out.println(String.valueOf(s)); :}
     |
           PRINTLN LPAREN SQRT LPAREN INTEGER_LITERAL:i RPAREN RPAREN SEMICOLON
           {:  :}
     |
           CLEAR LPAREN INTEGER_LITERAL:s RPAREN
                  {:
                     executorService.schedule(new Runnable() {
                     @Override
                     public void run() {
                           try {
                                  System.out.println("Clearing..." );
                                  for (int i = 0; i < 60; i++) { System.out.println(); }

                                  Runtime.getRuntime().exec("cls"); }
                                  catch (IOException e) {
                                        throw new RuntimeException(e);}
                           }
                      }, s.longValue(), TimeUnit.SECONDS);
                  :}
     |
           PRINTLN LPAREN expr:e RPAREN
           {: System.out.println(e + ""); :}
     |
           PRINTLN LPAREN exprd:e RPAREN
           {: System.out.println(e + ""); :}
     |     name LPAREN argument_list_opt RPAREN
     |     primary DOT IDENTIFIER LPAREN argument_list_opt RPAREN
     |     SUPER DOT IDENTIFIER LPAREN argument_list_opt RPAREN
     |     name DOT SUPER DOT IDENTIFIER LPAREN argument_list_opt RPAREN
     ;
```

تابع از پیش تعریف شده println():

```
method_invocation ::= //CHANGED
            PRINTLN LPAREN STRING_LITERAL:s RPAREN
            {: System.out.println(String.valueOf(s)); :}
```

```
            PRINTLN LPAREN expr:e RPAREN
            {: System.out.println(e + ""); :}
|
            PRINTLN LPAREN exprd:e RPAREN
            {: System.out.println(e + ""); :}
```

تابع از پیش تعریف شده clear(t):

```
CLEAR LPAREN INTEGER_LITERAL:s RPAREN
        {:
           executorService.schedule(new Runnable() {
           @Override
           public void run() {
                 try {
                        System.out.println("Clearing..." );
                        for (int i = 0; i < 60; i++) { System.out.println(); }

                        Runtime.getRuntime().exec("cls"); }
                        catch (IOException e) {
                              throw new RuntimeException(e);}
                 }
            }, s.longValue(), TimeUnit.SECONDS);
        :}
```

تابع از پیش تعریف شده SQRT:

```
| SQRT LPAREN INTEGER_LITERAL:x RPAREN {:
     int answr = x.intValue() / 2;
     if (x.intValue() == 1)
           RESULT = x.intValue();
     else{
           for(int j = answr; answr > 1; answr--){
                 if (answr * answr <= x.intValue()){
                 RESULT = answr;
                 break;
                 }
             }
     }
```

عملیات ریاضی:

عملیات ریاضی بر روی اعداد صحیح:

```
//CHANGED INTEGER
 expr      ::= expr:e PLUS factor1:f1
               {: RESULT = Integer.valueOf(e.intValue() + f1.intValue()); :}
               |
               expr:e MINUS factor1:f1
               {: RESULT = Integer.valueOf(e.intValue() - f1.intValue()); :}
               |
               factor1:f1
               {: RESULT = Integer.valueOf(f1.intValue()); :}
               ;

   factor1    ::= factor1:f1  MULT factor2:f2
               {: RESULT = Integer.valueOf(f1.intValue() * f2.intValue()); :}
               |
               factor1:f1  DIV factor2:f2
               {: RESULT = Integer.valueOf(f1.intValue() / f2.intValue()); :}
               |
               factor2:f2
               {: RESULT = Integer.valueOf(f2.intValue()); :}
               ;

    factor2    ::= factor2:f2  POW factor3:f3
                   {:

                    int base = f2.intValue();
                    int power = f3.intValue();
                    int result = 1;

                           for (power = f3.intValue(); power != 0; power--) {
                               result = result * base;
                           }
                   RESULT = result;
                    :}
                   |
                   factor3:f3
                   {: RESULT = Integer.valueOf(f3.intValue()); :}
                   ;


   factor3       ::= LPAREN expr:e RPAREN
                 {: RESULT = e; :}
                 |
                 INTEGER_LITERAL:a
                 {: RESULT = Integer.valueOf(a.intValue()); :}
                 |
                 IDENTIFIER:i
                 {: RESULT = var.get(i); :}
                 ;
```

عملیات ریاضی بر روی اعداد اعشاری:

```
//CHANGED DOUBLE
   exprd      ::= exprd:e PLUS factord1:f1
                  {: RESULT = Double.valueOf(e.doubleValue() + f1.doubleValue()); :}
                  |
                  exprd:e MINUS factord1:f1
                  {: RESULT = Double.valueOf(e.doubleValue() - f1.doubleValue()); :}
                  |
                  factord1:f1
                  {: RESULT = Double.valueOf(f1.doubleValue()); :}
                  ;

   factord1    ::= factord1:f1  MULT factord2:f2
                  {: RESULT = Double.valueOf(f1.doubleValue() * f2.doubleValue()); :}
                  |
                  factord1:f1  DIV factord2:f2
                  {: RESULT = Double.valueOf(f1.doubleValue() / f2.doubleValue()); :}
                  |
                  factord2:f2
                  {: RESULT = Double.valueOf(f2.doubleValue()); :}
                  ;

    factord2    ::= factord2:f2  POW factord3:f3
                     {:

                      double base = f2.doubleValue();
                      int power = f3.intValue();
                      double result = 1.0;

                             for (power = f3.intValue(); power != 0; power--) {
                                 result = result * base;
                             }
                     RESULT = result;
                      :}
                     |
                     factord3:f3
                     {: RESULT = Double.valueOf(f3.doubleValue()); :}
                     ;


   factord3      ::= LPAREN exprd:e RPAREN
                  {: RESULT = e; :}
                  |
                  FLOATING_POINT_LITERAL:b
                  {: RESULT = Double.valueOf(b.doubleValue()); :}
                  |
                  IDENTIFIERD:i
                  {: RESULT = vard.get(i); :}
                  ;
```

پشتیبانی از مقداردهی متغیرها:

به کمک Hash Map تعریف شده در اول فایل.

نحوه اجرا:

دو فایل jar. مذکور در ابتدای فایل را پس از نصب cup به عنوان Environment Variable تعریف کرده و سپس پوشه محتوای فایل‌های مورد نظر را در cmd باز می‌کنیم. مطابق فاز ۱ پروژه، ابتدا از روی فایل flex، به صورت زیر یک فایل java ایجاد می‌کنیم:

```
C:\Users\User\Desktop\Project\Compiler project - last edition>jflex java.flex
Reading "java.flex"
Constructing NFA : 1,008 states in NFA
Converting NFA to DFA :
.....................................................................................
.....................................................................................
.....................................................................................
452 states before minimization, 425 states in minimized DFA
Writing code to "Scanner.java"
```

خروجی این عملیات به صورت فایلی با پسوند java. در پوشه مربوطه ایجاد می‌شود.

سپس فایل‌های javaی مورد نیاز را از فایل cup که شامل گرامر مربوط به تحلیل‌گر نحوی است، استخراج می‌کنیم. این عملیات با دستور مربوطه و به صورت زیر انجام می‌شود:

```
C:\Users\User\Desktop\Project\Compiler project - last edition>java java_cup.Main "java12 (2).cup"
Warning : Terminal "CONST" was declared but never used
Warning : Terminal "GOTO" was declared but never used
------- CUP v0.11b beta 20140226 Parser Generation Summary -------
  0 errors and 2 warnings
  107 terminals, 162 non-terminals, and 384 productions declared,
  producing 664 unique parse states.
  2 terminals declared but not used.
  0 non-terminals declared but not used.
  0 productions never reduced.
  0 conflicts detected (0 expected).
  Code written to "parser.java", and "sym.java".
--------------------------------------------------- (CUP v0.11b beta 20140226)
```

خروجی این عملیات به صورت ۲ فایل به نام‌های sym.java و parser.java خواهد بود.

در مرحله بعد باید تمام فایل‌های جاوای ایجاد شده را کامپایل کنیم. این کار به صورت زیر انجام می‌شود:

```
C:\Users\User\Desktop\Project\Compiler project - last edition>javac *.java
Note: JavaParser.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
```

برای انجام عملیات Parsing بر روی یک فایل text ورودی نیاز است تا فایل پس از ایجاد یک فایل text مطابق استانداردهای مد نظر گرامر (برای جلوگیری از دریافت خطا)، آن را بر روی جاوایی که تایع main در آن است فراخوانی کنیم. این عملیات به صورت زیر انجام می‌شود:

```
C:\Users\User\Desktop\Compiler project>java JavaParser text1.txt
Parsing [text1.txt]
17.0
50
10
8
1024
hello world!
you are mature enough
something
you are young
I need some sleep
I want to sleep in while loop:D
HI its me again. just want to say i want to go to sleep.
4:01 AM
4
28
12
sqrt 16 is :
4
No errors.
Clearing...
```

فایل text ورودی نیز به صورت زیر بوده:

```
class A {
	public static void main(String[] args){
	int a = 25;
	float b = 2.5;
	println(6.2+10.8); // add two double numbers
//---------------------------
	println(a+25); //add a number with variable
	println(2*5);// multiply two numbers
//----POW-----
	println((3-1)**3); //more complex exression using power
	println((4/2)**2**5); //power precedendce
//------------String------------
	println("hello world!");// print string

//-----------If clause
	int age =23;
	if(age>18){
		println("you are mature enough");
	}
//----------If else
	age = 12;
	if(age>18){
		println("something");
		}
	else{
		println("you are young");
	}

//--------For statement
	  for(int i = 0;i <1;i++){
		println("I need some sleep");
	}
//--------while
	int i =0;
	while(i<1){
	    println("I want to sleep in while loop:D");
		i++;
	}
```

```
//--------do while
      int i =0;
      do{
            println("HI its me again. just want to say i want to go to sleep.");
            i++;
            }while(i<1);
//---------switch
      int a =1;
      switch(a) {
             case x:
              // code block
              default:
                  // code block
                  println("4:01 AM");
            }

//-------------Optional section
 int x=1;
 int y=2;
 println(2*x+y);
 //complex expression
 println(5**2 + x+y);

//-------------SQRT function
int root  = sqrt(49);//assignment
int root2 = sqrt(25);
println(root+root2);
//sqrt in print ln function
println("sqrt 16 is :");
println(sqrt(16));



//----------CLear screen
      clear(5);// clear console in 5 seconds.

}
}
```