پروژه کامپایلر

فاز دوم

فرشته نمازی 9901226800004

بهار خوبان 9901226810002

در بخش اول پروژه ارتباط بین پارسر و فایل فلکس را برقرار می کنیم. فایل فلکس در فاز اول نوشته شد و برای این فاز کمی تغییر داده شد.ترمینال ها و نان ترمینال های مورد نیاز:

```
106
107   /* Terminals */
108   terminal            TypeDef, MemDef, FOR, IN, RETURN ,IF ,ELSE, ELSIF,
109                       TRUE, FALSE, EntryPoint, SWITCH, CASE, WHILE,
110                       CONTINUE, BRAK, DOT, DO, CLEAR, PRINTLN, BREAK;
111
112   terminal            SEMICOLON, OPAR, CPAR, OBLOCK, CBLOCK, OBRACE, CBRACKET,
113                       CBRACE, OBRACKET,
114                       COLON, COMMA, DEC, POWER, ASSIGN, COMPARATOR , NEQ, EQ,
115
116                       PLUSPLUS, PLUS, MINUS, MINUSMINUS, MULT, OR, DIVIDE, NOT,
117                       NNOT, AND, MODULO, ANGBR, LTEQ, GTEQ, ASIGNN, Begin, END, LT;
118
```

```
119
120   // PRINTLN
121   terminal String     Identifier;
122   terminal String     StringConst;
123   terminal Class      Var;
124   terminal Integer    Type;
125   terminal Number    NumValue;
126   terminal Number    Time;       // our scanner provides numbers as integers
127
```

```
128   /* None Terminals */
129   non terminal                empty;
130   non terminal                MethodDeclare;
131   non terminal                MethodCall;
132   non terminal                Assignment;
133   non terminal Object         Expression;
134   non terminal                Loop;
135   non terminal                Statements;
136   non terminal                Statement;
137   non terminal                ForAssignment;
138   non terminal                Declaration;
139   non terminal                IfStatement;
140   non terminal                ElseStatement;
141   non terminal                ElseIfStatement;
142   non terminal                ElseIfStatements;
143   non terminal                DoStatement;
144   non terminal                WhileStatement;
145   non terminal                ForStatement;
146   non terminal                SwitchBlock;
147   non terminal                SwitchCases;
148   non terminal                SwitchCase;
149   non terminal Boolean        Condition;
150   non terminal                BaseName;
151   non terminal                ConditionalStatement;
152   non terminal                PowerStatement;
153   non terminal                CalculationStatement;
154   non terminal                PrintlnInput;
155   non terminal                Calculation;
156   non terminal                ClearStatement;
157   non terminal                PrintlnStatement;
158   non terminal                UpdateaAssignment;
159   non terminal TypeValuePair  AssignmentVal;
160   non terminal                EntryBlock;
161   non terminal                Code;
162   non terminal                SwitchStatement;
163   non terminal  ArrayList<Integer> MethodSignature;
164   non terminal  ArrayList<Integer> Signature;
165   non terminal                OneElseStatement;
166   non terminal  ArrayList<Integer> MethodArgs;
167   non terminal  ArrayList<Integer> Args;
168   non terminal  TypeValuePair Arg;
169
```

```
170
171   /* Calc Non-terminals */
172   non terminal Term, Factor, Logic, Comparison, CalculationList;
173   non terminal Double expr;
174
175   precedence left PLUS, MINUS;
176   precedence left MULT, DIVIDE;
177   precedence right POWER;
```

برای پارسر از گرامر LALR1 استفاده کردیم. در بخش اول و بالاترین قسمت یا تعریف تابع داریم یا entry block و با دیدن هر حالت در کد به گرامر مربوطه می رود.

```
181   Code            ::= MethodDeclare Code
182                       |  EntryBlock Code {: System.out.println("Entry Block"); :}
183                       | empty
184                       ;
185
186   EntryBlock      ::= EntryPoint OBLOCK Statements CBLOCK
187                       {: System.out.println(">> Code entry point"); :}
188                       ;
```

گرامر های مربوط به Statement و Decleration برای تعریف متغیر ها در اکثر بخش های کد استفاده شدند و ساختار ساده ای دارند. Statements می تواند چندتا یا حتی خالی باشد و در خود Statement به گرامر های محاسباتی و توابع و پرینت ... ارجاع می دهیم.

```
186
187   Statements      ::= Statement SEMICOLON Statements {: :}
188                       | ConditionalStatement
189                       | Loop
190                       | empty
191                       ;
192
193
194   Statement       ::= Assignment
195                       | MethodCall
196                       | CalculationStatement {:System.out.println("This is Calc Statement"); :}
197                       | ClearStatement
198                       | PrintlnStatement
199                       | Declaration
200                       ;
201
```

گرامرپرینت طوری نوشته شد که استرینگ یا هر مقدار مجازی که داخلش را بپذیرد.

```
370
371
372   PrintlnInput      ::= StringConst:i {: RESULT = new TypeValuePair("" + i, Constants.TypeString); System.out.println("Println Detected" + i ); :}
373                         | expr:i {: RESULT = new TypeValuePair("" + i, Constants.TypeInt); System.out.println("Println Detected" + i ); :}
374                         ;
375
376
377   PrintlnStatement ::= PRINTLN OPAR PrintlnInput CPAR ;
378
```

گرامر پاک کردن صفحه با کد جاوا با استفاده از تایمر و اسلیپ این طور است که با دیدن عدد داده شده در کد یک حلقه تا آن عدد پیمایش می کند و به همان اندازه صبر می کند. با اتمام پیمایش صفحه را تمیز می کند.

```
65          public void ClearScreen(Double SleepTime) throws IOException, InterruptedException
66                  {
67
68                  try {
69                      for (int j = 0; j < SleepTime; j++)
70                      {
71
72   // The main thread sleeps for the 1000 milliseconds, which is 1 sec
73   // whenever the loop runs
74                      Thread.sleep(1000);
75
76   // displaying the value of the variable
77                      System.out.println(j);
78                      }
79    //                  System.out.print("\033[H\033[2J");
80    //                  System.out.flush();
81                      System.out.println();
82                      System.out.println();
83                      System.out.println();
84                      System.out.println();
85                      System.out.println();
86                      System.out.println();
87                      System.out.println();
88
89                      }
90                  catch (Exception expn)
91                  {
92                      System.out.println(expn);
93                  }
94                  }
```

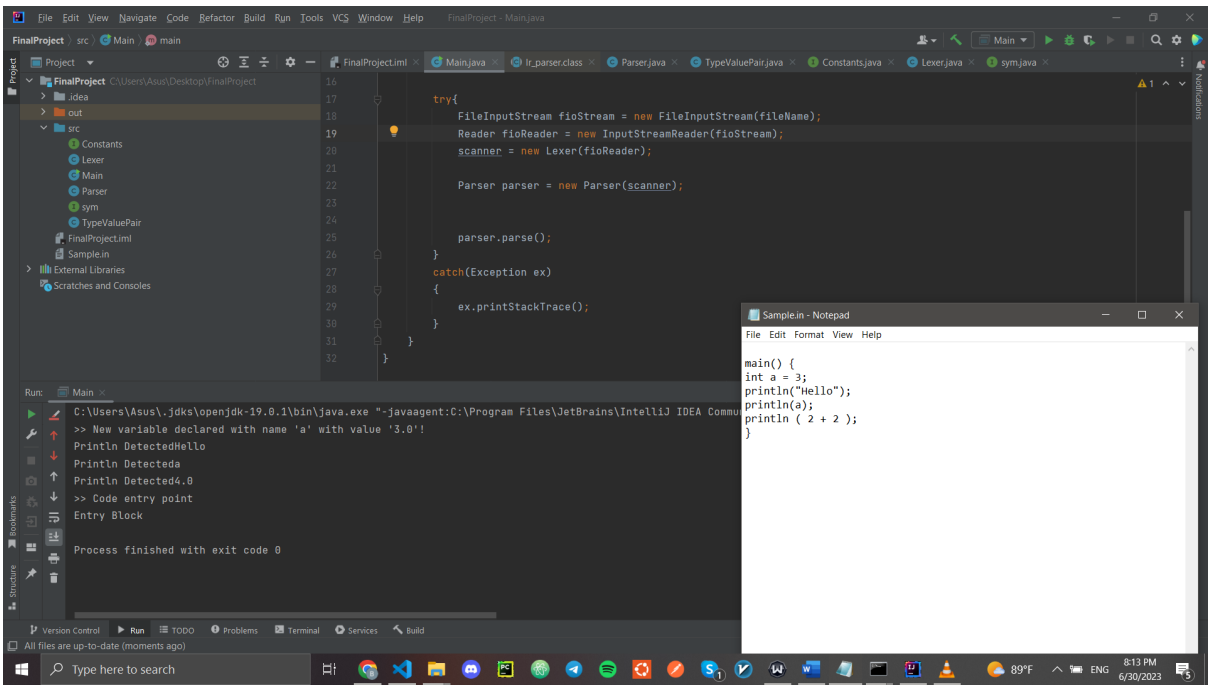
گرامر متد شامل نام و آرگومان و بدنه ای است که می تواند شامل استیتمنت ها باشد.

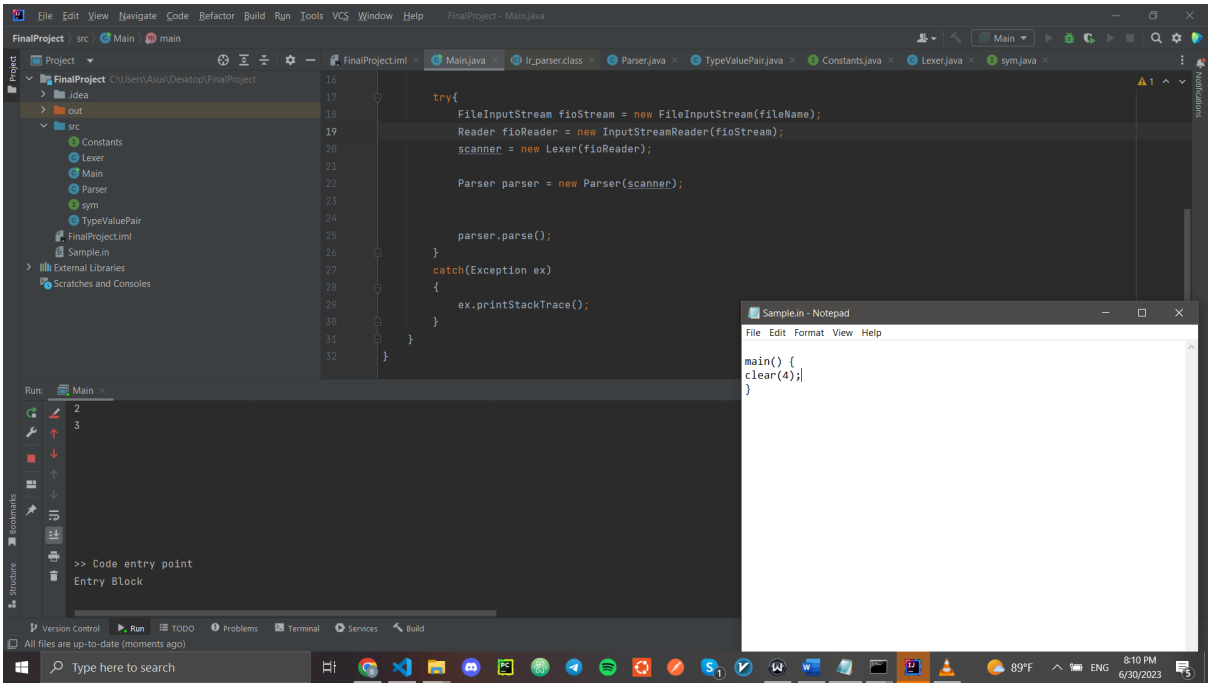
```
309 MethodDeclare  ::= Type Identifier:i OPAR MethodSignature:s CPAR OBLOCK Statements CBLOCK {:
310                if(s == null) sourceSignatures.put(i, null);
311                else sourceSignatures.put(i, s);
312                System.out.println(">> Method " + i.toString() + " Declared! with arg types: " + WriteArgs(s));
313                System.out.println("Method Declared");
314                 :}
315                 ;
316
317
318 MethodSignature ::= Signature:s {: RESULT = s; :}
319                  | empty {: RESULT = null; :}
320                  ;
321
322
323 Signature       ::= Signature:s COMMA Type:t Identifier {:
324                     s.add(t);
325                     RESULT = s;
326                  :}
327                  | Type:s Identifier {:
328                     ArrayList<Integer> list = new ArrayList<>();
329                     list.add(s);
330                     RESULT = list;
331                  :}
332                  ;
333
334 MethodCall     ::= Identifier:i OPAR MethodArgs:a CPAR {:
335                        System.out.println(">> Method " + i.toString() + " called!");
336                        ArgsCheck(sourceSignatures.get(i), a, i);
337                    :}
338                  | BaseName Identifier:i OPAR MethodArgs CPAR {: System.out.println(">> Method " + i.toString() + " called!"); :}
339                  ;
340
341 MethodArgs     ::= Args:i {: RESULT = i; :}
342                  | empty {: RESULT = new ArrayList<Integer>(); :}
343                  ;
```

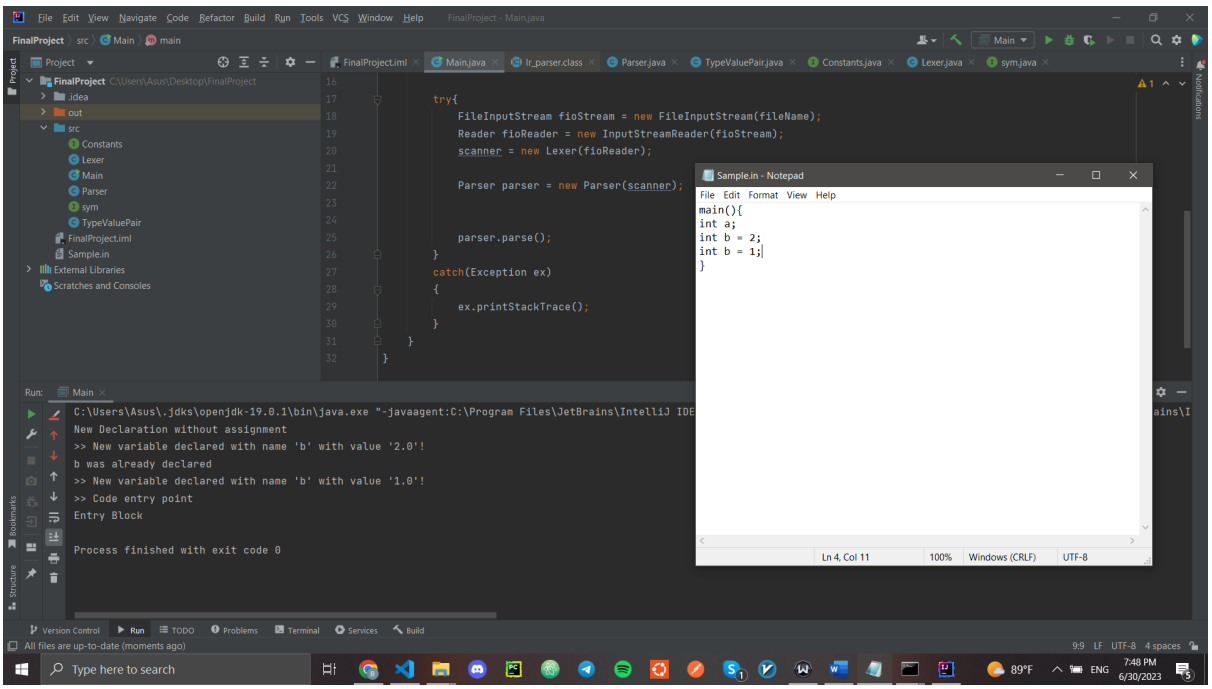
گرامر ریاضی به ترتیب اولویت آپریتورها نوشته شده و با کد جاوا هر آپریشن حساب می شود. هم می تواند اعداد را محاسبه کند و هم متغیر های عددی.

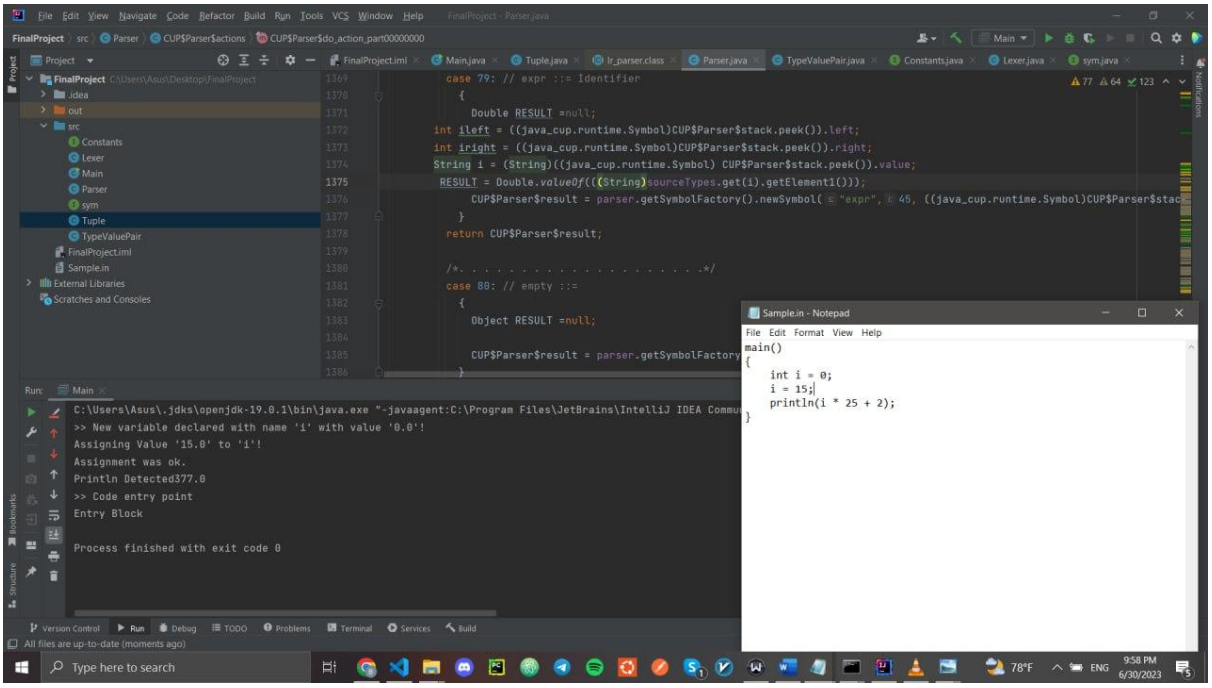
```
// ////////////////* Calculator Grammar *////////////////
CalculationStatement ::= CalculationList{: System.out.println("CalculationStatementt"); :};

CalculationList      ::= Calculation SEMICOLON
                       | CalculationList Calculation SEMICOLON;

Calculation          ::= expr;



expr      ::=   expr:e1 PLUS  expr:e2         {: RESULT = e1+e2;       :}
              | expr:e1 MINUS expr:e2        {: RESULT = e1-e2;       :}
              | expr:e1 MULT expr:e2        {: RESULT = e1*e2;       :}
              | expr:e1 DIVIDE expr:e2        {: RESULT = e1/e2;       :}
              | expr:e1 POWER expr:e2        {: RESULT = Math.pow(e1,e2);     :}
              |NumValue:i {: RESULT = i.doubleValue(); :}
              | Identifier:i {: RESULT = Double.valueOf(((String)sourceTypes.get(i).getElement1())); :};

 // add declared identifier
empty          ::= ;
```

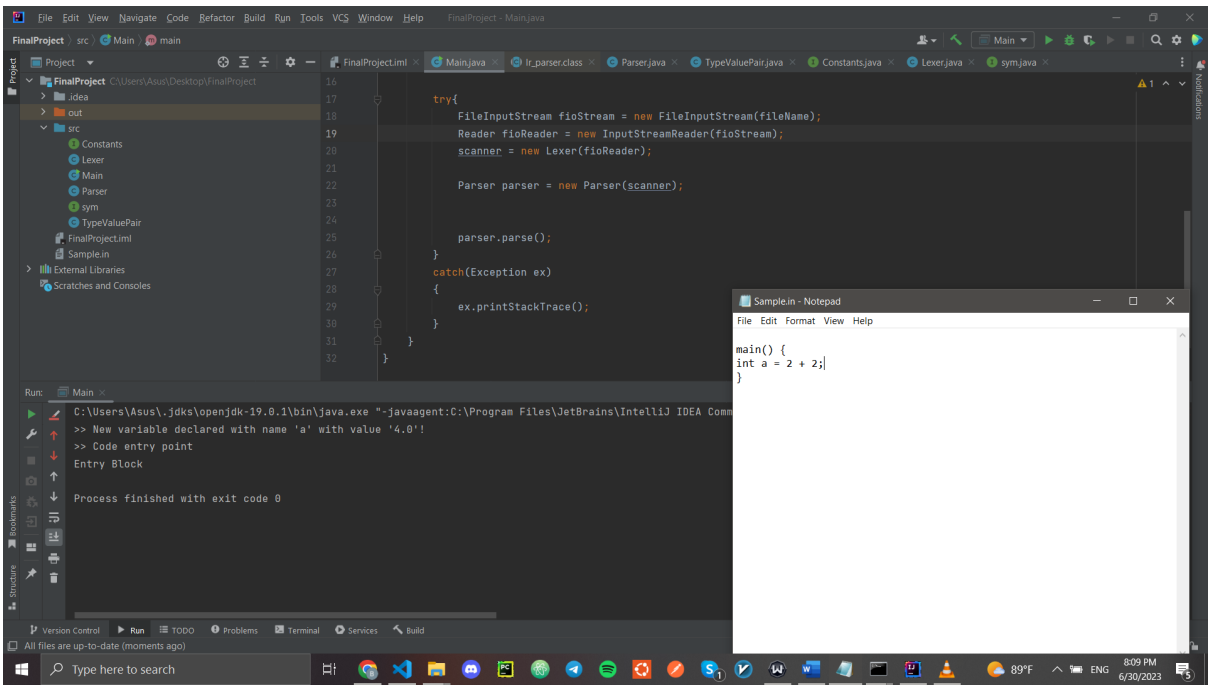
نمونه های تست شده:

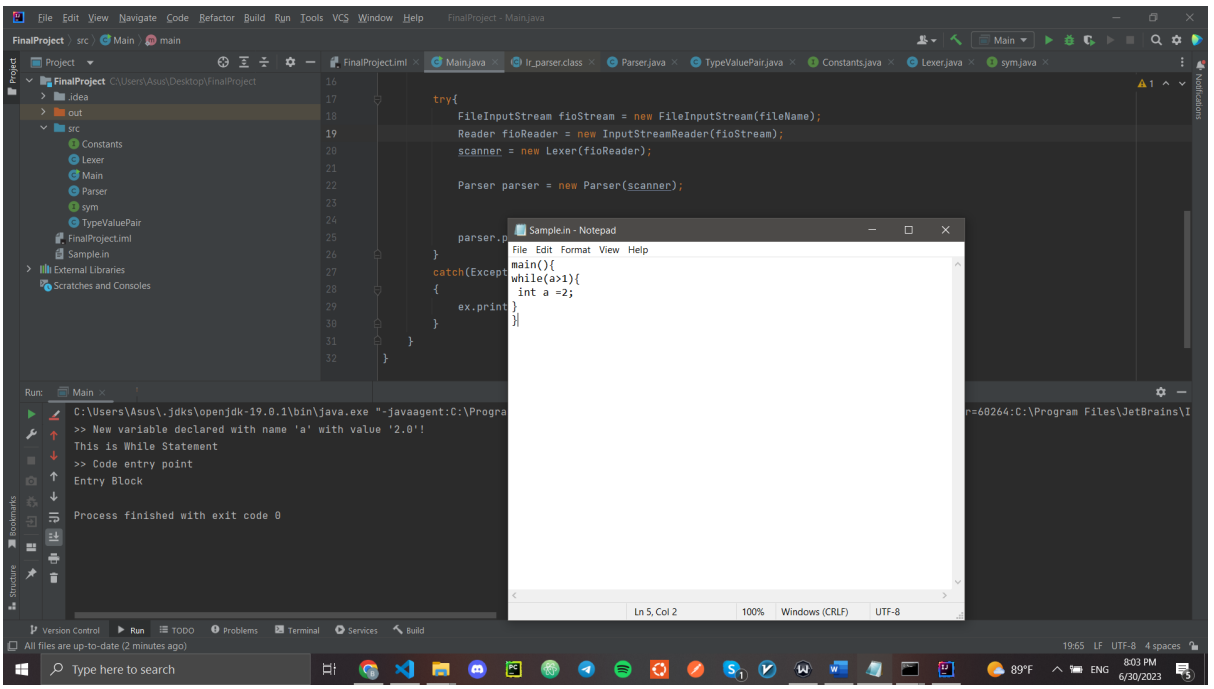

FinalProject - Main.java
FinalProject.iml
Main.java
lr_parser.class
Parser.java
TypeValuePair.java
Constants.java
Lexer.java
sym.java
try{
FileInputStream fioStream = new FileInputStream(fileName);
Reader fioReader = new InputStreamReader(fioStream);
scanner = new Lexer(fioReader);
Parser parser = new Parser(scanner);
parser.parse();
}
catch(Exception ex)
{
ex.printStackTrace();
}
}
}
Run: Main
C:\Users\Asus\.jdks\openjdk-19.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Commu
>> New variable declared with name 'a' with value '3.0'!
Println DetectedHello
Println Detecteda
Println Detected4.0
>> Code entry point
Entry Block
Process finished with exit code 0
Sample.in - Notepad
File Edit Format View Help
main() {
int a = 3;
println("Hello");
println(a);
println (2 + 2);
}
All files are up-to-date (moments ago)
Type here to search
89°F
ENG
8:13 PM
6/30/2023

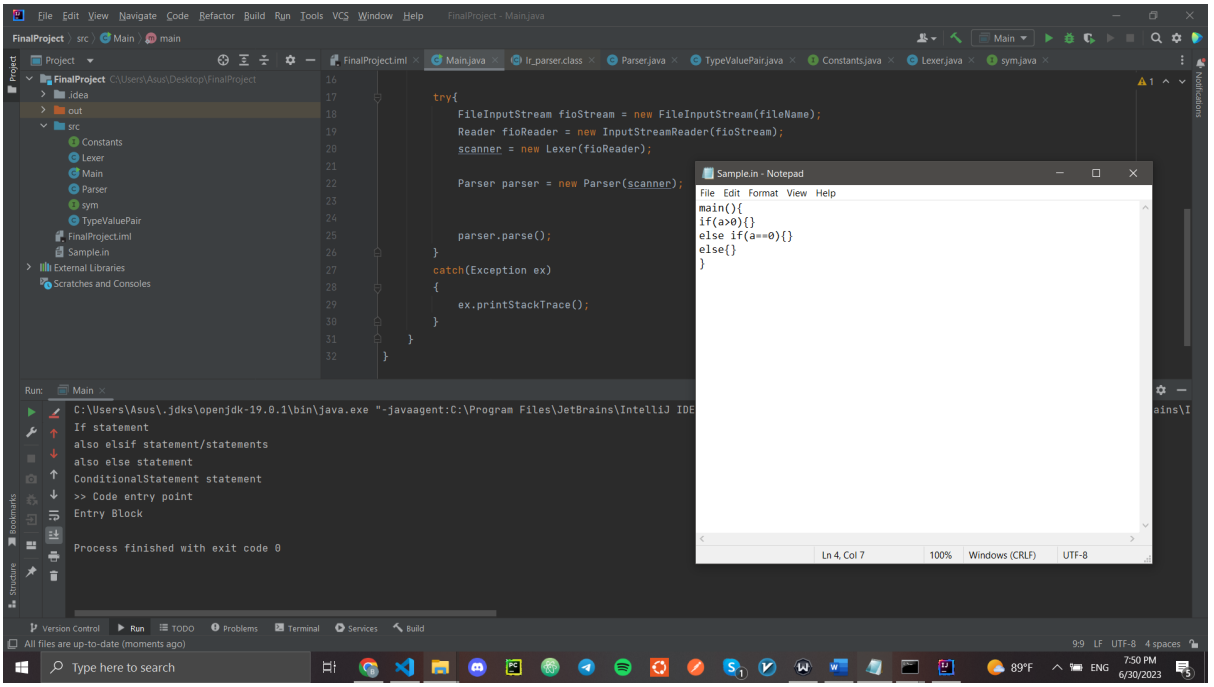

FinalProject - Main.java
FinalProject.iml
Main.java
lr_parser.class
Parser.java
TypeValuePair.java
Constants.java
Lexer.java
sym.java
try{
FileInputStream fioStream = new FileInputStream(fileName);
Reader fioReader = new InputStreamReader(fioStream);
scanner = new Lexer(fioReader);
Parser parser = new Parser(scanner);
parser.parse();
}
catch(Exception ex)
{
ex.printStackTrace();
}
}
}
Run: Main
2
3
>> Code entry point
Entry Block
Sample.in - Notepad
File Edit Format View Help
main() {
clear(4);
}
All files are up-to-date (moments ago)
Type here to search
89°F
ENG
8:10 PM
6/30/2023


Sample.in - Notepad
File Edit Format View Help
main(){
int a;
int b = 2;
int b = 1;
}
Ln 4, Col 11
100%
Windows (CRLF)
UTF-8
C:\Users\Asus\.jdks\openjdk-19.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDE
New Declaration without assignment
>> New variable declared with name 'b' with value '2.0'!
b was already declared
>> New variable declared with name 'b' with value '1.0'!
>> Code entry point
Entry Block
Process finished with exit code 0


Sample.in - Notepad
File Edit Format View Help
main()
{
int i = 0;
i = 15;
println(i * 25 + 2);
}
C:\Users\Asus\.jdks\openjdk-19.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Commu
>> New variable declared with name 'i' with value '0.0'!
Assigning Value '15.0' to 'i'!
Assignment was ok.
Println Detected377.0
>> Code entry point
Entry Block
Process finished with exit code 0


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
FinalProject - Main.java
FinalProject
Project
FinalProject C:\Users\Asus\Desktop\FinalProject
.idea
out
src
Constants
Lexer
Main
Parser
sym
TypeValuePair
FinalProject.iml
Sample.in
External Libraries
Scratches and Consoles
FinalProject.iml
Main.java
ir_parser.class
Parser.java
TypeValuePair.java
Constants.java
Lexer.java
sym.java
try{
FileInputStream fioStream = new FileInputStream(fileName);
Reader fioReader = new InputStreamReader(fioStream);
scanner = new Lexer(fioReader);
Parser parser = new Parser(scanner);
parser.parse();
}
catch(Exception ex)
{
ex.printStackTrace();
}
}
}
Run:
Main
C:\Users\Asus\.jdks\openjdk-19.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Comm
>> New variable declared with name 'a' with value '4.0'!
>> Code entry point
Entry Block
Process finished with exit code 0
Sample.in - Notepad
File Edit Format View Help
main() {
int a = 2 + 2;
}
Version Control
Run
TODO
Problems
Terminal
Services
Build
All files are up-to-date (moments ago)
Type here to search
89°F
ENG
8:09 PM
6/30/2023


Sample.in - Notepad
File Edit Format View Help
main(){
while(a>1){
int a =2;
}
}
Run: Main
C:\Users\Asus\.jdks\openjdk-19.0.1\bin\java.exe "-javaagent:C:\Progra
>> New variable declared with name 'a' with value '2.0'!
This is While Statement
>> Code entry point
Entry Block
Process finished with exit code 0
Ln 5, Col 2
100%
Windows (CRLF)
UTF-8


Sample.in - Notepad
File Edit Format View Help
main(){
if(a>0){}
else if(a==0){}
else{}
}
Run: Main
C:\Users\Asus\.jdks\openjdk-19.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDE
If statement
also elsif statement/statements
also else statement
ConditionalStatement statement
>> Code entry point
Entry Block
Process finished with exit code 0
Ln 4, Col 7
100%
Windows (CRLF)
UTF-8