پروژه درس کامپایلر دکتر فیضی

رضا محمدی - یاسان حسن زاد

1 - Build and Test

دستور زیر را برای تولید کد اسکنر اجرا می کنیم :

```
jflex Scanner.flex
```

دستور زیر را برای کامپایل کد اسکنر اجرا می کنیم :

```
javac Scanner.java
```

دستور زیر را برای اجرای اسکنر با دو فایل ورودی اجرا می کنیم :

```
java Scanner Test1.txt Test2.txt
```

می توان هر تعداد فایل تست دیگری هم نام برد و اسکنر به ترتیب برای هر کدام اجرا می شود.

تصویر از اجرای دستورات فوق در صفحه بعد قرار گرفته است.

```
MINGW64:/c/Users/rezam/Desktop/Compiler/Project/Phase1

rezam@DESKTOP-P0E6L2Q MINGW64 ~/Desktop/Compiler/Project/Phase1
$ jflex Scanner.flex
Reading "Scanner.flex"
Constructing NFA : 684 states in NFA
Converting NFA to DFA :
..........................................................................................................................
..........................................................................................................................
.....................................................................
478 states before minimization, 291 states in minimized DFA
Writing code to "Scanner.java"

rezam@DESKTOP-P0E6L2Q MINGW64 ~/Desktop/Compiler/Project/Phase1
$ javac Scanner.java

rezam@DESKTOP-P0E6L2Q MINGW64 ~/Desktop/Compiler/Project/Phase1
$ java Scanner Test1.txt Test2.txt
line: 1 col: 1 match: --0--
action [182] { System.out.println("     integer" + " ~~~ " + yytext() + " ~~~ line:" + yyline + " column:" + yycolumn); }
     integer ~~~ 0 ~~~ line:0 column:0
line: 1 col: 2 match: --,--
action [130] { System.out.println("     separator" + " ~~~ " + yytext() + " ~~~ line:" + yyline + " column:" + yycolumn); }
     separator ~~~ , ~~~ line:0 column:1
line: 1 col: 3 match: -- --
action [198] { /* ignore */ }
line: 1 col: 4 match: --7--
action [182] { System.out.println("     integer" + " ~~~ " + yytext() + " ~~~ line:" + yyline + " column:" + yycolumn); }
     integer ~~~ 7 ~~~ line:0 column:3
line: 1 col: 5 match: -- --
action [198] { /* ignore */ }
line: 1 col: 6 match: ---821--
action [183] { System.out.println("     negative integer" + " ~~~ " + yytext() + " ~~~ line:" + yyline + " column:" + yycolumn); }
     negative integer ~~~ -821 ~~~ line:0 column:5
line: 1 col: 10 match: -- --
action [198] { /* ignore */ }
line: 1 col: 11 match: ---0.039--
action [185] { System.out.println("     negative float" + " ~~~ " + yytext() + " ~~~ line:" + yyline + " column:" + yycolumn); }
     negative float ~~~ -0.039 ~~~ line:0 column:10
line: 1 col: 17 match: -- --
action [198] { /* ignore */ }
line: 1 col: 18 match: --0.4--
action [184] { System.out.println("     float" + " ~~~ " + yytext() + " ~~~ line:" + yyline + " column:" + yycolumn); }
     float ~~~ 0.4 ~~~ line:0 column:17
line: 1 col: 21 match: --,--
action [130] { System.out.println("     separator" + " ~~~ " + yytext() + " ~~~ line:" + yyline + " column:" + yycolumn); }
     separator ~~~ , ~~~ line:0 column:20
line: 1 col: 22 match: -- --
action [198] { /* ignore */ }
line: 1 col: 23 match: --+.015--
action [184] { System.out.println("     float" + " ~~~ " + yytext() + " ~~~ line:" + yyline + " column:" + yycolumn); }
     float ~~~ +.015 ~~~ line:0 column:22
line: 1 col: 28 match: -- --
action [198] { /* ignore */ }
line: 1 col: 29 match: --+88--
action [182] { System.out.println("     integer" + " ~~~ " + yytext() + " ~~~ line:" + yyline + " column:" + yycolumn); }
     integer ~~~ +88 ~~~ line:0 column:28
line: 1 col: 32 match: --,--
action [130] { System.out.println("     separator" + " ~~~ " + yytext() + " ~~~ line:" + yyline + " column:" + yycolumn); }
```

2 - Definitions

```
%public
%standalone
%class Scanner
```

اسم کلاس جاوا را تعیین میکند و standalone برای این استفاده می شود که میخواهیم به تنهایی از اسکنر استفاده کنیم و تحلیلگر نحوی نداریم.

```
%caseless
%unicode
%debug
```

با caseless مشخص می کنیم که اسکنر حساس به حروف بزرگ و کوچک نیست.

```
%line
%column
```

مشخص می کنیم که می خواهیم از متغیر های yyline و yycolumn در action ها استفاده کنیم.

```
DIGIT = [0-9]
```

عبارت منظم مربوط به ارقام

```
INTEGER = "+"?{DIGIT}{DIGIT}*
```

عبارت منظم مربوط به اعداد صحیح مثبت

```
NEG_INTEGER = "-"{DIGIT}{DIGIT}*
```

عبارت منظم مربوط به اعداد صحیح منفی

```
FLOAT =
"+"?({DIGIT}*"."{DIGIT}{DIGIT}*|{DIGIT}{DIGIT}*"."{DIGIT}*)
```

عبارت منظم مربوط به اعداد اعشاری مثبت

```
NEG_FLOAT = "-
"({DIGIT}*"."{DIGIT}{DIGIT}*|{DIGIT}{DIGIT}*"."{DIGIT}*)
```

عبارت منظم مربوط به اعداد اعشاری منفی

```
ALPHABET = [A-Za-z]
```

عبارت منظم مربوط به حروف الفبا

```
IDENTIFIER = {ALPHABET}({ALPHABET}|{DIGIT}|_)*
```

عبارت منظم مربوط به شناسه ها

```
INCREMENTAL_STATEMENT = {IDENTIFIER}"++"|"++"{IDENTIFIER}
```

عبارت منظم مربوط به دستورات افزایش مقدار متغیر

```
DECREMENTAL_STATEMENT = {IDENTIFIER}"--"|"--"{IDENTIFIER}
```

عبارت منظم مربوط به دستورات کاهش مقدار متغیر

```
INPUT_CHARACTER = [^\r\n]
```

عبارت منظم مربوط به حروفی که line terminator نیستند

```
LINE_TERMINATOR = \r|\n|\r\n
```

عبارت منظم مربوط به حروف خاتمه دهنده خط

```
WHITE_SPACE = {LINE_TERMINATOR}|[ \t\f]
```

عبارت منظم مربوط به فضاهای خالی

```
COMMENT =
{TRADITIONAL_COMMENT}|{END_OF_LINE_COMMENT}|{DOCUMENTATION_
COMMENT}
```

عبارت منظم مربوط به کامنت ها، همانطور که مشخص است سه نوع کامنت در زبان داریم

```
TRADITIONAL_COMMENT = "/*"[^*]+"*/"|"/*""*"+"/"
```

عبارت منظم مربوط به کامنت های مرسوم جاوا که با */ شروع و به /* ختم می شوند

```
END_OF_LINE_COMMENT =
"//"{INPUT_CHARACTER}*{LINE_TERMINATOR}?
```

عبارت منظم مربوط به کامنت های تک خطی که کل خط را اشغال می کنند یا در انتهای خطی که شامل دستورات دیگر است می آیند

```
DOCUMENTATION_COMMENT = "/**"{COMMENT_CONTENT}"*"+"/"
```

عبارت منظم مربوط به کامنت های javadoc

```
COMMENT_CONTENT = ([^*]|\*+[^/*])*
```

عبارت منظم مربوط به محتوای کامنت javadoc

3 - Rules

برخی rule ها با یک رشته مشخص match می شوند، مثل :

```
/* reserved */
"compiler"
"dxt.init()"
/* keywords */
"abstract"
"boolean"
/* operators */
"+"
"&"
```

برخی دیگر با عبارات با قاعده ای که در بخش قبلی توضیح دادیم match می شوند، مثل :

```
{INTEGER}
{NEG_FLOAT}
{IDENTIFIER}
{INCREMENTAL_STATEMENT}
```

برای این موارد action مشابه زیر در نظر گرفته شده که توکن شناسایی شده و شماره خط و ستون را مشخص می کند، مثل :

```
System.out.println("integer" + " ~~~ " + yytext() + " ~~~
line:" + yyline + " column:" + yycolumn);
```

با rule های زیر فضاهای خالی و کامنت ها نادیده گرفته می شوند :

```
/* whitespaces */
{WHITE_SPACE}                   { /* ignore */ }
/* comments */
{COMMENT}                       { /* ignore */ }
```

در انتها rule زیر هر کاراکتر دیگری را غیر معتبر معرفی می کند :

```
.    { System.out.println("illegal character" + " ~~~ " +
yytext() + " ~~~ line:" + yyline + " column:" + yycolumn);
}
```