پروژه نهایی کامپایلر – فاز اول (تحلیل‌گر لغوی)

بنیامین رمضانی (980122680008)

زهرا صداقت (990122681003)

توضیحات کد JFlex:

با توجه به این که در فاز اول پروژ، یک تحلیل‌گر لغوی (Lexical Analyzer) برای یک زبان شبه‌جاوا مطلوب است که ابتدا عناصر کد و در ادامه قوانین مورد نظر (طبق توضیحات صورت پروژه)، بررسی خواهند شد.

در بخش اول کد، متغییر comment_count را برای پیمایش بر روی کاراکترهای هر کامنت، با مقدار اولیه صفر تعریف می‌کنیم.

```
%{
  private int comment_count = 0 ;
%}
```

سپس آپشن‌های مورد نظر را به تناسب، اضافه می‌کنیم:

```
%public
%class subst
%caseless
%ignorecase
%standalone
%line
%char
%column
%state COMMENT
%unicode
```

public: نشان‌دهنده نوع کلاس

class subst: نام کلاس که subst می‌باشد.

caseless و **ignorecase**: برای حساس نبودن به حروف کوچک و بزرگ (Case Insensitive) طبق قانون اول

standalone: تولید فایل جاوایی با متد main که منجر به اجرای آن بدون کمک فایل دیگری می‌شود.

line: برای دریافت و چاپ شماره خط

char: برای دریافت و چاپ کاراکترها

column: برای دریافت و چاپ شماره ستون

state COMMENT: برای شناسایی و چاپ اطلاعات مربوط به هر کامنت

unicode: فایل‌هایی که توسط این اسکنر اسکن شوند، بر اساس Unicode خواهند بود. (از Unicode encoding استفاده می‌کنند.)

در ادامه ماکروها و آپشن‌های زیر را با فرمت‌های خواسته شده تعریف می‌کنیم:

```
ALPHA=[A-Za-z]
DIGIT=[0-9]
Identifier = {ALPHA}({ALPHA}|{DIGIT}|_)*
NONNEWLINE_WHITE_SPACE_CHAR=[\ \t\b\012]
WHITE_SPACE_CHAR=[\n\r\ \t\b\012]
InputCharacter = [^\r\n]
STRING_TEXT=(\\\"|[^\n\r\"]|\\{WHITE_SPACE_CHAR}+\\)*
LineTerminator = \r|\n|\r\n

COMMENT_TEXT = {TraditionalComment} | {EndOfLineComment} | {DocumentationComment}
TraditionalComment = "/*" [^*] ~"*/" | "/*" "*"+ "/"
EndOfLineComment = "//" {InputCharacter}* {LineTerminator}?
DocumentationComment = "/*" "*"+ [^/*] ~"*/"
```

ALPHA: تمام حروفی که توسط پارسر پذیرش می‌شوند. این دسته‌بندی شامل تمام حروف کوچک و بزرگ انگلیسی می‌باشد.

DIGIT: تمام اعداد قابل پذیرش که ادامه به کمک این تعریف، اعداد صحیح و اعشاری تعریف خواهند شد.

Identifier: شناسه که نشان‌دهنده هر فرمت ممکنی برای نام‌گذاری متغیرها، کلاس‌ها و... است.

NONNEWLINE_WHITE_SPACE_CHAR: هر حالتی از کاراکترهای فاصله، تب و... به جز خط جدید (\n)

WHITE_SPACE_CHAR: تمام کاراکترهایی که منجر به ایجاد فضای خالی می‌شوند. (اجتماع ماکروی NONENEWLINE_WHITE_SPACE و InputChar)

InputCharacter: هر نوع کاراکتر به جز کاراکتر خط جدید

STRING_TEXT: تمام حالت‌هایی که برای پذیرش یک رشته مجاز هستند.

LineTerminator: حالت‌هایی که نشان‌دهنده انتقال پارسر به خط بعد هستند.

COMMENT_TEXT: تعیین کننده تمام حالات قابل پذیرش برای کامنت که شامل TraditionalComment، EndOfLineComment و DocumentationComment می‌باشد.

TraditionalComment: کامنت‌های بلوکی جاوا که با */ شروع شده و با /* به اتمام می‌رسند.

EndOflineComment: کامنت‌های خطی جاوا که با // شروع می‌شوند و با سر خط رفتن به اتمام می‌رسند.

در بخش بعدی، از کد jflex که با <YYINITIAL> و براکت‌های باز و بسته مشخص شده است، به تعریف انواع actionهای مرتبط با ماکروها و خواسته صورت پروژه می‌پردازیم. هر تعریف و اکشنی در این بخش، داخل تعریف کلاس قرار می‌گیرد.

کلیدواژه‌ها (Keywords):

ابتدا کلیدواژه‌های زبان مد نظر را تعریف می‌کنیم و در بخش action هر کدام (طبق خواسته صورت پروژه)، شماره خط و ستون آن را در خروجی چاپ می‌کنیم.

کلیدواژه‌ها تعریف شده عبارتند از:

```
//Keywords
"abstract"      {System.out.println("Keyword: \"abstract\" "+  "line " + (yyline + 1) + " | " + "column " + (yycolumn + 1));}
"boolean"       {System.out.println("Keyword: \"boolean\" "+  "line " + (yyline + 1) + " | " + "column " + (yycolumn + 1));}
"break"         {System.out.println("Keyword: \"break\" "+  "line " + (yyline + 1) + " | " + "column " + (yycolumn + 1));}
"catch"         {System.out.println("Keyword: \"catch\" "+  "line " + (yyline + 1) + " | " + "column " + (yycolumn + 1));}
"char"          {System.out.println("Keyword: \"char\" "+  "line " + (yyline + 1) + " | " + "column " + (yycolumn + 1));}
"continue"      {System.out.println("Keyword: \"continue\" "+  "line " + (yyline + 1) + " | " + "column " + (yycolumn + 1));}
"default"       {System.out.println("Keyword: \"default\" "+  "line " + (yyline + 1) + " | " + "column " + (yycolumn + 1));}
"do"            {System.out.println("Keyword: \"do\" "+  "line " + (yyline + 1) + " | " + "column " + (yycolumn + 1));}
"double"        {System.out.println("Keyword: \"double\" "+  "line" + (yyline + 1) + " | " + "column " + (yycolumn + 1));}
"else"          {System.out.println("Keyword: \"else\" "+  "line" + (yyline + 1) + " | " + "column " + (yycolumn + 1));}
"enum"          {System.out.println("Keyword: \"enum\" "+  "line" + (yyline + 1) + " | " + "column " + (yycolumn + 1));}
"extends"       {System.out.println("Keyword: \"extends\" "+  "line" + (yyline + 1) + " | " + "column " + (yycolumn + 1));}
"final"         {System.out.println("Keyword: \"final\" "+  "line" + (yyline + 1) + " | " + "column " + (yycolumn + 1));}
"float"         {System.out.println("Keyword: \"float\" "+  "line" + (yyline + 1) + " | " + "column " + (yycolumn + 1));}
"for"           {System.out.println("Keyword: \"for\" "+  "line" + (yyline + 1) + " | " + "column " + (yycolumn + 1));}
"int"           {System.out.println("Keyword: \"int\" "+  "line" + (yyline + 1) + " | " + "column" + (yycolumn + 1));}
"if"            {System.out.println("Keyword: \"if\" "+  "line" + (yyline + 1) + " | " + "column" + (yycolumn + 1));}
"implements"    {System.out.println("Keyword: \"implements\" "+  "line" + (yyline + 1) + " | " + "column" + (yycolumn + 1));}
"import"        {System.out.println("Keyword: \"import\" "+  "line" + (yyline + 1) + " | " + "column" + (yycolumn + 1));}
"interface"     {System.out.println("Keyword: \"interface\" "+  "line" + (yyline + 1) + " | " + "column" + (yycolumn + 1));}
"long"          {System.out.println("Keyword: \"long\" "+  "line" + (yyline + 1) + " | " + "column" + (yycolumn + 1));}
"new"           {System.out.println("Keyword: \"new\" "+  "line" + (yyline + 1) + " | " + "column" + (yycolumn + 1));}
"package"       {System.out.println("Keyword: \"package\" "+  "line" + (yyline + 1) + " | " + "column" + (yycolumn + 1));}
"private"       {System.out.println("Keyword: \"private\" "+  "line" + (yyline + 1) + " | " + "column" + (yycolumn + 1));}
"protected"     {System.out.println("Keyword: \"protected\" "+  "line" + (yyline + 1) + " | " + "column" + (yycolumn + 1));}
"public"        {System.out.println("Keyword: \"public\" "+  "line" + (yyline + 1) + " | " + "column" + (yycolumn + 1));}
"return"        {System.out.println("Keyword: \"return\" "+  "line" + (yyline + 1) + " | " + "column" + (yycolumn + 1));}
"short"         {System.out.println("Keyword: \"short\" "+  "line" + (yyline + 1) + " | " + "column" + (yycolumn + 1));}
"static"        {System.out.println("Keyword: \"static\" "+  "line" + (yyline + 1) + " | " + "column" + (yycolumn + 1));}
"switch"        {System.out.println("Keyword: \"switch\" "+  "line" + (yyline + 1) + " | " + "column" + (yycolumn + 1));}
"this"          {System.out.println("Keyword: \"this\" "+  "line" + (yyline + 1) + " | " + "column" + (yycolumn + 1));}
"throws"        {System.out.println("Keyword: \"throws\" "+  "line" + (yyline + 1) + " | " + "column" + (yycolumn + 1));}
"try"           {System.out.println("Keyword: \"try\" "+  "line" + (yyline + 1) + " | " + "column" + (yycolumn + 1));}
"var"           {System.out.println("Keyword: \"var\" "+  "line" + (yyline + 1) + " | " + "column" + (yycolumn + 1));}
"void"          {System.out.println("Keyword: \"void\" "+  "line" + (yyline + 1) + " | " + "column" + (yycolumn + 1));}
"while"         {System.out.println("Keyword: \"while\" "+  "line" + (yyline + 1) + " | " + "column" + (yycolumn + 1));}
```

پس از شناسایی هر کلیدواژه، به فرمت زیر و با اطلاعات خوسته شده در خروجی چاپ می‌شود:

Keyword: “Keyword Name” line x | column y

عملگرها (Operators):

عملگرها (Operators) را هم همانند کلیدواژه‌ها تعریف می‌کنیم:

```
//Operators
"+"                 {System.out.println("Operator: \"Addition\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"-"                 {System.out.println("Operator: \"Subtraction\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"*"                 {System.out.println("Operator: \"Multiplication\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"/"                 {System.out.println("Operator: \"Division\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"%"                 {System.out.println("Operator: \"Modulus\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }

"="                 {System.out.println("Operator: \"Assignment\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"+="                {System.out.println("Operator: \"Addition Assignment\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"-="                {System.out.println("Operator: \"Subtraction Assignment\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"*="                {System.out.println("Operator: \"Multiplication Assignment\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"/="                {System.out.println("Operator: \"Division Assignment\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"%="                {System.out.println("Operator: \"Modulus Assignment\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"+="                {System.out.println("Operator: \"Addition Assignment\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"<="                {System.out.println("Operator: \"Less than or Equal to\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
">="                {System.out.println("Operator: \"Greater than or Equal to\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"<"                 {System.out.println("Operator: \"Less than\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
">"                 {System.out.println("Operator: \"Greater that\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"=="                {System.out.println("Operator: \"Equal to\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"!="                {System.out.println("Operator: \"Not Equal\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
":="                {System.out.println("Operator: \"Sign :=\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }

":"                 {System.out.println("Operator: \"Colon\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
";"                 {System.out.println("Operator: \"Semicolon\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"'"                 {System.out.println("Operator: \"Single Quotation\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
","                 {System.out.println("Operator: \"Comma\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"."                 {System.out.println("Operator: \"Period\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }

"["                 {System.out.println("Operator: \"Opening bracket\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"]"                 {System.out.println("Operator: \"Closing bracket\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"("                 {System.out.println("Operator: \"Opening paranthesis\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
")"                 {System.out.println("Operator: \"Closing parenthesis\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"{"                 {System.out.println("Operator: \"Opening brace\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"}"                 {System.out.println("Operator: \"Closing brace\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"\""                {System.out.println("Operator: \"Double quotation\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"<>"                {System.out.println("Operator: \"Sign <>\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }

"++"                {System.out.println("Operator: \"Increment\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"--"                {System.out.println("Operator: \"Decrement\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"**"                {System.out.println("Operator: \"Power Statement\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }

"&"                 {System.out.println("Operator: \"And\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"|"                 {System.out.println("Operator: \"Or\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"&&"                {System.out.println("Operator: \"Logical And\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"||"                {System.out.println("Operator: \"Logical Or\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
"!"                 {System.out.println("Operator: \"Logical Not\" line " + (yyline + 1) + " | " + "column " + (yycolumn + 1)); }
```

پس از شناسایی هر عملگر، به فرمت زیر و با اطلاعات خوسته شده در خروجی چاپ می‌شود:

Operator: “Operator Name” line x | column y

*عملگر توان (POWER Statement) نیز در همین بخش تعریف شده است.

فضاهای خالی (Whitespaces):

برای شناسایی و چاپ خروجی در برخورد پارسر با هر کاراکتر خط جدید، فاصله، تب و... این action تعریف می‌شود:

```
//Whitespace
{NONNEWLINE_WHITE_SPACE_CHAR}+ {System.out.println("Whitespace: line " + (yyline + 1) + " | " + "column " + (yycolumn + 1));}
```

پس از شناسایی هر یک از کاراکترهای مرتبط، خطی با اطلاعات مذکور در خروجی چاپ خواهد شد:

Whitespace: line x |column y

رشته (String):

برای action رشته‌های ورودی یا STRING_TEXT خواهیم داشت:

```
//String
\"{STRING_TEXT}\" {
    String str =  yytext().substring(1,yylength()-1);
    System.out.println ("String: " + "\"" + str + "\"" + " line " + (yyline + 1) + " | " + "column " + (yycolumn + 1));}
```

رشته‌های شناسایی شده در ورودی به فرمت زیر در خروجی چاپ خواهند شد:

String: “String Name” line x | column y

کلمات رزرو شده (Reserved Words):

برای action کلمات رزرو شده یا Reserved Word، بنا بر سه کلمه خواسته شده CLEAR، inumber و println خواهیم داشت:

```
//Reserved Words
"clear"          {System.out.println("Reserved word: \"clear\" line " + (yyline + 1) +  " | "+ "column " + (yycolumn + 1));}
"inumber"        {System.out.println("Reserved word: \"inumber\" line " + (yyline + 1) +  " | "+ "column " + (yycolumn + 1));}
"println"        {System.out.println("Reserved word: \"println\" line " + (yyline + 1) +  " | "+ "column " + (yycolumn + 1));}
```

سه کلمه مذکورپس از شناسایی در ورودی، به فرمت زیر در خروجی چاپ خواهند شد:

Reserved Word: “Reserved Word Name” line x | column y

شناسه‌ها (Identifiers):

برای action شناسه ها یا Identifiers خواهیم داشت:

```
//Identifier
{Identifier}      {System.out.println("Identifier: " + "\"" + yytext() + "\"" + " line " + (yyline + 1) +  " | "+ "column " + (yycolumn + 1)); }
```

شناسه‌های شناسایی شده در ورودی به فرمت زیر در خروجی چاپ خواهند شد:

Identifier: “Identifier Name” line x | column y

اعداد صحیح و اعشاری منفی و مثبت:

برای شناسایی اعداد صحیح و اعشاری منفی و مثبت، بدیهی است که اعداد منفی با علامت منفی و اعداد مثبت باید در دو حالت بدون علامت مثبت و با علامت مثبت شناسایی شوند.

```
//Positive Integer
("+"{DIGIT}+ | {DIGIT}+)                    {System.out.println("Positive Integer: " + "\"" + yytext() + "\"" + " line " + (yyline + 1) +  " | "+ "column " + (yycolumn + 1)); }
//Negative Integer
"-"{DIGIT}+                                 {System.out.println("Negative Integer: " + "\"" + yytext() + "\"" + " line " + (yyline + 1) +  " | "+ "column " + (yycolumn + 1)); }
//Positive Float
("+"{DIGIT}+"."{DIGIT}+ | {DIGIT}+"."{DIGIT}+)  {System.out.println("Positive Float: " + "\"" + yytext() + "\"" + " line " + (yyline + 1) +  " | "+ "column " + (yycolumn + 1)); }
//Negative Float
"-"{DIGIT}+"."{DIGIT}+                      {System.out.println("Negative Float: " + "\"" + yytext() + "\"" + " line " + (yyline + 1) +  " | "+ "column " + (yycolumn + 1)); }
```

اعداد صحیح مثبت شناسایی شده در ورودی به فرمت زیر در خروجی چاپ خواهند شد:

Positive Integer: “number” line x | column y

اعداد صحیح منفی شناسایی شده در ورودی به فرمت زیر در خروجی چاپ خواهند شد:

Nagative Integer: “number” line x | column y

اعداد اعشاری مثبت شناسایی شده در ورودی به فرمت زیر در خروجی چاپ خواهند شد:

Positive Float: “number” line x | column y

اعداد اعشاری منفی شناسایی شده در ورودی به فرمت زیر در خروجی چاپ خواهند شد:

Nagative Float: “number” line x | column y

کامنت‌ها (Comments):

کامنت‌ها همانگونه که در ابتدای کد JFelx، تحت عنوان ماکرو و به فرمت کامنت‌های زبان جاوا تعریف شده‌اند، شناسایی خواهند شد:

```
//Comment
<COMMENT> {
  "/*" { comment_count++; }
  "*/" { if (--comment_count == 0) yybegin(YYINITIAL); }
  {COMMENT_TEXT} {System.out.println("Comment: " + "\n" + yytext() + " line " + (yyline + 1) +  " | "+ "column " + (yycolumn + 1)); }
}
```

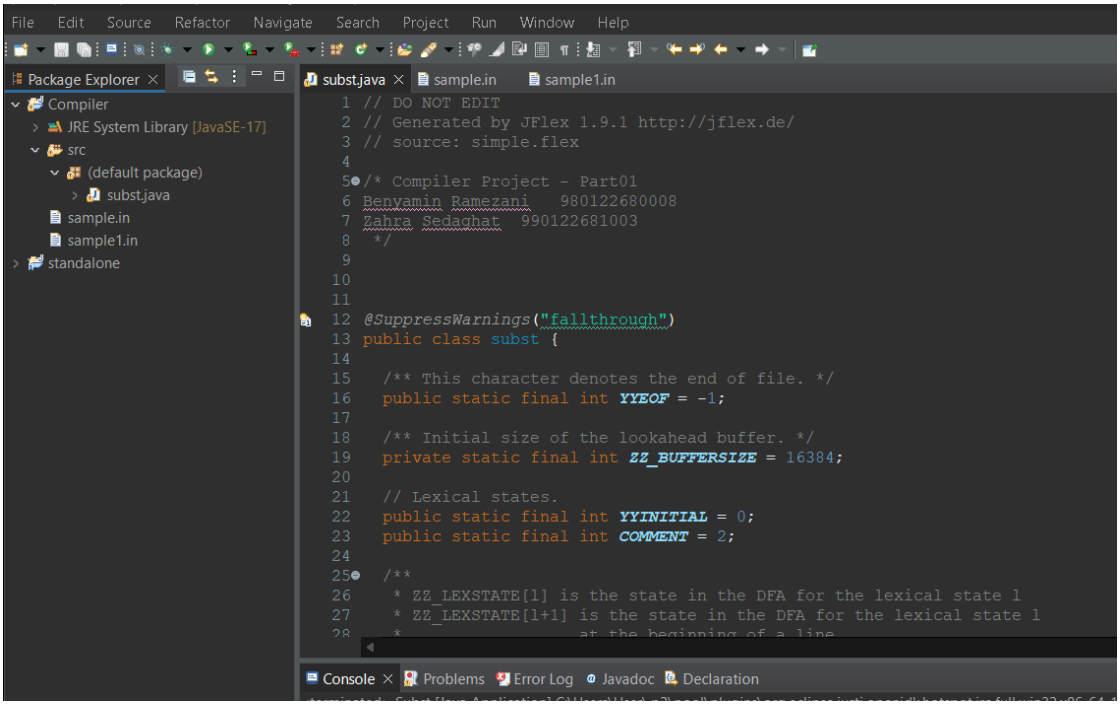
نحوه اجرا:

برای دریافت کد جاوای مربوطه، پس از نصب JFlex، ابتدا در cmd به دایرکتوری‌ای که فایل simple.flex قرار دارد می-رویم و سپس با وارد کردن دستورات مربوطه (به شکل زیر) خروجی subst.java تولید می‌شود.

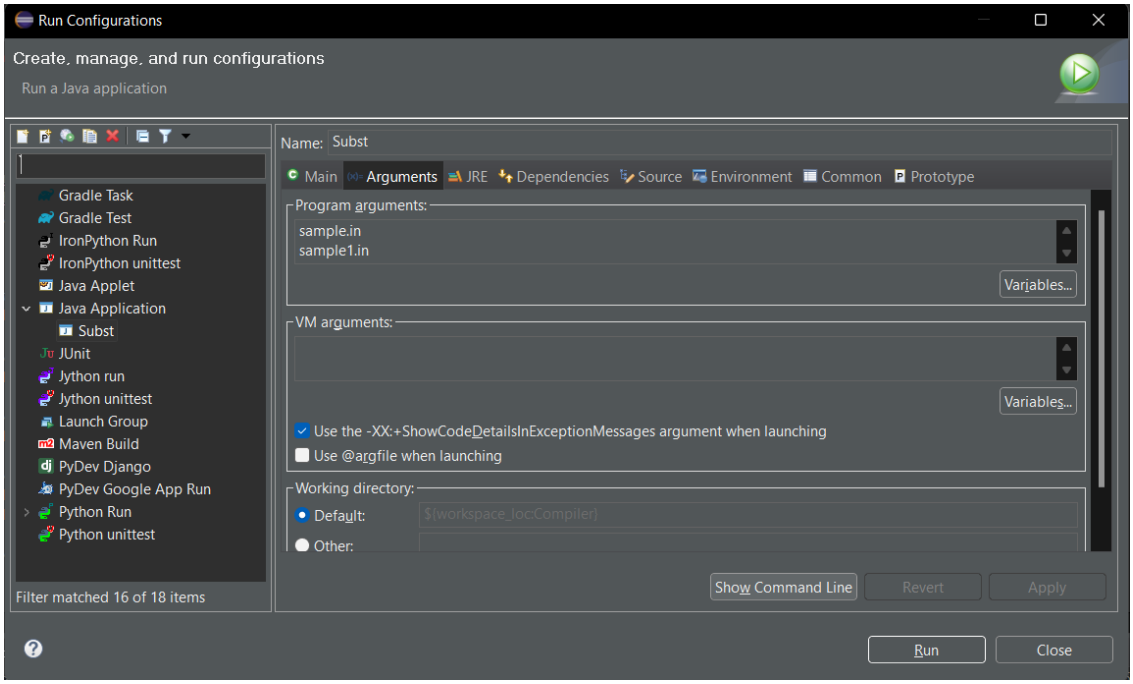
```
PS C:\Users\User\Desktop\Compiler project> jflex .\simple.flex
Reading ".\simple.flex"
Constructing NFA : 538 states in NFA
Converting NFA to DFA :
.....................................................................
.....................................................................
.............................
347 states before minimization, 327 states in minimized DFA
Writing code to ".\subst.java"
PS C:\Users\User\Desktop\Compiler project>
```

سپس فایل subst.java که طی مراحل مذکور تولید شده را در Eclipse باز می‌کنیم. در این فایل کد جاوای زبان شبه-جاوایی که طی دستورات مراحل قبل تعریف کرده‌ایم را خواهیم داشت.

برای تست این کد یک یا چند فایل sample ایجاد می‌کنیم و اسامی فایل یا فایل‌های تولید شده را در قسمت Run Configuration و سپس بخش argument مربوط به subst.java اضافه می‌کنیم. با این کار، پس از هر بار اجرای این کد، سمپل یا سمپل‌های تعریف شده به عنوان ورودی به کد داده خواهند شد و خروجی متناسب با محتوای هر سمپل را مشاهده خواهیم کرد.



*در این جا دو فایل سمپل برای پروژه به نام‌های sample.in و sample1.in تعریف شده‌اند.



*همان طور که مشخص است، نام هر دو فایل به عنوان آرگیومنت‌های subst.java ذکر شده‌اند.

محتوای هر سمپل و خروجی تولید شده از هر کدام در پوشه پروژه وجود دارند.