STRIGER

SER 502-EMERGING LANGUAGES AND PROGRAMMING PARADIGMS PROJECT GROUP 9

Github: https://github.com/ntarunasy/SER502-Striger-Team9/tree/main

TEAM MEMBERS

- VENKATA SAI TARUN NUKA
- SRI VIKAS GANUGU
- NITIN SURYA MOTURU

Striger Features

- It supports three types of data types: Integer, String and Boolean.
- It supports two types of printing of variables: print(variable) and print variable.
- It has two types of conditional statements:

1. if_condition: a normal if else condition. It also has a elif part.
2. If_then_else:It is also similar to above one, but here there won't be elif part.

- It has three types of loops: for_loop, while_loop and for_inrange.
- The arithmetic operations it support are addition, subtraction, multiplication, division.
- The Boolean operations it support are 'true', false, ==, not, and >,<,>=,<=,!='.
- Variables name contain small and capital letters, along with numbers. But variable names shouldn't start with number

Components of a Striger Program

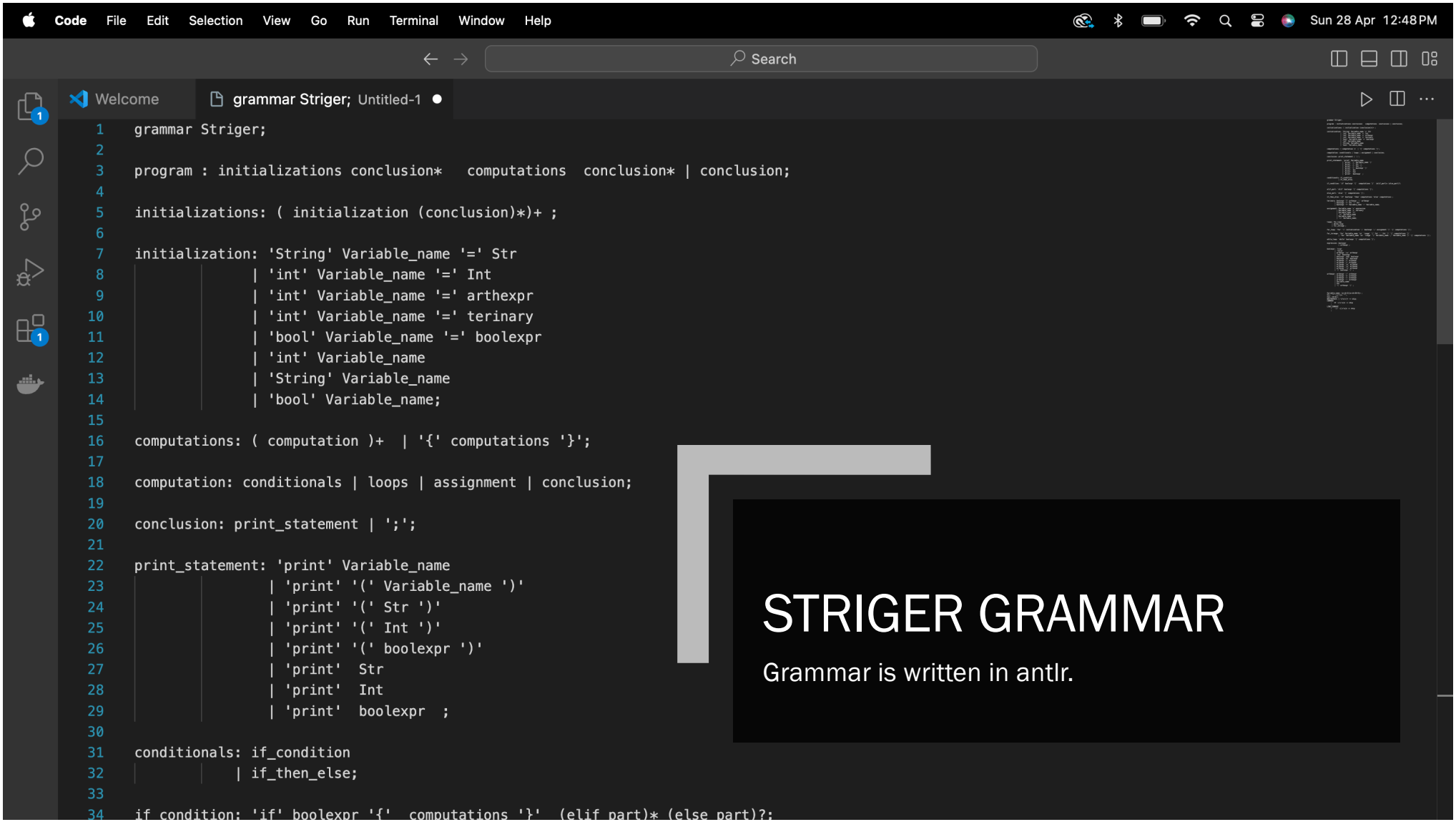
- *1. Initialization: The initialization part of the program consists of defining new variables[a unit that helps store and access values]. The variables are defined with a name and a type(mandatory)[Variable type is a generalization of the values assigned to the variable].*
- *Syntax: int x = 23*
- *int is type*
- *X is name*
- *23 is the value assigned to the variable x*

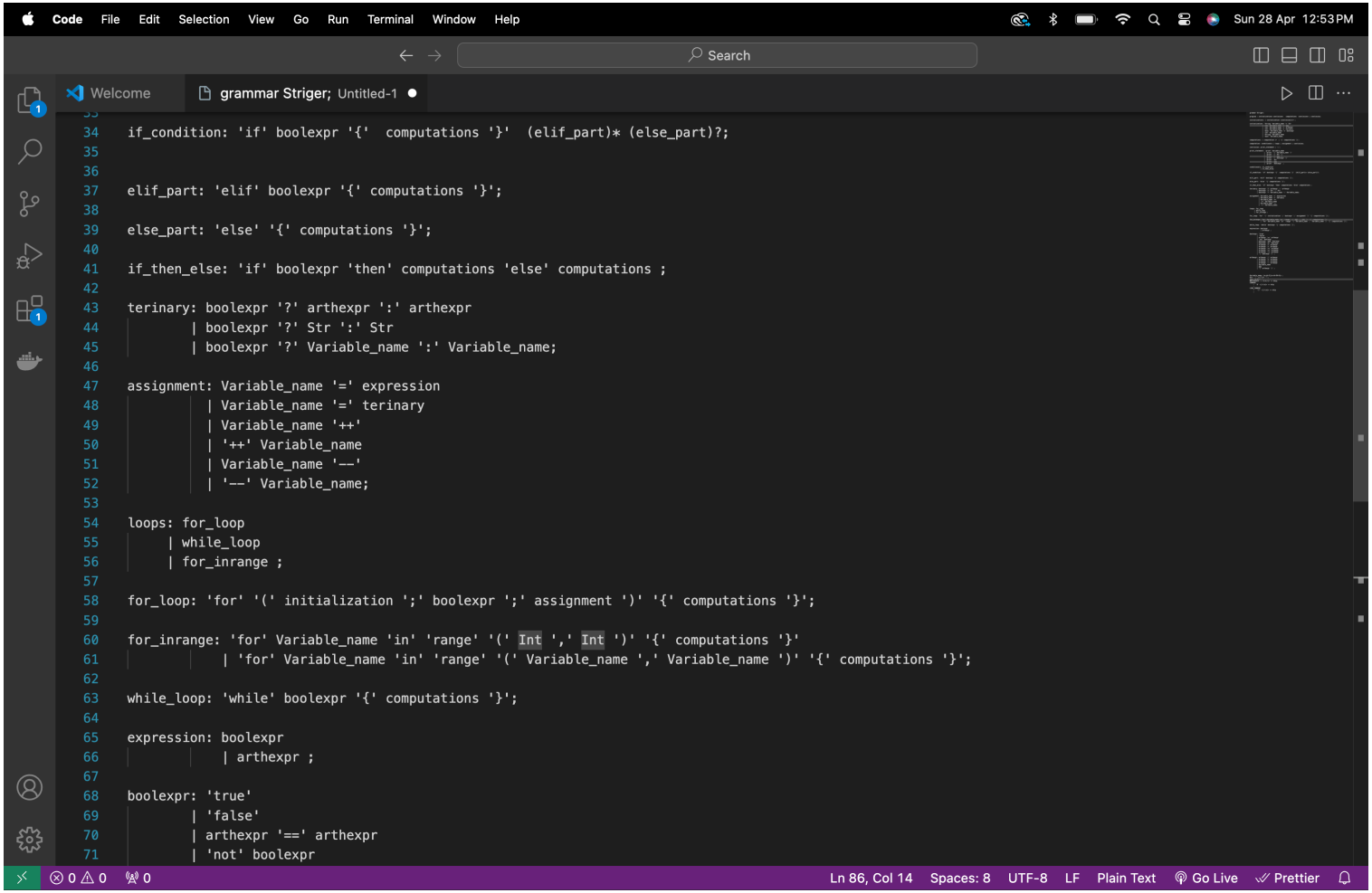
Components of a Striger Program(contd..)

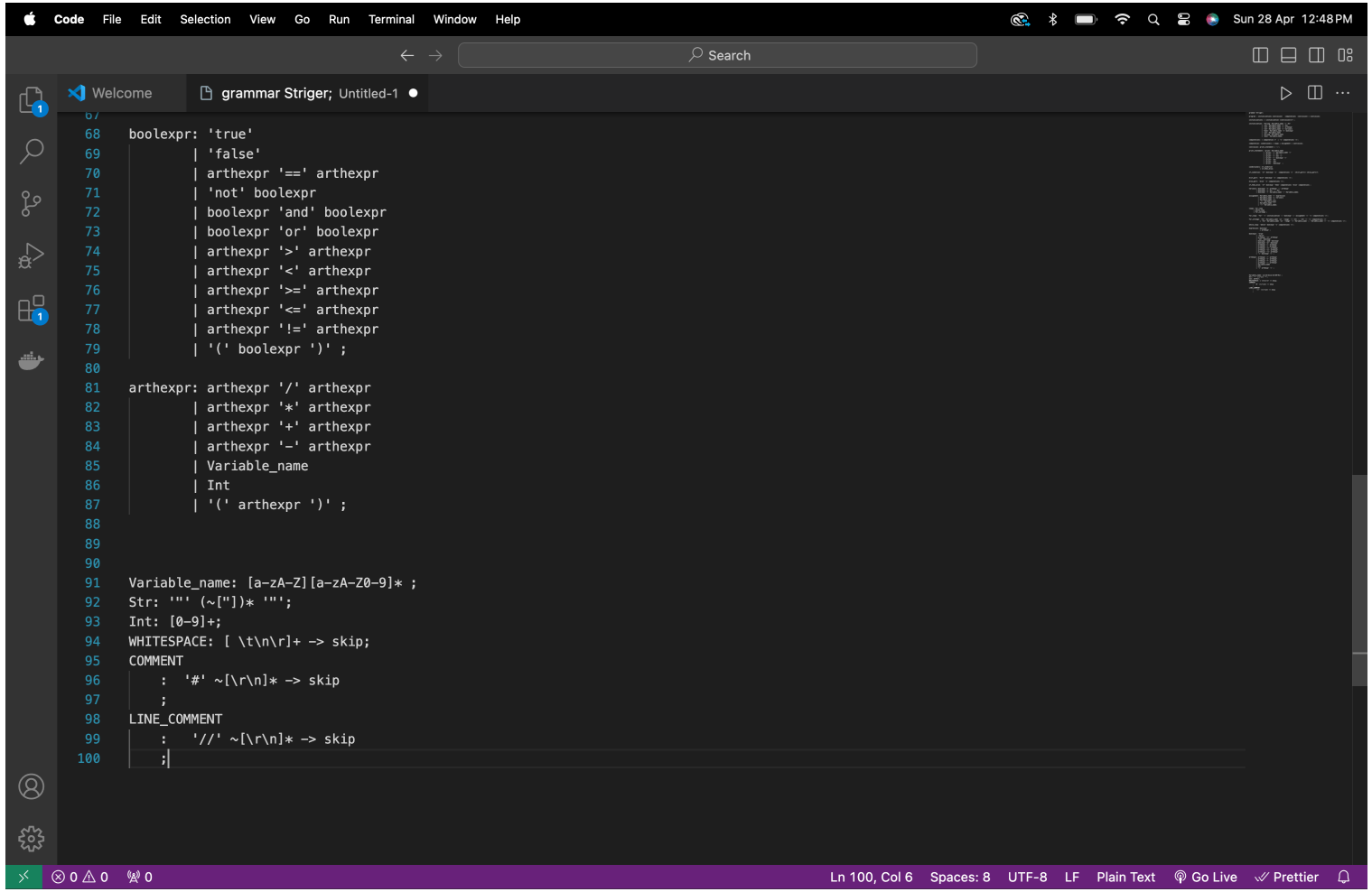
- *2. Computation: The computation part of the program is the main part of the program where all calculations and manipulations happen. This part consists of conditionals and loops:*
- *a. Conditionals: These are conditions as the name suggests that compute values if a condition is met(True or False).*
- *Example: “if” sky is light blue then it is day, “else” it is night.*
- *Syntax: if (boolean_expression):*
- *Arithmetic_expression*
- *Else:*
- *Arithmetic_expression*
- *b. Loops: These are lines of the program that repeat a part of the program when a condition is met until the condition fails.*
- *Example: while time>9am and time<5pm, you have to work*
- *Syntax:*
- *while(boolean_expression):*
- *arithmetic_expression*

Components of a Striger Program(contd..)

- *3. Conclusion: This part of the program is an optional part of the program where the results from the computation are displayed or returned for further computation.*
- *Example:*
- *print(x) → 23 (x is the variable defined in a previous example for initialization).*
- *print('2312312asafg')*
- *print(231)*
- *print(2<5)--> true.*

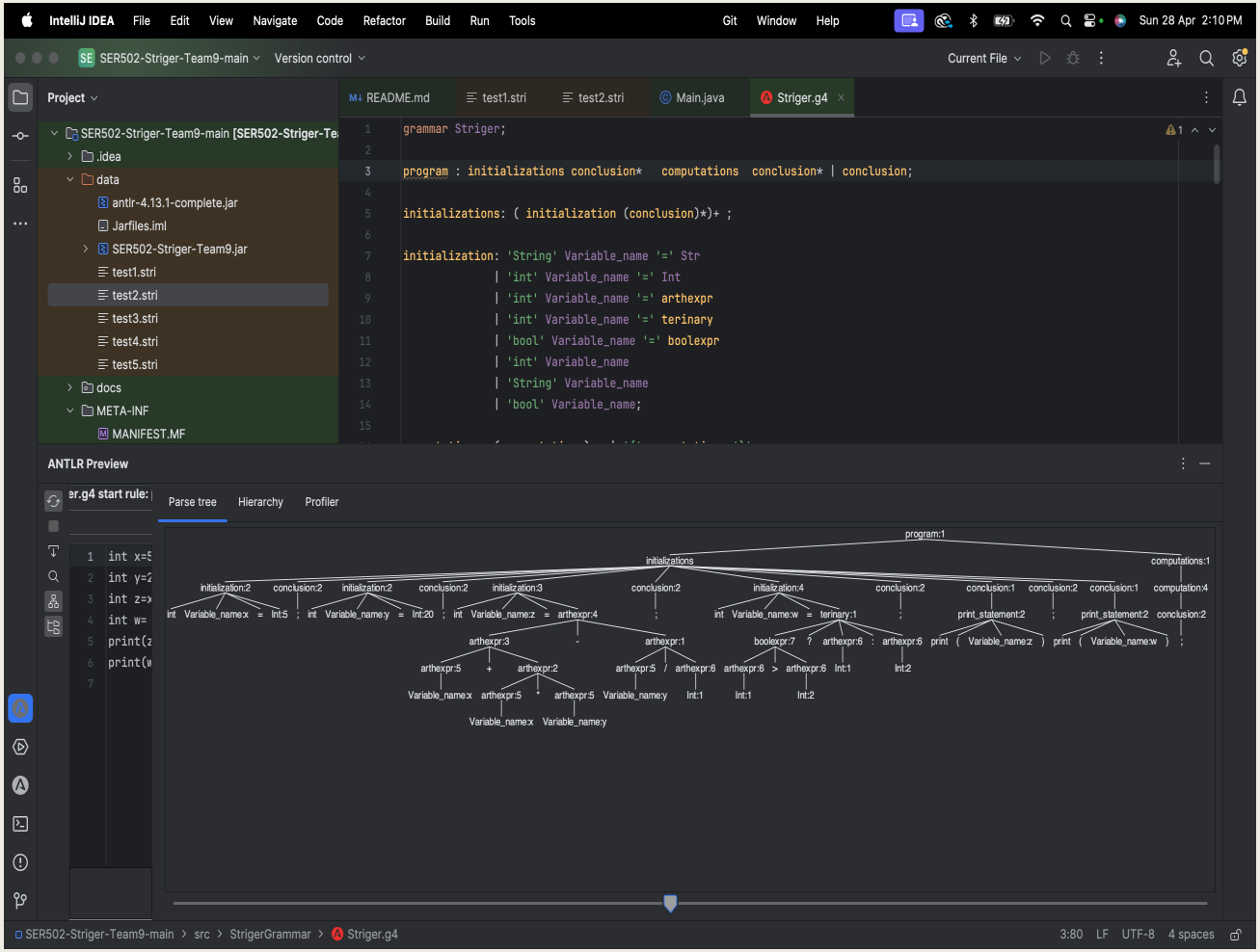

Code File Edit Selection View Go Run Terminal Window Help
Sun 28 Apr 12:48 PM
Search
Welcome
grammar Striger; Untitled-1
grammar Striger;
program : initializations conclusion* computations conclusion* | conclusion;
initializations: (initialization (conclusion)*)+ ;
initialization: 'String' Variable_name '=' Str
| 'int' Variable_name '=' Int
| 'int' Variable_name '=' arthexpr
| 'int' Variable_name '=' terinary
| 'bool' Variable_name '=' boolexpr
| 'int' Variable_name
| 'String' Variable_name
| 'bool' Variable_name;
computations: (computation)+ | '{' computations '}';
computation: conditionals | loops | assignment | conclusion;
conclusion: print_statement | ';';
print_statement: 'print' Variable_name
| 'print' '(' Variable_name ')'
| 'print' '(' Str ')'
| 'print' '(' Int ')'
| 'print' '(' boolexpr ')'
| 'print' Str
| 'print' Int
| 'print' boolexpr ;
conditionals: if_condition
| if_then_else;
if_condition: 'if' boolexpr '{' computations '}' (elif_part)* (else_part)?;
STRIGER GRAMMAR
Grammar is written in antlr.


Code File Edit Selection View Go Run Terminal Window Help
Sun 28 Apr 12:53 PM
Search
Welcome
grammar Striger; Untitled-1
if_condition: 'if' boolexpr '{' computations '}' (elif_part)* (else_part)?;
elif_part: 'elif' boolexpr '{' computations '}';
else_part: 'else' '{' computations '}';
if_then_else: 'if' boolexpr 'then' computations 'else' computations ;
terinary: boolexpr '?' arthexpr ':' arthexpr
| boolexpr '?' Str ':' Str
| boolexpr '?' Variable_name ':' Variable_name;
assignment: Variable_name '=' expression
| Variable_name '=' terinary
| Variable_name '++'
| '++' Variable_name
| Variable_name '--'
| '--' Variable_name;
loops: for_loop
| while_loop
| for_inrange ;
for_loop: 'for' '(' initialization ';' boolexpr ';' assignment ')' '{' computations '}';
for_inrange: 'for' Variable_name 'in' 'range' '(' Int ',' Int ')' '{' computations '}'
| 'for' Variable_name 'in' 'range' '(' Variable_name ',' Variable_name ')' '{' computations '}';
while_loop: 'while' boolexpr '{' computations '}';
expression: boolexpr
| arthexpr ;
boolexpr: 'true'
| 'false'
| arthexpr '==' arthexpr
| 'not' boolexpr
0 0 0
Ln 86, Col 14 Spaces: 8 UTF-8 LF Plain Text Go Live Prettier

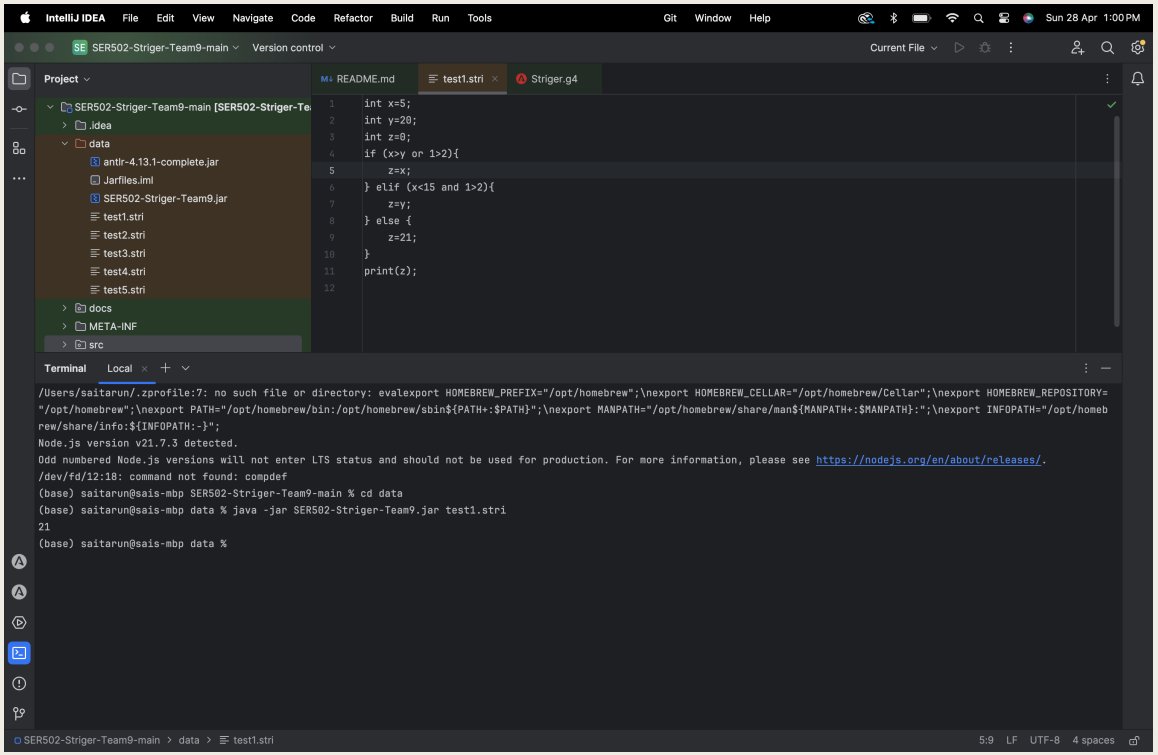

Code File Edit Selection View Go Run Terminal Window Help
Sun 28 Apr 12:48 PM
Search
Welcome
grammar Striger; Untitled-1
boolexpr: 'true'
| 'false'
| arthexpr '==' arthexpr
| 'not' boolexpr
| boolexpr 'and' boolexpr
| boolexpr 'or' boolexpr
| arthexpr '>' arthexpr
| arthexpr '<' arthexpr
| arthexpr '>=' arthexpr
| arthexpr '<=' arthexpr
| arthexpr '!=' arthexpr
| '(' boolexpr ')' ;
arthexpr: arthexpr '/' arthexpr
| arthexpr '*' arthexpr
| arthexpr '+' arthexpr
| arthexpr '-' arthexpr
| Variable_name
| Int
| '(' arthexpr ')' ;
Variable_name: [a-zA-Z][a-zA-Z0-9]* ;
Str: '"' (~["])* '"';
Int: [0-9]+;
WHITESPACE: [\t\n\r]+ -> skip;
COMMENT
: '#' ~[\r\n]* -> skip
;
LINE_COMMENT
: '//' ~[\r\n]* -> skip
;
Ln 100, Col 6
Spaces: 8
UTF-8
LF
Plain Text
Go Live
Prettier

Sample Code

```
int x=5;
int y=20;
int z=0;
if (x>y or 1>2){
    z=x;
} elif (x<15 and 1>2){
    z=y;
} else {
    z=21;
}
print(z);
```

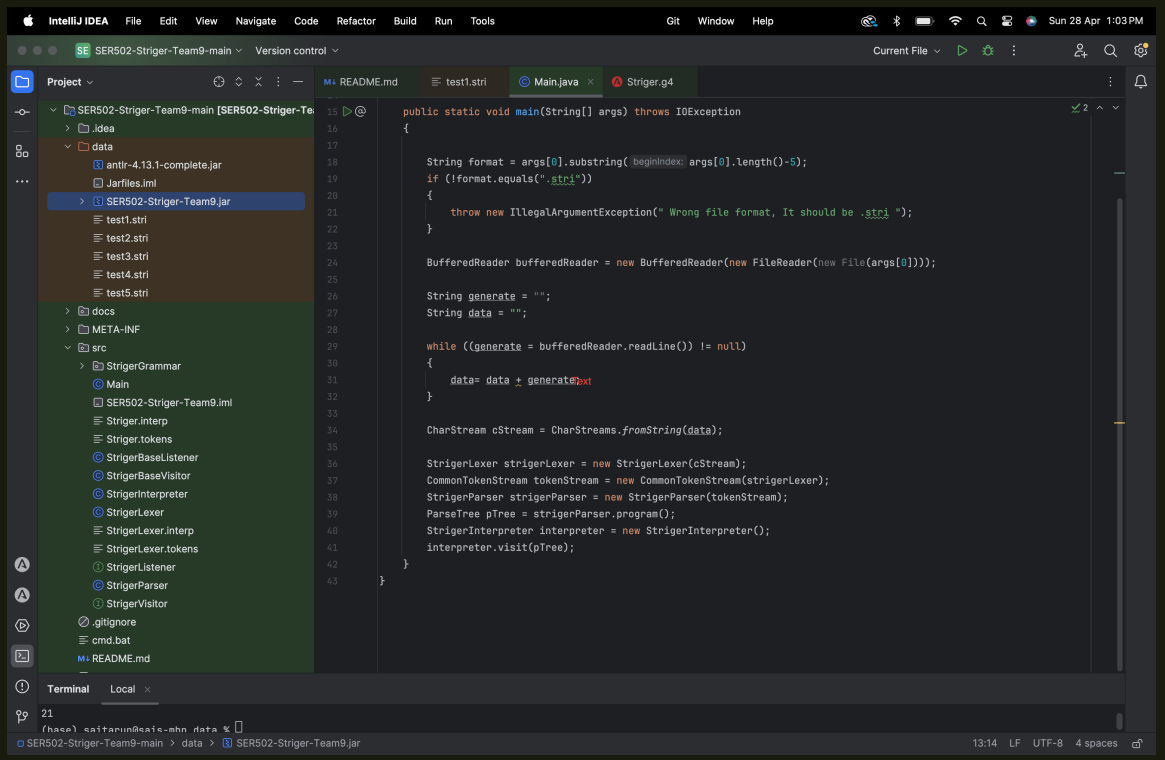

IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help Sun 28 Apr 2:10 PM
SER502-Striger-Team9-main Version control Current File
Project
SER502-Striger-Team9-main [SER502-Striger-Te
.idea
data
antlr-4.13.1-complete.jar
Jarfiles.iml
SER502-Striger-Team9.jar
test1.stri
test2.stri
test3.stri
test4.stri
test5.stri
docs
META-INF
MANIFEST.MF
README.md
test1.stri
test2.stri
Main.java
Striger.g4
grammar Striger;
program : initializations conclusion* computations conclusion* | conclusion;
initializations: (initialization (conclusion)*)+ ;
initialization: 'String' Variable_name '=' Str
| 'int' Variable_name '=' Int
| 'int' Variable_name '=' arthexpr
| 'int' Variable_name '=' terinary
| 'bool' Variable_name '=' boolexpr
| 'int' Variable_name
| 'String' Variable_name
| 'bool' Variable_name;
ANTLR Preview
er.g4 start rule:
Parse tree
Hierarchy
Profiler
program:1
initializations
computations:1
SER502-Striger-Team9-main src StrigerGrammar Striger.g4
3:80 LF UTF-8 4 spaces

PARSE TREE

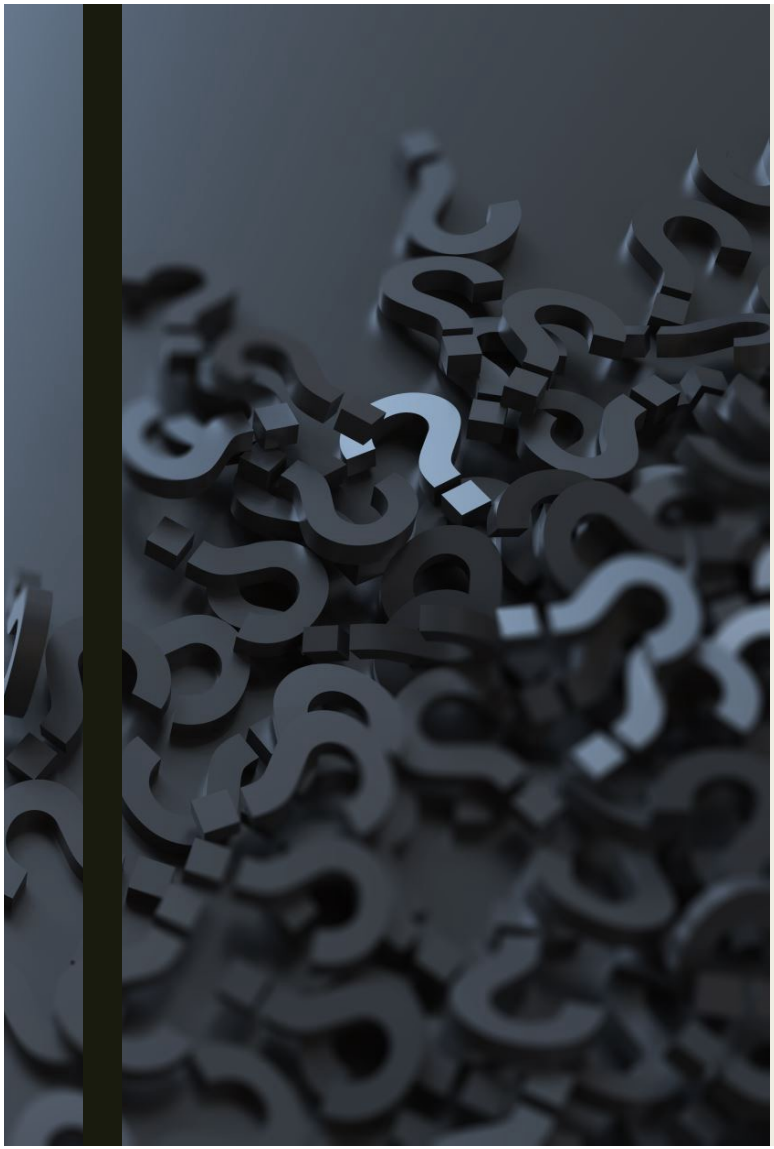


Sample Run and Output

To run the code, the command is
java -jar SER502-Striger-Team9.jar file_name.stri



INTERPRETER RUNTIME



Future Implementation

- Additional large datatypes like double and long.
- Providing input during run time
- Arrays and lists
- Functions

THANK YOU