BAGEL

Team 16 – SER 502 : Final Project

Ishan Dikshit (1211479279)
Sanchit Narang(1211092893)
Sonali Umesh(1211223881)

OVERVIEW

- Language Introduction
- Features
- Tools Used
- Information about Lexical Analysis and Parsing
- Runtime
- Sample Programs
- Demonstration Of Our Language

INTRODUCTION

- Our Language is mainly inspired from Python and Java
- Easy to code, because of familiar keywords used.
- Runtime is written in Java
- Parse tree gives a detailed view of the program for better understanding.

FEATURES

Datatypes:
- Integer
- Boolean

Mathematical Operators:
- Addition
- Subtraction
- Multiplication
- Division

Comparision Operators:
- Equals
- GreaterThan
- LessThan
- LessThanEqual
- GreaterThanEqual
- NotEqual

Decision Constructs

- if condition { statements; }
- if condition { statements; } else (statements;}

Loop Constructs

- while condition (statements;}

TOOLS USED

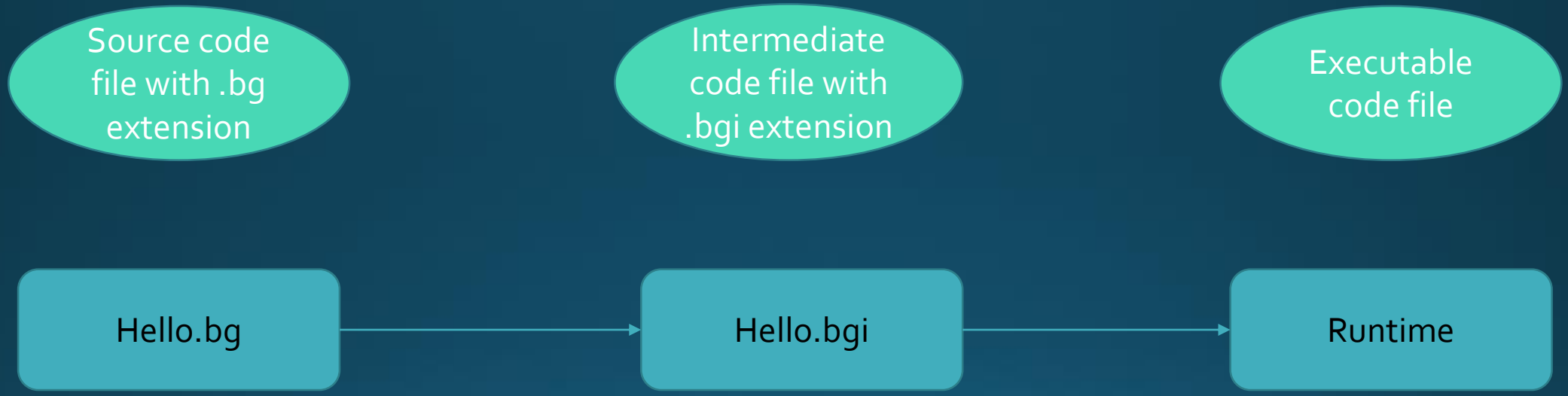
Compiler:

- Java Based
- Lexer and Parser – Built using *ANTLR v4.7 in Eclipse*
- Grammar – Using .g4 file on ANTLR
- Intermediate code generated using the ListenerClass (which is automatically generated by ANTLR)

Runtime/Interpreter:

- Java Based
- Runs the intermediate code generated by the compiler and gives the output

How Bagel Works?



GRAMMAR

```
60  float_literal : (('+'|'-')? DIGIT'.'DIGIT+);
61
62  operator : (ADDITION_OPERATOR | SUBTRACTION_OPERATOR | MULTIPLICATION_OPERATOR | DIVISION_OPERATOR);
63
64  declaration_statement : DATATYPE ' ' identifier ;
65
66  term : integer_literal | float_literal | identifier | BOOLEAN_KEYWORDS ;
67
68  basic_expression : term | (term (' ')?('\n')?operator(' ')?('\n')? term) ;
69
70  relational_expression : (basic_expression (' ')?('\n')? COMPARISON_KEYWORDS (' ')?('\n')? basic_expression) ;
71
72  complex_expression : basic_expression |relational_expression  ;
73
74  condition : complex_expression ;
75
76  return_statement : PRINT_KEYWORD (' ')? (QUOTE)? complex_expression (QUOTE)?;
77
78  while_loop : (WHILE_KEYWORD (' ')?('\n')? condition (' ')?('\n')? OPEN_BRACE (' ')?('\n')? statements (' ')?('\n')? CLOSE_BRACE) ;
79
80  if_statement: IF_KEYWORD (' ')?('\n')? condition (' ')?('\n')? OPEN_BRACE (' ')?('\n')? statements (' ')?('\n')? CLOSE_BRACE ((' ')?('\n')?
81
82  else_statement : (ELSE_KEYWORD (' ')?('\n')? OPEN_BRACE (' ')?('\n')? statements (' ')?('\n')? CLOSE_BRACE) ;
83
84  ifelse_statement : if_statement ((' ')?('\n')? else_statement(' ')?('\n')?)? ;
85
86  construct_statement : (ifelse_statement | while_loop) ;
87
88  assignment_statement : term (' ')?('\n')? ASSIGNMENT_KEYWORD (' ')?('\n')? complex_expression ;
89
90  other_statement : (assignment_statement | declaration_statement | return_statement | basic_expression | term | relational_expression) ;
91
92  statements : ((construct_statement | other_statement) (' ')? ('\n')?)* ';' ;
93
94  program : statements;
```

LEXER & PARSER USING ANTLR

Lexer:

- ANTLR reads the input and divides it into tokens based on the grammar provided.

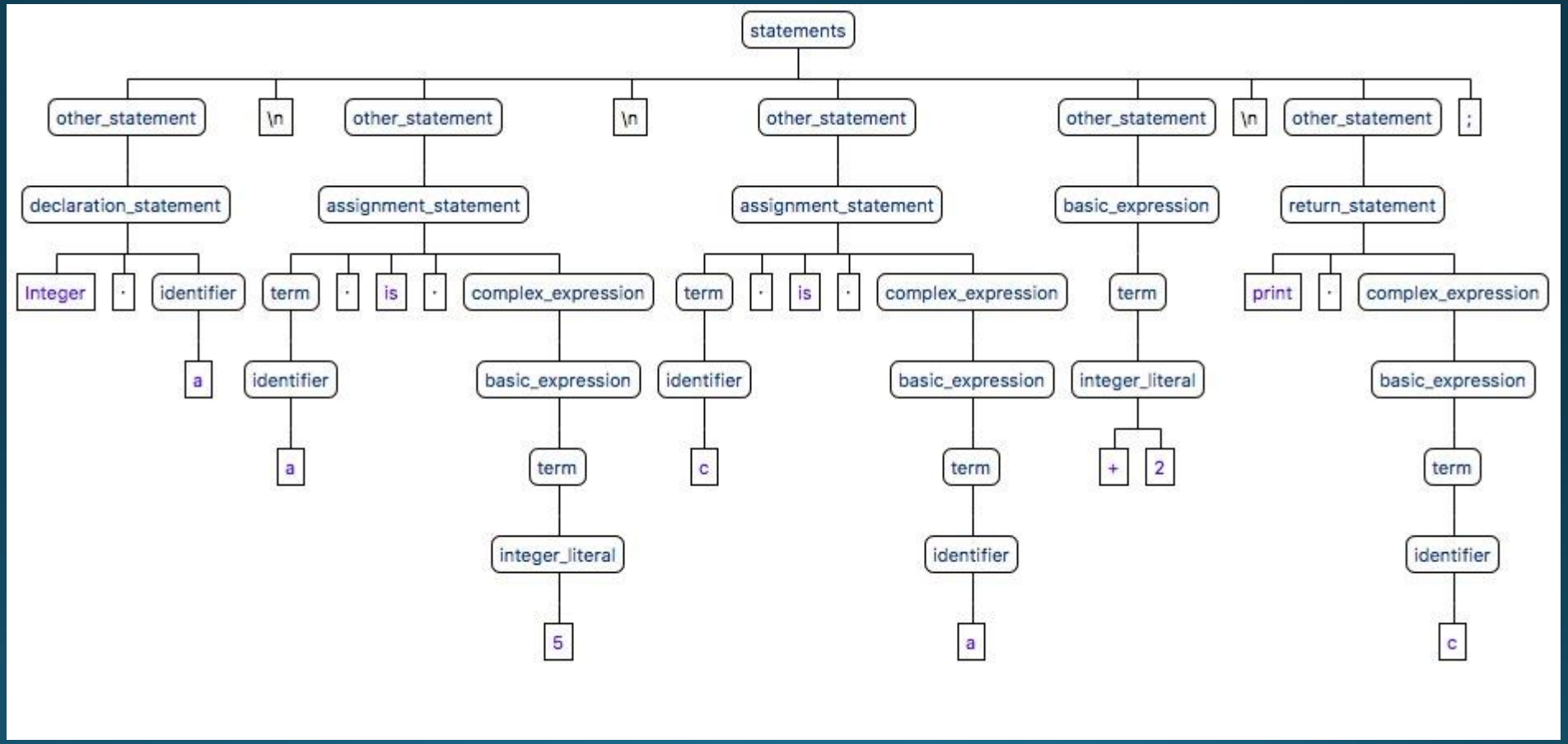
Parser:

- The tokens generated from the lexer go as input to the parser.
- The tokens are parsed and a parse tree is generated.
- Each node in the tree is a grammar NonTerminal / Terminal

File Input → *Lexical Analysis* → Tokens → *Parser* → Parse Tree → Intermediate Code

Parse Tree

```
Integer a
a is 5
c is a+2
print c;
```



Intermediate Code Operations

- DECLARE – Declares the data type of the variable
- STORE – Gets input value
- PUSH – Saves the value into the declared variable
- GET – Retrieves the variable from the stack
- OPERATOR – Shows which operation is being performed
- COMPARATOR – Shows the comparision operation being performed
- CONDITIONNOTTRUE – It tells what has to be done next when the condition fails
- JUMP – Jumps the lines in the code until it encounters the given label

RUNTIME

- Bagel’s runtime is completely based on Java
- It internally uses stack and hash map
- The intermediate code contains prefix expressions. Hence, the runtime is designed accordingly, such that it processes those prefix expressions to give the output

Sample High-Level and Intermediate Code

```
Integer a
Integer b
a is 10
b is 5
if
a GreaterThan b
{ print "AGreater"; }
else
{ print "BGreater"; };
```

```
DECLARE Integer a
DECLARE Integer b
STORE 10
PUSH a
STORE 5
PUSH b
GET a
GET b
COMPARE GreaterThan
CONDITIONNOTTRUE JUMP LABELELSE
SCOPEBEGIN
PRINT "AGreater"
IFSCOPEEND
JUMP ELSESCOPEEND
LABELELSE
ELSESCOPEBEGIN
PRINT "BGreater"
ELSESCOPEEND
```

High-Level Code

```
Integer a
Integer b
a is 10
b is 5
if
a GreaterThan b
{ print "InsideIf" a is 2; }
else
{ print "InsideElse" b is 4; };
```

Intermediate Code

```
DECLARE Integer a
DECLARE Integer b
STORE 10
PUSH a
STORE 5
PUSH b
GET a
GET b
COMPARE GreaterThan
CONDITIONNOTTRUE JUMP LABELELSE
SCOPEBEGIN
PRINT "InsideIf"
STORE 2
PUSH a
IFSCOPEEND
JUMP ELSESCOPEEND
LABELELSE
ELSESCOPEBEGIN
PRINT "InsideElse"
STORE 4
PUSH b
ELSESCOPEEND
```

Output

THANK YOU!!!