Antephanie

By Anfal Al-Hussaini and Stephanie Motz

Java Arithmetic Operators	Antephanie Arithmetic Operators
Minus, Plus	-, [P] (Phosphorus)
Mult, Div	[Ts] (Tennessine), [Dy] (Dysprosium)
Increment, Decrement	[In] (Indium), [Md] (Mendelevium)

Java Assignment Operators	Antephanie Assignment Operators
Plus Equal, Minus Equal	[P=], [-=]
Times Equal, Divide Equal	[Ts=], [Dy=]

Java Comparison Operators	Antephanie Comparison Operators
Equal, Equal Equal, Bang Equal	=, ==, [No=]
Less, Less Than	<, <=
Greater, Greater Than	>, >=

Java Logical Operators	Antephanie Logical Operators
And	[Am] (Americium)
Or	[O] (Oxygen)
Not	[No] (Nobelium)

Java Decision-Making	Antephanie Decision-Making
If	Reactant
Else	Product
Switch	Experiment
Case	Reaction
Break	Spill

Java Looping	Antephanie Looping
For	For
While	While
Do	Do

Literals & Comments	Antephanie Literals & Comments
Var	Independent
Identifier	Indentifier
Print	Formula
Start Comments	#PERIODIC
End Comments	PERIODIC#
Single Line Comments	%

Steps to Building the Parser

- **Found a Parser Generator**
- **Define Grammar**: Started by defining the grammar of 'Antephanie'
- **Choose Parsing Technique**: Recursive Top-Down Approach (starts at the beginning and follows the rules step by step until it understands the whole language)
- **Implement Lexer**
- **Implement Parser**: Analyze token stream from lexer and construct a parse tree or AST representing the structure of the input.
- **Handle Errors**: Provide informative error messages and suggestions for corrections.
- **Test**: Write test cases to verify that the parser behaves correctly. Test both the Lexer and Parser components thoroughly.
- **Documentation**

Steps to Building the Interpreter

- Work with defined syntax of language
- Implement the Lexer to tokenize input
- Develop the parser to generate AST from tokens
- Design the evaluator to traverse and interpret the AST
- Implement the evaluator to execute operations based on AST nodes
- Handle errors and exceptions in parsing and evaluation
- Test the interpreter with various inputs and test cases

Output at each level:

LEXER/SCANNER:

User input: 1 [P] 2

Output: [INT: 1; PLUS; INT:2; EOF]

PARSER:

User input: 1 [P] 2

Output: (+ 1.0 2.0)

INTERPRETER/EVALUATOR:

User input: 1 [P] 2

Output: 3

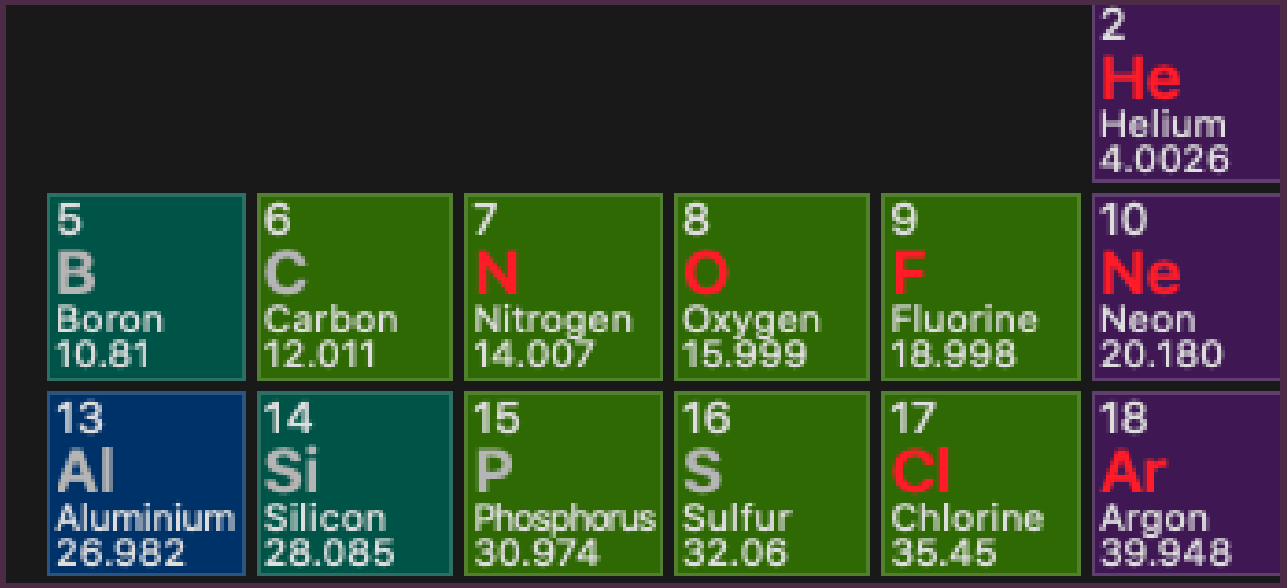
BNF Grammar Example

```
expression     → assignment

assignment     → "VAR" IDENTIFIER "=" assignment
|   |   |   |   | logic_or

logic_or       → logic_and ( "||" logic_and )*

logic_and      → logic_not ( "&&" logic_not )*

logic_not      → "!" logic_not
|   |   |   |   | equality

equality       → comparison ( ( "!=" | "==" ) comparison )*

comparison     → term ( ( ">" | ">=" | "<" | "<=" ) term )*

term           → factor ( ( "-" | "+" ) factor )*

factor         → unary ( ( "/" | "*" ) unary )*
```

```
unary          → ( "-" ) unary | primary
|   |   |   |   | primary "++"
|   |   |   |   | primary "--"

primary        → NUMBER | "(" term ")"
|   |   |   |   | if_elif_else
|   |   |   |   | for_loop
|   |   |   |   | while_loop


if_elif_else   → "IF" expression "THEN" expression
|   |   |   |   | ("ELSEIF" expression "THEN" expression)*
|   |   |   |   | ("ELSE" expression)?

for_loop       → "FOR" IDENTIFIER "IN" range "DO" expression

range          → IDENTIFIER ":" IDENTIFIER

while_loop     → "WHILE" "(" expression ")" "DO" expression
```

Sample Code- Interpreter

- Input:
  - INDEPENDENT element = 14
  - REACTANT element > 1 [Am] element < 18 PRODUCT element – 6
- Output:
  - 14
  - 8



Demo

Examples of Operators in Use:

Arithmetic

```
Input your Chemical Reaction: INDEPENDENT element = 50
50
Input your Chemical Reaction: element [Dy] 5
10
Input your Chemical Reaction:
```

Comparison/Logical

```
Input your Chemical Reaction: INDEPENDENT x = 5
5
Input your Chemical Reaction: x > 3
true
Input your Chemical Reaction: x > 3 [Am] x < 6
true
```

Decision Making

```
Input your Chemical Reaction: INDEPENDENT weight = 50
50
Input your Chemical Reaction: REACTANT weight < 100 [O] weight > 51 PRODUCT 0
0
Input your Chemical Reaction:
```

Demo of If/Then Statements

DEBUG CONSOLE OUTPUT TERMINAL PORTS GITLENS

Run: Antephanie

```
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) MacBook-Air-7:Antephanie- stephaniemotz$  /usr/bin/env /Library/Java/JavaVirtualMachines/adoptopenjdk-16.jdk/Contents/Home/bin/java -XX:+ShowCodeDetailsInExcepti
onMessages -cp /Users/stephaniemotz/Library/Application\ Support/Code/User/workspaceStorage/52a94ddb39be12e75c018c8c08680cfa/redhat.java/jdt_ws/Antephanie-_d4ae4af1/bin
 src.Antephanie
Input your Chemical Reaction:
```

OUTLINE
TIMELINE
JAVA PROJECTS

Interpreter* 1↓ 0↑ 0 1 0 Java: Ready You, 3 days ago Ln 17, Col 1 Tab Size: 4 UTF-8 LF Java

Demo of If/Then Statements

```
(base) MacBook-Air-7:Antephanie- stephaniemotz$ /usr/bin/env /Library/Java/JavaVirtualMachines/adoptopenj
dk-16.jdk/Contents/Home/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/stephaniemotz/Library/
Application\ Support/Code/User/workspaceStorage/52a94ddb39be12e75c018c8c08680cfa/redhat.java/jdt_ws/Anteph
anie-_d4ae4af1/bin src.Antephanie
Input your Chemical Reaction:
```

Demo of If/Then Statements (ELIF/ELSE)

```
(base) MacBook-Air-7:Antephanie- stephaniemotz$ /usr/bin/env /Library/Java/JavaVirtualMachines/adoptopenj
dk-16.jdk/Contents/Home/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/stephaniemotz/Library/
Application\ Support/Code/User/workspaceStorage/52a94ddb39be12e75c018c8c08680cfa/redhat.java/jdt_ws/Anteph
anie-_d4ae4af1/bin src.Antephanie
Input your Chemical Reaction:
```

For Loop Example Code

```
(base) MacBook-Air-7:Antephanie- stephaniemotz$  /usr/bin/env /Library/Java/JavaVirtualMachines
/adoptopenjdk-16.jdk/Contents/Home/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/
stephaniemotz/Library/Application\ Support/Code/User/workspaceStorage/52a94ddb39be12e75c018c8c0
8680cfa/redhat.java/jdt_ws/Antephanie-_d4ae4af1/bin src.Antephanie
Input your Chemical Reaction: INDEPENDENT h20 = 0
0
Input your Chemical Reaction: FOR i = 0 TO 10 PRODUCT INDEPENDENT h20 = h20 + 1
11
```

```
(base) MacBook-Air-7:Antephanie- stephaniemotz$  /usr/bin/env /Library/Java/JavaVirtualMachines/adoptopenjdk-16.jdk/Contents/Home/bin/java -XX:+ShowCodeDetailsInExcepti
onMessages -cp /Users/stephaniemotz/Library/Application\ Support/Code/User/workspaceStorage/52a94ddb39be12e75c018c8c08680cfa/redhat.java/jdt_ws/Antephanie-_d4ae4af1/bin
 src.Antephanie
Input your Chemical Reaction: INDEPENDENT x = 1
1
Input your Chemical Reaction: FOR i = 0 TO 10 STEP 2 PRODUCT INDEPENDENT x = x + 1
7
Input your Chemical Reaction:
```

Error Handling:

```
(base) MacBook-Air-7:Antephanie- stephaniemotz$  /usr/bin/env /Library/Java/JavaVirtualMachines/adoptopenjdk-16.jdk/Contents/Home/bin/java -XX:+ShowCodeDetailsInExcepti
onMessages -cp /Users/stephaniemotz/Library/Application\ Support/Code/User/workspaceStorage/52a94ddb39be12e75c018c8c08680cfa/redhat.java/jdt_ws/Antephanie-_d4ae4af1/bin
 src.Antephanie
Input your Chemical Reaction: INDEPENDENT x = 7
7
Input your Chemical Reaction: FOR i = 1 - 9
Error: Incorrect For-Loop Formatting. Try Again
```

Future Implementations

- Add back in the actual periodic elements and get them working. Minus (-) = [S]
- Printing a number would link it to the matching periodic element (1-103)
- File input instead of command line

Anfal

- Favorite Part:
  - Creating the tokens
- Least Favorite:
  - Figuring out the for loop

Stephanie

- Favorite Part:
  - Designing the syntax
- Least Favorite:
  - Taking the time to understand the generators