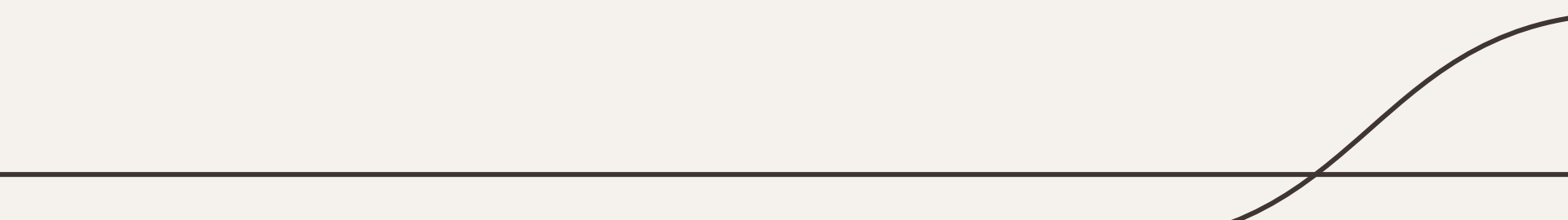




# Lexical Analyser

Team Compilyashki



---

# Our team: Compilyashki

## Polina Pushkareva

Responsible for  
organizational aspects  
and report

## Matthew Rusakov

Responsible for testing of  
lexical analyser



## Aliia Bogapova

Responsible for the  
lexical analyser coding

# Technologies



## Project F

Interpreter for lisp-like  
(functional) language



## Hand-based parser in Java



## Hand-based lexer in Java


# Description of the implementation



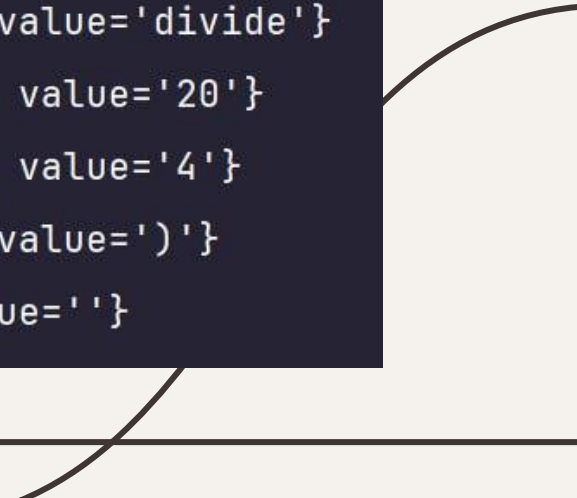
The flexer reads the input string, tokenizes it into individual components (tokens), and recognizes various language constructs such as parentheses, keywords, operators, numbers, and literals. The `Flexer` class contains an inner `Token` class and a set of enumerations (`TokenType`) representing different possible token types. The `tokenize()` method processes the input character by character, handling special cases like quoted elements, numbers, and keywords, converting them into tokens. The lexical analyzer also includes specific methods for parsing numbers, identifiers, and quoted expressions.

# A few examples of work

```
String input = "(divide 20 4)";
```



```
Token{type=LPAREN, value='('}  
Token{type=DIVIDE, value='divide'}  
Token{type=INTEGER, value='20'}  
Token{type=INTEGER, value='4'}  
Token{type=RPAREN, value=')'}  
Token{type=EOF, value=''}
```

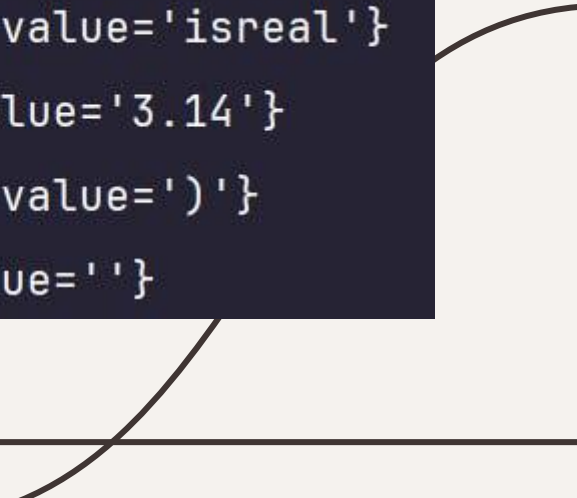


# A few examples of work

```
String input = "(isreal 3.14)";
```




```
Token{type=LPAREN, value='('}  
Token{type=ISREAL, value='isreal'}  
Token{type=REAL, value='3.14'}  
Token{type=RPAREN, value=')'}  
Token{type=EOF, value=''}
```

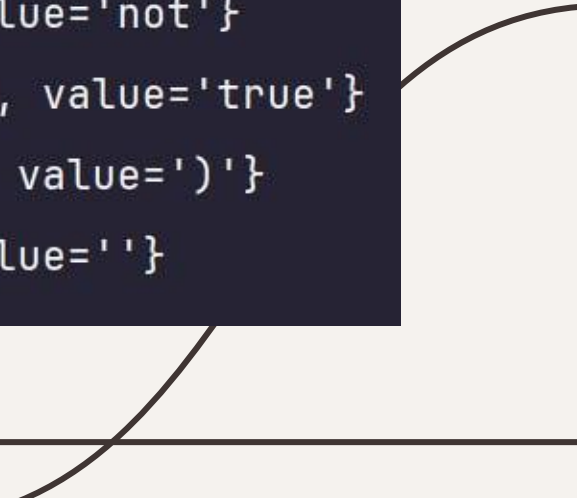


# A few examples of work

```
String input = "(not true)";
```

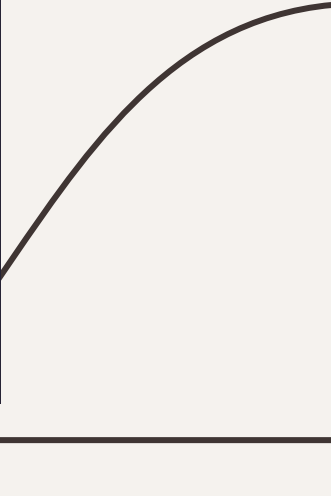



```
Token{type=LPAREN, value='('}  
Token{type=NOT, value='not'}  
Token{type=BOOLEAN, value='true'}  
Token{type=RPAREN, value=')'}  
Token{type=EOF, value=''}
```



# A few examples of work

```
String input = "(func powerOfThree (x) (times (square x) x))";
```



```
Token{type=RPAREN, value=')'}  
Token{type=LPAREN, value='('}  
Token{type=TIMES, value='times'}  
Token{type=LPAREN, value='('}  
Token{type=ATOM, value='square'}  
Token{type=ATOM, value='x'}  
Token{type=RPAREN, value=')'}  
Token{type=ATOM, value='x'}  
Token{type=RPAREN, value=')'}  
Token{type=RPAREN, value=')'}  
Token{type=EOF, value=''}
```



# A few examples of work

```
String input = "(setq subtractTwo (lambda (x) (minus x 2)))";
```



```
Token{type=LPAREN, value='('}  
Token{type=SETQ, value='setq'}  
Token{type=ATOM, value='subtractTwo'}  
Token{type=LPAREN, value='('}  
Token{type=LAMBDA, value='lambda'}  
Token{type=LPAREN, value='('}  
Token{type=ATOM, value='x'}  
Token{type=RPAREN, value=')'}  
Token{type=LPAREN, value='('}  
Token{type=MINUS, value='minus'}  
Token{type=ATOM, value='x'}  
Token{type=INTEGER, value='2'}  
Token{type=RPAREN, value=')'}  
Token{type=RPAREN, value=')'}  
Token{type=RPAREN, value=')'}  
Token{type=EOF, value=''}
```

The image features a light gray background with decorative wavy lines in the corners, creating a frame-like effect. The text is centered and reads:

**Thanks for  
attention!**

**Link to GitHub:**

[https://github.com/MattWay224/F24CompilerConstruction\\_Compilyashki](https://github.com/MattWay224/F24CompilerConstruction_Compilyashki)