Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра интеллектуальных информационных технологий

**Лабораторная работа №1 по курсу «Языковые процессоры интеллектуальных систем» на тему:**

**«**Разработка языка для работы с множествами**»**

Выполнил Кривецкий А.Э.

студент группы

121702:

Проверил: Соколович М.Г.

**МИНСК 2024**

1. **Требования к разрабатываемому языку:**
   1. Встроенные типы не менее трех
   2. Возможность инициализация переменных всех типов при объявлении: <тип>? <имя\_переменной> = <выражение>
      1. Инициализирующее выражение может быть константным
   3. Встроенные операции не менее 10 штук
   4. Встроенные функции
      1. Встроенные функции ввода\вывода для работы со встроенными типами read() write()
   5. Использование сложных выражений (составных и со скобками)
   6. Блочный оператор
   7. Управляющие структуры
      1. Условный оператор (if-then-else)
      2. Операторы цикла (while и until)
      3. Оператор цикла с итерациями (for)
   8. Пользовательские подпрограммы
      1. Передача и возврат параметров
   9. Задание локальной и глобальной области видимости для имен переменных
   10. Объявление переменных
       1. Явное
   11. Преобразование типов
       1. Явное, например, a = (int) b
   12. Оператор присваивания
       1. Многоцелевой, например, a, b = c, d
   13. Структуры, ограничивающие область видимости
       1. Подпрограммы и блочные операторы
   14. Маркер блочного оператора
       1. Явные, например, { } или begin end
   15. Условные операторы
       1. Двух вариантный оператор if-then-else
   16. Перегрузка подпрограмм
       1. Присутствует
   17. Передача параметров в подпрограмму
       1. По значению и результату
   18. Допустимое место объявления подпрограмм:
       1. В начале программы
   19. Встроенные типы: element, set
   20. Операции: переопределить +, -, \*, \ и т.д. для встроенных типов
2. **Вариант целевого кода:**
   1. Байт-код JVM, формат промежуточного кода ассемблер для JVM
3. **Грамматика языка:**
   1. Лексер:

lexer grammar MyLexer;

// Logical Operators

AND : '&&' ;

OR : '||' ;

NOT : '!' ;

// Relational Operators

GT : '>' ;

LT : '<' ;

NEQ : '!=' ;

GTE : '>=' ;

LTE : '<=' ;

EQ : '==' ;

// Assignment Operators

ASSIGN : '=' ;

PLUS\_ASSIGN : '+=' ;

MINUS\_ASSIGN : '-=' ;

MULT\_ASSIGN : '\*=' ;

DIV\_ASSIGN : '/=' ;

MOD\_ASSIGN : '%=' ;

// Delimiters

COMMA : ',' ;

SEMI : ';' ;

LPAREN : '(' ;

RPAREN : ')' ;

LCURLY : '{' ;

RCURLY : '}' ;

COLON : ':';

DOT : '.';

// Keywords

CLASS : 'class' ;

LOCAL : 'local' ;

GLOBAL : 'global' ;

CONSTRUCTOR : 'constructor' ;

FUNCTION : 'function' ;

FINAL : 'final' ;

VOID : 'void' ;

RETURN : 'return' ;

IF : 'if' ;

ELSE : 'else' ;

SWITCH : 'switch' ;

CASE : 'case' ;

FOR : 'for' ;

WHILE : 'while' ;

BREAK : 'break' ;

CONTINUE : 'continue' ;

// Arithmetic Operators

PLUS : '+' ;

MINUS : '-' ;

MULT : '\*' ;

DIV : '/' ;

MOD : '%' ;

// Data Types

STRING\_TYPE : 'string' ;

INT\_TYPE : 'int' ;

FLOAT\_TYPE : 'float' ;

BOOLEAN\_TYPE : 'boolean' ;

ELEMENT\_TYPE : 'element' ;

ELEMENT\_SET\_TYPE : 'elementSet' ;

// Literals

BOOLEAN : 'true' | 'false' ;

INT : [0-9]+ ;

FLOAT : [0-9]+ '.' [0-9]\* | '.' [0-9]+ ;

STRING : '"' ( ~["\\] | '\\' . )\* '"' ;

// Identifier

ID : [a-zA-Z\_][a-zA-Z\_0-9]\* ;

// Whitespace and Comments

WS : [ \t\n\r\f]+ -> skip ;

LINE\_COMMENT : '//' ~[\r\n]\* -> skip ;

BLOCK\_COMMENT : '/\*' .\*? '\*/' -> skip ;

* 1. Парсер:

parser grammar MyParser;

options { tokenVocab=MyLexer; } // Указание на использование созданного Lexer

// Точка входа

program : (functionDeclaration | statement | classDeclaration)\* EOF ;

// Правило для объявления функции

functionDeclaration

: (GLOBAL | LOCAL) FUNCTION type ID LPAREN parameterList? RPAREN block

;

// Правило для типа данных

type

: VOID

| INT\_TYPE

| STRING\_TYPE

| FLOAT\_TYPE

| BOOLEAN\_TYPE

| ELEMENT\_TYPE

| ELEMENT\_SET\_TYPE

;

// Список параметров

parameterList

: parameter (COMMA parameter)\*

;

//Параметр

parameter

: type ID

;

// Блок кода

block

: LCURLY statement\* RCURLY

;

// Оператор

statement

: variableDeclaration

| assignment

| methodCall

| returnStatement

| voidReturnStatement

| ifStatement

| switchStatement

| forStatement

| whileStatement

| breakStatement

| continueStatement

;

//Оператор выхода

breakStatement

: BREAK SEMI

;

//Оператор продолжения

continueStatement

: CONTINUE SEMI

;

// Условный оператор if

ifStatement

: IF LPAREN comparativeExpression RPAREN block (ELSE block)?

;

// Оператор switch

switchStatement

: SWITCH LPAREN expression RPAREN LCURLY caseStatement\* RCURLY

;

// Оператор case

caseStatement

: CASE expression COLON statement

;

// Оператор for

forStatement

: FOR LPAREN (variableDeclaration | expression)? comparativeExpression SEMI expression RPAREN block

;

// Оператор while

whileStatement

: WHILE LPAREN comparativeExpression RPAREN block

;

// Объявление переменной

variableDeclaration

: LOCAL? (type | ID) ID ASSIGN expression SEMI

;

// Присваивание

assignment

: ID (ASSIGN | PLUS\_ASSIGN | MINUS\_ASSIGN | MULT\_ASSIGN | DIV\_ASSIGN | MOD\_ASSIGN) expression SEMI

;

// Вызов метода

methodCall

: (expression (DOT ID LPAREN argumentList? RPAREN)

| ID LPAREN argumentList? RPAREN

) SEMI

;

// Список аргументов

argumentList

: expression (COMMA expression)\*

;

//Сравнительное выражение

comparativeExpression

: expression GT expression

| expression LT expression

| expression NEQ expression

| expression GTE expression

| expression LTE expression

| expression EQ expression

| comparativeExpression AND comparativeExpression

| comparativeExpression OR comparativeExpression

| NOT comparativeExpression

;

// Выражение

expression

: primary (DOT ID)?

| ID DOT ID LPAREN argumentList? RPAREN

| ID DOT methodCall

| expression PLUS expression

| expression MINUS expression

| expression MULT expression

| expression DIV expression

| expression MOD expression

| expression PLUS\_ASSIGN expression

| expression MINUS\_ASSIGN expression

| expression MULT\_ASSIGN expression

| expression DIV\_ASSIGN expression

| expression MOD\_ASSIGN expression

| typeCast

;

// Приведение типов

typeCast

: LPAREN type RPAREN expression

;

// Первичное выражение

primary

: INT

| FLOAT

| BOOLEAN

| STRING

| ELEMENT\_TYPE LPAREN argumentList? RPAREN

| ELEMENT\_SET\_TYPE LPAREN argumentList? RPAREN

| ID

| ID LPAREN argumentList? RPAREN

| LPAREN expression RPAREN

;

//Возврат со значением

returnStatement

: RETURN expression SEMI

;

// Возврат без значения для функций типа void

voidReturnStatement

: RETURN SEMI

;

//Декларация класса

classDeclaration

: (GLOBAL | LOCAL) CLASS ID classBlock

;

//Внутренность класса

classBlock

: LCURLY (constructor variableDeclaration\* functionDeclaration\*)+ RCURLY

;

//Конструктор

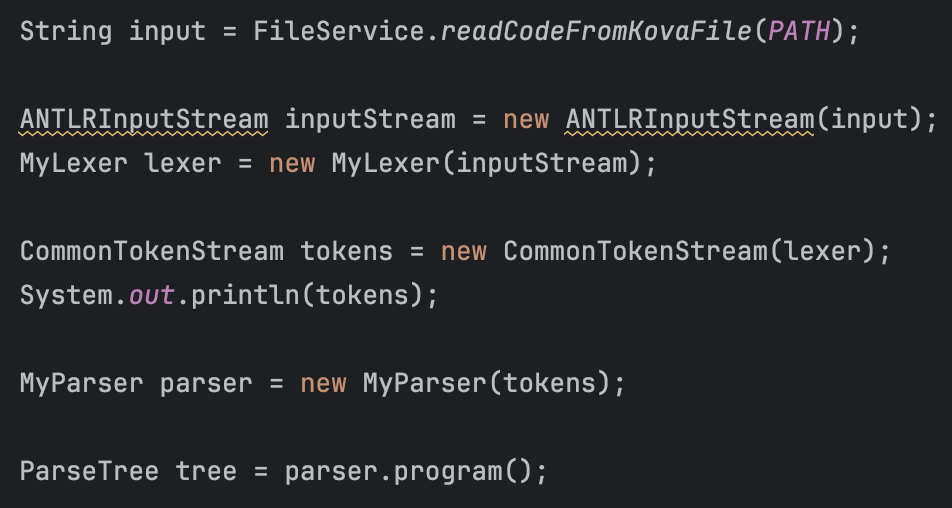
constructor

: (GLOBAL | LOCAL) CONSTRUCTOR LPAREN parameterList? RPAREN block

;

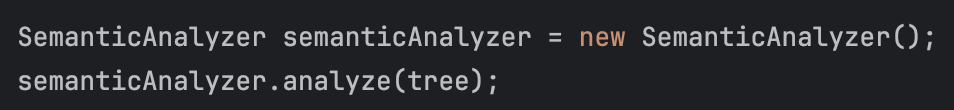
1. **Синтаксический анализатор:**

Синтаксический анализатор разработан на языке программирования Java:



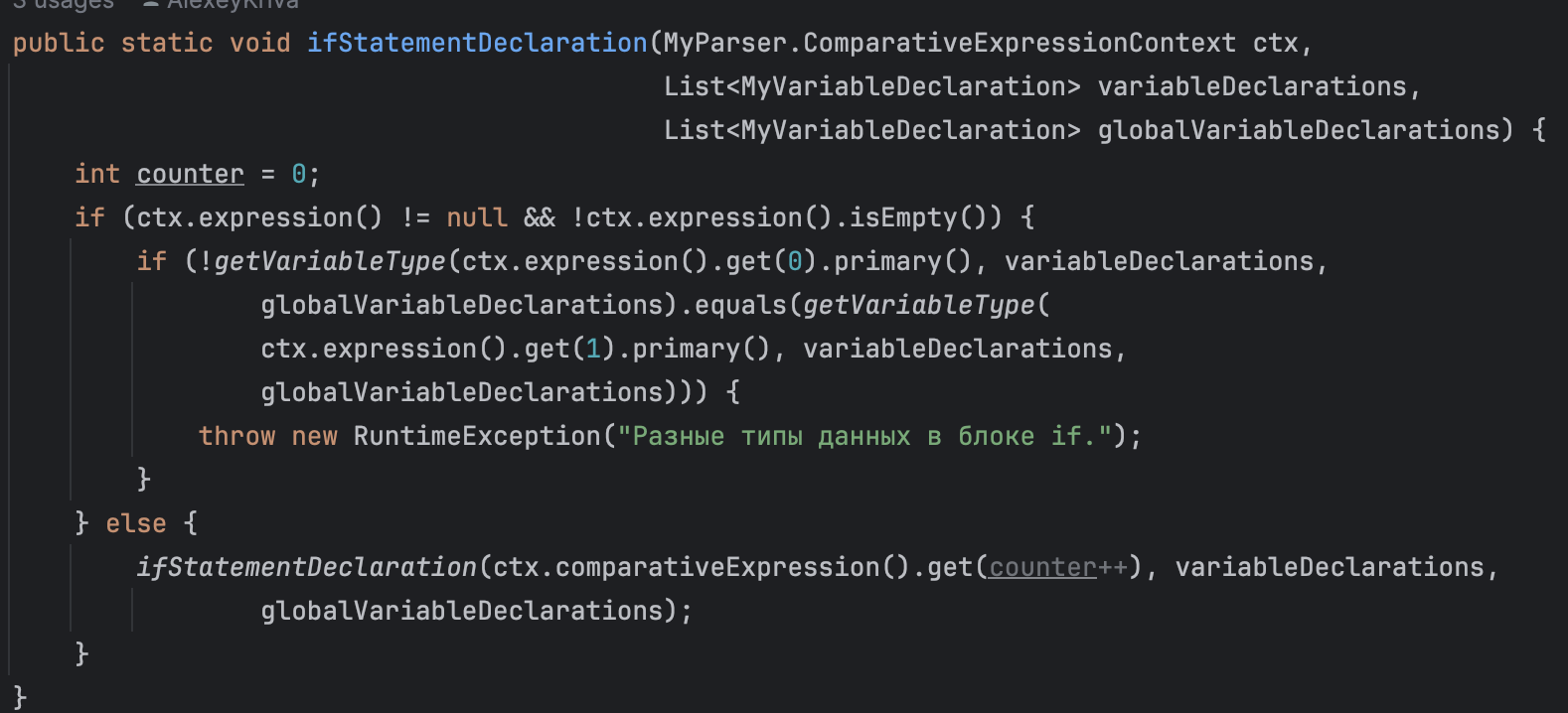
1. **Семантический анализатор:**

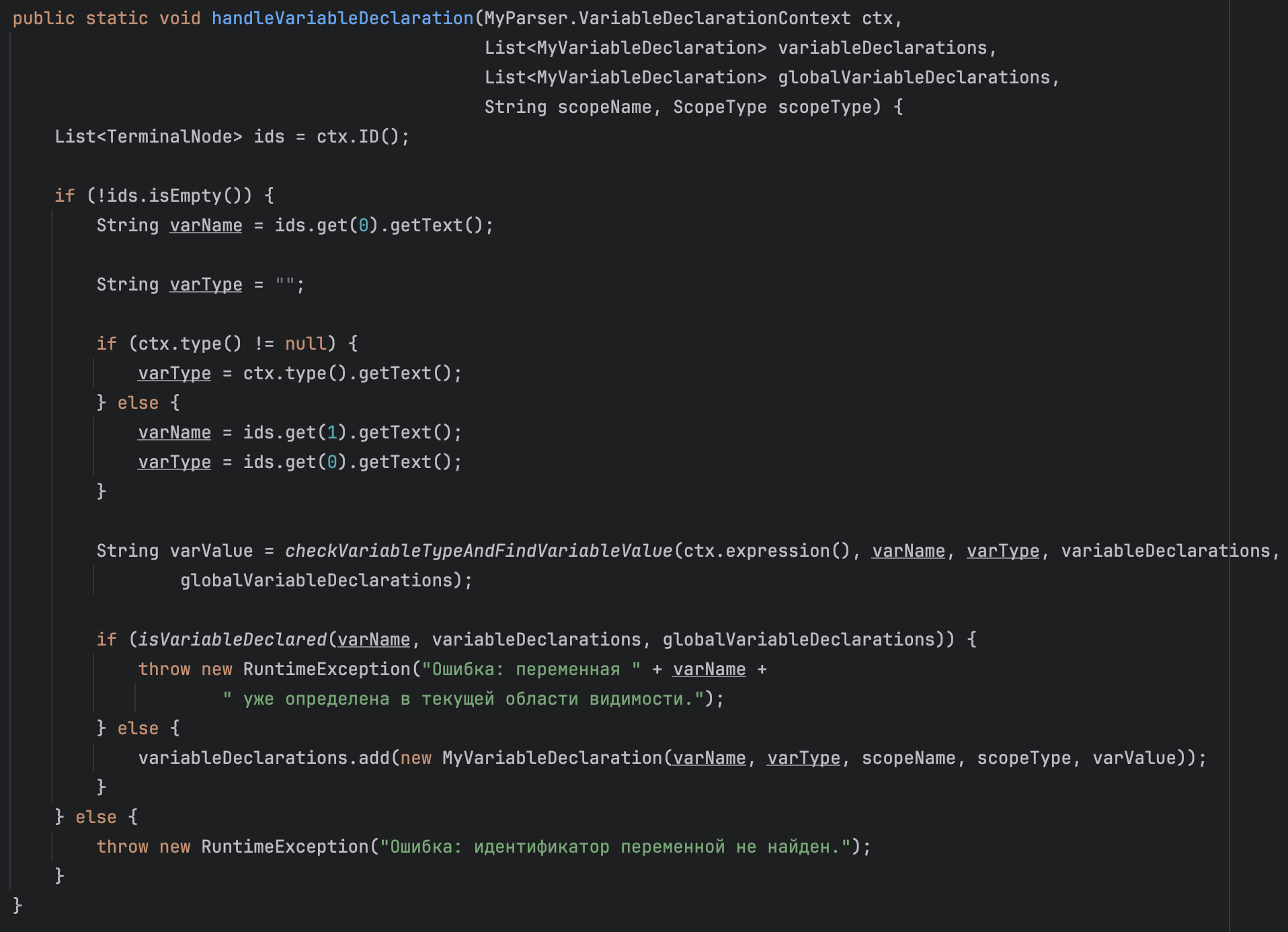
Семантический анализатор разработан на языке программирования Java:

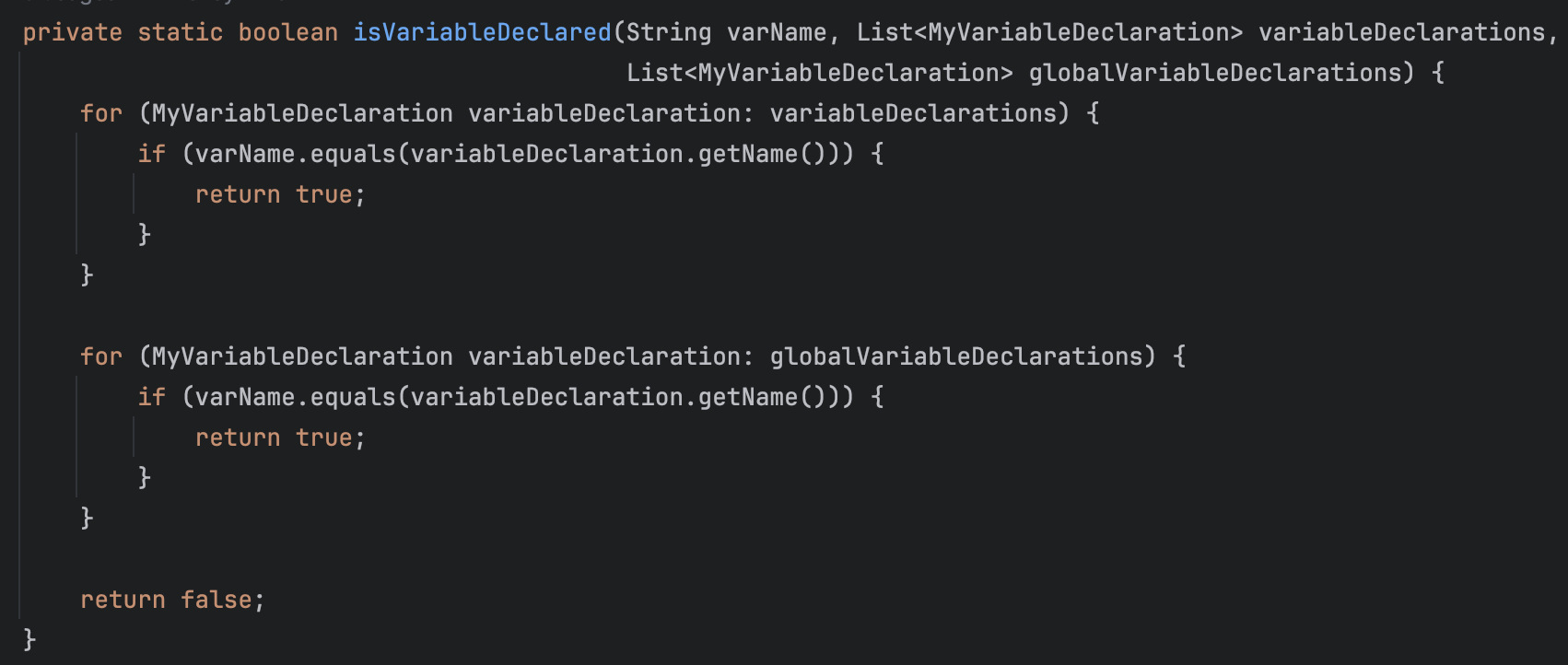


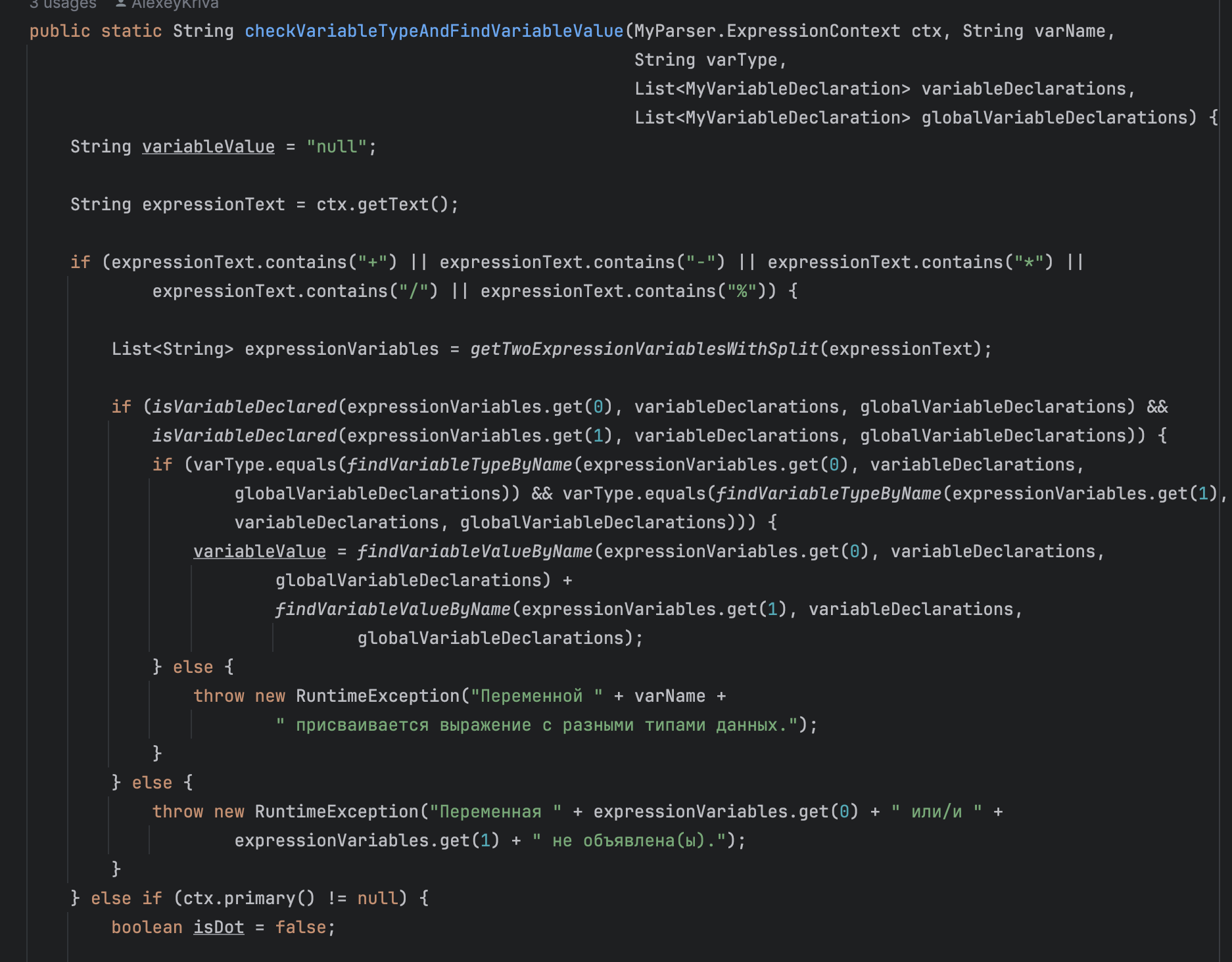


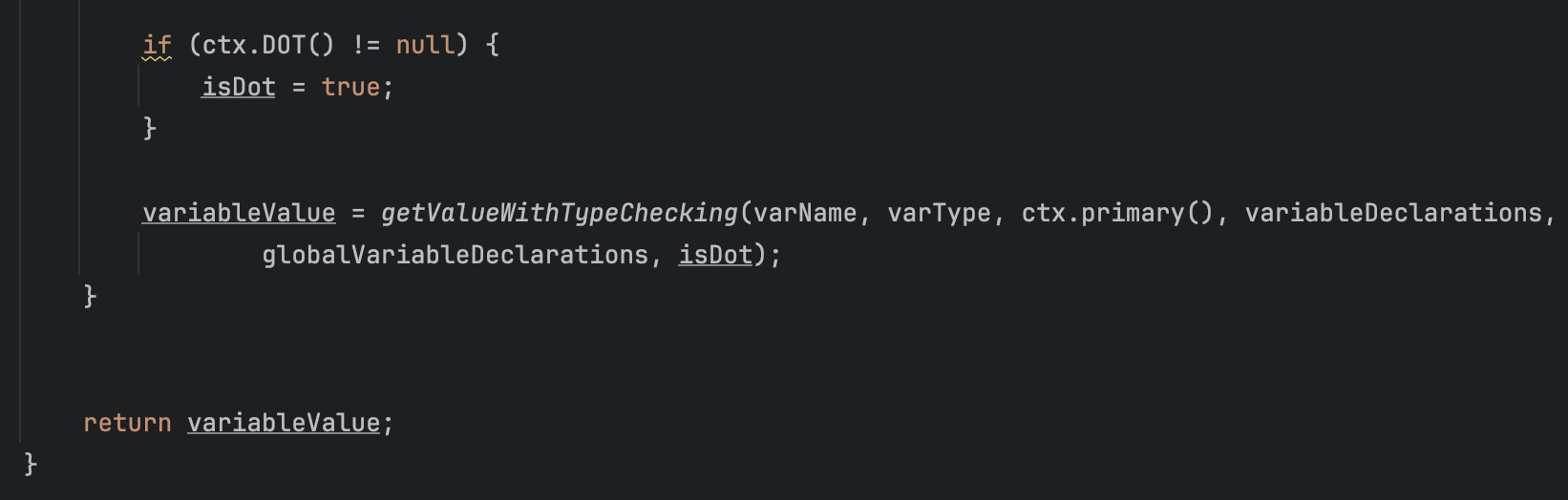




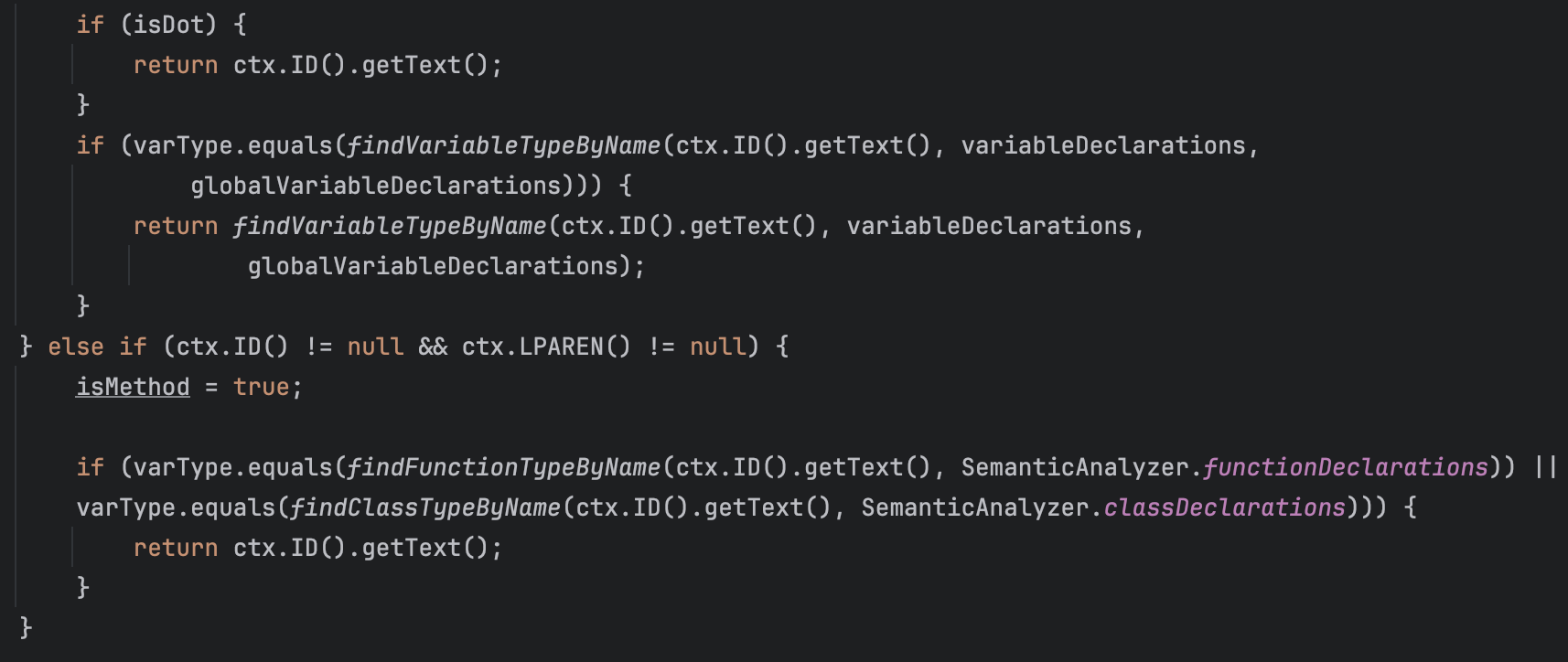






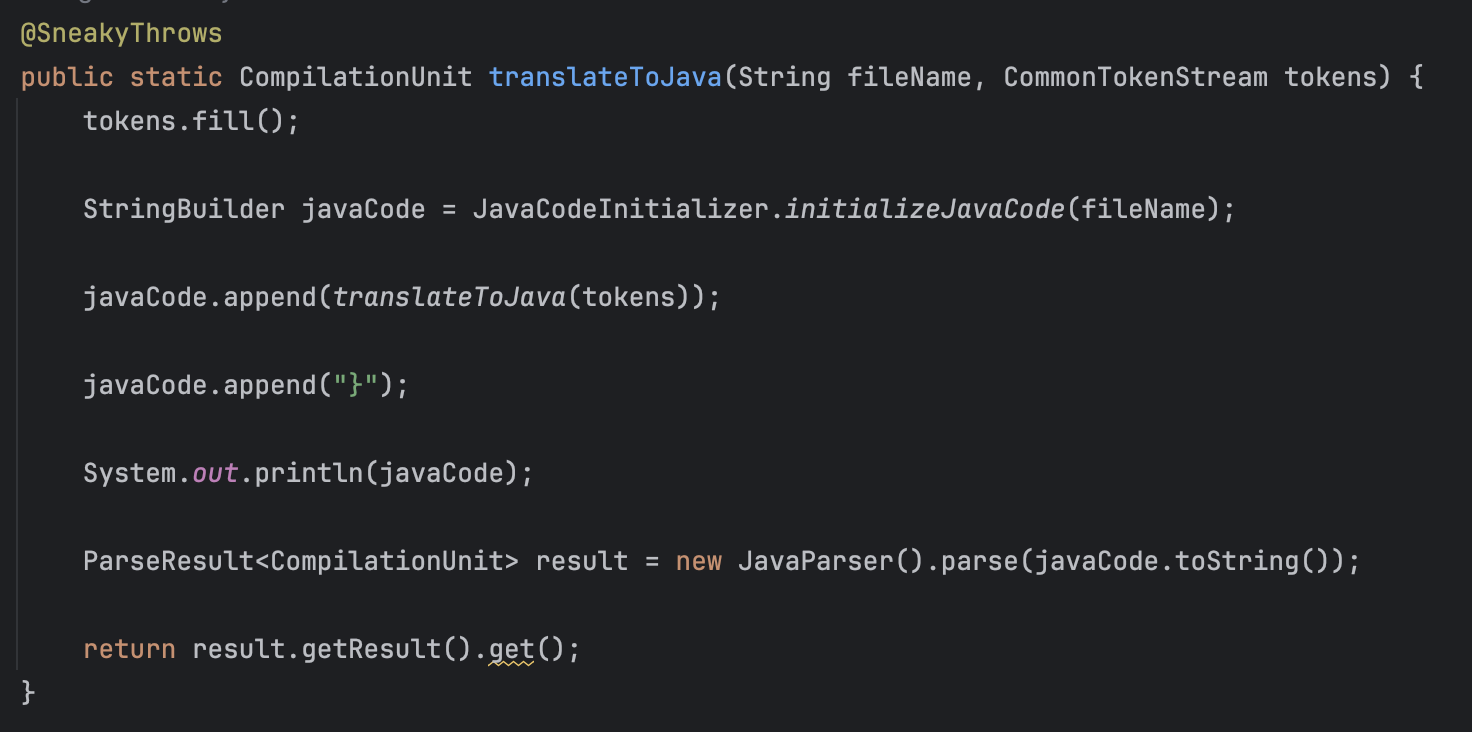






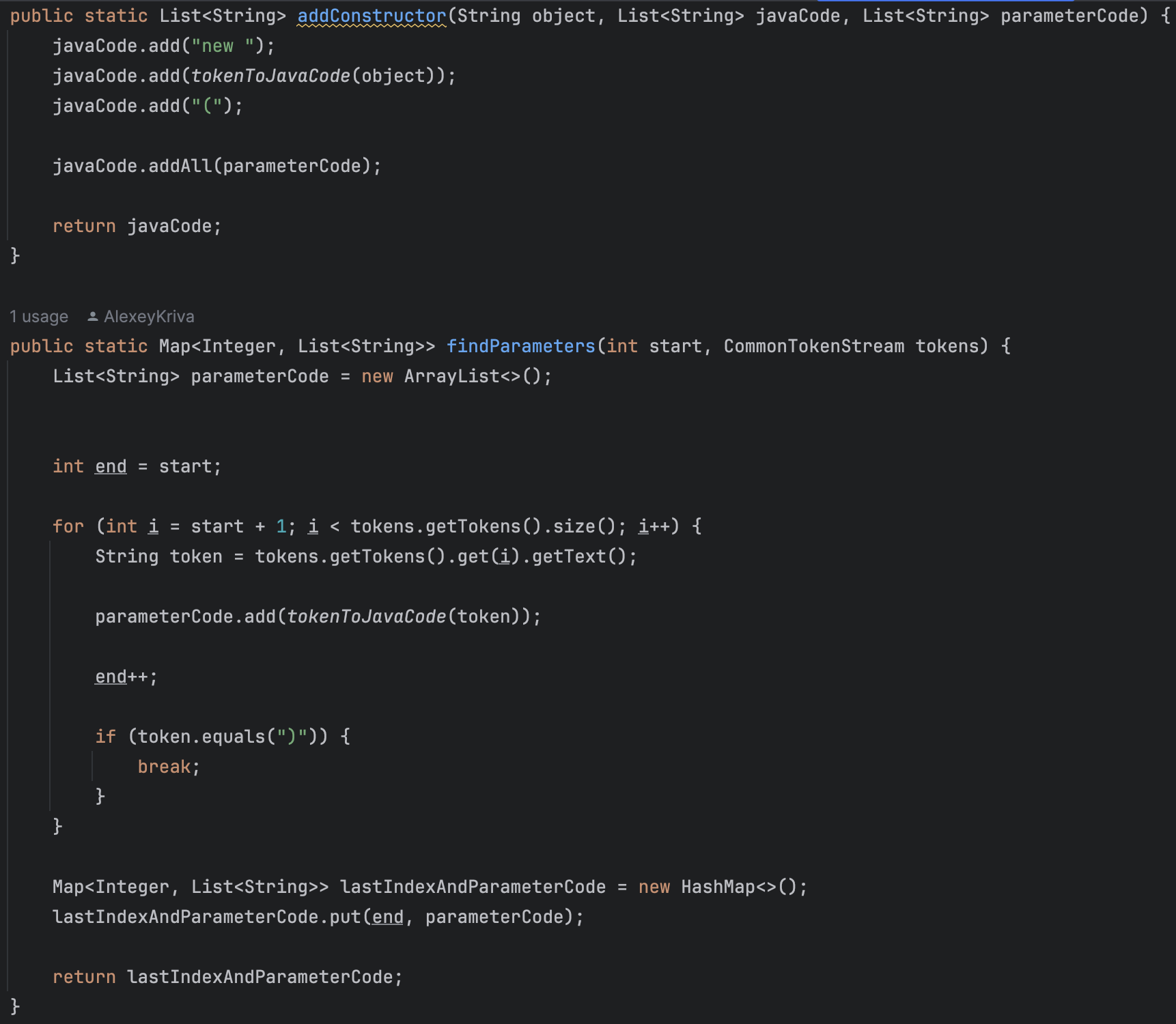
1. **Транслятор исходного кода на язык программирования Java:**

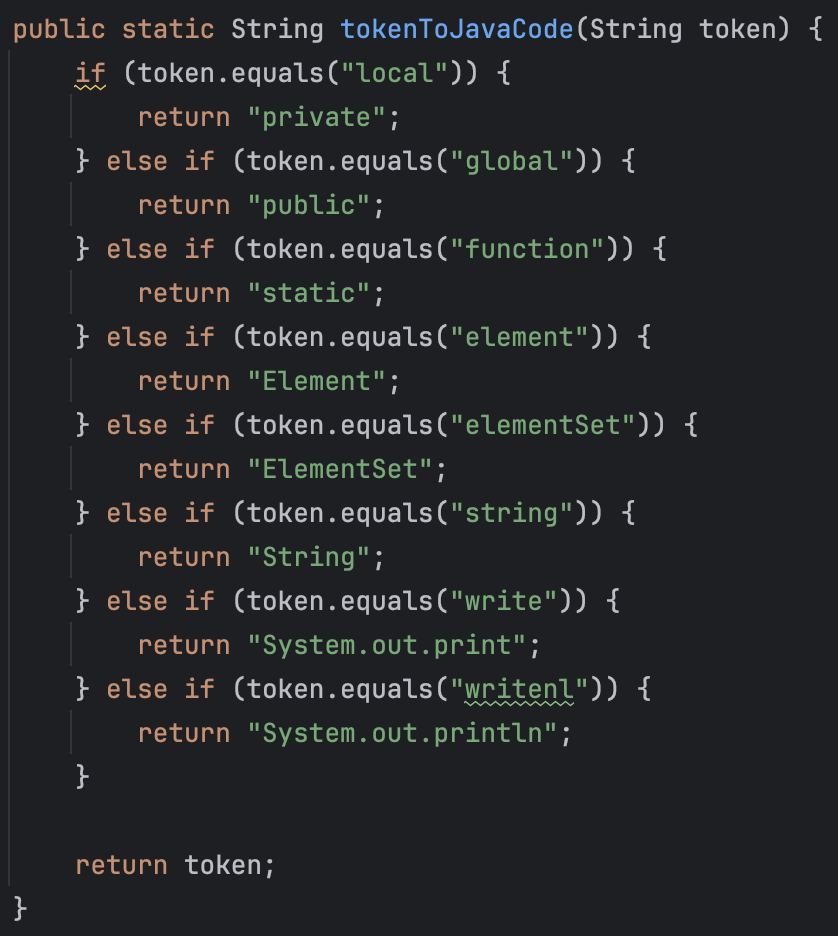
****

****

****

****

****

****

1. **Пример кода:**

**local function elementSet plus(elementSet elements) {**

**for (int i = 0; i < 20; i += 2) {**

**string strName1 = (string) i;**

**elements.addElement(element(i, strName1));**

**}**

**return elements;**

**}**

**local function elementSet minus(elementSet elements) {**

**for (int i = 30; i >= 0; i -= 3) {**

**string strName2 = (string) i;**

**elements.addElement(element(i, strName2));**

**}**

**return elements;**

**}**

**global function int sum(elementSet elements) {**

**int summa = 0;**

**for (int i = 0; i < elements.getSize(); i += 1) {**

**element current = elements.get(i);**

**summa += current.getValue();**

**}**

**return summa;**

**}**

**local function element max(elementSet elements) {**

**element maxElement = elements.get(0);**

**int i = 1;**

**while (i < elements.getSize()) {**

**int maxVal = maxElement.getValue();**

**element currentElementMax = elements.get(i);**

**int valMax = currentElementMax.getValue();**

**if (maxVal < valMax) {**

**maxElement = currentElementMax;**

**}**

**i += 1;**

**}**

**return maxElement;**

**}**

**local function element min(elementSet elements) {**

**element minElement = elements.get(0);**

**int j = 1;**

**while (j < elements.getSize()) {**

**int minVal = minElement.getValue();**

**element currentElementMin = elements.get(j);**

**int valMin = currentElementMin.getValue();**

**if (minVal > valMin) {**

**minElement = currentElementMin;**

**}**

**j += 1;**

**}**

**return minElement;**

**}**

**global function void main() {**

**elementSet elements = elementSet();**

**elements.addElement(element(10, "first"));**

**plus(elements);**

**minus(elements);**

**writenl("Elements of set:");**

**writenl(elements);**

**writenl("Total sum:");**

**int totalSum = sum(elements);**

**write(totalSum);**

**writenl();**

**writenl("Max element");**

**element maximum = max(elements);**

**write(maximum);**

**writenl();**

**writenl("Min element");**

**element minimum = min(elements);**

**write(minimum);**

**}**

1. **Вывод:**

В ходе данной лабораторной работы были получены знания и навыки разработки грамматики языка, лексеров и парсеров. Также были разработаны синтаксический и семантический анализаторы, которые помогают выявить ошибки в коде перед компиляцией или же транслированием. Также был разработан транслятор исходного языка на Java.