Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 11

по курсу «Языки и методы программирования»

«Разработка парсеров на языке Java»

Студент группы ИУ9-22Б Лавров Р. Д.

Преподаватель Посевин Д. П.

Москва 2025

1 Задание

В ходе лабораторной работы нужно разработать программу, выполняющую синтаксический анализ текста по одной из LL(1)-грамматик, БНФ которых приведены в таблицах 1–6. Текст может содержать символы перевода строки. В записи БНФ терминальные символы IDENT, NUMBER и STRING означают идентификаторы, числа и строки, соответственно. Идентификатор – это последовательность букв и цифр, начинающаяся с буквы. Число – это непустая последовательность десятичных цифр. Строка – это обрамлённая кавычками произвольная последовательность символов, не содержащая кавычек и символов перевода строки. Программа должна выводить в стандартный поток вывода последовательность правил грамматики, применение которых даёт левый вывод введённого из стандартного потока ввода текста. Если вывод не может быть построен, программа должна выводить сообщение «syntax error at (line, col)», где line и col – координаты ошибки в тексте.

Вариант 35:

35	<Parcel> ::= <Call> <Tail> <Tail>::= : <Parcel> | ε <Call>::= (IDENT <Arg>) <Arg>::= NUMBER | STRING | <Parcel>	(f 4):(e (g 6))	Лавров Р Д

2 Результаты

Исходный код 1– 2.

Листинг 1 — Класс Parser

```
1  public class Parser {
2      private char sym;
3      private String string;
4      private int pos;
5
6      public Parser(String string){
7          this.string = string;
8          this.pos = 0;
9          this.sym = string.charAt(0);
10     }
```

Листинг 2 — Класс Parser (продолжение 1)

```
    private void Next(){
        if (this.pos < string.length()-1){
            this.sym = string.charAt(++pos);
        }
    }

    // <Parcel> ::= <Call> <Tail>
    public boolean Parcel(){
        System.out.println("<Parcel> ::= <Call> <Tail>");
        boolean resp;
        SkipSpace();
        resp = Call();
        if (!resp) {
            return false;
        }
        return Tail();
    }

    // <Tail> ::= : <Parcel> | e
    public boolean Tail(){
        if (this.sym == ':') {
            System.out.println("<Tail>   ::= :<Parsel>");
            Next();
            return Parcel();
        }
        System.out.println("<Tail>   ::= e");
        return true;
    }

    // <Call>::= (IDENT <Arg>)
    private boolean Call(){
        boolean resp;
        if (this.sym == '(') {
            System.out.println("<Call>   ::= (IDENT <Arg>)");
            Next();
            resp = Ident();
            SkipSpace();
            resp = Arg();
            if (!resp || this.sym != ')') {
                return GiveError(this.pos);
            }
            Next();
            return true;
        }
        return GiveError(this.pos);
    }

    private boolean Ident(){
        String ident = "";
        while (Character.isDigit(this.sym) || Character.isLetter(this.sym)) {
            ident += this.sym;
            Next();
        }
        System.out.printf("IDENT = %s\n", ident);
        return true;
    }
```

Листинг 3 — Класс Parser (продолжение 2)

```
    // <Arg>::= NUMBER | STRING | <Parsel>
    private boolean Arg(){
        if (Character.isDigit(this.sym)){
            System.out.println("<Arg>    ::= NUMBER");
            return Numberr();
        }else if (Character.isLetter(this.sym)) {
            System.out.println("<Arg>    ::= STRING");
            return Stringg();
        }else if (this.sym == '('){
            System.out.println("<Arg>    ::= <Parsel>");
            return Parcel();
        }
        return GiveError(this.pos);
    }

    private boolean Numberr(){
        String ident = "";
        while (Character.isDigit(this.sym)) {
            ident += this.sym;
            Next();
        }
        System.out.printf("NUMBER = %s\n", ident);
        return true;
    }

    private boolean Stringg(){
        String ident = "";
        while (Character.isLetter(this.sym)) {
            ident += this.sym;
            Next();
        }
        System.out.printf("STRING = %s\n", ident);
        return true;
    }

    private void SkipSpace() {
        while (this.sym == ' '){
            Next();
        }
    }

    private boolean GiveError(int index){
        System.out.printf("Ошибка в позиции %d\n", index);
        return false;
    }

    public static void main(String[] args) {
        String string = "(f 4):(e (g 6))";
        Parser parser = new Parser(string);
        boolean resp = parser.Parcel();
    }
}
```

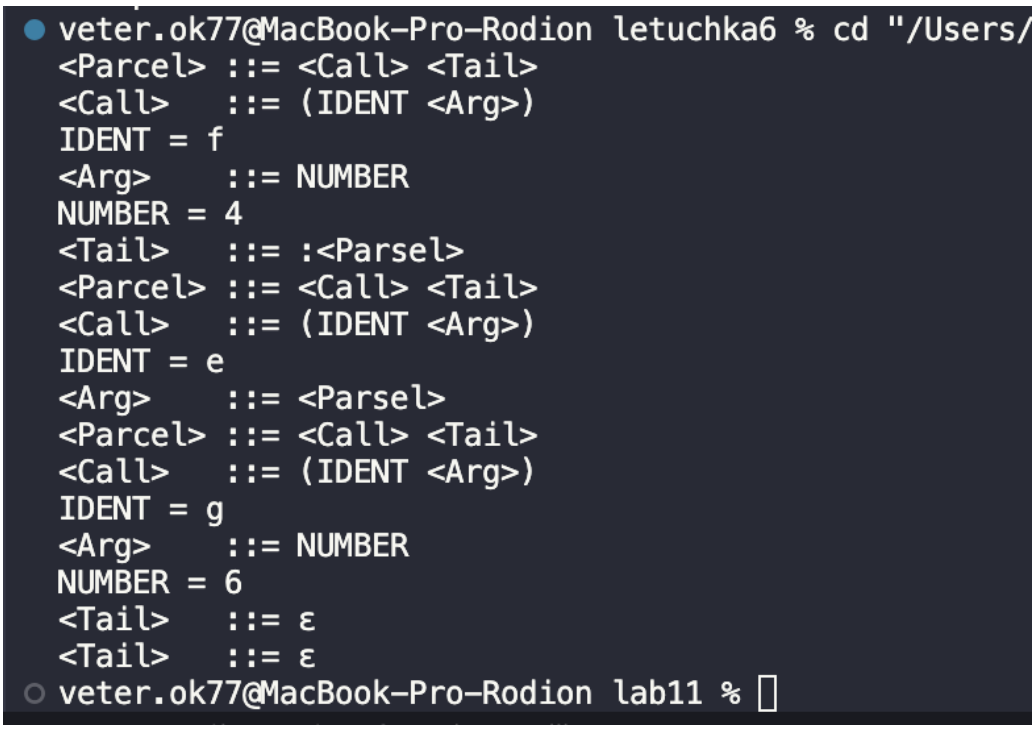

Рис. 1 — Результат работы

3 Вывод

Я смог написать парсер ещё на одном языке программирования (это уже третий)