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 ИУ9-21Б Горбунов А. Д.

Преподаватель Посевин Д. П.

Москва 2023

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.

<Decl>::= <Enum> <VarsOpt> ;

<VarsOpt>::= <Vars> | E

<Enum>::= enum IDENT <List>

<List>::= <Item> <LTail>

<LTail> ::= , <List> | E

<Item>::= IDENT <ITail>

<ITail> ::= = NUMBER | E

<Vars>::= IDENT <VTail>

<VTail> ::= , <Vars> | E

2 Результаты

Исходный код программы представлен в листинге 1, 2, 4

Листинг 1 — Test.java

```
1  public class Test {
2      public static void main(String[] args) {
3          System.out.println("\nCorrect stitch:");
4          String input = "enum Day
5          { SUN = 0 , MON , TUE , WED , THU , FRI , SAT } first , last ;";
6          Parser.parse(input);
7
8          System.out.println("\nError: skipped 0");
9          input = "enum Day
10         { SUN =  , MON , TUE , WED , THU , FRI , SAT } first , last ;";
11         Parser.parse(input);
12
13         System.out.println("\nError: skipped =");
14         input = "enum Day
15         { SUN  0 , MON , TUE , WED , THU , FRI , SAT } first , last ;";
16         Parser.parse(input);
17
18         System.out.println("\nError: skipped ,");
19         input = "enum Day
20         { SUN = 0 , MON  TUE , WED , THU , FRI , SAT } first , last ;";
21         Parser.parse(input);
22
23         System.out.println("\nError: skipped TUE");
24         input = "enum Day
25         { SUN = 0 , MON ,  , WED , THU , FRI , SAT } first , last ;";
26         Parser.parse(input);
27
28         System.out.println("\nError: skipped }");
29         input = "enum Day
30         { SUN = 0 , MON , TUE , WED , THU , FRI , SAT  first , last ;";
31         Parser.parse(input);
32
33         System.out.println("\nError: skipped {");
34         input = "enum Day
35         SUN = 0 , MON , TUE , WED , THU , FRI , SAT } first , last ;";
36         Parser.parse(input);
37
38         System.out.println("\nError: skipped ;");
39         input = "enum Day
40         { SUN = 0 , MON , TUE , WED , THU , FRI , SAT } first , last ";
41         Parser.parse(input);
42
43         System.out.println("\nError: skipped enum");
44         input = " Day
45         { SUN = 0 , MON , TUE , WED , THU , FRI , SAT  first , last ;";
46         Parser.parse(input);
47
48         System.out.println("\nError: skipped Day");
49         input = "enum
50         { SUN = 0 , MON , TUE , WED , THU , FRI , SAT  first , last ;";
51         Parser.parse(input);
52     }
53 }
```

Листинг 2 — класс Lexer.java

```
public class Lexer {
    public static ArrayList<String> tokenize(String input) {
        ArrayList<String> tokens = new ArrayList<String>();
        String[] words = input.trim().split("\\s+");
        for (String word : words) {
            if (!word.matches("enum") && (word.matches("[a-zA-Z]+") ||
   word.matches("[a-zA-Z]+[\\,]"))) {
                tokens.add("IDENT");
            } else if (word.matches("[0-9]+")) {
                tokens.add("NUMBER");
            } else {
                tokens.add(word);
            }
        }
        return tokens;
    }
}
```

Листинг 3 — класс Parser.java

```
import java.util.ArrayList;
public class Parser {
    private static int pos;
    private static ArrayList<String> tokens;
    public static void parse(String input) {
        tokens = Lexer.tokenize(input);
        pos = 0;
        try {
            decl();
            System.out.println("Parsing successful!");
        } catch (Exception e) {
            System.out.println("Parsing error: " + e.getMessage());
        }
    }
    public static void decl() throws Exception {
        enumDef();
        varsOpt();
        match(";");
    }
    public static void enumDef() throws Exception {
        match("enum");
        match("IDENT");
        match("{");
        list();
        match("}");
    }
    public static void list() throws Exception {
        item();
        ltail();
    }
    public static void ltail() throws Exception {
        if (tokens.get(pos).equals(",")) {
            match(",");
            list();
        }
    }
```

Листинг 4 — класс Parser.java(продолжение)

```
    public static void item() throws Exception {
        match("IDENT");
        itail();
    }
    public static void itail() throws Exception {
        if (tokens.get(pos).equals("=")) {
            match("=");
            match("NUMBER");
        }
    }
    public static void varsOpt() throws Exception {
        if (tokens.get(pos).startsWith("IDENT")) {
            vars();
        }
    }
    public static void vars() throws Exception {
        match("IDENT");
        vtail();
    }
    public static void vtail() throws Exception {
        if (tokens.get(pos).equals(",")) {
            match(",");
            vars();
        }
    }
    public static void match(String expected) throws Exception {
        if (pos < tokens.size() && tokens.get(pos).equals(expected)) {
            pos++;
        } else {
            throw new Exception("Expected " + expected + " but found " +
    tokens.get(pos) + " (Index:" + pos + ")");
        }
    }
}
```

Результат запуска представлен на рисунке 1, 2, 3

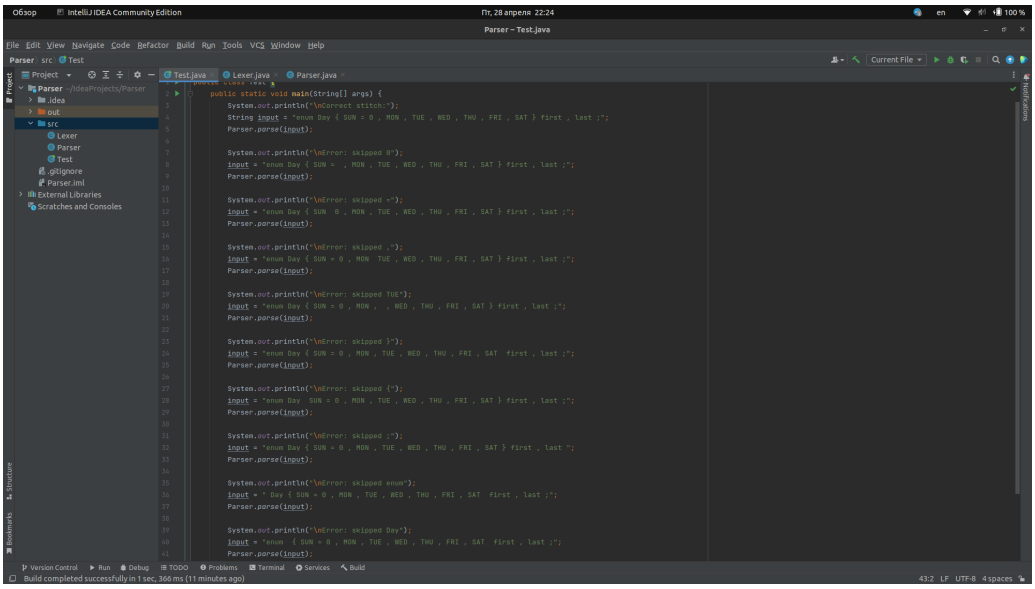

Рис. 1 — Реализация Test.java

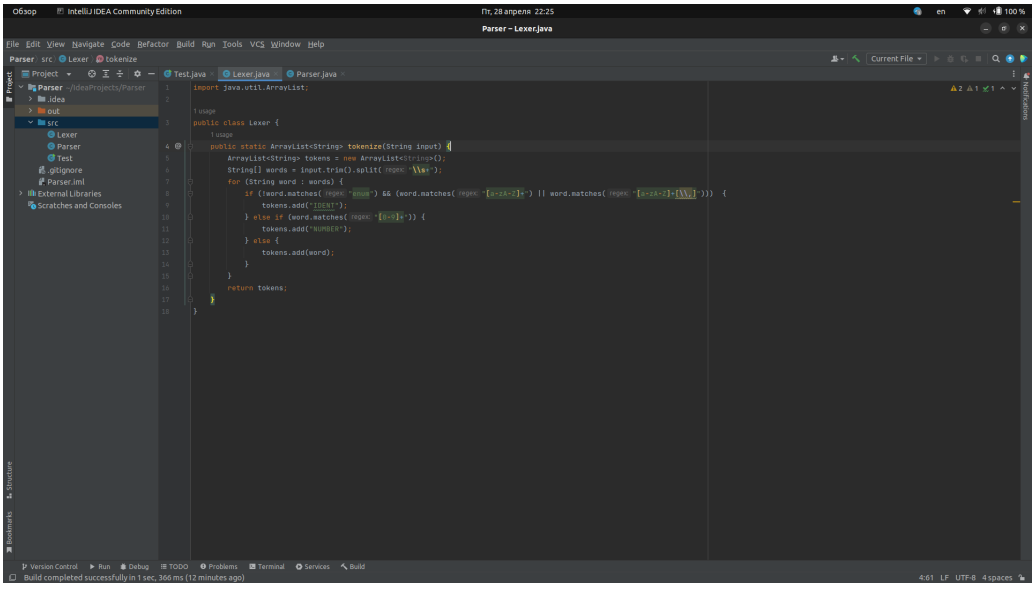

Рис. 2 — Реализация Leser.java

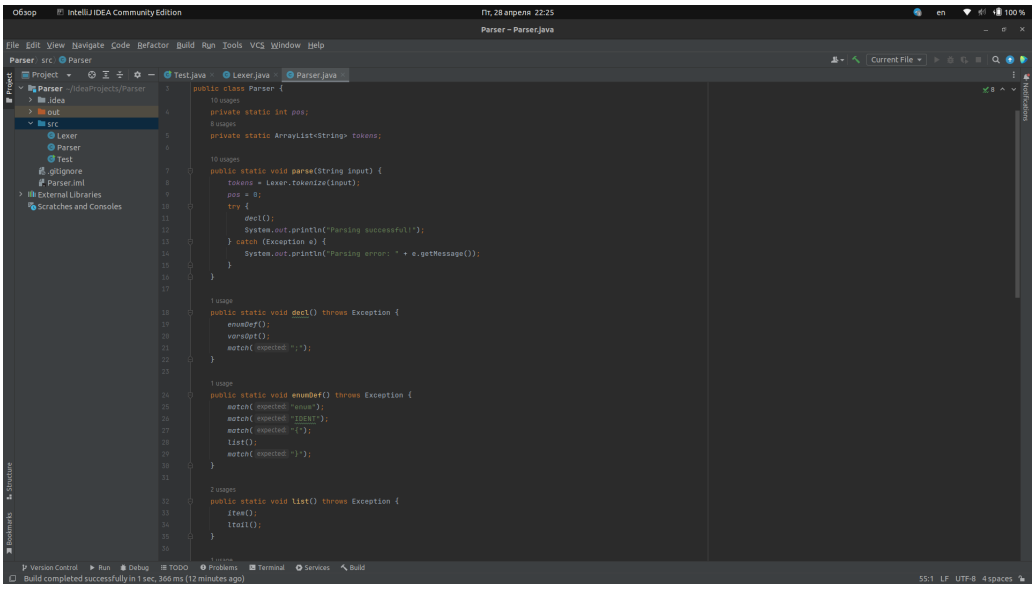

Рис. 3 — Реализация Parser.java

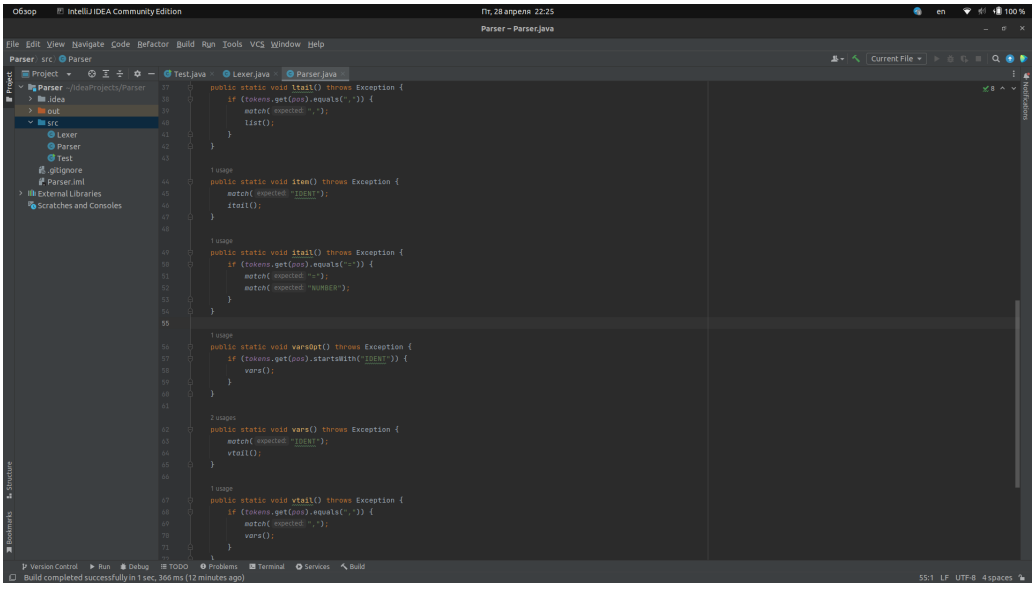

Рис. 4 — Реализация Parser(продолжение).java

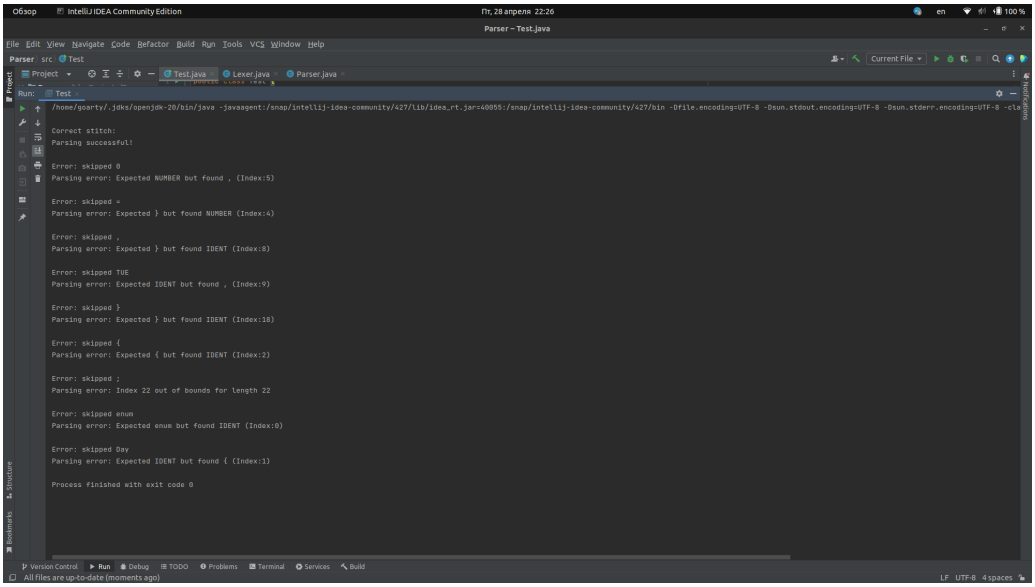

Рис. 5 — Работа программы