Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа № 3

По дисциплине ЯПИС

За шестой семестр

Тема: «Проектирование лексического анализатора  
с использованием регулярных выражений»

Вариант 7

Выполнил:

Студент 3 курса

Группы ИИ-16(2)

Пешко А.С.

Проверил:

Слинко Е.В.

Брест, 2021

**Цель работы:** изучение основных понятий теории регулярных грамматик, ознакомление с назначением и принципами работы лексических анализаторов (сканеров), получение практических навыков построения сканера на примере заданного простейшего входного языка.

**Ход работы**

**Задание:** Для выполнения лабораторной работы требуется написать программу, которая выполняет лексический анализ входного текста в соответствии с заданием и порождает таблицу лексем с указанием их типов и значений. Текст на входном языке задается в виде символьного (текстового) файла. Программа должна выдавать сообщения о наличие во входном тексте ошибок, которые могут быть обнаружены на этапе лексического анализа.

Длину идентификаторов и строковых констант считать ограниченной 32 символами. Программа должна допускать наличие комментариев неограниченной длины во входном файле. Форму организации комментариев предлагается выбрать самостоятельно.

Для организации лексического разбора следует использовать регулярные выражения.

**Вариант 7:** Входной язык содержит последовательность вызовов процедур, разделенных символом ; (точка с запятой). Вызов процедуры должен состоять из имени процедуры и списка параметров. В качестве параметров могут выступать идентификаторы, строковые константы, заключенные в двойные кавычки и одиночные символы, заключенные в одинарные кавычки.

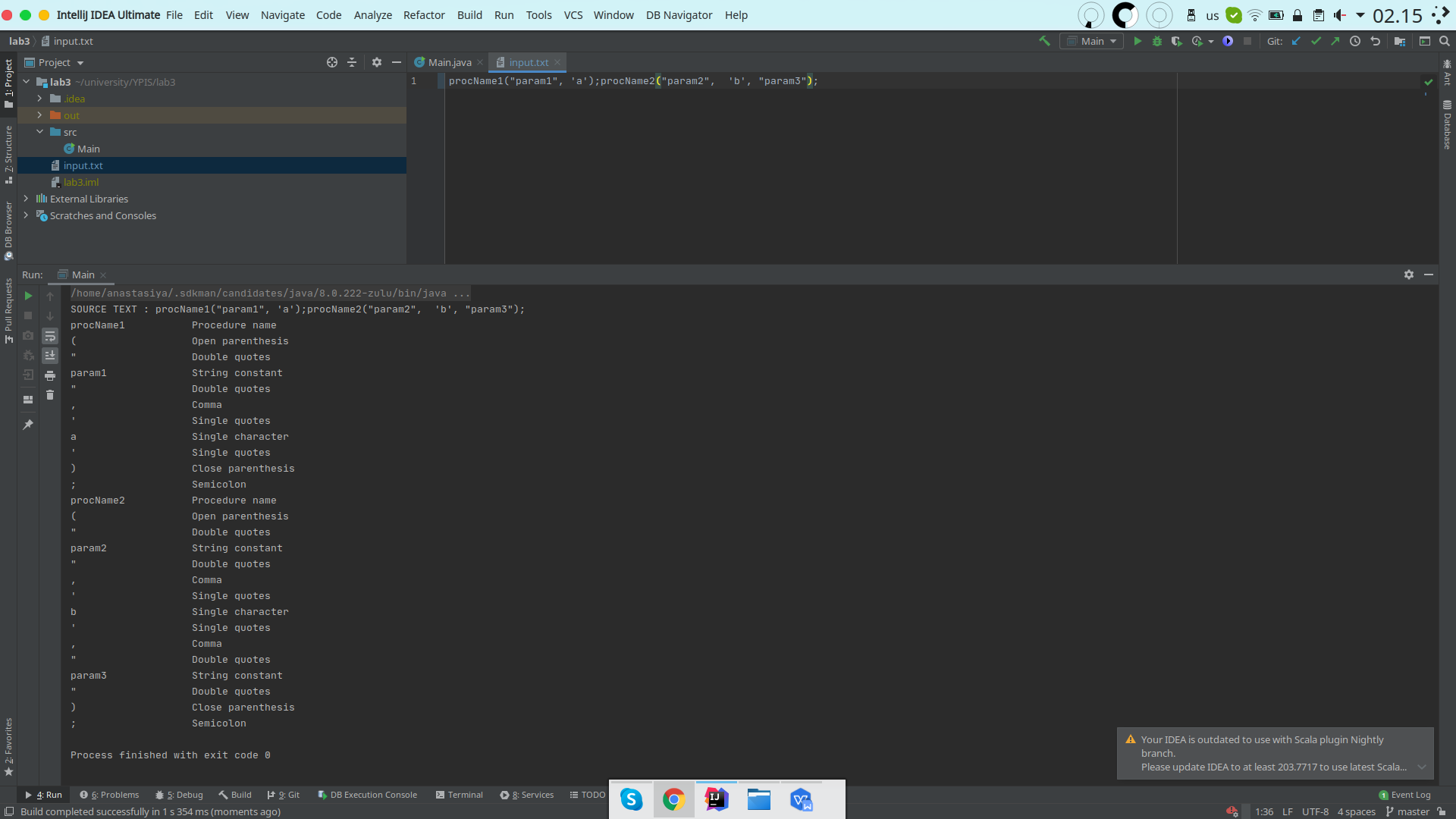
Проект с исходным кодом можно найти в репозитории по ссылке: **[https://github.com/anapiashko/university/tree/master/YPIS/lab3](https://github.com/anapiashko/university/tree/master/YPIS/lab1)**

**Код программы:**

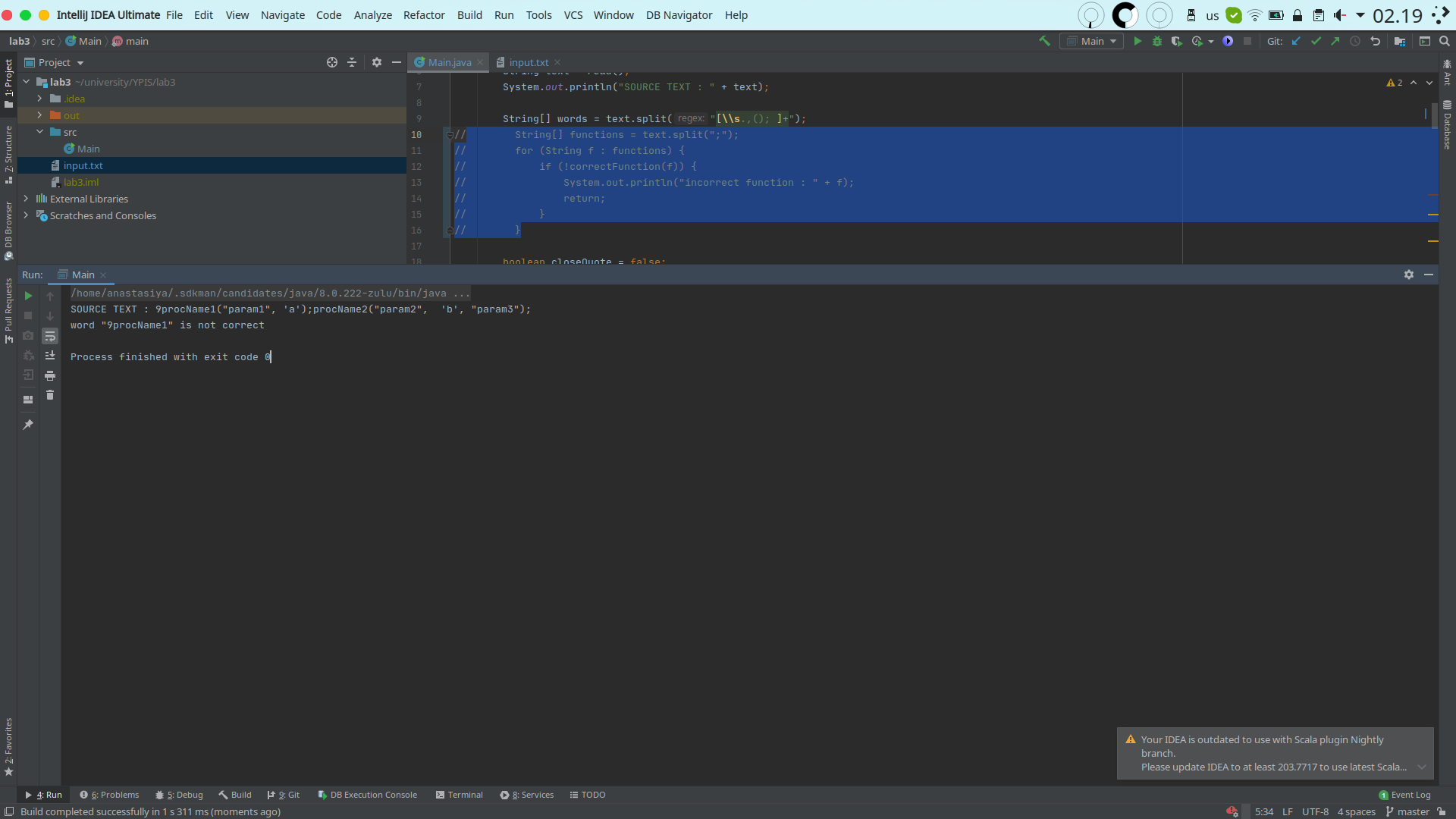
import java.io.\*;  
  
public class Main {  
 public static void main(String[] args) {  
  
 String text = read();  
 System.out.println("SOURCE TEXT : " + text);  
  
 String[] words = text.split("[.,(); ]+");  
 String[] functions = text.split(";");  
 for (String f : functions) {  
 if (!correctFunction(f)) {  
 System.out.println("incorrect function : " + f);  
 return;  
 }  
 }  
  
 boolean closeQuote = false;  
 int wordCount = 0;  
  
 for (int i = 0; i < text.length(); i++) {  
 char symbol = text.charAt(i);  
  
 // если слово начинается с буквы, то это имя процедуры, т.к. другие параметры заключены в ковычки  
 if (String.valueOf(symbol).matches("[a-zA-Z0-9\_]+")) {  
 if (correctWord(words[wordCount])) {  
 System.out.format("%-20s%s\n", words[wordCount], "Procedure name");  
 i += words[wordCount].length() - 1;  
 wordCount++;  
 } else {  
 System.out.println("word \"" + words[wordCount] + "\" is not correct");  
 return;  
 }  
 } else if (symbol == '(') System.out.format("%-20s%s\n", symbol, "Open parenthesis");  
 else if (symbol == ')') System.out.format("%-20s%s\n", symbol, "Close parenthesis");  
 else if (symbol == ';') System.out.format("%-20s%s\n", symbol, "Semicolon");  
 else if (symbol == ',') System.out.format("%-20s%s\n", symbol, "Comma");  
 else if (symbol == '\'') {  
 System.out.format("%-20s%s\n", symbol, "Single quotes");  
 if (!closeQuote) {  
 System.out.format("%-20s%s\n", text.charAt(++i), "Single character");  
 closeQuote = true;  
 wordCount++;  
 } else closeQuote = false;  
 } else if (symbol == '\"') {  
 System.out.format("%-20s%s\n", symbol, "Double quotes");  
 if (!closeQuote) {

String wordWithoutQuotes = words[wordCount].substring(1, words[wordCount].length() - 1);

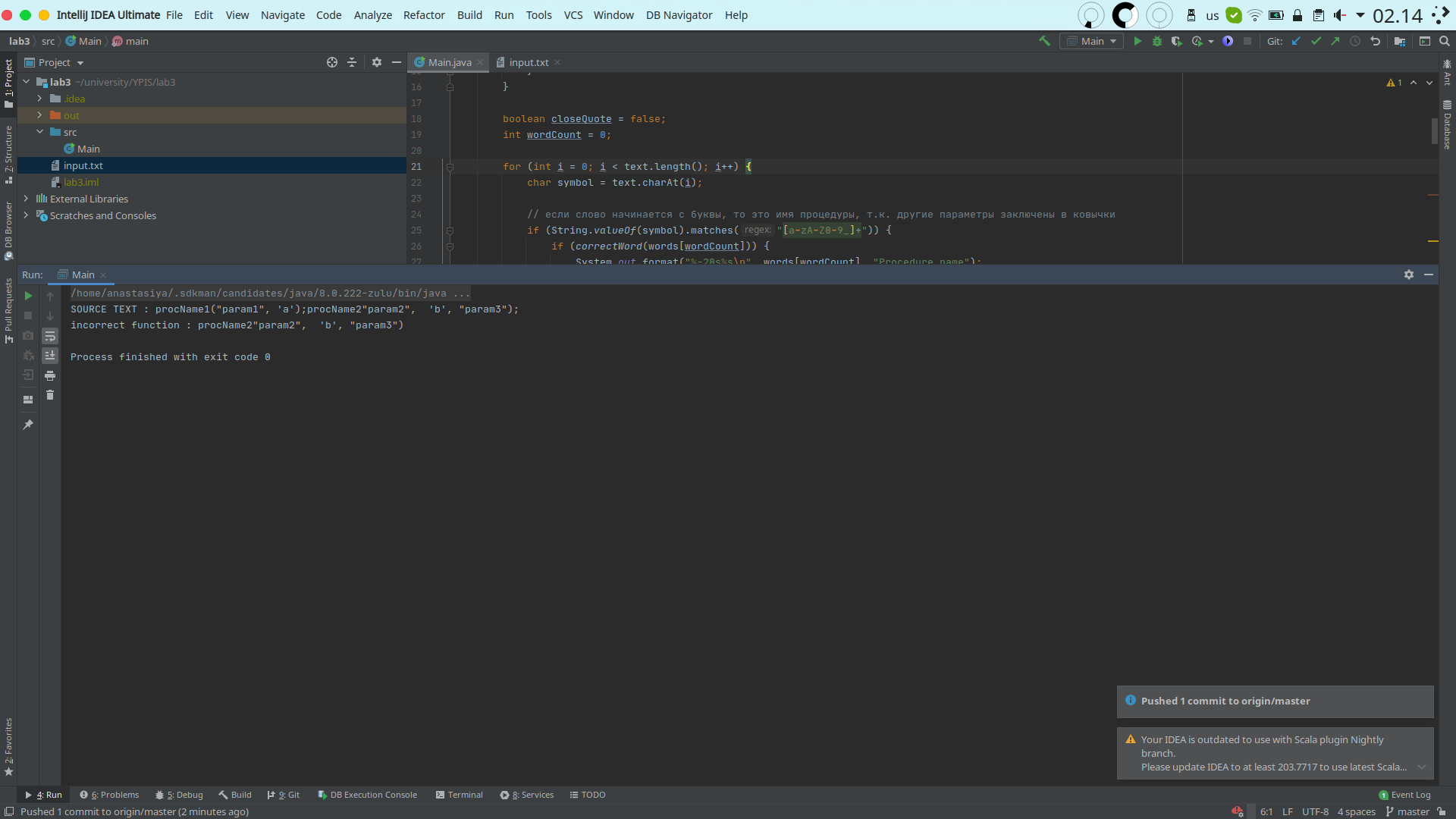
if (correctWord(wordWithoutQuotes )) {  
 System.out.format("%-20s%s\n", wordWithoutQuotes, "String constant");  
 closeQuote = true;  
 i += words[wordCount].length() - 2;  
 wordCount++;  
 } else {  
 System.out.println("word \"" + wordWithoutQuotes + "\" is not correct");  
 return;  
 }  
 } else {  
 closeQuote = false;  
 }  
 }  
 }  
 }  
  
 private static boolean correctWord(String word) {  
 String regex = "[a-zA-Z\_][a-zA-Z0-9]\*";  
 return word.matches(regex);  
 }  
  
 private static boolean correctFunction(String function) {  
 String regex = " \*([a-zA-Z\_][a-zA-Z0-9]\*) \*\\([a-zA-Z0-9 ,\"']\*\\) \*";  
 return function.matches(regex);  
 }  
  
 // построчное считывание файла  
 public static String read() {  
 String input = "";  
 try {  
 File file = new File("input.txt");  
 FileReader fr = new FileReader(file);  
 BufferedReader reader = new BufferedReader(fr);  
 input = reader.readLine();  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return input;  
 }  
}



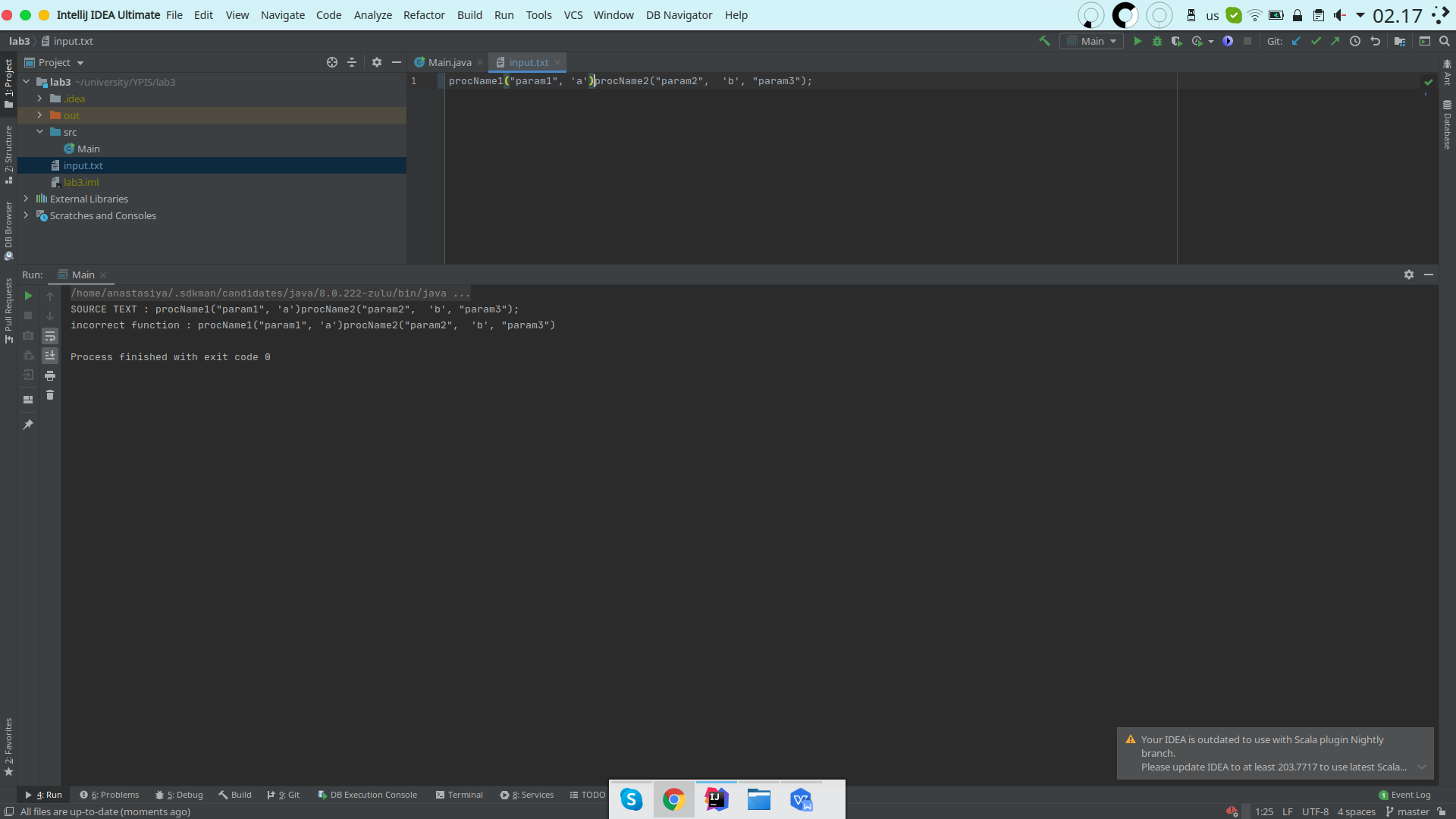
Лексема начинается с цифры



Пропущена открывающая скобка во второй функции



Пропущена точка с запятой между функциями



**Вывод**: изучили основные понятия теории регулярных грамматик, ознакомились с назначением и принципами работы лексических анализаторов (сканеров), получили практические навыки построения сканера на примере заданного простейшего входного языка.