**Министерство цифрового развития, связи и массовых коммуникаций**

**Российской Федерации Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский технический университет связи и информатики»**

Кафедра математической кибернетики и информационных технологий

Отчет по лабораторной работе №3

на тему «Методы поиска подстроки в строке»

по дисциплине «Структуры и алгоритмы обработки данных»

Выполнила: студентка группы БВТ1904

Кремер К.С.

# 

Москва 2021

**Цель работы**

Реализовать методы поиска подстроки в строке. Добавить возможность ввода строки и подстроки с клавиатуры. Предусмотреть возможность существования пробела. Реализовать возможность выбора опции чувствительности или нечувствительности к регистру. Оценить время работы каждого алгоритма поиска и сравнить его со временем работы стандартной функции поиска, используемой в выбранном языке программирования.

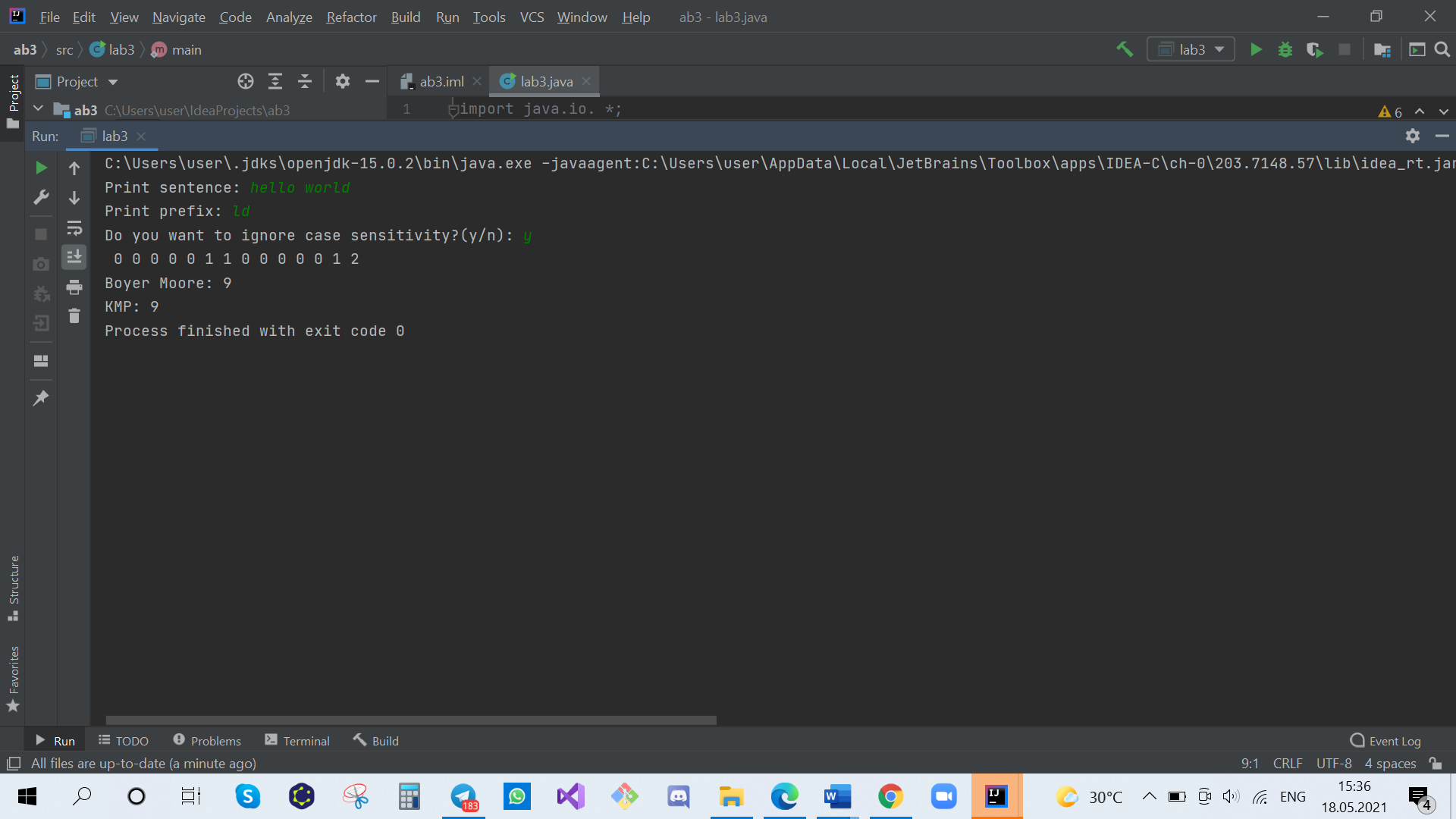
Алгоритмы:

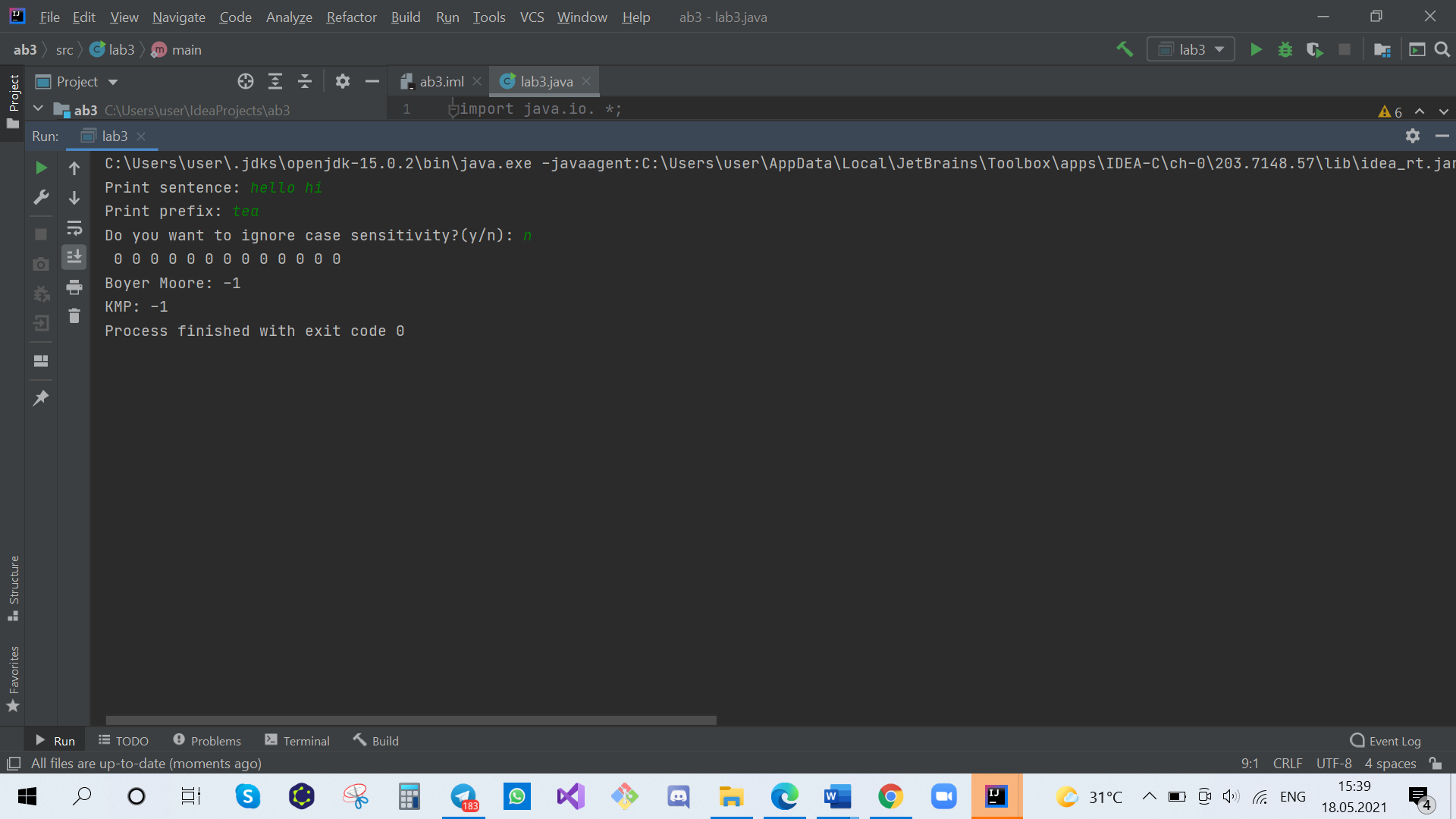
* Кнута-Морриса-Пратта
* Упрощенный Бойера-Мура

**Код программы**

import java.io. \*;  
import java.util.\*;  
import java.lang. \*;  
  
  
  
public class lab3 {  
 public static void main(String[] args){  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.print("Print sentence: ");  
 String line = in.nextLine();  
 System.*out*.print("Print prefix: ");  
 String pref = in.nextLine();  
 System.*out*.print("Do you want to ignore case sensitivity?(y/n): ");  
 String ignorance = in.nextLine();  
  
 if(ignorance.equals("y")){  
 line = line.toLowerCase();  
 pref = pref.toLowerCase();  
 }  
 int[] mas = *prefixFunction*(pref + "\*" + line);  
 for(int i = 0; i < mas.length; i++){  
 System.*out*.print(" " + mas[i]);  
 }  
 System.*out*.print("\nBoyer Moore: " + *Boyer\_Moore*(line, pref));  
 System.*out*.print("\nKMP: " + *KMP*(line, pref));  
 }  
  
 public static int[] prefixFunction(String s) {  
 int[] p = new int[s.length()];  
 int k = 0;  
 for (int i = 1; i < s.length(); i++) {  
 while (k > 0 && s.charAt(k) != s.charAt(i))  
 k = p[k - 1];  
 if (s.charAt(k) == s.charAt(i))  
 ++k;  
 p[i] = k;  
 }  
 return p;  
 }  
  
 public static String KMP(String s, String pref){  
 String line = pref + "\*" + s;  
 int[] p = *prefixFunction*(line);  
 String index = "";  
 for (int i = 0; i < p.length; i++){  
 if(p[i] == pref.length()){  
 index += (i - pref.length() \* 2) + " ";  
 }  
 }  
 if (index == "") index = "-1";  
 return index;  
 }  
  
 public static int Boyer\_Moore(String s, String pref){  
 int[] ascii = new int[256];  
 int[] px = new int[pref.length()];  
 int state = 0;  
  
 if(pref.length() > s.length()){  
 return -1;  
 }  
  
 for(int i = 0; i < ascii.length; i++){  
 ascii[i] = pref.length();  
 }  
  
 for(int i = pref.length() - 1; i >= 0; i--){  
 if(ascii[pref.charAt(i)] == pref.length()){  
 ascii[pref.charAt(i)] = state;  
 }  
 px[i] = ascii[pref.charAt(i)];  
 state++;  
 }  
  
 int i = pref.length() - 1;  
 int j = i;  
 int k = i;  
 while(j >= 0 && i <= s.length() - 1){  
 j = pref.length() - 1;  
 k = i;  
 while(j >= 0 && s.charAt(k) == pref.charAt(j)){  
 k--;  
 j--;  
 }  
 i += ascii[s.charAt(i)];  
 }  
 if(k >= s.length() - pref.length()){  
 return -1;  
 }  
 else{  
 return k + 1;  
 }  
 }  
  
}

**Результат работы программы**





**Вывод**

В ходе данной лабораторной работы я реализовала методы поиска подстроки в строке. В результате проделанной работы я сделала вывод, что метод Бойера-Мура занимает наименее количество времени.