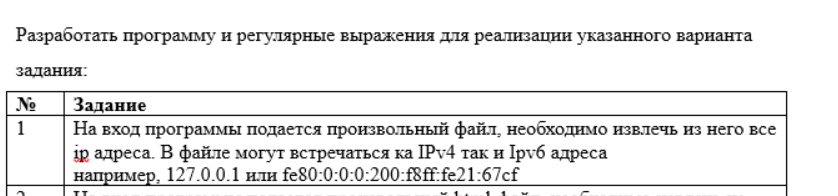
# Лабораторная работа № 4 студента группы ИТз-51 Черепченко Юрия Сергеевича Выполнение: \_\_\_\_\_\_\_\_\_\_\_ Защита: \_\_\_\_\_\_\_\_\_\_\_

# Регулярные выражения в Java.

**Цель работы:** изучение основ поиска строк при помощи регулярных выражений в Java.

**Вариант-1**

# Задание к лабораторной работе



# Ход работы

1. Разработать программу и регулярные выражения для реализации указанного варианта

Текст программы

ConsoleMain

package ru.bstu.itz.cherepchenko.lab1;  
  
  
import java.io.File;  
import java.io.IOException;  
import java.io.PrintWriter;  
import java.lang.reflect.Array;  
import java.time.LocalDate;  
import java.util.\*;  
  
public class ConsoleMain {  
  
 public String readFromFile(String path) {  
 try {  
 File file = new File(path);  
 if (!file.exists()) {  
 file.createNewFile();  
 PrintWriter printWriter = new PrintWriter(file);  
 printWriter.println("3 4 5");  
 printWriter.close();  
 }  
 Scanner scanner = new Scanner(file);  
 StringBuilder string = new StringBuilder();  
 while (scanner.hasNextLine()) {  
 string.append(scanner.nextLine() + "\n");  
 }  
 return string.toString();  
  
 } catch (IOException e) {  
 return "";  
 }  
 }  
  
 private void start() {  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.print("Введите путь к файлу: ");  
 String path = scanner.nextLine();  
  
 String fileText = readFromFile(path);  
 List<String> IPs = IpMatcher.*getAllIp*(fileText);  
  
 System.*out*.print("Найденные IP адреса: \n");  
 for (String ip : IPs)  
 System.*out*.print(ip + "\n");  
 }  
  
 public static void main(String[] args) {  
 new ConsoleMain().start();  
 }  
}

IpMatcher

package ru.bstu.itz.cherepchenko.lab1;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class IpMatcher {  
 public static List<String> getAllIp(String string) {  
 String IPV4\_REGEX = "((25[0-5]|2[0-4]\\d|[0-1]?\\d?\\d)(\\.(25[0-5]|2[0-4]\\d|[0-1]?\\d?\\d)){3})";  
 String IPV6\_HEX4DECCOMPRESSED\_REGEX = "(((?:[0-9A-Fa-f]{1,4}(?::[0-9A-Fa-f]{1,4})\*)?) ::((?:[0-9A-Fa-f]{1,4}:)\*)(25[0-5]|2[0-4]\\d|[0-1]?\\d?\\d)(\\.(25[0-5]|2[0-4]\\d|[0-1]?\\d?\\d)){3})";  
 String IPV6\_6HEX4DEC\_REGEX = "(((?:[0-9A-Fa-f]{1,4}:){6,6})(25[0-5]|2[0-4]\\d|[0-1]?\\d?\\d)(\\.(25[0-5]|2[0-4]\\d|[0-1]?\\d?\\d)){3})";  
 String IPV6\_HEXCOMPRESSED\_REGEX = "(((?:[0-9A-Fa-f]{1,4}(?::[0-9A-Fa-f]{1,4})\*)?)::((?:[0-9A-Fa-f]{1,4}(?::[0-9A-Fa-f]{1,4})\*)?))";  
 String IPV6\_REGEX = "((?:[0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4})";  
  
 String IPADDRESS\_PATTERN =  
 IPV4\_REGEX + "|" +  
 IPV6\_HEX4DECCOMPRESSED\_REGEX + "|" +  
 IPV6\_6HEX4DEC\_REGEX + "|" +  
 IPV6\_HEXCOMPRESSED\_REGEX + "|" +  
 IPV6\_REGEX;  
  
 Pattern pattern = Pattern.*compile*(IPADDRESS\_PATTERN);  
 Matcher matcher = pattern.matcher(string);  
  
 List<String> matches = new ArrayList<String>();  
 while (matcher.find()) {  
 matches.add(matcher.group(0));  
 }  
 return matches;  
 }  
}

Вывод: изучил основы поиска строк при помощи регулярных выражений в Java.