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 **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.04.01/07 **Интеллектуальные системы анализа, обработки и интерпретации больших данных**

О Т Ч Е Т

по лабораторной работе №6

Название: Коллекции

Дисциплина: Языка программирования для работы с большими данными

Студент	ИУ6-23М		Д.Д. Капитонов
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			П.В. Степанов
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2022

Вариант 1, номера 9 и 1.

9. Задан файл с текстом на английском языке. Выделить все различные слова. Слова, отличающиеся только регистром букв, считать одинаковыми. Использовать класс HashSet.

1. Определить множество на основе множества целых чисел. Создать методы для определения пересечения и объединения множеств.

Код программы:

```
package dan.lab;

//1.	Определить множество на основе множества целых чисел.
//	Создать методы для определения пересечения и объединения множеств.

import java.io.*;
import java.util.*;


public class Lab_9_1_1 {

   public static boolean check_in(ArrayList<Integer> m1, ArrayList<Integer> m2){
      return m1.get(m1.size()-1) >= m2.get(0) || m1.get(0) <= m2.get(m2.size()-1);
   }

   public static ArrayList<Integer> concat(ArrayList<Integer> m1, ArrayList<Integer> m2){
      System.out.println("Вариант 1 №1");
      ArrayList<Integer> ret = new ArrayList<>();
      int max= Math.max(m1.size(), m2.size());
      System.out.println(max);
      for (int i =0; i!= max; i++){
         if (m1.size()  > i && m2.size() > i){
            if (m1.get(i) == m2.get(i)){
               ret.add(m1.get(i));
            }  else{
               ret.add(m1.get(i));
               ret.add(m2.get(i));
            }
         } else{
            if (m1.size()  > i){
               ret.add(m1.get(i));
            } else {
               ret.add(m2.get(i));
            }
         }
      }
      return ret;
   }

   public static void main(String[] args) throws IOException {
      ArrayList<ArrayList<Integer>> mn_vo = new ArrayList<>();
```

```
        for (int j = 0; j!=6; j++){
            ArrayList<Integer> cash = new ArrayList<>();
            int i_1 = (int) (Math.random() * 10 + 1);
            for (int i =0; i!=i_1; i++){
                cash.add(3*(i_1 - i));
            }
            Collections.sort(cash);
            mn_vo.add(cash);
        }

        System.out.println(mn_vo);

        System.out.println(check_in(mn_vo.get(1), mn_vo.get(2)));
        System.out.println(concat(mn_vo.get(1), mn_vo.get(2)));

        System.out.println("Вариант 1 №9");

        File file = new File("eng_test.txt");
        BufferedReader br = new BufferedReader(new FileReader(file));
        String line = "";
        HashSet<String> str = new HashSet<>();
        while ((line = br.readLine())!= null){
            for (String word:line.split(" ")){
                if (!str.contains(word.toLowerCase())){
                    str.add(word.replace(".", "")
                            .replace(",","")
                            .replace(";", "").toLowerCase());
                }
            }
        }
        System.out.println(str);



    }
}
```

Результат выполнения программы:

```
C:\Users\dan-1\.jdks\corretto-1.8.0_322\bin\java.exe ...
[[3, 6], [3], [3, 6, 9, 12, 15, 18, 21, 24, 27, 30], [3, 6, 9], [3], [3, 6, 9]]
true
Вариант 1 №1
10
[3, 6, 9, 12, 15, 18, 21, 24, 27, 30]
Вариант 1 №9
[sigh, traveler, equally, about, stood, had, fair, diverged, down, somewhere, that, leaves, far, should, claim,

Process finished with exit code 0
```

Вариант 2, номера 9 и 10.

9. Дана матрица из целых чисел. Найти в ней прямоугольную подматрицу, состоящую из максимального количества одинаковых элементов. Использовать класс Stack.

10. На прямой гоночной трассе стоит N автомобилей, для каждого из которых известны начальное положение и скорость. Определить, сколько произойдет обгонов.

Код программы:

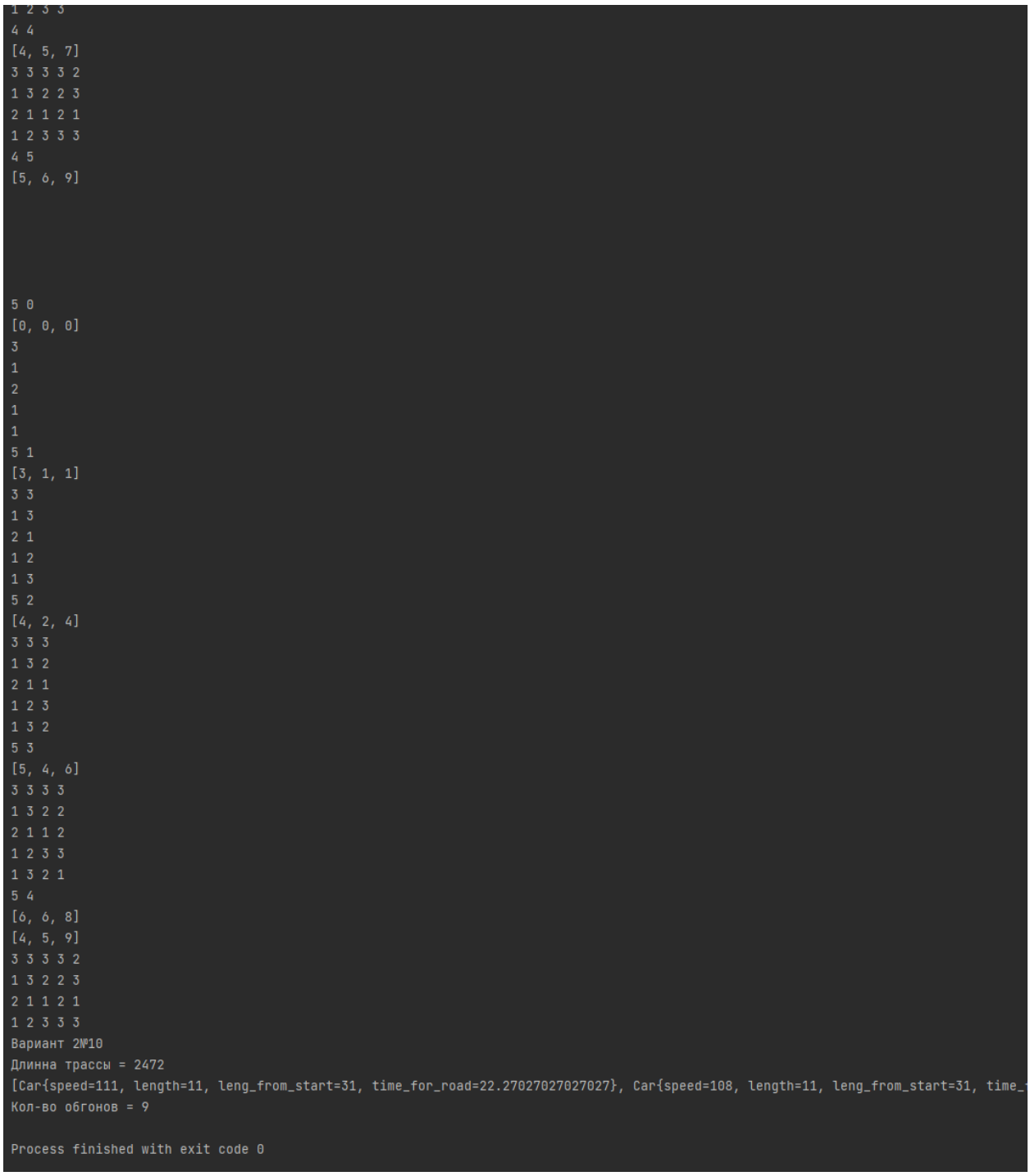
```
package dan.lab;
//
//9.	Дана матрица из целых чисел. Найти в ней прямоугольную подматрицу,
//	состоящую из максимального количества одинаковых элементов. Использовать класс Stack.
//
//	10.	На прямой гоночной трассе стоит N автомобилей, для каждого из которых
//	известны начальное положение и скорость. Определить, сколько произойдет обгонов.
//

import com.sun.org.apache.xalan.internal.xsltc.trax.SmartTransformerFactoryImpl;

import java.util.*;

public class Lab9_10_2 {

    public static void main(String[] args) {
        System.out.println("Вариант 2№9");
        int n = 5;
        HashSet<Integer> num_list = new HashSet<>();
        int[][] mass = new int[n][n];
        Stack<Integer> matr_stack = new Stack<>();
        for (int i = 0; i != n; i++) {
            for (int j = 0; j != n; j++) {
                mass[i][j] = (int) (Math.random() * 3 + 1);
                System.out.print(mass[i][j] + " ");
                matr_stack.push(mass[i][j]);
                num_list.add(mass[i][j]);
            }
            System.out.println("");
        }
        System.out.println("Num_list = " + num_list);
        System.out.println("Matr_stack = " + matr_stack);
```

```
        System.out.println(matr_stack.search(3));
        int[] max_p_i_j_c = new int[3];
        max_p_i_j_c[0] = 0;
        max_p_i_j_c[1] = 0;
        max_p_i_j_c[2] = 0;
        for (int num_add_i = 0; num_add_i != n + 1; num_add_i++) {
            for (int num_add_j = 0; num_add_j != n + 1; num_add_j++) {
                int[] mass_numbers_c = new int[3];
                if (num_add_i != n || num_add_j != n) {
                    for (int i = 0; i != num_add_i; i++) {
                        for (int j = 0; j != num_add_j; j++) {
                            switch (mass[i][j]) {
                                case 1:
                                    mass_numbers_c[0]++;
                                    break;
                                case 2:
                                    mass_numbers_c[1]++;
                                    break;
                                case 3:
                                    mass_numbers_c[2]++;
                                    break;
                            }

                            System.out.print(mass[i][j] + " ");
                        }
                        System.out.println("");
                    }
                    System.out.println(num_add_i + " " + num_add_j);
                    System.out.println(Arrays.toString(mass_numbers_c));
                    int max = Math.max(Math.max(mass_numbers_c[0],
mass_numbers_c[1]), Math.max(mass_numbers_c[1], mass_numbers_c[2]));
                    if (max > max_p_i_j_c[2]) {
                        max_p_i_j_c[2] = max;
                        max_p_i_j_c[0] = num_add_i;
                        max_p_i_j_c[1] = num_add_j;
                    }
                }
            }

        }
        System.out.println(Arrays.toString(max_p_i_j_c));

        for (int i = 0; i != max_p_i_j_c[0]; i++) {
            for (int j = 0; j != max_p_i_j_c[1]; j++) {
                System.out.print(mass[i][j] + " ");
```

```
            }
            System.out.println("");
        }
        System.out.println("Вариант 2№10");
        int road_lenght =  (int)(Math.random()*2000+1000);
        System.out.println("Длинна трассы = "+ road_lenght);
        ArrayList<Car> car_mass = new ArrayList<>();
        for (int i=0; i!=n; i++){
            car_mass.add(new
Car((int)(Math.random()*16+100),(int)(Math.random()*2+10),(int)(Math.ran
dom()*2+20), road_lenght));
        }
        System.out.println(car_mass);
        int count =0;
        for (int i=0; i!=n; i++){
            for (int j=0; j!=n; j++){
                if (car_mass.get(i).time_for_road > car_mass.get(j).time_for_road){
                    count++;
                }
            }
        }
        System.out.println("Кол-во обгонов = "+ count);



    }

    public static class Car{
        int speed;
        int length;
        int leng_from_start;
        double time_for_road;

        public Car(int speed, int length, int leng_from_start, int road_lenght) {
            this.speed = speed;
            this.length = length;
            this.leng_from_start = length + leng_from_start;
            this.time_for_road = time_count(this.speed, road_lenght);

        }

        public double time_count(int speed, int road_lenght){
            return (double) road_lenght/ (double)speed;
        }
```

```
    @Override
    public String toString() {
       return "Car{" +
             "speed=" + speed +
             ", length=" + length +
             ", leng_from_start=" + leng_from_start +
             ", time_for_road=" + time_for_road +
             '}';
    }
  }


}
```

Результат выполнения программы:



Вывод: научились работать с коллекциями.