Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №3

По дисциплине: «СПП»

Выполнил:

студент 3 курса

группы ПО-3

Горбун В.О.

Проверил:

Крощенко А.А.

2020

**Цель работы:** научиться создавать и использовать классы в программах на языке программирования Java.

**Вариант 7**

**Задание 1:** реализовать простой класс.

**Множество символов ограниченной мощности** – Предусмотреть возможность объединения двух множеств, вывода на печать элементов множества, а также метод, определяющий, принадлежит ли указанное значение множеству. Класс должен содержать методы, позволяющие добавлять и удалять элемент в/из множества. Конструктор должен позволить создавать объекты с начальной инициализацией. Мощность множества задается при создании объекта. **Реализацию множества осуществить на базе одномерного массива**. Реализовать метод equals, выполняющий сравнение объектов данного типа.

**Задание 2:** разработать автоматизированную систему на основе некоторой структуры данных, манипулирующей объектами пользовательского класса.

**Система оповещений на дорожном вокзале**

Автоматизированная информационная система на железнодорожном вокзале содержит сведения об отправлении поездов дальнего следования.

Составить программу, которая должна хранить расписание поездов в структурированном, отсортированном по времени отправления виде (используя бинарное дерево).

• Обеспечивает первоначальный ввод данных в информационную систему о текущем расписании из файла и формирование дерева;

• Печатает все расписание на экран по команде;

• Выводит информацию о поезде по номеру поезда;

• По названию станции назначения выводит данные обо всех поездах, которые следуют до этой станции;

• Список поездов, следующих до заданного пункта назначения и отправляющихся после заданного часа;

• Список поездов, отправляющихся до заданного пункта назначения и имеющих общие места;

• За 10, 5, 3 минуты до отправления поезда показывает информационное сообщение об отправлении поезда.

**Ход работы:**

**Задание 1:**

**Текст программы:**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Mnojestvo mnoj1 = new Mnojestvo();

System.out.println(mnoj1);

Mnojestvo mnoj2 = new Mnojestvo();

System.out.println(mnoj2);

mnoj1.combine(mnoj2);

System.out.println(mnoj1);

mnoj1.isElem("t");

mnoj1.addition();

System.out.println(mnoj1);

mnoj1.deletion();

System.out.println(mnoj1);

Mnojestvo mnoj3 = new Mnojestvo();

System.out.println(mnoj3);

mnoj1.equals(mnoj3);

}

}

class Mnojestvo {

int n; //мощность

String[] M;

Mnojestvo () {

System.out.println("Vvedite moschnost mnojestva:");

Scanner in = new Scanner(System.in); //считываем из консоли

n = in.nextInt();

in.nextLine();

if (n > 0) {

String[] Mn = new String[n];

System.out.println("Vvedite elementi (cherez probel):");

for (int i = 0; i < n; i++) {

Mn[i] = in.next();

}

M = Mn;

}

}

public void combine(Mnojestvo M2) { //объединение множеств

String[] Mn = new String[this.n + M2.n];

int i;

for (i = 0; i < this.n; i++) {

Mn[i] = this.M[i];

}

for (; i < this.n + M2.n; i++) {

Mn[i] = M2.M[i-this.n];

}

this.n = this.n + M2.n;

this.M = Mn;

}

public void isElem(String e) { //поиск элемента в множестве

boolean elem = false;

System.out.println("Poisk elementa " + e + " v mnojestve");

for(int i = 0; i < this.n; i++) if (this.M[i].equals(e)) elem = true;

if (elem == true) System.out.println("Element " + e + " prinadlejit mnojestvu");

else System.out.println("Element " + e + " ne prinadlejit mnojestvu");

}

public void addition() {

System.out.println("Vvedite poziciu");

Scanner in = new Scanner(System.in); //считываем из консоли

int pos = in.nextInt();

in.nextLine();

System.out.println("Vvedite element");

String elem = in.next();

pos--;

if (pos < 0) pos = 0;

String[] Mn = new String[this.n + 1];

for (int i = 0; i < this.n; i++) {

if (i == pos) {

Mn[i+1] = M[i];

Mn[i] = elem;

}

if (i < pos) Mn[i] = this.M[i];

if (i > pos) Mn[i + 1] = this.M[i];

}

if (pos >= this.n) Mn[this.n] = elem;

this.n = this.n + 1;

this.M = Mn;

}

public void deletion() {

Scanner in = new Scanner(System.in); //считываем из консоли

System.out.println("Vvedite element na udalenie");

String elem = in.next();

int pos=-1;

for(int i = 0; i < this.n; i++) if (this.M[i].equals(elem)) pos = i;

if (pos >= 0) {

String[] Mn = new String[this.n -1];

for (int i = 0; i < this.n -1; i++) {

if (i == pos) {

Mn[i] = M[i+1];

}

if (i < pos) Mn[i] = this.M[i];

if (i > pos) Mn[i] = this.M[i + 1];

}

this.n = this.n - 1;

this.M = Mn;

}

else System.out.println("V mnojestve netu takogo elementa");

}

boolean equals(Mnojestvo Mn) {

boolean res = true;

if (this.n != Mn.n) res = false;

if (res == true) {

for (int i = 0; i < this.n; i++) //если Mn.n больше, то и смысла в этом нету

if (this.M[i].equals(Mn.M[i]) == false) res = false;

}

if (res == true) System.out.println("Mnojestva ravni");

else System.out.println("Mnojestva ne ravni");

return res;

}

@Override

public String toString() {

String str = "Elementi mnojestva: ";

for (int i = 0; i < this.n; i++)

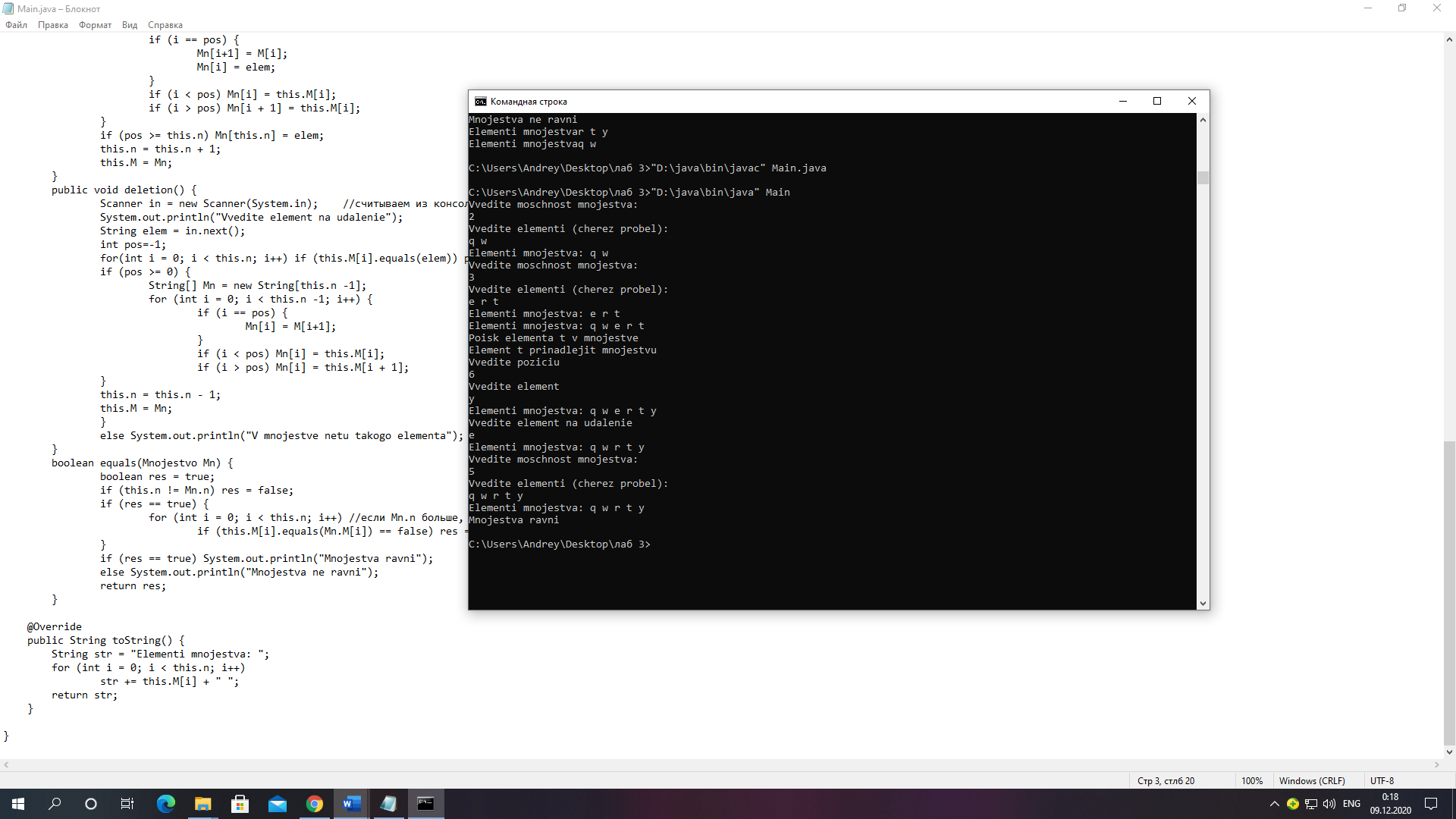
str += this.M[i] + " ";

return str;

}

}

**Результат:**



**Задание 2:**

**Текст программы:**

import java.io.\*;

import java.util.Scanner;

import java.io.FileNotFoundException;

class Tree {

public Tree left; //левое поддерево

public Tree right; //правое поддерево

public int time; //время отправления

public int timeH; //час отправления

public int timeM; //минуты отправления

public String station1; //станция 1

public String station2; //станция 2

public int number; //номер поезда

public String kind; //вид вагона (общий, плацкарт, купе)

public Tree(int h, int m, String s1, String s2, int n, String k) { // конструктор с инициализацией ключа

timeH = h;

timeM = m;

time = h\*100 + m;

station1 = s1;

station2 = s2;

number = n;

kind = k;

}

public void insert(Tree aTree) { //добавление нового поддерева

if (aTree.time < time)

if (left != null) left.insert(aTree);//рекурсивно добавить новое дерево в левое поддерево

else left = aTree;//если поддерева нет, то вставить на это место новое дерево

else

if (right != null) right.insert(aTree); //рекурсивно добавить новое дерево в правое поддерево

else right = aTree;//если поддерева нет, то вставить на это место новое дерево

}

public void traverse(TreeVisitor visitor) { //обход

if (left != null) left.traverse(visitor); //рекурсивно обойти левое поддерево

visitor.visit(this); //применить функцию печать к корневому узлу

if (right != null) right.traverse(visitor);//рекурсивно обойти правое поддерево

}

public void numbertrav(TreeVisitorN visitor) { //обход

if (left != null) left.numbertrav(visitor); //рекурсивно обойти левое поддерево

visitor.numbervisit(this); //применить функцию печать к корневому узлу

if (right != null) right.numbertrav(visitor); //рекурсивно обойти правое поддерево

}

public void stationtrav(TreeVisitorS visitor) { //обход

if (left != null) left.stationtrav(visitor); //рекурсивно обойти левое поддерево

visitor.stationvisit(this); //применить функцию печать к корневому узлу

if (right != null) right.stationtrav(visitor); //рекурсивно обойти правое поддерево

}

public void statimetrav(TreeVisitorST visitor) { //обход

if (left != null) left.statimetrav(visitor);//рекурсивно обойти левое поддерево

visitor.statimevisit(this); //применить функцию печать к корневому узлу

if (right != null) right.statimetrav(visitor); //рекурсивно обойти правое поддерево

}

public void stakindtrav(TreeVisitorSK visitor) { //обход

if (left != null) left.stakindtrav(visitor); //рекурсивно обойти левое поддерево

visitor.stakindvisit(this); //применить функцию печать к корневому узлу

if (right != null) right.stakindtrav(visitor); //рекурсивно обойти правое поддерево

}

public void timetrav(TreeVisitorT visitor) { //обход

if (left != null) left.timetrav(visitor); //рекурсивно обойти левое поддерево

visitor.timevisit(this); //применить функцию печать к корневому узлу

if (right != null) right.timetrav(visitor); //рекурсивно обойти правое поддерево

}

}

interface TreeVisitor {

public void visit(Tree node);

};

interface TreeVisitorN {

public void numbervisit(Tree node);

};

interface TreeVisitorS {

public void stationvisit(Tree node);

};

interface TreeVisitorST {

public void statimevisit(Tree node);

};

interface TreeVisitorSK {

public void stakindvisit(Tree node);

};

interface TreeVisitorT {

public void timevisit(Tree node);

};

class KeyPrinter implements TreeVisitor {

public void visit(Tree node) {

if (node.timeH < 10) System.out.print("0");

System.out.println(node.timeH + ":" + node.timeM + " " + node.station1 + " " + node.station2 + " " + node.number + " " + node.kind);

}

};

class NumberPrinter implements TreeVisitorN {

Scanner in = new Scanner(System.in);

int num = in.nextInt();

public void numbervisit(Tree node) {

if (node.number == num) {

System.out.println("time station1 station2 number vagons");

if (node.timeH < 10) System.out.print("0");

System.out.println(node.timeH + ":" + node.timeM + " " + node.station1 + " " + node.station2 + " " + node.number + " " + node.kind);

}

}

};

class StationPrinter implements TreeVisitorS {

Scanner in = new Scanner(System.in);

String st = in.nextLine();

public void stationvisit(Tree node) {

if (node.station2.equals(st)) {

System.out.println("time station1 station2 number vagons");

if (node.timeH < 10) System.out.print("0");

System.out.println(node.timeH + ":" + node.timeM + " " + node.station1 + " " + node.station2 + " " + node.number + " " + node.kind);

}

}

};

class StaTimePrinter implements TreeVisitorST {

Scanner in = new Scanner(System.in);

int tim = in.nextInt();

Scanner ins = new Scanner(System.in);

String st = ins.nextLine();

public void statimevisit(Tree node) {

if (node.station2.equals(st) && node.timeH > tim) {

System.out.println("time station1 station2 number vagons");

if (node.timeH < 10) System.out.print("0");

System.out.println(node.timeH + ":" + node.timeM + " " + node.station1 + " " + node.station2 + " " + node.number + " " + node.kind);

}

}

};

class StaKindPrinter implements TreeVisitorSK {

Scanner in = new Scanner(System.in);

String st = in.nextLine();

public void stakindvisit(Tree node) {

if (node.station2.equals(st) && node.kind.equals("obschiy")) {

System.out.println("time station1 station2 number vagons");

if (node.timeH < 10) System.out.print("0");

System.out.println(node.timeH + ":" + node.timeM + " " + node.station1 + " " + node.station2 + " " + node.number + " " + node.kind);

}

}

};

class TimePrinter implements TreeVisitorT {

Scanner in = new Scanner(System.in);

int t = in.nextInt();

public void timevisit(Tree node) {

if (node.time == t+10) {

System.out.println("Cherez 10 min train number " + node.number + " otpravlyaetsa");

}

if (node.time == t+5) {

System.out.println("Cherez 5 min train number " + node.number + " otpravlyaetsa");

}

if (node.time == t+3) {

System.out.println("Cherez 3 min train number " + node.number + " otpravlyaetsa");

}

}

};

class Trains {

public static void main(String args[]) throws java.io.FileNotFoundException{

Tree myTree;

int p1, p2, p5;

String p3, p4, p6;

boolean run = true;

File file = new File("datatrains.txt");

Scanner scan = new Scanner(file);

p1 = Integer.parseInt(scan.nextLine());

p2 = Integer.parseInt(scan.nextLine());

p3 = scan.nextLine();

p4 = scan.nextLine();

p5 = Integer.parseInt(scan.nextLine());

p6 = scan.nextLine();

myTree = new Tree(p1, p2, p3, p4, p5, p6);

while(scan.hasNextLine()) {

p1 = Integer.parseInt(scan.nextLine());

p2 = Integer.parseInt(scan.nextLine());

p3 = scan.nextLine();

p4 = scan.nextLine();

p5 = Integer.parseInt(scan.nextLine());

p6 = scan.nextLine();

myTree.insert(new Tree(p1, p2, p3, p4, p5, p6));

}

scan.close();

System.out.println("Enter 1 - show all info about trains");

System.out.println("Enter 2 - info about train by number");

System.out.println("Enter 3 - info about train by end station");

System.out.println("Enter 4 - info about train by time and end station");

System.out.println("Enter 5 - info about train with obschimi mestami by time");

System.out.println("Enter 6 - time");

System.out.println("Enter 0 - exit");

Scanner sc = new Scanner(System.in);

while (run) {

int ch = sc.nextInt();

if (ch == 1) {

System.out.println("time station1 station2 number vagons");

myTree.traverse(new KeyPrinter());

}

if (ch == 2) {

System.out.println("Write number of train:");

myTree.numbertrav(new NumberPrinter());

}

if (ch == 3) {

System.out.println("Write name of station:");

myTree.stationtrav(new StationPrinter());

}

if (ch == 4) {

System.out.println("Write number of hour and name of station:");

myTree.statimetrav(new StaTimePrinter());

}

if (ch == 5) {

System.out.println("Write name of station:");

myTree.stakindtrav(new StaKindPrinter());

}

if (ch == 6) {

System.out.println("Write time:");

myTree.timetrav(new TimePrinter());

}

if (ch == 0) run = false;

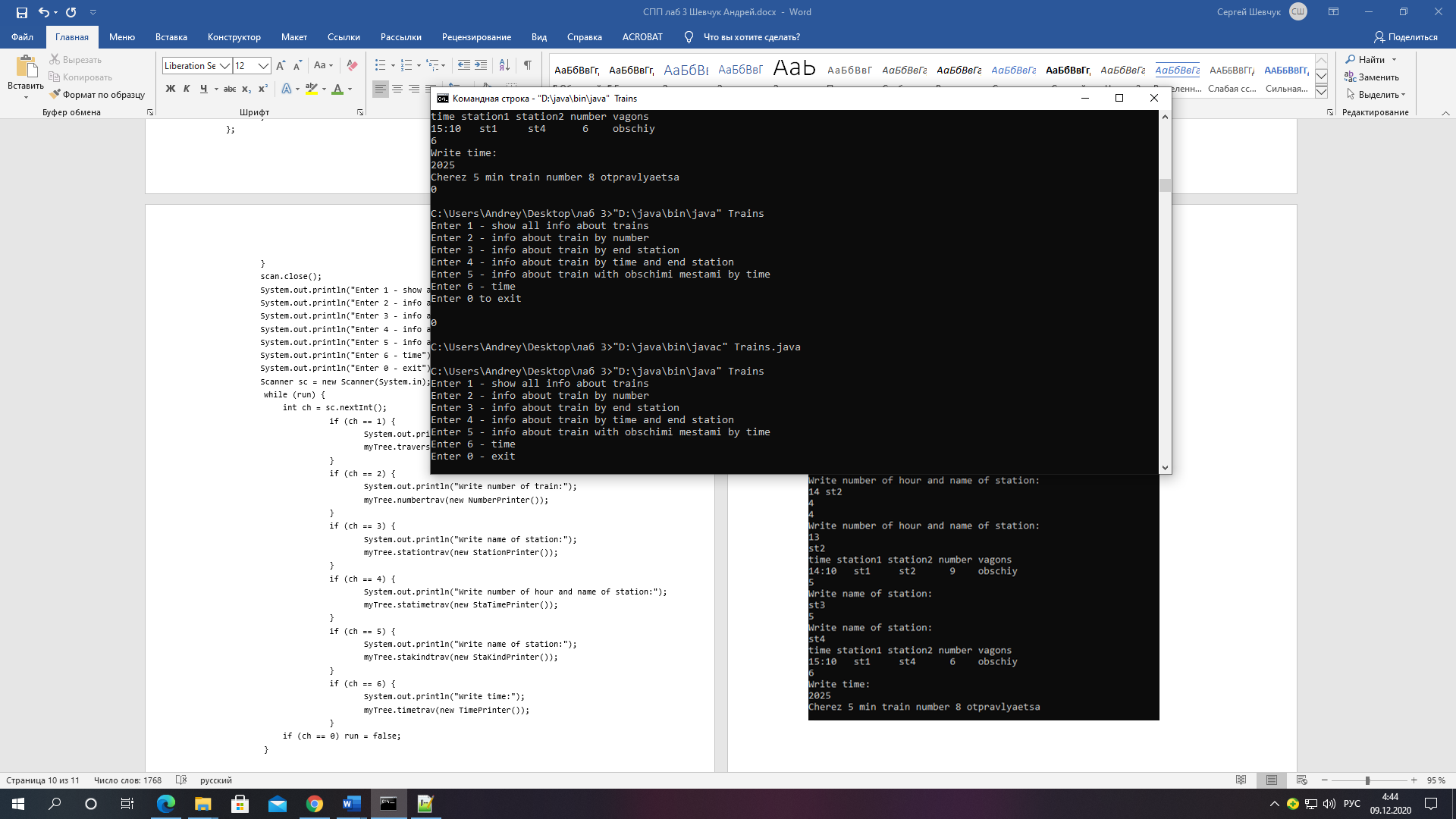
}

}

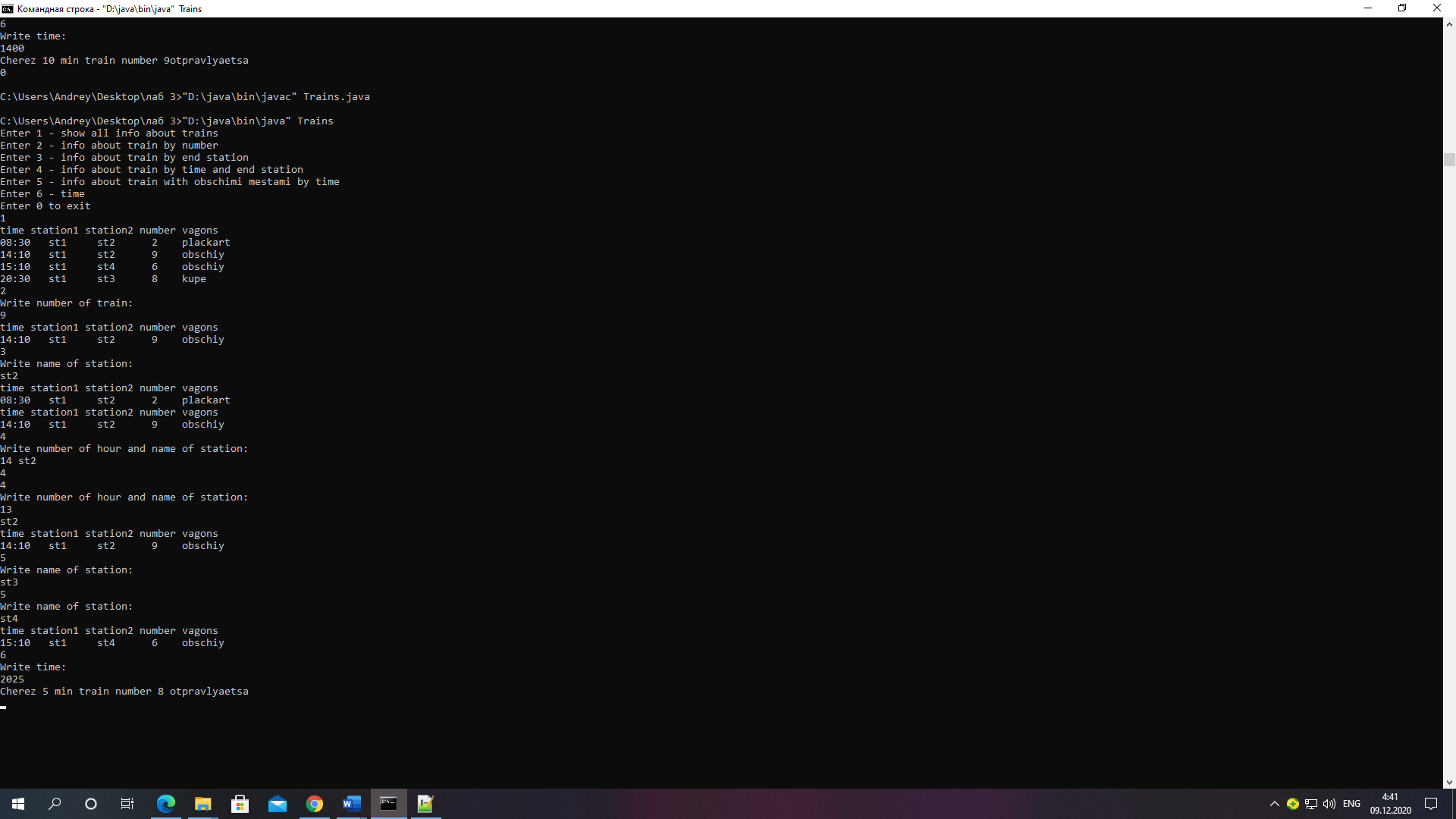
}

**Результат:**

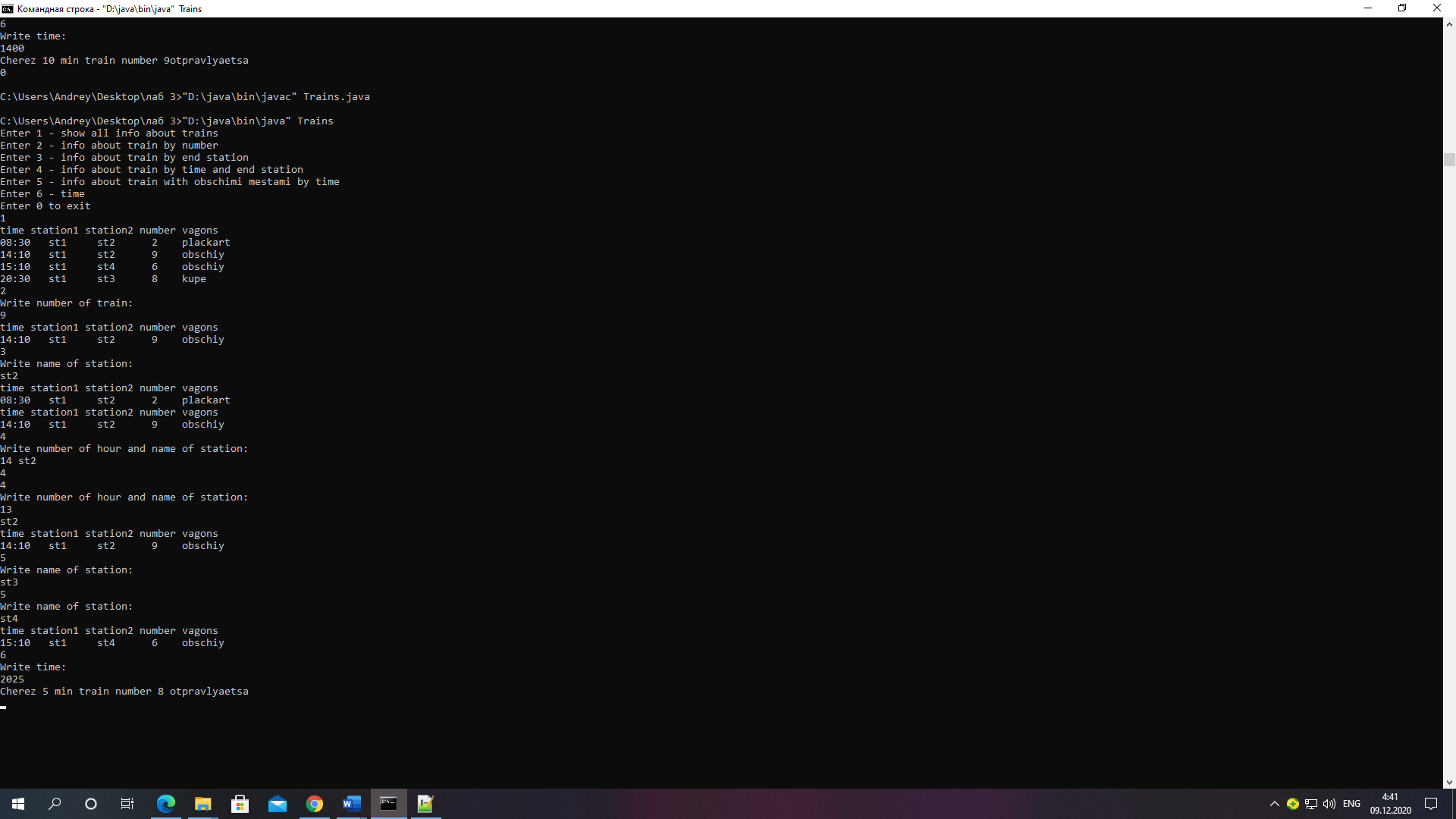
При запуске мы видим меню, предлагающее выбрать действие:



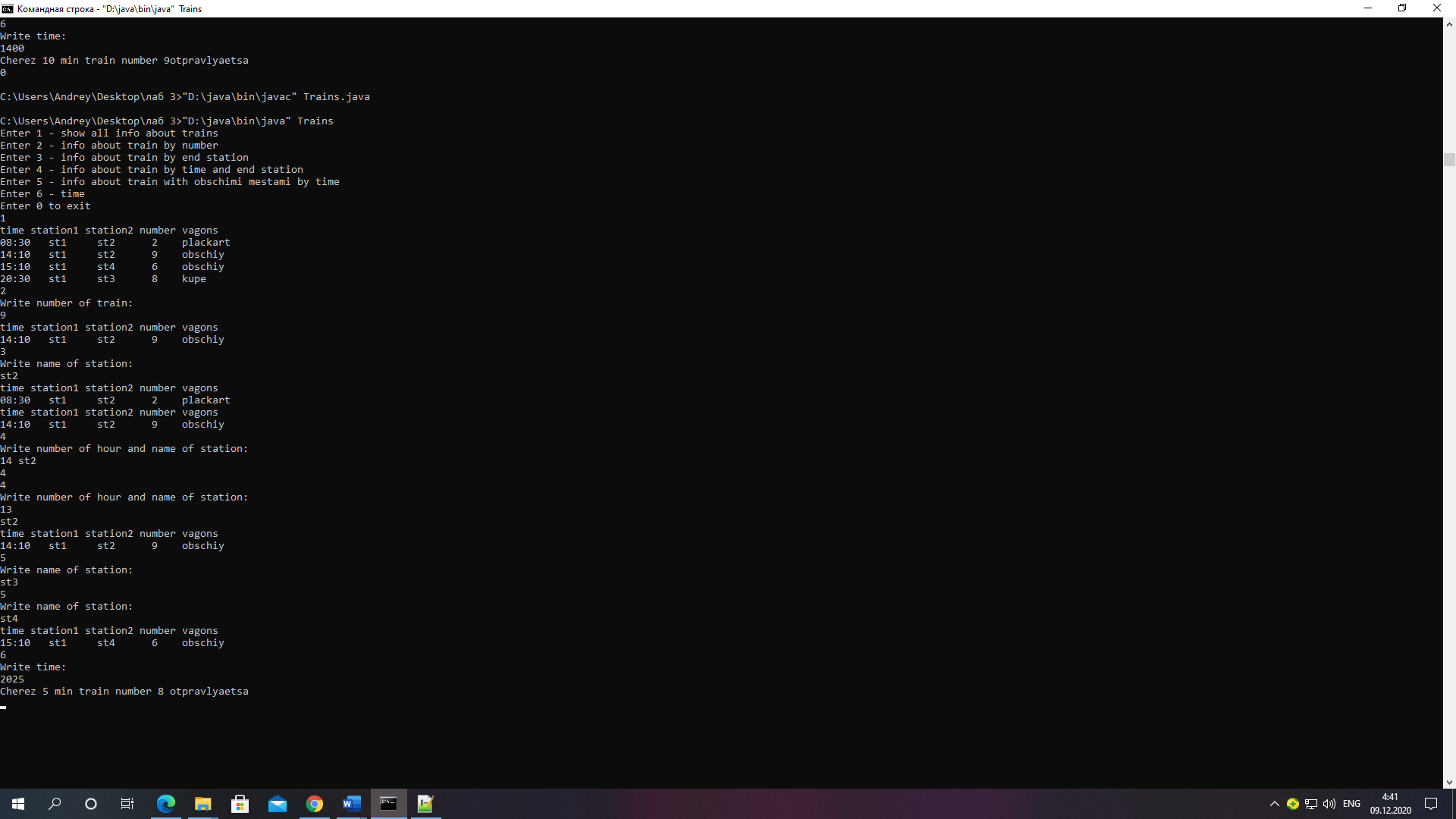
1) выводит расписание поездов на экран



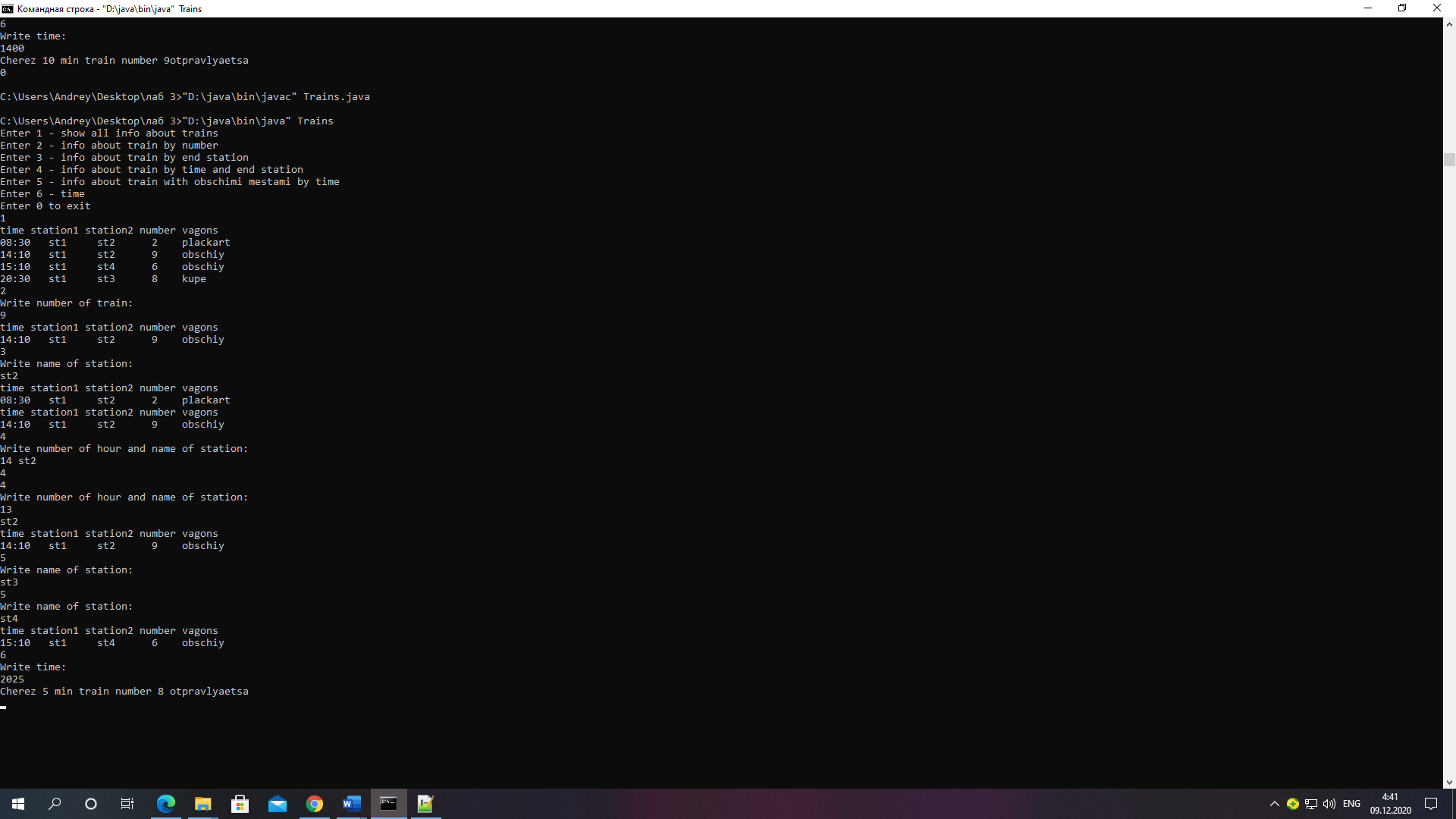
2) выводит информацию о поезде по номеру поезда



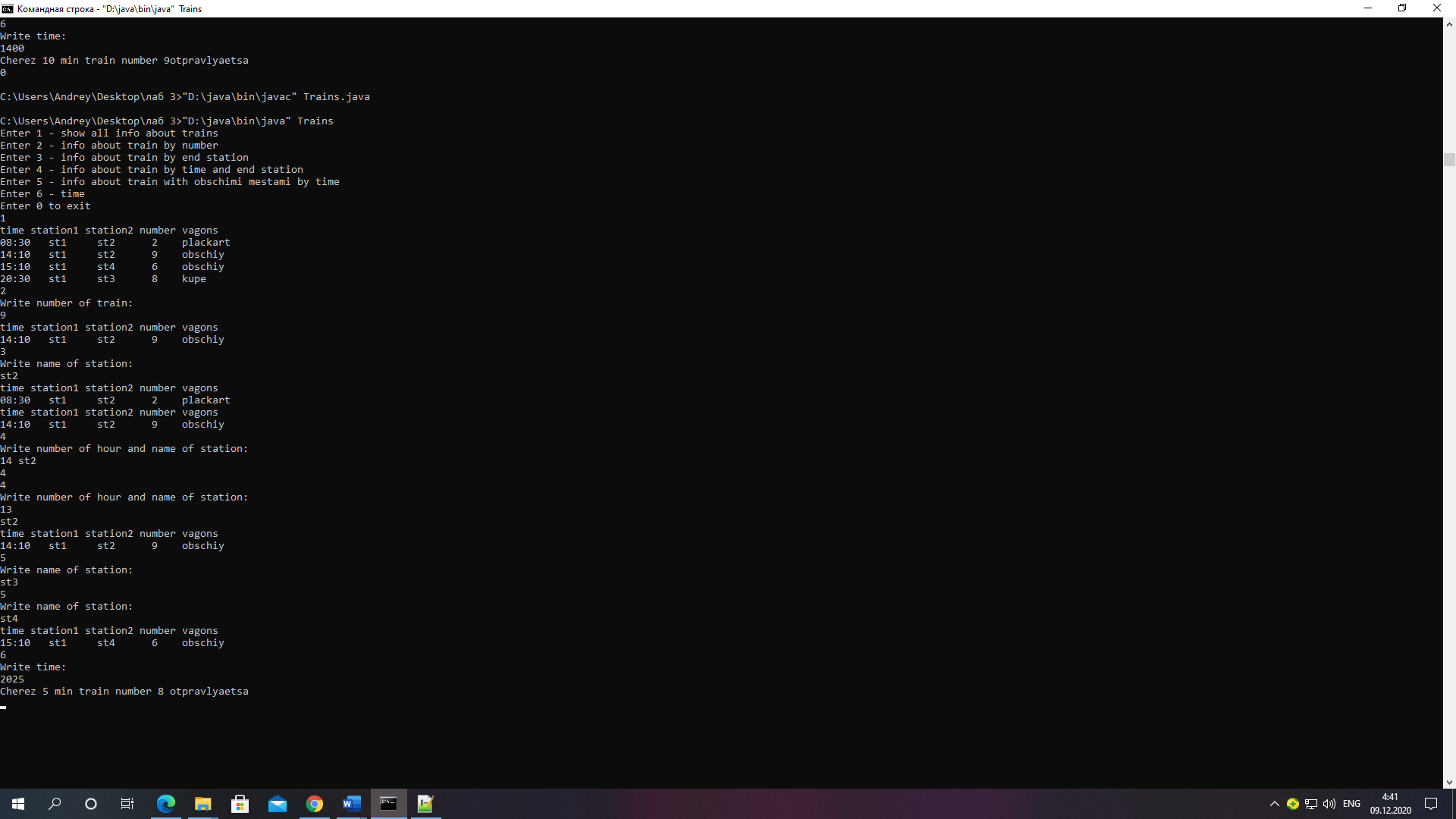
3) выводит данные обо всех поездах, которые следуют до этой станции назначения



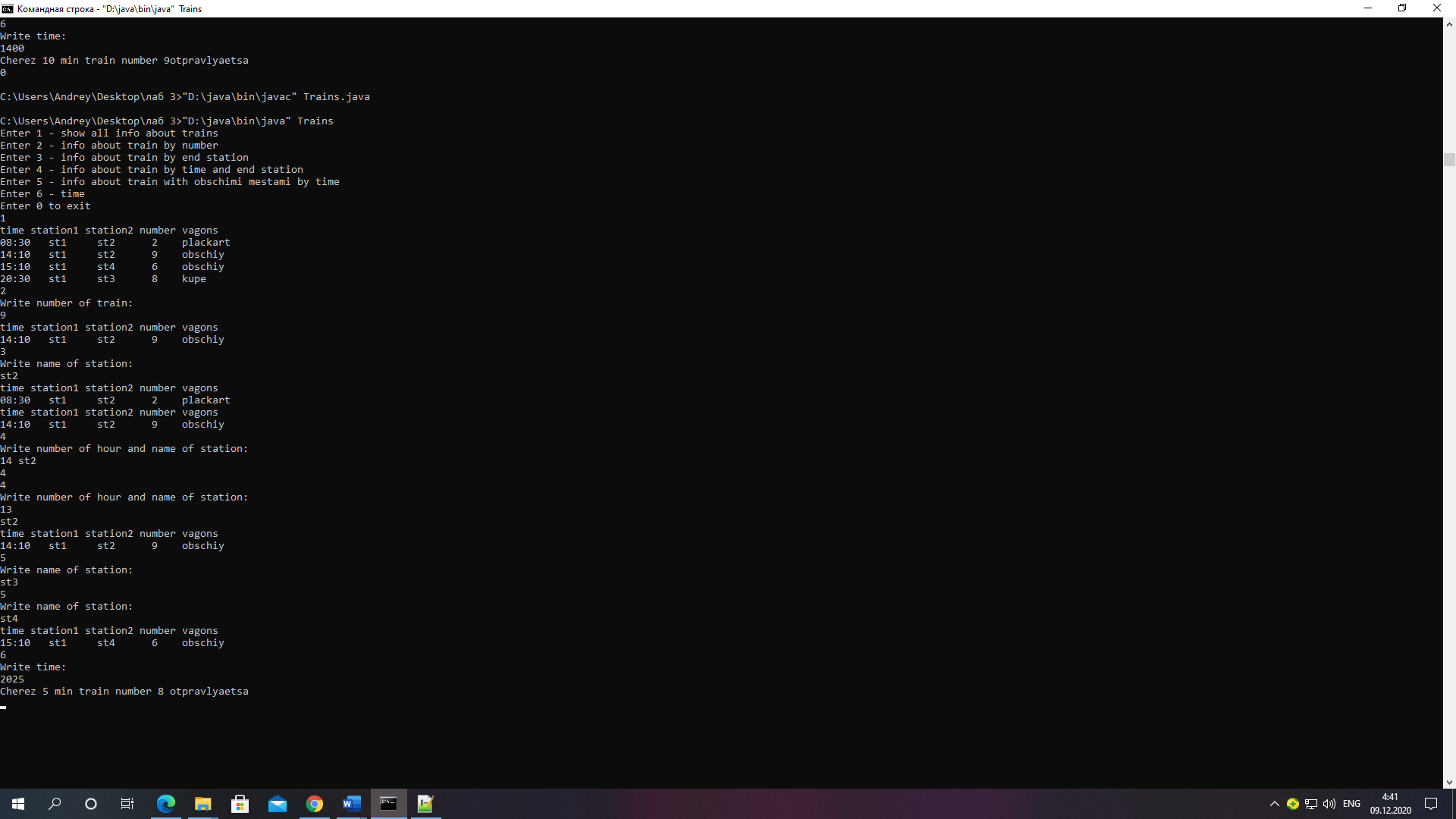
4) выводит список поездов, следующих до заданного пункта назначения и отправляющихся после заданного часа



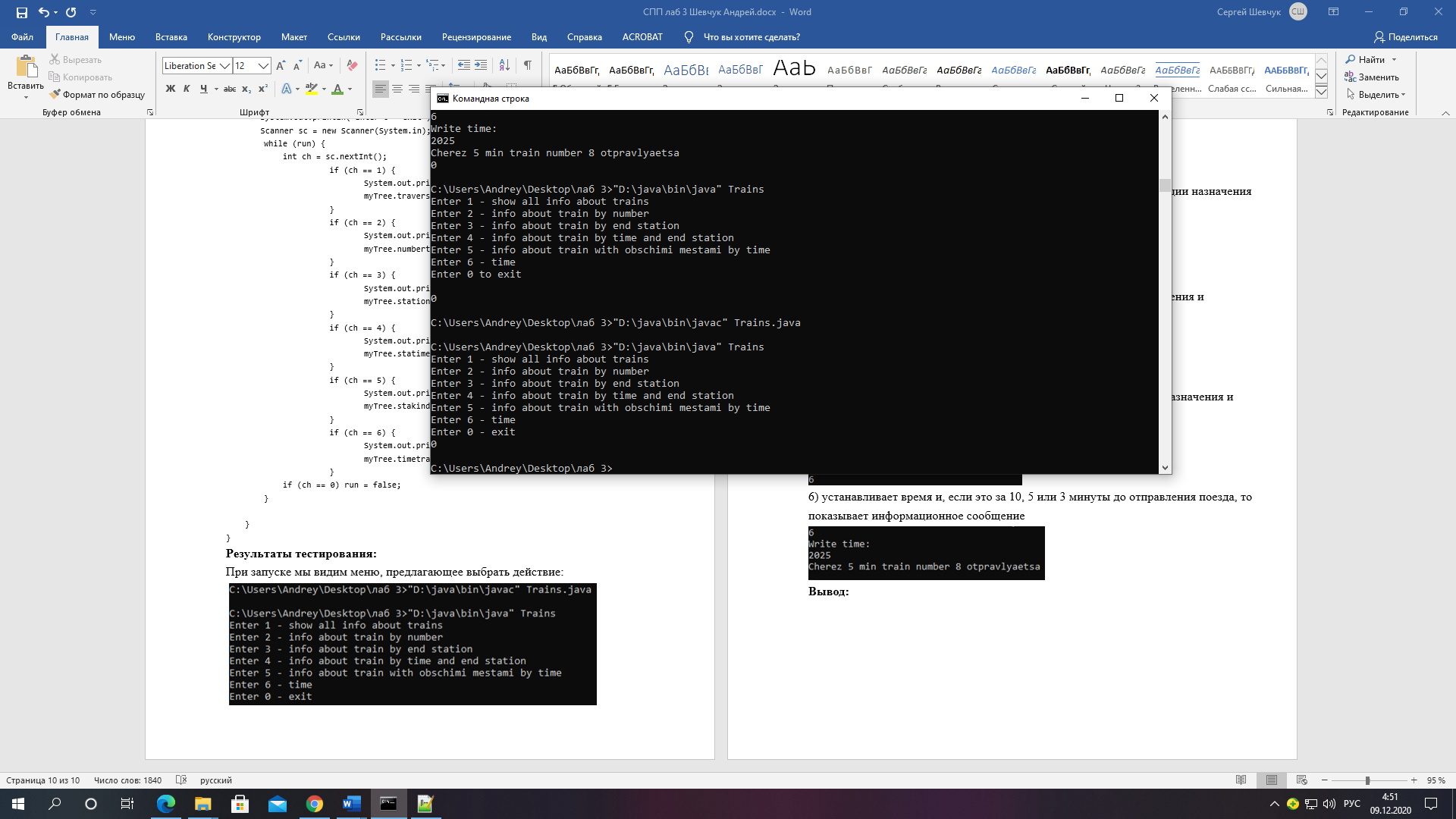
5) выводит список поездов, отправляющихся до заданного пункта назначения и имеющих общие места



6) устанавливает время и, если это за 10, 5 или 3 минуты до отправления поезда, то показывает информационное сообщение



7) при вводе «0» выходит из программы



**Вывод:** научился создавать и использовать классы в программах на языке программирования Java.