|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| *Федеральное государственное бюджетное образовательное учреждение высшего образования* ***«МИРЭА – Российский технологический университет»***  **РТУ МИРЭА** |

**Институт информационных технологий (ИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИиППО)**

**Дисциплина «Программирование на языке Джава»**

**ОТЧЕТ**

**ПО ПРАКТИЧЕСКОМУ ЗАНЯТИЮ №19-20**

Выполнил студент группы ИНБО-02-20 Маркарянц Р.Д.

Принял Степанов П.В.

Практические работы выполнены «\_\_\_»\_\_\_\_\_\_\_2021г.

«\_\_\_\_\_\_\_\_\_\_\_\_\_\_» «\_\_\_»\_\_\_\_\_\_\_2021г.

Отметка о выполнении

**Москва – 2021 г.**

## **Задание**

1. Напишите генератор «красивых» автомобильных номеров. Используйте правила генерации номеров для получения более 2 млн номеров:

* X, Y, Z — различные буквы автомобильного номера ("А", "В", "Е", "К", "М", "Н", "О", "Р", "С", "Т", "У", "Х"), N — цифры, R — регион (от 01 до 199);
* XNNNYZR — пример, A111BC197, Y777HC66

1. Используя генератор «красивых» номеров сгенерируйте N-номеров и проведите поиск номера, введенного из консоли, с помощью методов:

* прямым перебором по ArrayList, (array.contains())
* бинарным поиском по сортированному ArrayList, (Collections.binarySearch())
* Копировать
* поиском в TreeSet. (setTree.содержит())

## **Ход Работы**

В ходе выполнения работы были получены следующие исходные коды:

package ru.mirea.inbo0220.markaryants.pr19;

import java.util.\*;

public class Generator {

static Map<Integer, String> *numb* = new HashMap<>();

static int[] *masDigits* = new int[]{0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

public static String genNumber() {

int a = (int) (Math.*random*() \* 10 + 0);

String str = *numb*.get((int)(Math.*random*() \* 12 + 0)) + *masDigits*[a]+*masDigits*[a]+*masDigits*[a] +

*numb*.get((int)(Math.*random*() \* 12 + 0))+*numb*.get((int)(Math.*random*() \* 12 + 0))+ (int)(Math.*random*() \* 199 + 1) ;

return str;

}

public static void main(String[] args) {

*numb*.put(0, "А");

*numb*.put(1, "В");

*numb*.put(2, "Е");

*numb*.put(3, "К");

*numb*.put(4, "М");

*numb*.put(5, "Н");

*numb*.put(6, "О");

*numb*.put(7, "Р");

*numb*.put(8, "С");

*numb*.put(9, "Т");

*numb*.put(10, "У");

*numb*.put(11, "Х");

ArrayList<String> array = new ArrayList<>();

TreeSet<String> tree = new TreeSet<>();

HashSet<String> hash = new HashSet<>();

Scanner in = new Scanner(System.*in*);

int N = in.nextInt();

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

String k = *genNumber*();

array.add(k);

tree.add(k);

hash.add(k);

System.*out*.println(k);

}

String str = in.next();

long beginTime = System.*nanoTime*();

boolean b = array.contains(str);

long elapsedTime = System.*nanoTime*() - beginTime;

if (b) System.*out*.println("Поиск перебором: номер найден, поиск занял " + elapsedTime + "нс");

else System.*out*.println("Поиск перебором: номер не найден, поиск занял " + elapsedTime +"нс");

Collections.*sort*(array);

beginTime = System.*nanoTime*();

int cls;

cls = Collections.*binarySearch*(array, str);

elapsedTime = System.*nanoTime*() - beginTime;

if (cls >=0) System.*out*.println("Бинарный поиск: номер найден, поиск занял " + elapsedTime + "нс");

else System.*out*.println("Бинарный поиск: номер не найден, поиск занял " + elapsedTime +"нс");

beginTime = System.*nanoTime*();

b = tree.contains(str);

elapsedTime = System.*nanoTime*() - beginTime;

if (b) System.*out*.println("Поиск в TreeSet: номер найден, поиск занял " + elapsedTime + "нс");

else System.*out*.println("Поиск в TreeSet: номер не найден, поиск занял " + elapsedTime +"нс");

beginTime = System.*nanoTime*();

b = hash.contains(str);

elapsedTime = System.*nanoTime*() - beginTime;

if (b) System.*out*.println("Поиск в HashSet: номер найден, поиск занял " + elapsedTime + "нс");

else System.*out*.println("Поиск в HashSet: номер не найден, поиск занял " + elapsedTime +"нс");

}

}

## **Вывод**

Выполнив практическую работу, я изучил методы поиска в различных структурах данных на языке Java.