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 Лукьяненко Д.В.

Принял Степанов П.В.

Практическая работа выполнена «___»_____2021 г.

«_____» «___»_____2021 г.
Отметка о выполнении

Москва – 2021 г.

СОДЕРЖАНИЕ

Цель работы

Цель данной практической работы – Реализовать генератор «красивых» автомобильных номеров.

Задание

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

- X, Y, Z — различные буквы автомобильного номера ("A", "B", "E", "K", "M", "H", "O", "P", "C", "T", "У", "X"), N — цифры, R — регион (от 01 до 199);
- XNNNYZR — пример, A111BC197, Y777HC66

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())
- поиском в HashSet, (setHash.contains())
- поиском в TreeSet. (setTree.contains())

Измерьте и сравните длительность каждого метода поиска. Формат вывода результатов поиска:

- Поиск перебором: номер <найден/не найден>, поиск занял 34нс
- Бинарный поиск: номер <найден/не найден>, поиск занял 34нс
- Поиск в HashSet: номер <найден/не найден>, поиск занял 34нс
- Поиск в TreeSet: номер <найден/не найден>, поиск занял 34нс

Репозиторий

Ссылка: https://github.com/neluckoff/mirea-java-lessons/tree/master/src/ru/luckoff/mirea/exercies19and20

Выполнение работы

В процессе выполнения поставленных заданий я реализовал ArrayList, HashSet и TreeSet, и заполнял их одновременно одинаковыми номерами.

Для генерации номеров используются две функции – первая генерирует красивые номера с одинаковыми буквами, например A128AA197, а вторая – красивые номера с одинаковыми цифрами.

Также была предусмотрена особенность регионов, ведь когда регион пятый, он пишется, как 05, а не 5.

Код выполненной работы

```
public class CarNumberGenerator {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);

        String[] letter = {"A", "B", "E", "K", "M", "H", "O", "P", "C", "T", "Y", "X"};
        Arrays.sort(letter);

        ArrayList<String> arrayList = new ArrayList<>();
        HashSet<String> hashSet = new HashSet<>();
        TreeSet<String> treeSet = new TreeSet();
```

Рисунок 1 – Создание массива букв и списка с номерами

```
for (String a123 : letter) {
    for (int reg = 1; reg <= 199; reg++) {
        for (int j = 1; j <= n; j++) {
            if (reg < 10) {
                arrayList.add(String.format("%s%03d%s%s%02d", a123, j, a123, a123, reg));
                hashSet.add(String.format("%s%03d%s%s%02d", a123, j, a123, a123, reg));
                treeSet.add(String.format("%s%03d%s%s%02d", a123, j, a123, a123, reg));
            } else {
                arrayList.add(String.format("%s%03d%s%s%d", a123, j, a123, a123, reg));
                hashSet.add(String.format("%s%03d%s%s%d", a123, j, a123, a123, reg));
                treeSet.add(String.format("%s%03d%s%s%d", a123, j, a123, a123, reg));
            }
        }
    }
}
```

Рисунок 2 – Генерация номеров с одинаковыми буквами

```
for (int i = 111; i <= 999; i += 111) {
    for (String a1 : letter) {
        for (String a2 : letter) {
            for (String a3 : letter) {
                for (int reg = 1; reg <= 199; reg++) {
                    if (reg < 10) {
                        arrayList.add(String.format("%s%03d%s%s%02d", a1, i, a2, a3, reg));
                        hashSet.add(String.format("%s%03d%s%s%02d", a1, i, a2, a3, reg));
                        treeSet.add(String.format("%s%03d%s%s%02d", a1, i, a2, a3, reg));
                    } else {
                        arrayList.add(String.format("%s%03d%s%s%d", a1, i, a2, a3, reg));
                        hashSet.add(String.format("%s%03d%s%s%d", a1, i, a2, a3, reg));
                        treeSet.add(String.format("%s%03d%s%s%d", a1, i, a2, a3, reg));
                    }
                }
            }
        }
    }
}
```

Рисунок 3 – Генерация номеров с одинаковыми цифрами

```
System.out.print("Введите номер для поиска - ");
String search = in.next();

//Поиск перебором
long m = System.nanoTime();
if (arrayList.contains(search)) {
    System.out.println("Поиск перебором: номер " + search + " найден, поиск занял " +
            (System.nanoTime() - m) + "нс");
} else System.out.println("Поиск перебором: номер " + search + " не найден, поиск занял " +
        (System.nanoTime() - m) + "нс");
```

Рисунок 4 – Подсчет времени поиска перебором

```
//Бинарный поиск
Collections.sort(arrayList);
long m1 = System.nanoTime();
boolean result = arrayList.contains(search);
if (result) {
    System.out.println("Бинарный поиск: номер " + search + " найден, поиск занял " +
            (System.nanoTime() - m1) + "нс");
} else System.out.println("Бинарный поиск: номер " + search + " не найден, поиск занял " +
        (System.nanoTime() - m1) + "нс");
```

Рисунок 5 – Подсчет времени бинарного поиска

```
//Поиск в HashSet
long m2 = System.nanoTime();
if (hashSet.contains(search)) {
    System.out.println("Поиск в HashSet: номер " + search + " найден, поиск занял " +
            (System.nanoTime() - m2) + "нс");
} else System.out.println("Поиск в HashSet: номер " + search + " не найден, поиск занял " +
        (System.nanoTime() - m2) + "нс");
```

Рисунок 6 – Подсчет времени поиска в HashSet

```
//Поиск в TreeSet
long m3 = System.nanoTime();
if (treeSet.contains(search)) {
    System.out.println("Поиск в TreeSet: номер " + search + " найден, поиск занял " +
            (System.nanoTime() - m3) + "нс");
} else System.out.println("Поиск в TreeSet: номер " + search + " не найден, поиск занял " +
        (System.nanoTime() - m3) + "нс");
```

Рисунок 7 – Подсчет времени поиска в TreeSet

Тестирование программы

Так как показать вывод всего списка сгенерированных номеров будет достаточно сложно, на рисунке 8 я покажу лишь конец этого списка.

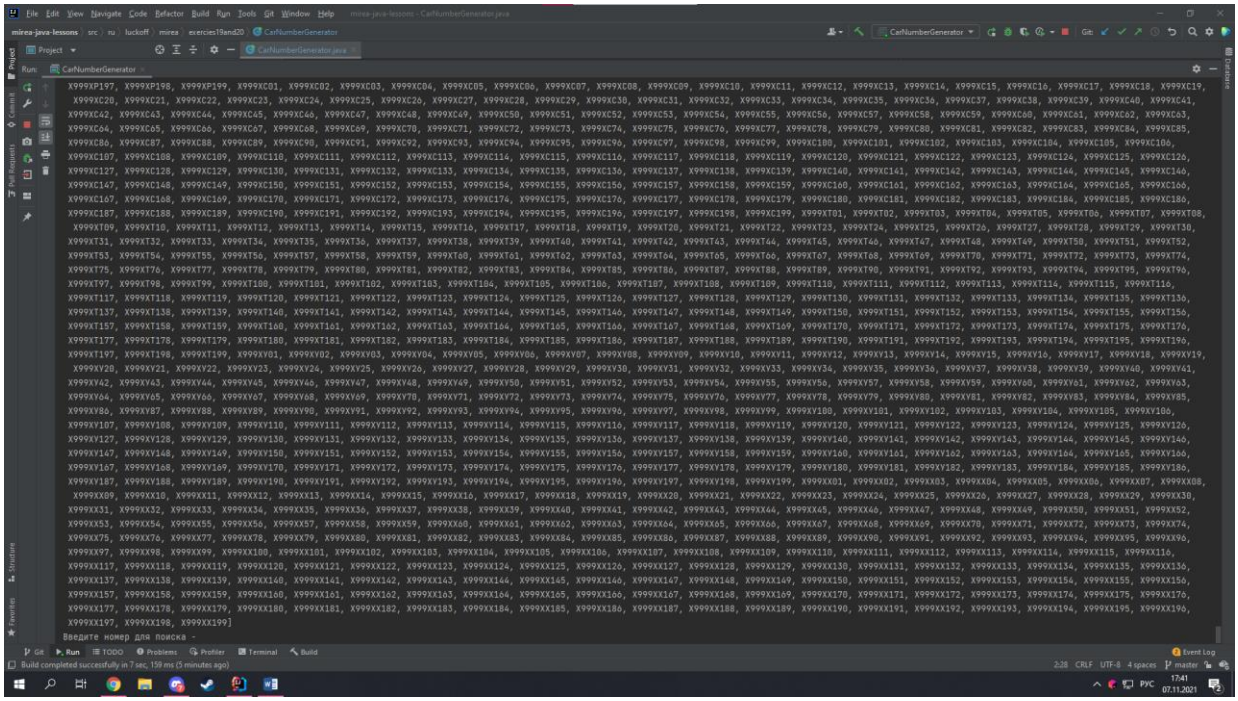

Рисунок 8 – Вывод полученного списка красивых номеров

```
X999XX97, X999XX98, X999XX99, X999XX100, X999XX101, X999XX102, X999XX103, X999XX104, X999XX105, X9
X999XX117, X999XX118, X999XX119, X999XX120, X999XX121, X999XX122, X999XX123, X999XX124, X999XX125,
X999XX137, X999XX138, X999XX139, X999XX140, X999XX141, X999XX142, X999XX143, X999XX144, X999XX145,
X999XX157, X999XX158, X999XX159, X999XX160, X999XX161, X999XX162, X999XX163, X999XX164, X999XX165,
X999XX177, X999XX178, X999XX179, X999XX180, X999XX181, X999XX182, X999XX183, X999XX184, X999XX185,
X999XX197, X999XX198, X999XX199]
```

Рисунок 9 – Более детальный конец списка

```
Введите номер для поиска - X999XY112
Поиск перебором: номер X999XY112 найден, поиск занял 106663100нс
Бинарный поиск: номер X999XY112 найден, поиск занял 240308200нс
Поиск в HashSet: номер X999XY112 найден, поиск занял 13600нс
Поиск в TreeSet: номер X999XY112 найден, поиск занял 31000нс

Process finished with exit code 0
```

Рисунок 10 – Результат поисков и их время

Вывод

В результате выполнения данной практической работы я смог создать программу, генерирующую «красивые» автомобильные номера, которая может создать более 2 млн. таких номеров. Помимо этого я узнал различие между ArrayList, HashSet и TreeSet, и научился искать в них объект. Как оказалось, в больших файлах быстрее всего работает поиск в HashSet.