Домашнее задание

Задание 1:
Реализуйте метод, который поменяет ключи и значения в HashMap местами. На вход в метод поступает HashMap<Integer, String>, надо вернуть HashMap<String, Integer>. Выведите результат

Задание 2:
Реализуйте метод, в котором создаются ArrayList, LinkedList и заполняются 1 000 000 случайными элементами одного и того же типа. После из ArrayList и LinkedList 1000 раз выбираем элемент по случайному индексу. Замерьте время для ArrayList и LinkedList. Сравните результаты и предположите почему они могут отличаться.

Задание3:
Реализуйте метод, который на вход примет ArrayList строк и удаляет из него все дубликаты, не используя метод contains(), можно использовать другие коллекции, которые были изучены на уроке

Решение задания 1:

```
public static void main(String[] args) {
    HashMap<Integer, String> map = new HashMap<>();
    map.put(10, "Igor");
    map.put(20, "Anton");

    System.out.println("Before: " + map);
    System.out.println("After: " + inverseKeysAndValues(map));
}

1 usage
public static HashMap<String, Integer> inverseKeysAndValues(HashMap<Integer, String> mp) {
    HashMap<String, Integer> resultMap = new HashMap<>();

    for (Map.Entry<Integer, String> entry : mp.entrySet()) {
        resultMap.put(entry.getValue(), entry.getKey());
    }

    return resultMap;
}
```

```
Before: {20=Anton, 10=Igor}
After: {Anton=20, Igor=10}
```

Решение задания 2:

```java
public static void compareLists() {
    ArrayList<Double> arrayList = new ArrayList<>();
    LinkedList<Double> linkedList = new LinkedList<>();

    final int N = 1000000;
    final int M = 1000;

    for (int i = 0; i < N; i++) {
        arrayList.add(Math.random());
        linkedList.add(Math.random());
    }

    long startTime = System.currentTimeMillis();
    for (int i = 0; i < M; i++) {
        arrayList.get((int) (Math.random() * (N - 1)));
    }
    System.out.println(System.currentTimeMillis() - startTime);

    startTime = System.currentTimeMillis();
    for (int i = 0; i < M; i++) {
        linkedList.get((int) (Math.random() * (N - 1)));
    }
    System.out.println(System.currentTimeMillis() - startTime);
}
```

Если вызвать этот метод, то в консоли можно увидеть, например, следующее:

```
0
2046
```

Т.е. время получения 1000 случайных элементов по индексу из LinkedList гораздо больше чем аналогичные операции для ArrayList. Это связано с тем что элементы ArrayList имею свои индексы, по которым можно быстро получить соответствующий элемент из памяти. В случае с LinkedList нам нужно каждый раз итерироваться по связному списку с самого начала и искать нужный нам индекс элемента, поэтому это выполняется значительно дольше.

Решение задания 3:

```java
public static void main(String[] args) {
    ArrayList<String> strs = new ArrayList<>();
    strs.add("s1");
    strs.add("s1");
    strs.add("s3");
    strs.add("s2");
    strs.add("s3");
    System.out.println("Before: " +  strs);
    System.out.println("After : " + removeDuplicates(strs));
}

1 usage
public static ArrayList<String> removeDuplicates(ArrayList<String> input) {
    LinkedHashSet<String> set = new LinkedHashSet<>();

    set.addAll(input);
    input.clear();

    input.addAll(set);
    return input;
}
```

```
Before: [s1, s1, s3, s2, s3]
After : [s1, s3, s2]
```

Для решения нам нужно использовать множество, т.е. коллецию, которая реализует Set: HashSet, LinkedHashSet, TreeSet. Все элементы множества уникальны, поэтому когда мы будет заполнять множество элементами ArrayList, в которых есть дубликаты - дубликаты сохранены не будут. Дальше просто очищаем входной ArrayList от всех элементов, заполняем его уникальными элементами и возвращаем обновленный ArrayList