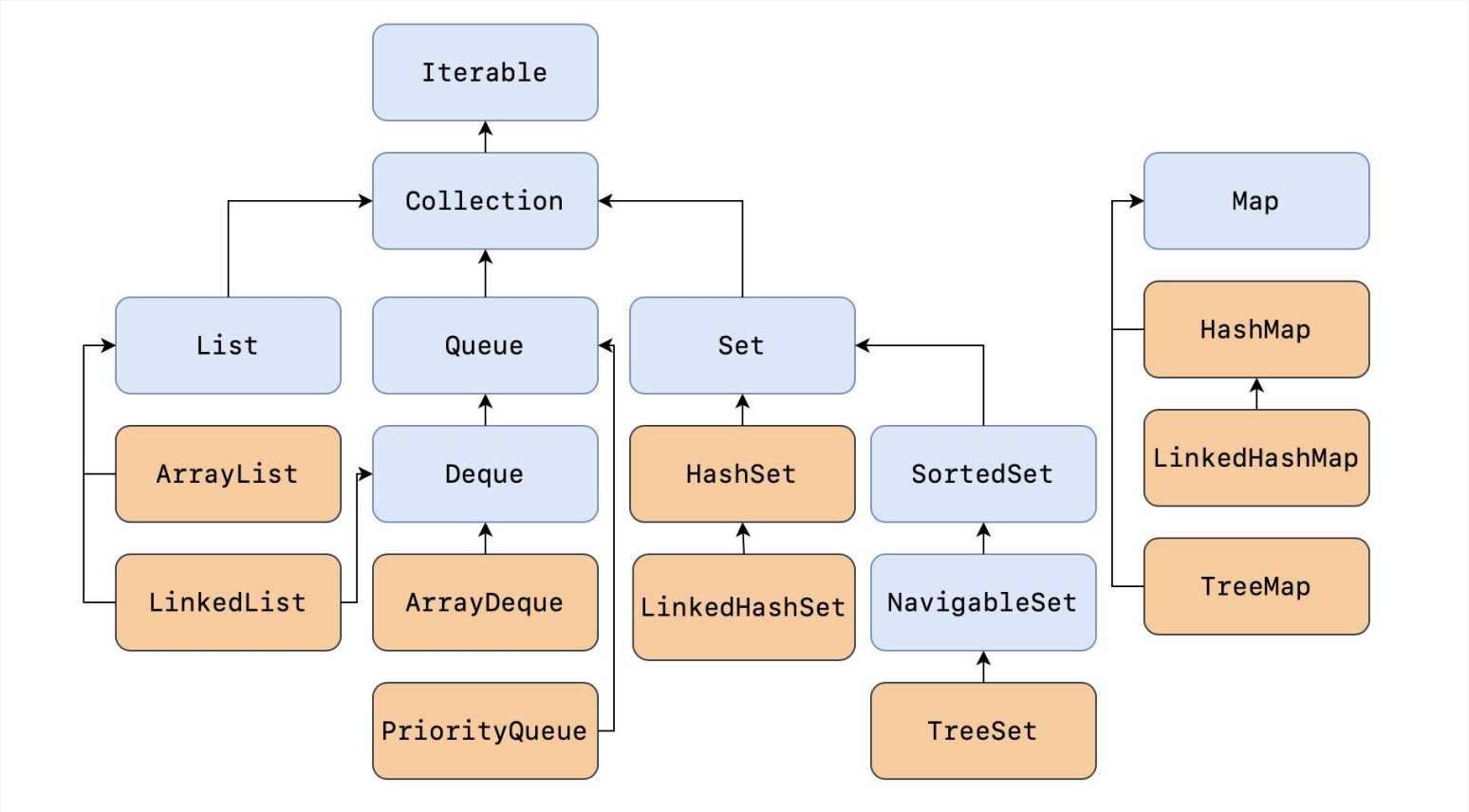
Хекслет

Коллекции в Java. Map. HashMap.

Наставник: Теплинская Мария Георгиевна
Группа: java-167
Дата: 14.01.2023

Иерархия коллекций



Ассоциативный массив

Ассоциативный массив - это абстракция, которая позволяет хранить данные в виде пар “*ключ-значение*” и поддерживает операции поиска, добавления и

Массив

0	Маша
1	Петя
2	Вова
3	Катя
4	Слава
5	Миша

Ассоциативный массив

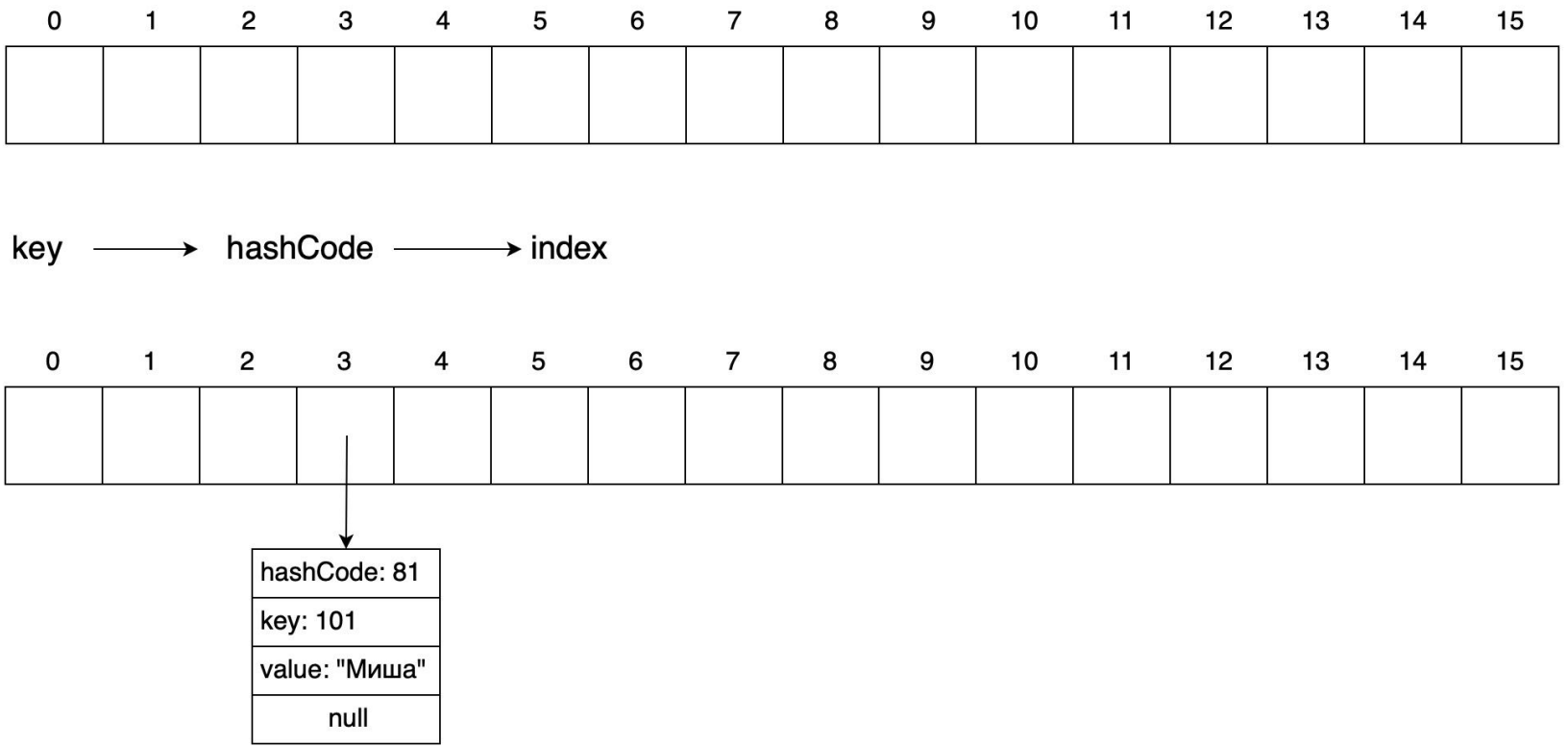
key	value
4510772544	Иванов Владислав Андреевич
4510781216	Смирнова Анна Николаевна
4510782433	Одинцов Владимир Алексеевич
4510788652	Тихонова Светлана Антоновна
4510782458	Воронцов Алексей Владимирович

HashMap

HashMap - это конкретная реализация коллекции Map, основанная на структуре данных под названием ***хэш-таблица***.

Хэш-таблица - структура данных, реализующая интерфейс ассоциативного массива, которая обеспечивает быструю вставку и удаление элементов.

Структура HashMap



Методы equals(..) и hashCode(..)

hashCode(..) и equals(..) - методы класса ***Object***, переопределяемые при необходимости в классах-наследниках.

Метод equals(..) служит для определения равенства объектов.

Метод hashCode(..) преобразует данные объекта в некоторое число типа int.

Согласно документации Oracle методы equals(..) и hashCode(..) связаны контрактом. Если переопределяется метод equals(..), должен быть переопределен и hashCode(..).

Правило

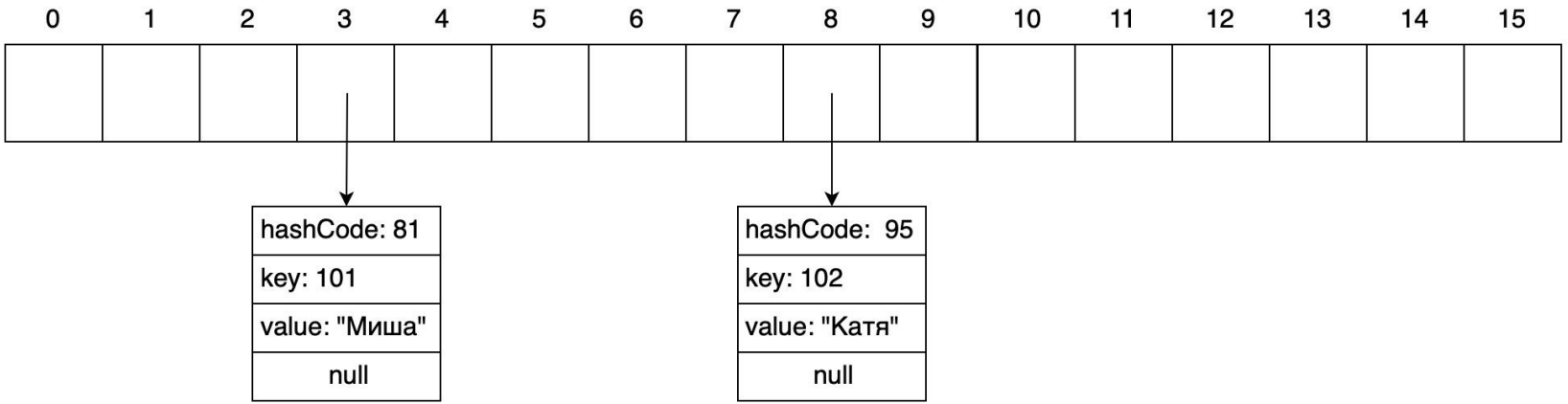
Если **a.equals(b) == true**, то a.hashCode() == b.hashCode()

Если **a.equals(b) == false**, то мы ничего не можем сказать про хэш-коды.

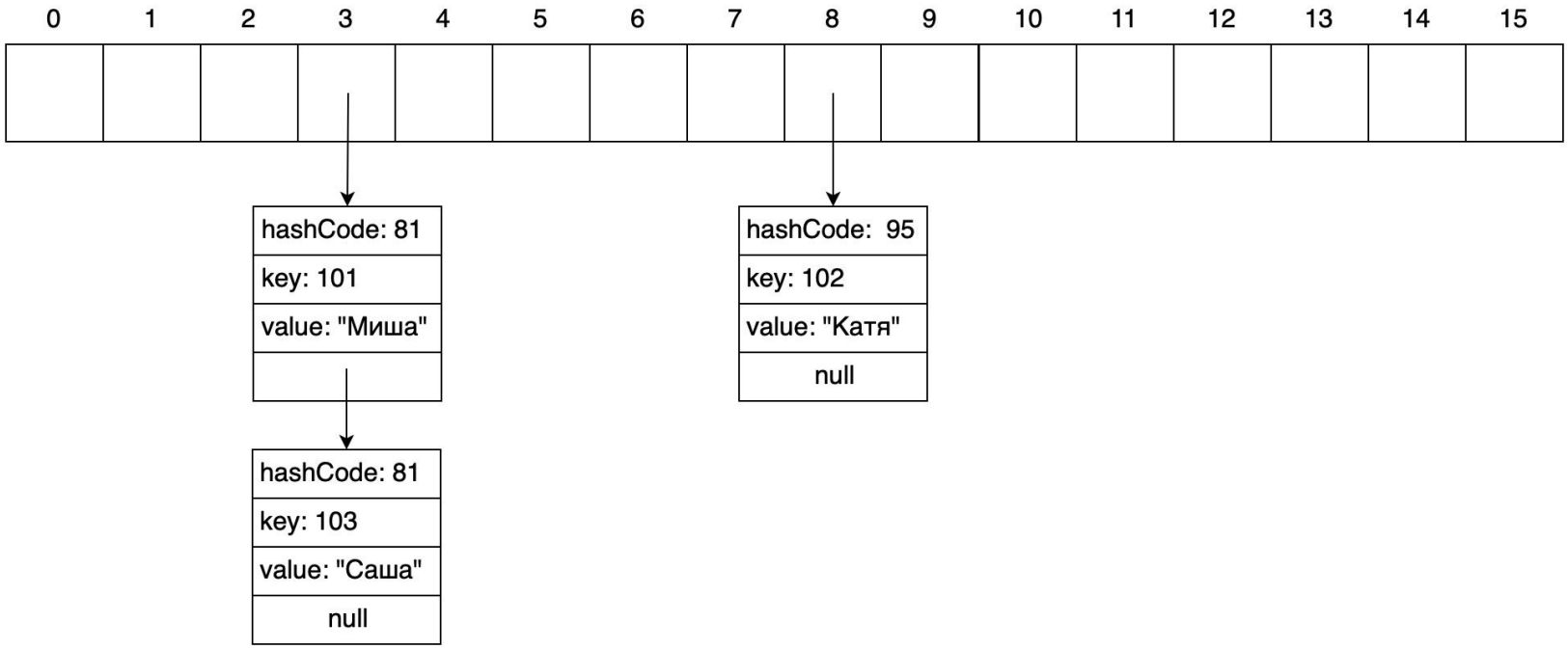
Если **a.hashCode() == b.hashCode()**, то мы ничего не можем сказать про равенство объектов, нам нужно проверять их на equals(..).

Если **a.hashCode() != b.hashCode()**, то на equals(..) можно не проверять, объекты точно не равны.

Добавление элемента



Коллизии


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
hashCode: 81
key: 101
value: "Миша"
hashCode: 81
key: 103
value: "Саша"
null
hashCode: 95
key: 102
value: "Катя"
null

capacity и loadFactor

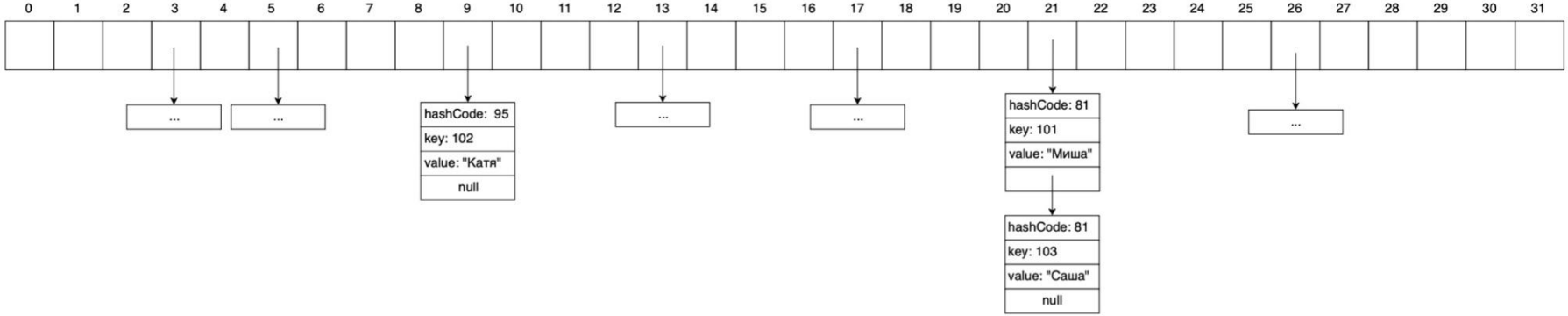
capacity - количество *buckets* в HashMap

loadFactor - коэффициент заполненности, после превышения которого идет расширение мапы.

threshold = capacity * loadFactor

коллизия - ситуация, при которой хэш-функция вернула такое значение, что в массиве по этому индексу уже есть элемент.

Rehashing


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
...
...
hashCode: 95
key: 102
value: "Катя"
null
...
...
hashCode: 81
key: 101
value: "Миша"
hashCode: 81
key: 103
value: "Саша"
null
...

Объявление HashMap

```
Map<String, String> map1 = new HashMap<>();

Map<String, String> map2 = new HashMap<>( initialCapacity: 64);

Map<String, String> map3 = new HashMap<>( initialCapacity: 32,  loadFactor: 0.99f);

Map<String, String> map4 = new HashMap<>(map3);
```

Основные операции

```
Map<String, String> map = new HashMap<>();
map.put("4510755678", "Linus Torvalds");
map.put("4510755667", "Donald Knuth");
map.put("4510755434", "Larry Page");
map.put("4510755434", "James Gosling");
map.put(null, "Anders Hejlsberg");
map.put("4510755322", null);

System.out.println(map);

System.out.println(map.size());
System.out.println(map.isEmpty());
System.out.println(map.containsKey("4510755667"));
System.out.println(map.containsKey(null));
System.out.println(map.containsValue("Donald Knuth"));
System.out.println(map.containsValue("Larry Page"));
System.out.println(map.containsValue(null));

System.out.println(map.get("4510755434"));
System.out.println(map.get("4510755315"));
System.out.println(map.get("4510755322"));
System.out.println(map.getOrDefault( key: "4510755315",  defaultValue: "No such value!"));
System.out.println(map.replace("4510755667", "Bjarne Stroustrup"));
```

Итерация по Map

```
Map<String, String> map = new HashMap<>();
map.put("4510755678", "Linus Torvalds");
map.put("4510755667", "Donald Knuth");
map.put("4510755434", "James Gosling");
map.put(null, "Anders Hejlsberg");
map.put("4510755322", null);

System.out.println(map);

for (String key: map.keySet()) {
    System.out.println(key + ":" + map.get(key));
}

for (String value: map.values()) {
    System.out.println(value);
}

for (Map.Entry<String, String> entry: map.entrySet()) {
    if (entry.getValue() != null)
        entry.setValue(entry.getValue().toUpperCase());
    System.out.println(entry.getKey() + ":" + entry.getValue());
}
```

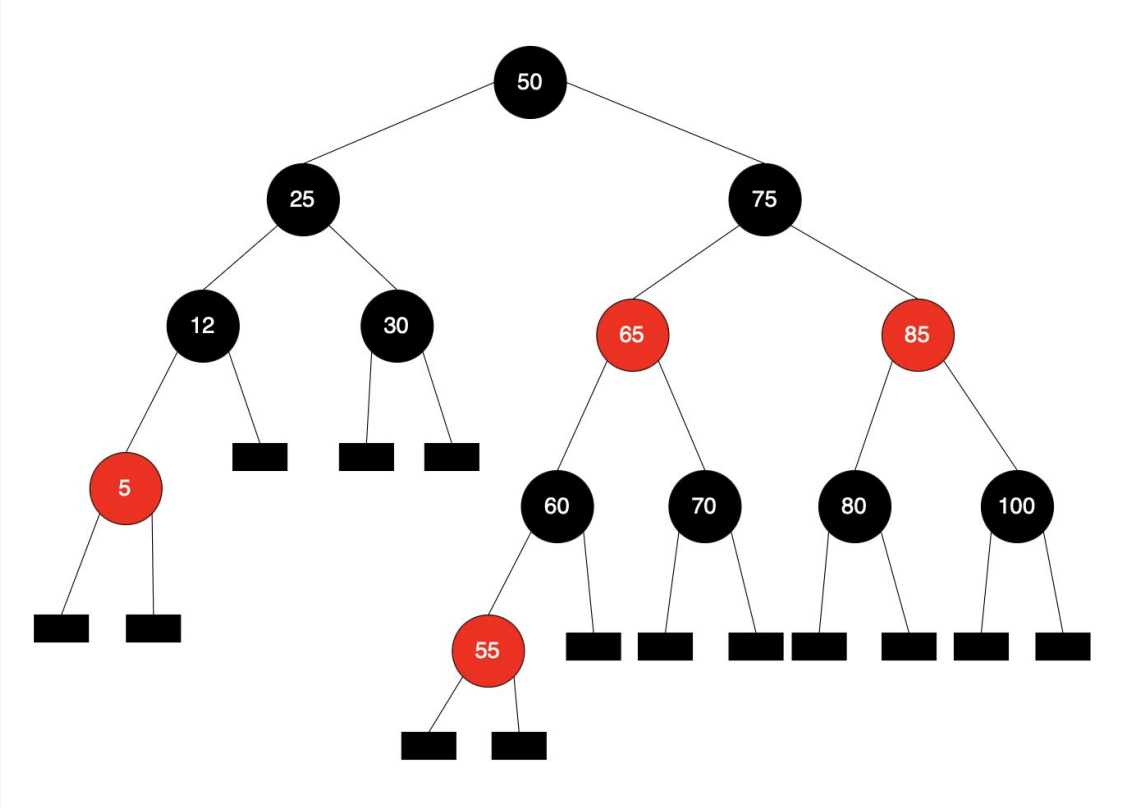
LinkedHashMap

```
Map<String, String> lhm = new LinkedHashMap<>( initialCapacity: 16,  loadFactor: 0.75f,  accessOrder: false);
lhm.put("4510755678", "Linus Torvalds");
lhm.put("4510755667", "Donald Knuth");
lhm.put("4510755434", "James Gosling");
lhm.put(null, "Anders Hejlsberg");
lhm.put("4510755322", null);

System.out.println(lhm);

Map<String, String> lhm2 = new LinkedHashMap<>( initialCapacity: 16,  loadFactor: 0.75f,  accessOrder: true);
lhm2.put("4510755678", "Linus Torvalds");
lhm2.put("4510755667", "Donald Knuth");
lhm2.put("4510755434", "James Gosling");
lhm2.put(null, "Anders Hejlsberg");
lhm2.put("4510755322", null);
lhm2.get("4510755678");

System.out.println(lhm2);
```

TreeMap

```
Map<String, String> map = new HashMap<>();
map.put("4510755678", "Linus Torvalds");
map.put("4510755667", "Donald Knuth");
map.put("4510755434", "James Gosling");
//map.put(null, "Anders Hejlsberg");
//map.put("4510755322", null);

System.out.println(map);

Map<String, String> tm = new TreeMap<>();
tm.putAll(map);

System.out.println(tm);

Map<String, String> tm2 = new TreeMap<>(Comparator.reverseOrder());
tm2.putAll(map);

System.out.println(tm2);
```

Красно-черное дерево (RB-Tree)



IdentityHashMap, WeakHashMap

WeakHashMap в качестве ключа использует WeakReference. Если на ключ больше никто не ссылается, то он чистится Garbage Collector-ом и запись удаляется из WeakHashMap.
Используется при построении разного рода кэшей.

IdentityHashMap отличается тем, что ключи сравниваются не по equals, а по “==”, а вместо сравнения по hashCode используется сравнение по identityHashCode.

Спасибо за внимание

Заполняйте анкету после активности!
Вам несложно,
Хекслету - полезно.