Map

Интерфейс Map

Map который хранит пары ключ - значение и не может содержать повторяющихся ключей.

При добавлении элемента по существующему ключу происходит запись нового элемента по ключу вместо старого.

Порядок элементов в Map зависит от реализации интерфейса.

Интерфейс Map

Всегда переопределяйте методы equals() и hashCode() для своих объектов, так как они могут использоваться в методе contains(), а также для более эффективного расположения элементов в карте, в результате чего поиск происходит быстрее.

Интерфейс Map

Интерфейс Map предоставляет три способа для доступа к данным:

* используя Set из ключей (метод map.keySet())

* коллекцию из значений (метод map.values())

* Set из пары ключ-значение (метод map.entrySet()).

Интерфейс Map

Основные **методы Map**:

- void **clear()** - удаляет все пары "ключ-значение" из вызывающего отображения
- boolean **containsKey(Object k)** - возвращает значение true, если вызывающее отображение содержит ключ k. В противном случае возвращает false.
- boolean **containsValue(Object v)** - возвращает значение true, если вызывающее отображение содержит значение v. В противном случае возвращает false
- Set<Map.Entry<K, V>> **entrySet()** - возвращает набор, содержащий все значения отображения. Набор содержит объекты интерфейса Map.Entry.

Интерфейс Map

Основные **методы Map**:

- V **get(Object k)** - возвращает значение, ассоциированное с ключом k. Возвращает значение null, если ключ не найден.
- boolean **isEmpty()** - возвращает значение true, если вызывающее отображение пусто. В противном случае возвращает false
- Set<K> **keySet()** - возвращает набор, содержащий ключи вызывающего отображения. Метод представляет ключи вызывающего отображения в виде набора
- V **put(K k, V v)** - помещает элемент в вызывающее отображение, переписывая любое предшествующее значение, ассоциированное с ключом.

Интерфейс Map

Основные **методы Map**:

- void **putAll(Map<? extends K, ? extends V> m)** - помещает все значения из m в отображение
- V **remove(Object k)** - удаляет элемент, ключ которого равен k
- **int size()** - возвращает количество пар "ключ-значение" в отображении
- Collection<V> **values()** - возвращает коллекцию, содержащую значения отображения.

Интерфейс Map

Доступ к элементам Map:

```
for (Map.Entry entry : hashMap.entrySet()) {
    System.out.println("Key: " + entry.getKey() + " Value: "
        + entry.getValue());
}
```

Интерфейс Map

Реализации интерфейса:

HashMap - структура данных для хранения связанных вместе пар "ключ-значение".
TreeMap - отображение с отсортированными ключами.
EnumMap - отображение с ключами, относящимися к перечисляемому типу
LinkedHashMap - отображение с запоминанием порядка, в котором добавлялись элементы
WeakHashMap - отображение со значениями, которые могут удаляться сборщиком мусора, если они больше не используются
IdentityHasMap - отображение с ключами, сравниваемыми с помощью операции == вместо метода equals()

HashMap

HashMap — основан на хэш-таблицах, реализует интерфейс Map (что подразумевает хранение данных в виде пар ключ/значение).
Ключи и значения могут быть любых типов, в том числе и null.
Данная реализация не дает гарантий относительно порядка элементов с течением времени.
Разрешение коллизий осуществляется с помощью метода цепочек.

HashMap. Создание объекта

Map<String, String> hashmap = new HashMap<String, String>();

Новоявленный объект hashmap, содержит ряд свойств:

- table — Массив типа Entry[]
- loadFactor — Коэффициент загрузки
- threshold — Предельное количество элементов, при достижении которого, размер хэш-таблицы увеличивается вдвое.
- size — Количество элементов HashMap-а;

HashMap. Добавление элементов

При добавлении элемента, последовательность шагов следующая:

1. Сначала ключ проверяется на равенство null. Если это проверка вернула true, будет вызван метод putForNullKey(value)
2. Далее генерируется хэш на основе ключа. Для генерации используется метод hash(hashCode), в который передается key.hashCode().

HashMap. Добавление элементов

3. С помощью метода indexFor(hash, tableLength), определяется позиция в массиве, куда будет помещен элемент.
4. Теперь, зная индекс в массиве, мы получаем список (цепочку) элементов, привязанных к этой ячейке. Хэш и ключ нового элемента поочередно сравниваются с хэшами и ключами элементов из списка и, при совпадении обоих этих параметров, значение элемента перезаписывается.

HashMap. Добавление элементов

3. С помощью метода indexFor(hash, tableLength), определяется позиция в массиве, куда будет помещен элемент.
4. Теперь, зная индекс в массиве, мы получаем список (цепочку) элементов, привязанных к этой ячейке. Хэш и ключ нового элемента поочередно сравниваются с хэшами и ключами элементов из списка и, при совпадении обоих этих параметров, значение элемента перезаписывается.

HashMap. Добавление элементов

5. Если все же в данной позиции уже есть объект (ключи разные, а hashCode совпадает), проверяется следующий атрибут. Если он возвращает null и текущий объект Entry становится следующим звеном в списке. Если следующая переменная не null, процедура повторяется для следующей, пока не найдет null.

HashMap.

1. Структура данных для хранения в объекте Entry это массив с именем table и типом Entry.
2. hashCode() Ключа требуется для вычисления позиции объекта Entry.
3. equals() Ключа используется для проверки уникальности ключа в карте(map).
4. hashCode() и equals() Значения не используется в методах get() и set() в HashMap.
5. Хеш-код для ключей со значением null это всегда 0.И такой объект Entry всегда будет храниться в нулевой позиции массива.

TreeMap

Класс TreeMap **расширяет класс AbstractMap** и **реализует интерфейс NavigatebleMap**.

Объекты сохраняются в отсортированном порядке по возрастанию.

Время доступа и извлечения элементов достаточно мало, что делает класс TreeMap блестящим выбором для хранения больших объемов отсортированной информации, которая должна быть быстро найдена.

TreeMap основан на Красно-Черном дереве, вследствие чего TreeMap сортирует элементы по ключу в естественном порядке или на основе заданного вами компаратора.

При попытке добавить null-элемент в TreeMap происходит исключение NullPointerException.