1. **Что такое Map и чем он отличается от Collection ?**

Map хранит элементы, парами “ключ - значение”. В Collection представленны в виде совокупности некоторых элементов.

1. **Что такое HashMap ?**

Ассоциативный массив. Хранит обьекты в виде пар “ключ-значение”.

Основан на хэш-таблицах, элементы не упорядочены. Не синхронизированн.

Не поддерживается реализация интерфейса Interable. Но похожая возможность работы можно организовать, применяя методы **keySet(), values(), entrySet().**

Только что созданный обьект HashMap содержит в себе:

**Массив Entry [] table** (это хранилище ссылок на цепочки значений, на связанные списки),

**loadFactor** – поле содержит коэффициент загрузки (стандартно 0.75),

**threshold** – содержит рассчитанное предельное ко-во элементов, при достижении которого происходит увеличение (x2) и переопределение hashMap (capacity \* loadFactor),

**size** - кол-во элементов hashMapa. Изначально состоит из 16 позиций(корзин).

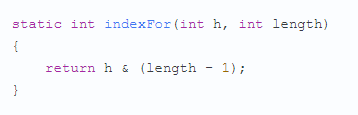
После добавления пары в массив Entry [] table, элемент будет состоять из 4 подэлементов – это **hash** (hashCode вычесленный по key), **key** (сам наш ключ в оригинале), **next** (указатель на следующий узел под данным ключом), **value** (значение, передаётся вторым параметром после key).

Можно указать емкость и коэффициент загрузки для нового HashMap, используя конструкторы HashMap(capacity) и HashMap(capacity, loadFactor). Максимальная емкость, которую можно установить, равна половине максимального значения int (1073741824).

Если добавляю null элемент – он добавляется в entry[0] позицицю.

**Порядок добавления:**

* Ключ проверяется на равенство null. Если true вызывается метод **putForNullKey(value)**;
* Генерируется хэш на основе ключа, используется метод **hash(hashCode)** в который передаётся key.hashCode(), особенность этого метода в том – что уменьшает кол-во коллиций.
* С помощью метода **indexFor (hash,tableLength)** определяется позиция из возможных (со старта 16) куда будет помещён обьект, hash сжимается с учётом tableLength и уже из возможных выбирается индекс.



* Зная индекс мы получаем цепочку элементов привязанных к этой ячейке. HashCode и key нового элемента поочерёдно сравниваются, с HashCode и key элементов из этой цепочки. Если есть совпадение – перезаписыается значение.
* Если НЕТУ – то создаётся ещё одно звено под данным индексом методом **addEntry(hash, key, value, index);**

1. **Что такое TreeMap ?**

Реализует интерфейс NavigatebleMap, SortedMap и создаёт коллекцию, которая для хранения обьектов использует “красно-чёрное дерево”. Обьекты хранятся в отсортированном порядке по возрастанию, алфавиту или по своему способу сортировки.

Не синхронизирован. Классы Integer, String, Double и т.п. реализуют интерфейс Comparable. Если вы создали **собственный** класс для ключей и не реализовали интерфейс Comparable (и не используете Comparator), то при попытке добавления объекта в коллекцию будет брошено исключение java.lang.ClassCastException!

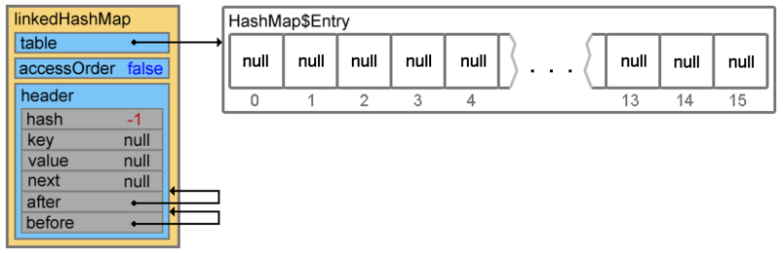
Частая ошибка при использовании TreeMap, в том, что для **класса ключа** не определяется метод hashcode(). Если в данном случае использовать метод map.get(new MyCustomKey()), то результат может быть непредсказуем. Поэтому всегда рекомендуется реализовать метод hashCode() для класса ключей.

1. **Что такое LinkedHashMap ?**

Хранит элементы типа “ключ-значение” в том порядке в котором были добавлены.

Принцип работы такой же как и у HashMap, плюс – каждый элемент содержит ссылку на следующий и предыдущий элемент.

Только что созданный **LinkedHashMap** содержит помимо всего имеющегося в HashMap ( table, loadFactory, threshold, size) ещё и **header** (голова двухсвязного списка, при инициал – указывает на самого себя) , **accessOrder** ( указывает каким образом будет осуществляться доступ к элементам при использовании итератора, по порядку добавления или по порядку доступа к элементам(к кому чаще достукивались – в конец, чем реже – в начало)) используя методы get и put. Метод removeEldestEntry() может быть преопределён, к примеру для построения кэширующий структур на основе Map, после того как он начнёт возвращать true – самый старый элемент будет удалён при достижении максимального размера коллекции.



1. **Что может быть ключем в Map ?**

Null значение, ссылочные типы, классы обёртки

1. **Почему String лучше использовать в качестве ключа в Map чем другой объект ?**

Строки неизменны, и hashCode кэшируется в момент их создания, и следовательно не пересчитывается, обрабатываются они быстрее чем другие варианты в качестве ключа.

1. **Могу ли я добавить null как ключ в Map ?**

Да можно

При создании пары, первым делом выполняется проверка на null значение и вызывается соответствующий метод.

1. **Могу я добавить дубликат в ключ в Map ? Почему ? Что будет если я попытаюсь добавить дубликат в ключ**

Если ключи будут одинаковы – то будут сравнены ключи и хэшкод, если совподёт – то значение перезапишется. Но дубликат ключа не создаётся.

1. **Могу я добавить дубликат в значение в Map ? Почему ?**

Дубликат значения можно добавить, но при этом должны попасть по одному ключу, что бы хэшкод совпадал а ключи отличались.

1. **Могу я добавить null в значение в Map ?**

Сколько угодно null значений можно добавлять ( уточнение 2 ?!)

1. **Могу я использовать HashMap в многопоточности ?**

Нет нельзя, HashMap не синхронизирован.

1. **Сколько корзин по умолчанию содержится в HashMap ?**

16 корзин

1. **Как добавить элемент в Map ?**

Передать значения в метод put(), putAll(), putIfAbsent().

1. **Как сделать перебор по всем ключам в Map ? Какой метод отвечает за это ?**

Есть метод keyset() возвращает множество ключей, их можно пропустить через foreach

1. **Как сделать перебор по всем значениям в Map ? Какой метод отвечает за это ?**

Метод values() – вернёт коллекцию, по которой точно так же можно пробежаться

Но не факт что определённому элементу будет соответствовать какой то конкретный ключ.

1. **Как проверить ключ и значение на наличие в Map ?**

Методы containsKey и containsValue – возвращают true или false в зависимости есть или нету переданный ключ\значение в параметры данных методов.

1. **Напишите цикл перебора по всем ключам и значениям одновременно? Какой метод для этого используется ?**

Существует метод entrySet() – который вернёт множество пар ключ-значение

Set <Map.Entry <Integer, String>> set = map.entrySet();

/100% уверенность что к определённому ключу - принадлежит определённое значение:

**for**(Map.Entry <Integer, String> entry : set) {

**int** key = entry.getKey();

value = entry.getValue();

System.***out***.println("key: " + key + ", value: " + value);

}

1. **Как в HashMap хранятся пары ключ-значение ?**

В массиве типа Entry

1. **Что происходит, когда корзины заполняются ? (опишите 2 метода + какой становится размер корзин)**

Если был достигнут максимально допустимый размер коллекции (он рассчитывается как **size** \* **loadFactor**) – то происходит увеличение вдвое и перераспределение элементов HashMap, запускается метод **resize()** который отвечает за создание большей HashMap и метод transfer() – который перебирает элементы из старой HashMap и помещает в новый HM с учётом нового размера. Размер с 16 ячеек в 32 ячейки после достижения в начальной (с 16) 12 элементов.

1. **Меняется ли размер HashMap при удалении элементов ?**

Размер не изменяется.

1. **Сложность добавления в начало, середину и конец для HashMap, TreeMap и LinkedHashMap.**

**HashMap** – в лучшем случае если нет коллизий и хэш-функция хорошо распределила, удаление, поиск за O(1). Добавление за всегда за O(1). В худшем, когда все значения попадают по одному ключу – O(n).

Хорошо если зададим начальный размер, если знаем сколько будет элементов, тк в таком случае не будет каждый раз удваиваться HashMap (если вставляем сразу много элементов).

**LinkedHashMap** - выполнение операций add(), contains(), remove() будет за время — O(1).И уступает по производительности родительскому классу HashMap. Плюс дополнительное место в памяти для хранения элементов и их связей. Все остальное схоже с hashMap.

**TreeMap** – хранится в виде “красно-чёрного дерева”, гарантирует скорость доступа log (n) для операций containsKey, get, put и remove.

1. **Куда по умолчанию добавляются новые элементы в HashMap ? (середина, начало, конец)**Начало цепочки.
2. **Как работает HashMap ? (1 слово)**

Entry. ???

1. **Можно добавить null в TreeMap ?**

При попытке добавить null элемент в TreeMap вылетит исключение типа NullPointerException