**VI. Сравнение стандартных и альтернативных коллекций**

5.1 Какие вообще бывают коллекции

Для начала рассмотрим какие основные коллекции встречаются в программировании (не только в Java, а вообще):  
1) **Вектор (Список)**— элементы коллекции упорядочены, можно обойти все элементы по очереди или обратиться по индексу,  
— *Массив* — реализация вектора, когда данные находятся в памяти непосредственно друг за другом  
— *Динамический массив* — реализация массива, когда размер массива может увеличиваться во время выполнения,  
— *Односвязный список* — реализация списка, когда каждый элемент списка содержит значение и ссылку на  
следующий элемент, в отличии от массива более структурно гибок,  
— *Двусвязный список* — реализация списка когда каждый элемент списка содержит значение и ссылку на  
следующий и предыдущий элемент,  
2) **Стек (Stack)** — коллекция, реализующая принцип хранения «LIFO» («последним пришёл — первым вышел»). В  
стеке постоянно доступен элемент добавленый последним, если он ещё не удален/извлечен.  
3) **Очередь (Queue)** — коллекция, реализующая принцип хранения «FIFO» («первым пришёл— первым вышел»). В  
очереди постоянно доступен только добавлен самым первым из имеющихся и ещё не удален/извлечен.  
4) **Двухсторонняя очередь (Double-ended queue)** — очередь, которая позволяет добавлять и извлекать данные и  
с начала очереди и с конца.  
5) **Очередь с приоритетом (англ. priority queue)** — очередь, позволяющая добавить новый элемент и извлечь  
максимум. Все данные хранятся в порядке убывания приоритета,  
— *куча (heap)* — одна из реализаций очереди с приоритетом, с помощью дерева,  
6) **Ассоциативный массив (словарь), (Associative array, Dictionary)** — неупорядоченная коллекция, хранящая  
пары «ключ— значение»  
— *Хеш-таблица (hashtable)* — реализация ассоциативного массива, построенная на вычислении хеша значения,  
— *Хеш-таблица со связями один ко многим (Multimap или multihash)* — реализация Хеш-таблицы, которая хранит  
отношение ключ и много значений,  
— *Двух-сторонняя хеш-таблица (bi-map)* — реализация Хеш-таблицы, которая позволяет получать как значение  
по ключу, так и ключ по значению,  
— *Упорядоченная хеш-таблица (hashtable)* — хеш-таблица, возвращающая элементы в порядке добавления,  
— *Отсортированная хеш-таблица (hashtable)* — хеш-таблица, возвращающая элементы отсортированном порядке,   
7) **Множество**— неупорядоченная коллекция, хранящая набор уникальных значений и поддерживающая аналогичные  
операциям с математическими множествами,  
— *Мультимножество*— неупорядоченная коллекция, аналогичная множеству, но допускающая наличие в коллекции  
одновременно двух и более одинаковых значений,  
— *Упорядоченное множество*— коллекция, аналогичная множеству, но возвращает элементы в порядке добавления,  
— *Отсортированное множество*— коллекция, аналогичная множеству, но возвращает элементы в отсортированном порядке,   
8)**Битовый массив** — то есть массив значений 1 или 0,  
9) **Множество зарытых или открытых отрезков** — то есть структура хранящая и работающая с геометрическими  
интервалами,  
10) **Деревья** — структура данных, хранящая данные в виде дерева,  
11) **Кеши** — коллекции для работы с устаревающими за определенное время данными,  
  
Давайте посмотрим какие из данных сущностей соответствуют каким коллекциям и интерфейсам Java и альтернативных библиотек:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Название** | **List** | **Set** | **Map** | **Query Dequery и т.п. в JDK** | **guava** | **apache** | **gs-collections** |
| 1) **Вектор (Список)** *Динамический массив* | [ArrayList](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html) | [LinkedHashSet](https://docs.oracle.com/javase/8/docs/api/java/util/LinkedHashSet.html) | [LinkedHashMap](https://docs.oracle.com/javase/8/docs/api/java/util/LinkedHashMap.html) | [ArrayDeque](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayDeque.html) |  | [TreeList](http://commons.apache.org/proper/commons-collections/javadocs/api-release/org/apache/commons/collections4/list/TreeList.html) | [FastList](http://www.goldmansachs.com/gs-collections/javadoc/6.1.0/com/gs/collections/impl/list/mutable/FastList.html) |
| *Двусвязный список* | [LinkedList](https://docs.oracle.com/javase/8/docs/api/java/util/LinkedList.html) | [LinkedHashSet](https://docs.oracle.com/javase/8/docs/api/java/util/LinkedHashSet.html) | [LinkedHashMap](https://docs.oracle.com/javase/8/docs/api/java/util/LinkedHashMap.html) | [LinkedList](https://docs.oracle.com/javase/8/docs/api/java/util/LinkedList.html) |  | [NodeCachingLinkedList](http://commons.apache.org/proper/commons-collections/javadocs/api-release/org/apache/commons/collections4/list/NodeCachingLinkedList.html) |  |
| 2) **Стек (Stack)** | [LinkedList](https://docs.oracle.com/javase/8/docs/api/java/util/LinkedList.html) |  |  | [ArrayDeque](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayDeque.html) |  |  | [ArrayStack](http://www.goldmansachs.com/gs-collections/javadoc/6.1.0/com/gs/collections/impl/stack/mutable/ArrayStack.html) |
| 3) **Очередь (Queue)** | [LinkedList](https://docs.oracle.com/javase/8/docs/api/java/util/LinkedList.html) |  |  | [ArrayDeque](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayDeque.html) |  | [CircularFifoQueue](http://commons.apache.org/proper/commons-collections/javadocs/api-release/org/apache/commons/collections4/queue/CircularFifoQueue.html) |  |
| 4) **Двухсторонняя очередь** | [LinkedList](https://docs.oracle.com/javase/8/docs/api/java/util/LinkedList.html) |  |  | [ArrayDeque](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayDeque.html) |  |  |  |
| 5) **Очередь с приоритетом** *куча (heap)* |  |  |  | [PriorityQueue](https://docs.oracle.com/javase/8/docs/api/java/util/PriorityQueue.html) |  |  |  |
| 6) **Ассоциативный массив (словарь)** |  |  | [HashMap](https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html) |  |  | [HashedMap](http://commons.apache.org/proper/commons-collections/javadocs/api-release/org/apache/commons/collections4/map/HashedMap.html) | [UnifiedMap](http://www.goldmansachs.com/gs-collections/javadoc/6.1.0/com/gs/collections/impl/map/mutable/UnifiedMap.html) |
| *Хеш-таблица (hashtable)* |  | [HashSet](https://docs.oracle.com/javase/8/docs/api/java/util/HashSet.html) | [HashMap](https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html) |  |  | [HashedMap](http://commons.apache.org/proper/commons-collections/javadocs/api-release/org/apache/commons/collections4/map/HashedMap.html) | [UnifiedMap](http://www.goldmansachs.com/gs-collections/javadoc/6.1.0/com/gs/collections/impl/map/mutable/UnifiedMap.html) |
| *Хеш-таблица со связями один ко многим* |  |  |  |  | [Multimap](http://google.github.io/guava/releases/snapshot/api/docs/com/google/common/collect/Multimap.html) | [MultiMap](http://commons.apache.org/proper/commons-collections/javadocs/api-release/org/apache/commons/collections4/MultiMap.html) | [Multimap](http://www.goldmansachs.com/gs-collections/javadoc/6.1.0/com/gs/collections/api/multimap/Multimap.html) |
| *Двух-сторонняя хеш-таблица* |  |  |  |  | [HashBiMap](http://google.github.io/guava/releases/snapshot/api/docs/com/google/common/collect/HashBiMap.html) | [BidiMap](http://commons.apache.org/proper/commons-collections/javadocs/api-release/org/apache/commons/collections4/BidiMap.html) | [HashBiMap](http://www.goldmansachs.com/gs-collections/javadoc/6.1.0/com/gs/collections/impl/bimap/mutable/HashBiMap.html) |
| *Упорядоченная хеш-таблица* |  | [LinkedHashSet](https://docs.oracle.com/javase/8/docs/api/java/util/LinkedHashSet.html) | [LinkedHashMap](https://docs.oracle.com/javase/8/docs/api/java/util/LinkedHashMap.html) |  |  | [LinkedMap](http://commons.apache.org/proper/commons-collections/javadocs/api-release/org/apache/commons/collections4/map/LinkedMap.html) |  |
| *Отсортированная хеш-таблица* |  | [TreeSet](https://docs.oracle.com/javase/8/docs/api/java/util/TreeSet.html) | [TreeMap](https://docs.oracle.com/javase/8/docs/api/java/util/TreeMap.html) |  |  | [PatriciaTrie](http://commons.apache.org/proper/commons-collections/javadocs/api-release/org/apache/commons/collections4/trie/PatriciaTrie.html) | [TreeSortedMap](http://www.goldmansachs.com/gs-collections/javadoc/6.1.0/com/gs/collections/impl/map/sorted/mutable/TreeSortedMap.html) |
| 7) **Множество** | [HashSet](https://docs.oracle.com/javase/8/docs/api/java/util/HashSet.html) |  |  |  |  |  | [UnifiedSet](http://www.goldmansachs.com/gs-collections/javadoc/6.1.0/com/gs/collections/impl/set/mutable/UnifiedSet.html) |
| *Мультимножество* |  |  |  |  | [HashMultiset](http://google.github.io/guava/releases/snapshot/api/docs/com/google/common/collect/HashMultiset.html) | [HashBag](http://commons.apache.org/proper/commons-collections/javadocs/api-release/org/apache/commons/collections4/bag/HashBag.html) | [HashBag](http://www.goldmansachs.com/gs-collections/javadoc/6.1.0/com/gs/collections/impl/bag/mutable/HashBag.html) |
| *Упорядоченное множество* |  | [LinkedHashSet](https://docs.oracle.com/javase/8/docs/api/java/util/LinkedHashSet.html) |  |  |  |  |  |
| *Отсортированное множество* |  | [TreeSet](https://docs.oracle.com/javase/8/docs/api/java/util/TreeSet.html) |  |  |  | [PatriciaTrie](https://commons.apache.org/proper/commons-collections/javadocs/api-release/org/apache/commons/collections4/trie/PatriciaTrie.html) | [TreeSortedSet](http://www.goldmansachs.com/gs-collections/javadoc/6.1.0/com/gs/collections/impl/set/sorted/mutable/TreeSortedSet.html) |
| 8)**Битовый массив** |  |  |  | [BitSet](http://docs.oracle.com/javase/8/docs/api/java/util/BitSet.html) |  |  |  |
| 9) **Множество зарытых или открытых отрезков** |  |  |  |  | [RangeSet](http://google.github.io/guava/releases/snapshot/api/docs/com/google/common/collect/RangeSet.html) [RangeMap](http://google.github.io/guava/releases/snapshot/api/docs/com/google/common/collect/RangeMap.html) |  |  |
| 10) **Деревья** |  | [TreeSet](https://docs.oracle.com/javase/8/docs/api/java/util/TreeSet.html) | [TreeMap](https://docs.oracle.com/javase/8/docs/api/java/util/TreeMap.html) |  |  | [PatriciaTrie](https://commons.apache.org/proper/commons-collections/javadocs/api-release/org/apache/commons/collections4/trie/PatriciaTrie.html) | [TreeSortedSet](http://www.goldmansachs.com/gs-collections/javadoc/6.1.0/com/gs/collections/impl/set/sorted/mutable/TreeSortedSet.html) |
| 11)**Кеши** |  |  | [LinkedHashMap](https://docs.oracle.com/javase/8/docs/api/java/util/LinkedHashMap.html) [WeakHashMap](https://docs.oracle.com/javase/8/docs/api/java/util/WeakHashMap.html) |  | [LoadingCache](http://docs.guava-libraries.googlecode.com/git/javadoc/com/google/common/cache/LoadingCache.html) |  |  |

VII. Заключение

Я специально не буду делать выводов какая библиотека хуже или лучше, так как во многом это дело вкуса, но в любом случае в альтернативных коллекциях можно найти много полезных коллекций и методов, если знать где искать. Спасибо, за то что дочитали (или долистали) до конца, надеюсь вы сумели найти в этой статье что-нибудь для себя полезное.  
  
Исходные коды всех примеров можно найти на [github'е](https://github.com/Vedenin/java_in_examples).  
  
**Источники, которые использовались для написания статьи**:  
1. [Обзор java.util.concurrent.\* tutorial](http://habrahabr.ru/company/luxoft/blog/157273/)  
2. [Trove library: using primitive collections for performance](http://java-performance.info/primitive-types-collections-trove-library/)  
3. [Java performance tuning tips](http://java-performance.com/)  
4. [Large HashMap overview](http://java-performance.info/hashmap-overview-jdk-fastutil-goldman-sachs-hppc-koloboke-trove-january-2015/)  
5. [Memory consumption of popular Java data types](http://java-performance.info/memory-consumption-of-java-data-types-1/)  
6. И, естественно, официальная документация, javadoc и исходные коды всех рассмотренных библиотек  
  
**P.P.S.** Так же советую посмотреть мой opensource проект [useful-java-links](https://github.com/Vedenin/useful-java-links/tree/master/link-rus) — возможно, наиболее полная коллекция полезных Java библиотек, фреймворков и русскоязычного обучающего видео. Так же есть аналогичная [английская версия](https://github.com/Vedenin/useful-java-links/) этого проекта и начинаю opensource подпроект [Hello world](https://github.com/Vedenin/useful-java-links/tree/master/helloworlds) по подготовке коллекции простых примеров для разных Java библиотек в одном maven проекте (буду благодарен за любую помощь).

Общее оглавление 'Шпаргалок'

Только зарегистрированные пользователи могут участвовать в опросе. [Войдите](https://habr.com/ru/auth/login/), пожалуйста.

Какие библиотеки альтернативных коллекций вы используете в своей работе?

Начало формы

* Guava
* Apache Commons Collections
* Trove
* GS-collections
* Fastutil
* HPPC
* Koloboke
* GlazedLists collections
* Concurrent Building Blocks
* Functional Java
* Primitive Collections for Java
* Apache Jakarta Commons Primitives
* Chronicle-Queue
* Использую свои реализации коллекций
* Использую только стандартный фреймворк коллекций JDK
* Другая библиотека (напишите, пожалуйста какая в комментариях)

Конец формы

Проголосовал 421 пользователь. Воздержались 212 пользователей.