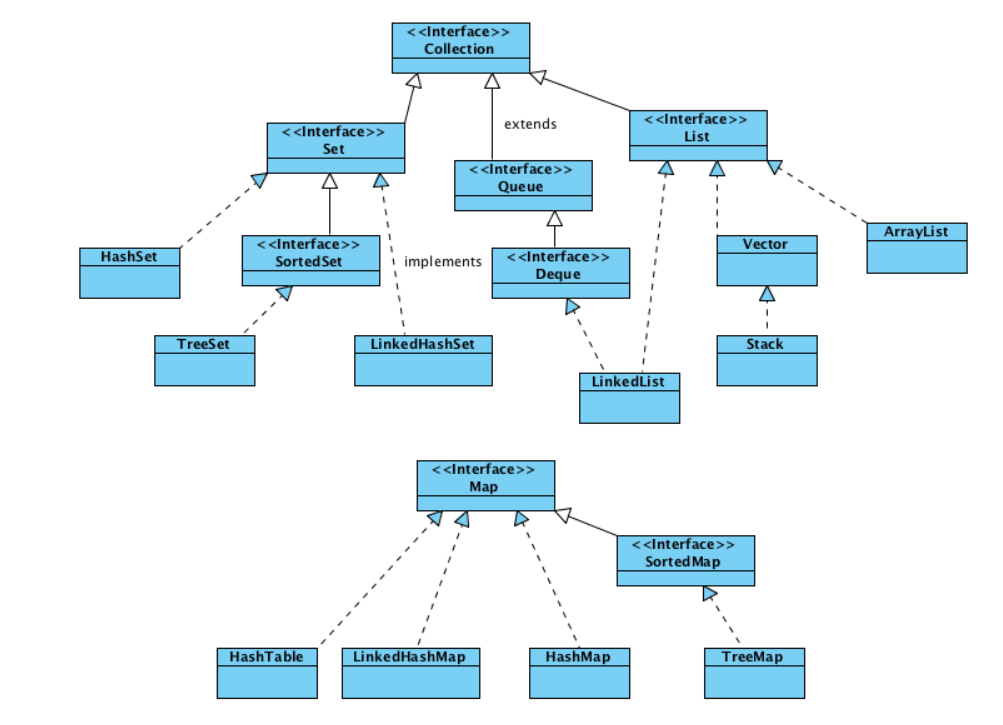
**Java Collection**

1. Основные интерфейсы коллекций и их иерархия (List, Set, Queue). Какие бывают коллекции? В чём особенности разных видов коллекций? Когда какие стоит применять?

Иерархия коллекций



* **java.util.List** — интерфейс для операций с коллекцией, которая является списком. Список обладает следующими важными признаками:
  1. Список может включать одинаковые элементы
  2. Элементы в списке хранятся в том порядке, в котором они туда помещались. Самопроизвольных перемещений элементов в нем не происходит — только с вашего ведома. Например, вы можете добавить элемент на какую-то позицию и тогда произойдет сдвиг других элементов.
  3. Можно получить доступ к любому элементу по его порядковому номеру/индексу внутри списка

Т.е. если вам требуется, чтобы коллекция обладала такими свойствами — выбирайте класс, который реализует интерфейс **java.util.List**

* **java.util.Set** — интерфейс для хранения множества. В отличии от **java.util.List** этот интерфейс как раз не может иметь одинаковые элементы и порядок хранения элементов в множестве может меняться при добавлении/удалении/изменении элемента. Может возникнуть вопрос, зачем такиая коллекция нужна — это удобно в случае, когда вы создаете набор уникальных элементов из какой-то группы элементов
* **java.util.Queue** — интерфейс предлагает работать с коллекцией как с очередью, т.е. коллекция имеет метод для добавления элементов в один конец и метод для получения элемента с другого конца — т.е. настоящая очередь по принципу FIFO — First In First Out — если первым пришел, то первым и уйдешь. Для широкого круга задач такая конструкция работы с коллекцией бывает достаточно удобной структурой.

1. Что такое интерфейс Map? Представляет ли он коллекцию?

Интерфейс **Map<K, V>** представляет отображение или иначе говоря словарь, где каждый элемент представляет пару "ключ-значение". При этом все ключи уникальные в рамках объекта Map. Такие коллекции облегчают поиск элемента, если нам известен ключ - уникальный идентификатор объекта.

Map не является Collection. Так как Map - это совокупность пар "ключ"-"значение". Соответственно некоторые методы интерфейса Collection нельзя использовать в Map. Например, метод remove(Object o) в интерфейсе Collection предназначен для удаления элемента, тогда как такой же метод remove(Object key) в интерфейсе Map - удаляет элемент по заданному ключу.

1. Сравнить ArrayList и LinkedList.

Отличие заключается в способе хранения данных. ArrayList хранит в виде массива, а LinkedList - в виде списка (двунаправленного).  
В ArrayList быстрее происходит сортировка, т.к. для ее выполнения данные списка копируются в массив (а копировать из массива ArrayList в массив для сортировки быстрее). При большом числе операций добавления и удаления LinkedList должен быть более удачным выбором, т.к. при этих операциях не приходится перемещать части массива.  
Если при добавлении в ArrayList превышается его объем, размер массива увеличивается, новая емкость рассчитывается по формуле (oldCapacity \* 3) / 2 + 1, поэтому лучше указывать размер при создании или, если он не известен, использовать LinkedList (но это может быть существенно при слишком уж больших объемах данных).

1. Сравнить HashMap и Hashtable.

Некоторые методы HashTable синхронизированы (могут быть выполнены только одним потоком одновременно), поэтому она медленнее HashMap.

Различия (кратко):

* HashTable синхронизирована, а HashMap нет.
* HashTable не позволяет иметь null ключи или значения. HashMap позволяет иметь один null ключ и сколько угодно null значений.
* У HashMap есть подкласс LinkedHashMap, который добавляет возможности по итерации. Если вам нужна эта функциональность, то можно легко переключаться между классами.

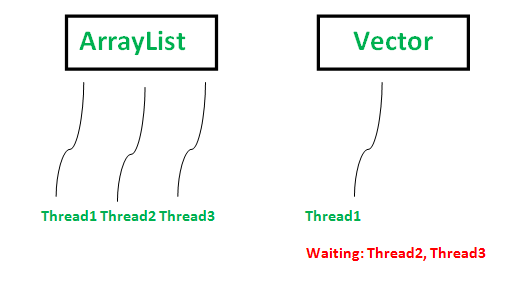
Общее замечание – не рекомендуется использовать HashTable даже в многопоточных приложениях. Для этого есть ConcurrentHashMap.

1. Как устроены HashSet, TreeMap, TreeSet?

Класс HashSet реализует интерфейс Set, основан на хэш-таблице, а также поддерживается с помощью экземпляра HashMap. В HashSet элементы не упорядочены, нет никаких гарантий, что элементы будут в том же порядке спустя какое-то время. Операции добавления, удаления и поиска будут выполняться за константное время при условии, что хэш-функция правильно распределяет элементы по «корзинам», о чем будет рассказано далее.

Несколько важных пунктов о HashSet:

* Т.к. класс реализует интерфейс Set, он может хранить только уникальные значения;
* Может хранить NULL – значения;
* Порядок добавления элементов вычисляется с помощью хэш-кода;
* HashSet также реализует интерфейсы Serializable и Cloneable.

1. Принцип работы и реализации HashMap. Изменения HashMap в java8.
2. Чем отличается ArrayList от Vector?
3. **Синхронизация :** Vector синхронизируется**,**что означает, что только один поток одновременно может получить доступ к коду, в то время как arrayListнесинхронизирован**,**что означает, что несколько потоков могут работать в arrayList одновременно. Например, если один поток выполняет операцию добавления, то может быть другой поток, выполняющий операцию удаления в многопоточной среде.  
   [](https://media.geeksforgeeks.org/wp-content/uploads/ArrayList-vs-Vector-Java.png)
4. **Производительность:** ArrayListбыстрее**,**так как он не синхронизирован, в то время как векторные операции дают более медленную производительность, поскольку они синхронизированы (потокобезопасны).
5. **Рост данных:** ArrayList и Vector растут и уменьшаются динамически**,**чтобы поддерживать оптимальное использование хранилища, но способ их изменяемого размера отличается. ArrayList увеличивается на 50% от текущего размера массива, если количество элементов превышает его емкость, в то время как вектор увеличивается на 100%, что по существу удваивает текущий размер массива.
6. **Траверс:** Vector может использовать как [перечисление (быстрее), так и итератор](https://www.geeksforgeeks.org/iterators-in-java/)для обхода элементов вектора, в то время как ArrayList может использовать только итератор для обхода.
7. **Области применения :** В большинстве случаев программисты предпочитают ArrayList векторному, поскольку [ArrayList можно явно синхронизировать с помощью Collections.synchronizedList.](https://www.geeksforgeeks.org/synchronization-arraylist-java/)

*Примечание: ArrayList предпочтительнее, когда нет особых требований к использованию вектора.*

1. Особенности интерфейса Set.

Уникальность элементов.

90. Как добавляются объекты в HashSet?

**Добавление**

Когда приходит новый элемент, **хэш-код (**целое число, которое является уникальным идентификатором содержимого объекта) ключа определяет корзину для элемента. В корзине идет проверка, есть ли у нее элементы. Если нету, то корзина получает ссылку нового элемента, если есть, то происходит прохождение по списку элементов и сравнивание элементов в списке. Проверяется равенство hashcode. Так как мы не можем однозначно судить на счет эквивалентности, зная о случаях коллизии, проводится еще сравнивание ключей методом equals.

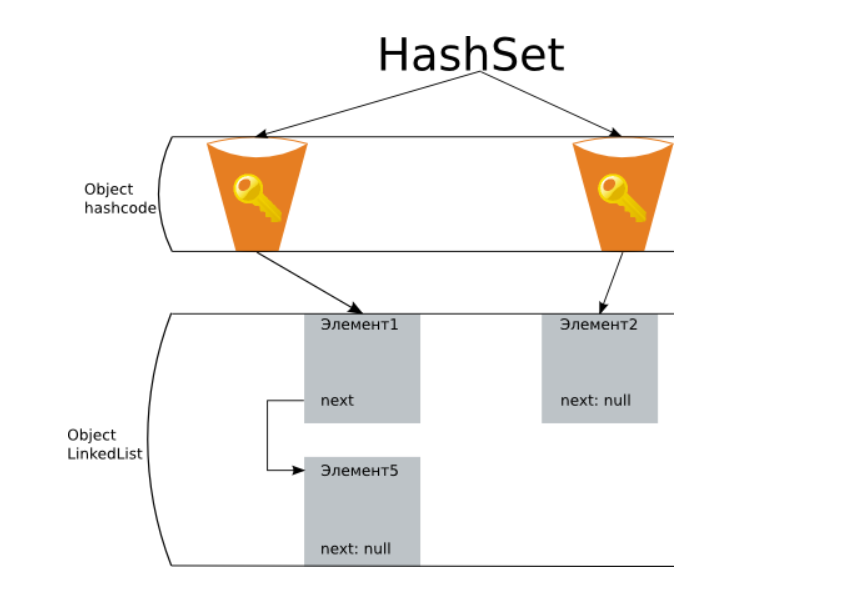
 (ключом является сам объект)

**Если оба равны**: идет перезапись

**Если не равен equals**: добавляется элемент в список

**Поэтому:**  
1. Если у нас плохой алгоритм расчета хэш-кода, то мы получим обычный связной список и потеряем все преимущества данной структуры. (Добавление, поиск и удаление элементов выполняется за константное время)

2. Важно переопределять метод hashCode & equals для корректной работы.



91. Какими способами можно отсортировать коллекцию? (привести три способа)

92. Как правильно удалить элемент из коллекции при итерации в цикле?

93. Как правильно удалить элемент из ArrayList (или другой коллекции) при поиске этого элемента в цикле?

94.Коллекции из пакета concurrent. Их особенности.

95.Что происходит при добавлении в ArrayList нового элемента и как это реализовано.

96.Thread-safe and non-thread safe collections.

97.Метод для преобразования потоко-небезопасной коллекции в потоко-безопасную.

98.Написать метод в котором проверяется HashMap на наличие в нем некоторого значения, и его извлечения, если такого значения нет, надо добавить значение с пустой строкой и её вернуть. Написать код, чтобы он был как можно более эффективным (меньше затратных действий).

99.Какие потокобезопасные коллекции более «быстрые» – legacy(Vector, HashTable) или из пакета concurrent?

100.Если в коллекцию часто добавлять элементы, и удалять, какую лучше использовать? Почему? Как они устроены?

101.Как быстро получить копию коллекции. Записать код преобразования.