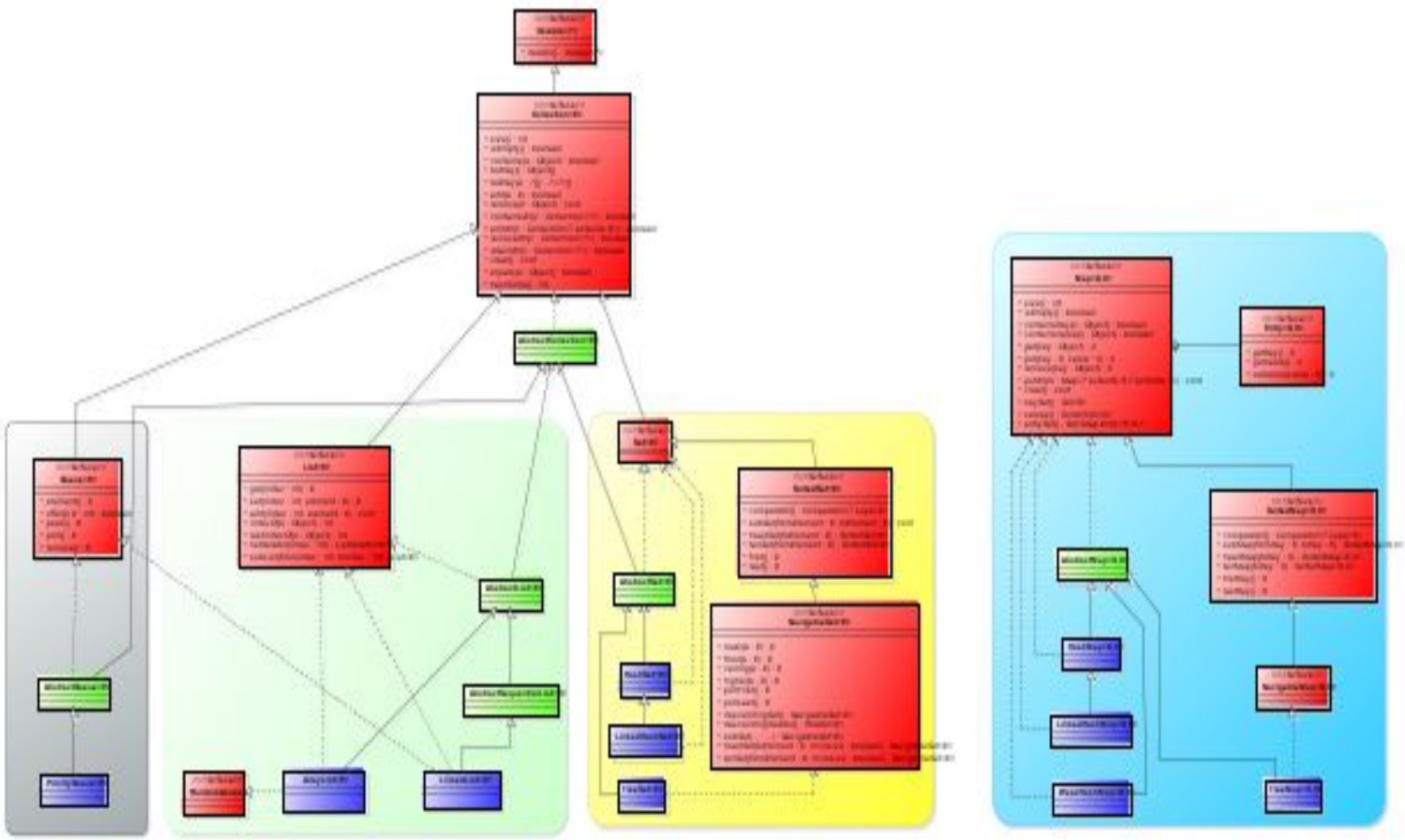
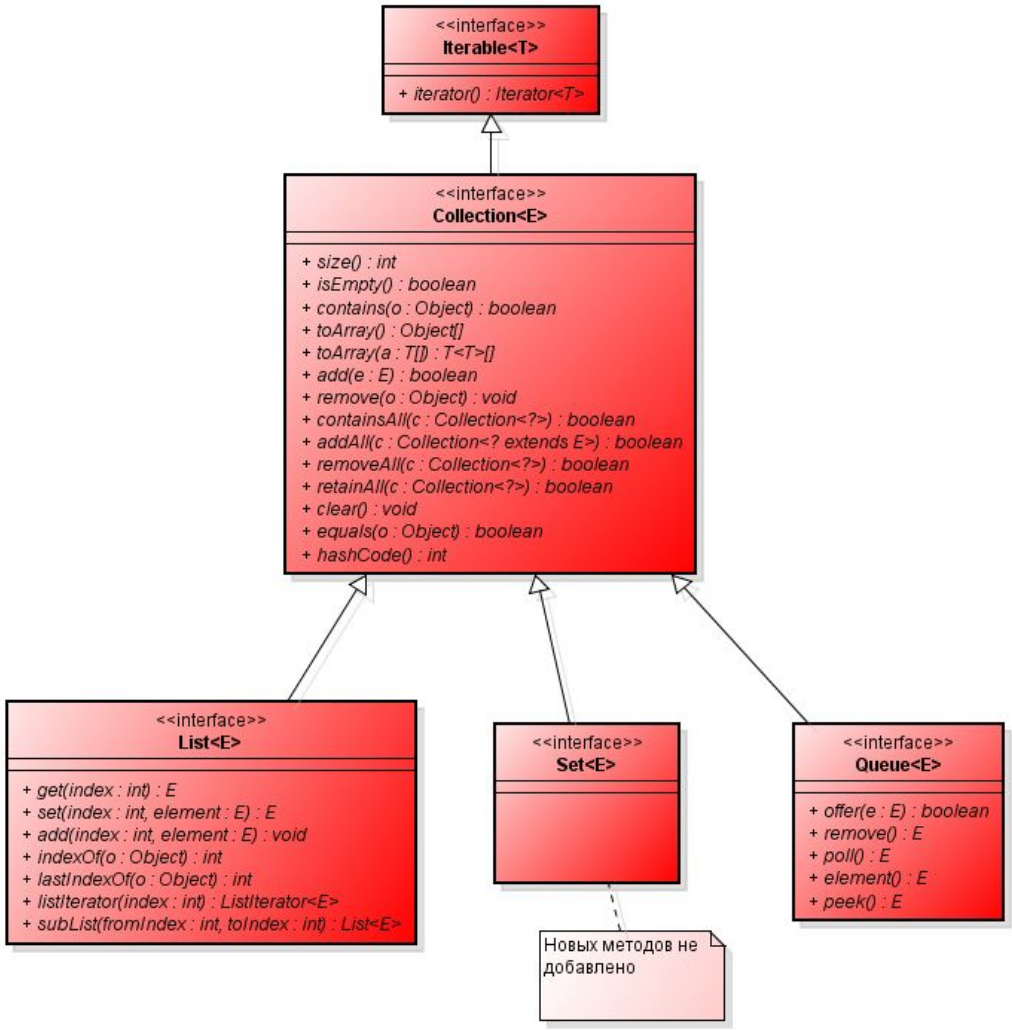
Java Collection



Collection



Интерфейс **Collection** расширяет интерфейс **Iterable**, у которого есть только один метод iterator().
Итератор- объект, абстрагирующийся за единым интерфейсом доступ к элементам коллекции (без вникания в суть ее реализации).

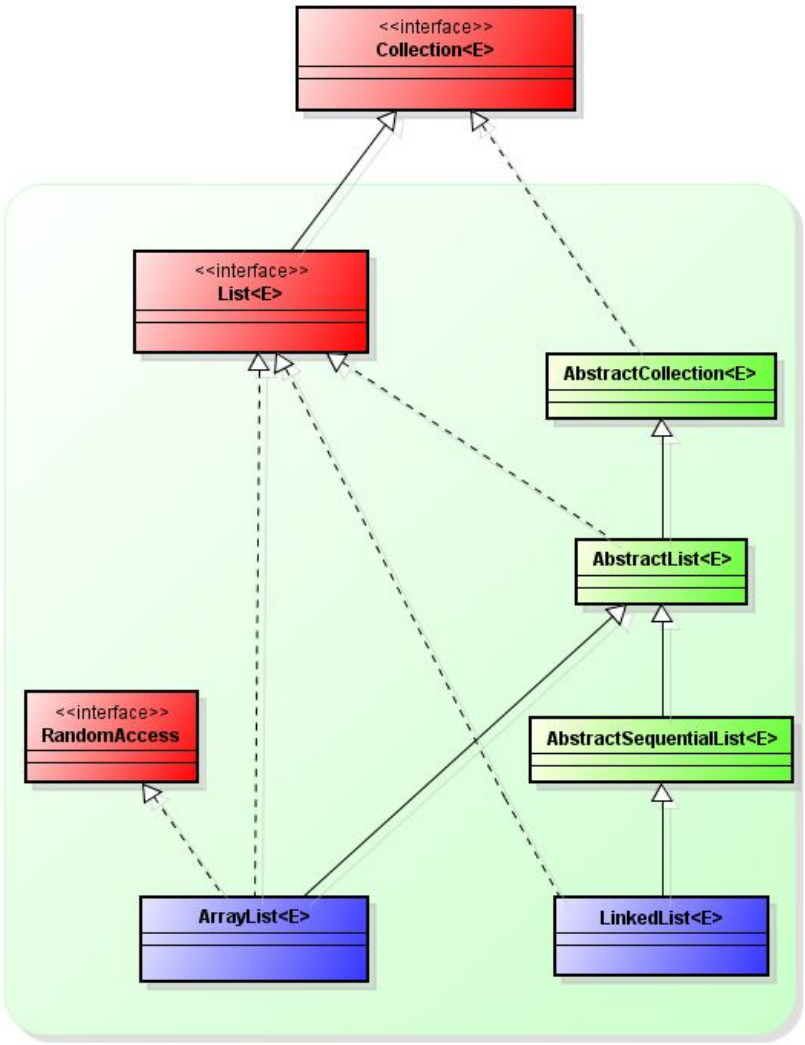
1. **List** - Представляет собой неупорядоченную коллекцию, в которой допустимы дублирующие значения. Элементы такой коллекции пронумерованы, начиная от нуля, к ним можно обратиться по индексу.
2. **Set** - описывает неупорядоченную коллекцию, не содержащую повторяющихся элементов.
3. **Queue** - очередь. Это коллекция, предназначенная для хранения элементов в порядке, нужном для их обработки. В дополнение к базовым операциям интерфейса

интерфейс Collection

Данный интерфейс является базовым для всех "коллекционных" интерфейсов.

- ***add***(E e) - добавить элемент;
- ***addAll***(Collection<?extends E> c) - добавить элементы из другой коллекции;
- ***clear***() - удалить все элементы;
- ***contains***(Object o) - находится ли указанный объект в коллекции;
- ***containsAll***(Collection<?> c) - содержатся ли указанные объекты в коллекции;
- ***equals***(Object o) - сравнение коллекции с указанным объектом;
- ***isEmpty***() - является ли коллекция пустой;
- ***iterator***() - возвращает итератор для прохода по элементам этой коллекции;
- ***remove***(Object o) - удаляет указанный объект из коллекции, если он есть там;
- ***removeAll***(Collection<?> c) - удалить указанные объекты;
- ***retainAll***(Collection<?> c) - оставить в коллекции только указанные объекты;
- ***size***() - размер коллекции в элементах;
- ***toArray***() - возвращает массива объектов, содержащий все элементы коллекции;
- ***toArray***(T[] a) - возвращает массива объектов, содержащий все элементы коллекции. Если аргумента a null, то создается новый массив в который копируются элементы.

List



1. **ArrayList** - инкапсулирует в себе обычный массив, длина которого автоматически увеличивается при добавлении новых элементов. Время доступа к элементу по индексу минимально.

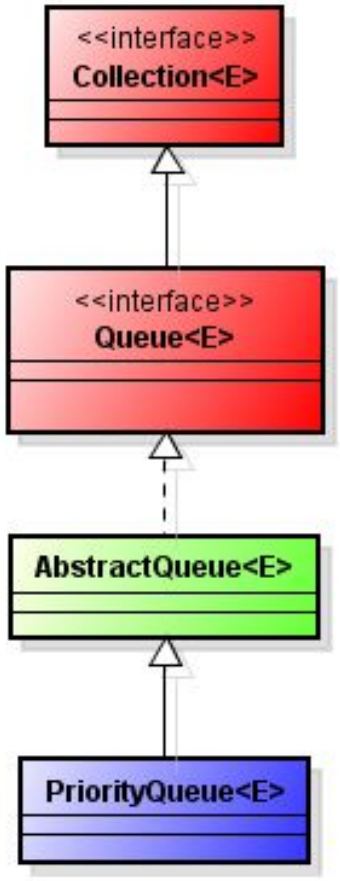
2. **LinkedList** - структура данных, состоящая из узлов, каждый из которых содержит как собственно данные, так и две ссылки («связки») на следующий и предыдущий узел списка. Доступ к произвольному элементу осуществляется за линейное время.

интерфейс List

Данный интерфейс предназначен для работы с упорядоченной коллекцией, т. е. есть возможность обратиться к элементу по его индексу.

- ***add***(int ind, E e) - добавляет элемент в указанную позицию;
- ***addAll***(int ind, Collection<? extends E> c) - добавляет элементы в указанную позицию;
- ***get***(int ind) - возвращает элемент в указанной позиции;
- ***indexOf***(Object o) - возвращает индекс указанного объекта, или -1 если его нет в списке;
- ***lastIndexOf***(Object o) - найти последнее вхождение указанного объекта, или -1 если его нет в списке;
- ***listIterator***() - списочный итератор для прохода по всем элементам с возможностью вставки или замены;
- ***listIterator***(int ind) - списочный итератор с указанной позиции;
- ***remove***(int index) - возвращает элемент в указанной позиции, удаляя его;
- ***set***(int index, E el) - заменяет элемент в указанной позиции новым элементом;
- ***subList***(int fromInd, int toInd) - возвращает часть списка, т.е. элементы в диапазоне [fromIndex; toIndex).

Queue



PriorityQueue - единственная прямая реализация интерфейса **Queue** (не считая **LinkedList**, который больше является списком, чем очередью). Эта очередь упорядочивает элементы либо по их натуральному порядку (используя интерфейс **Comparable**), либо с помощью интерфейса **Comparator**, полученному в конструкторе.

интерфейсы Queue, Deque (1)

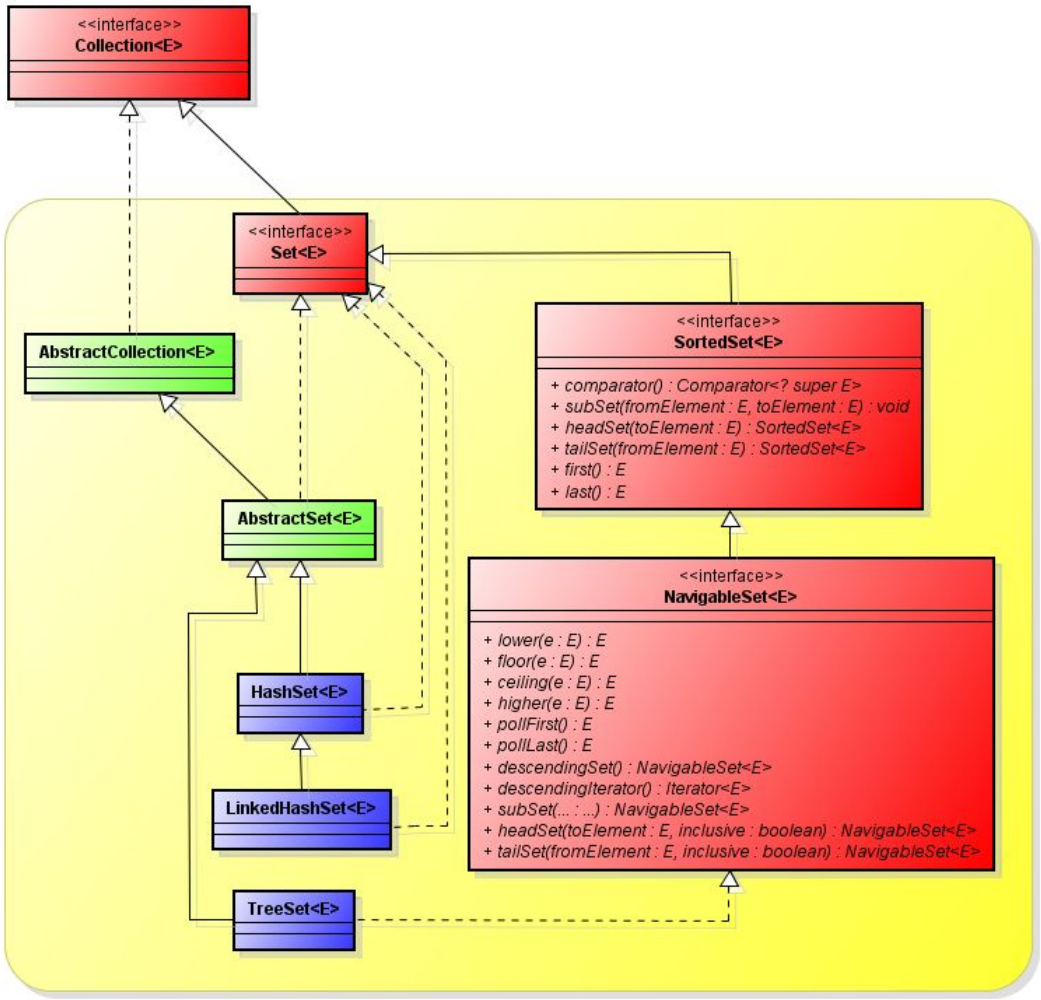
Данные расширения Collection реализует очередь и двустороннюю очередь (double ended queue), которую можно использовать и как стек. Ниже представлены методы Deque, первые шесть наследуются от Queue.

- ***add***(E e) - истина, если элемент e добавлен, если места нет возникает исключение IllegalStateException;
- ***offer***(E e) - истина, если элемент e добавлен, не бросает исключений;
- ***element***() - возвращает элемент из начала очереди;
- ***peek***() - возвращает элемент из начала очереди или возвращает null, когда очередь пуста;
- ***remove***() - возвращает элемент из начала очереди и удаляет его;
- ***poll***() - возвращает элемент из начала очереди и удаляет его, или возвращает null, когда очередь пуста.
- ***addFirst***(E e) - добавляет элемент в начало очереди;
- ***addLast***(E e) - добавляет элемент в конец очереди;
- ***contains***(Object o) - истина, если указанный элемент есть в очереди;
- ***descendingIterator***() - возвращает итератор для прохода по элементам в обратном порядке;
- ***iterator***() - возвращает итератор для прохода по элементам этой коллекции;

интерфейсы Queue, Deque (2)

- ***getFirst***() - возвращает первый элемент очереди без его удаления;
- ***getLast***() - возвращает последний элемент очереди без его удаления;
- ***offerFirst***(E e) - вставляет элемент в начало очереди, пока есть место;
- ***offerLast***(E e) - вставляет элемент в конец очереди, пока есть место;
- ***peekFirst***() - возвращает первый элемент без его удаления или null, когда очередь пуста;
- ***peekLast***() - возвращает последний элемент без его удаления или null, когда очередь пуста;
- ***pollFirst***() - возвращает первый элемент, удаляя его, или null, когда очередь пуста;
- ***pollLast***() - возвращает последний элемент, удаляя его, или null, когда очередь пуста;
- ***removeFirst***() - возвращает первый элемент, удаляя его;
- ***removeFirstOccurrence***(Object o) - удаляет первое появление указанного объекта в очереди;
- ***removeLast***() - возвращает последний элемент, удаляя его;
- ***removeLastOccurrence***(Object o) - удаляет последнее появление указанного объекта в очереди;
- ***size***() - размер очереди в элементах;
- ***pop***() - удаляет элемент из стека представленного этой очередью;
- ***push***(E e) - добавляет элемент в стек представленного этой очередью.

Set



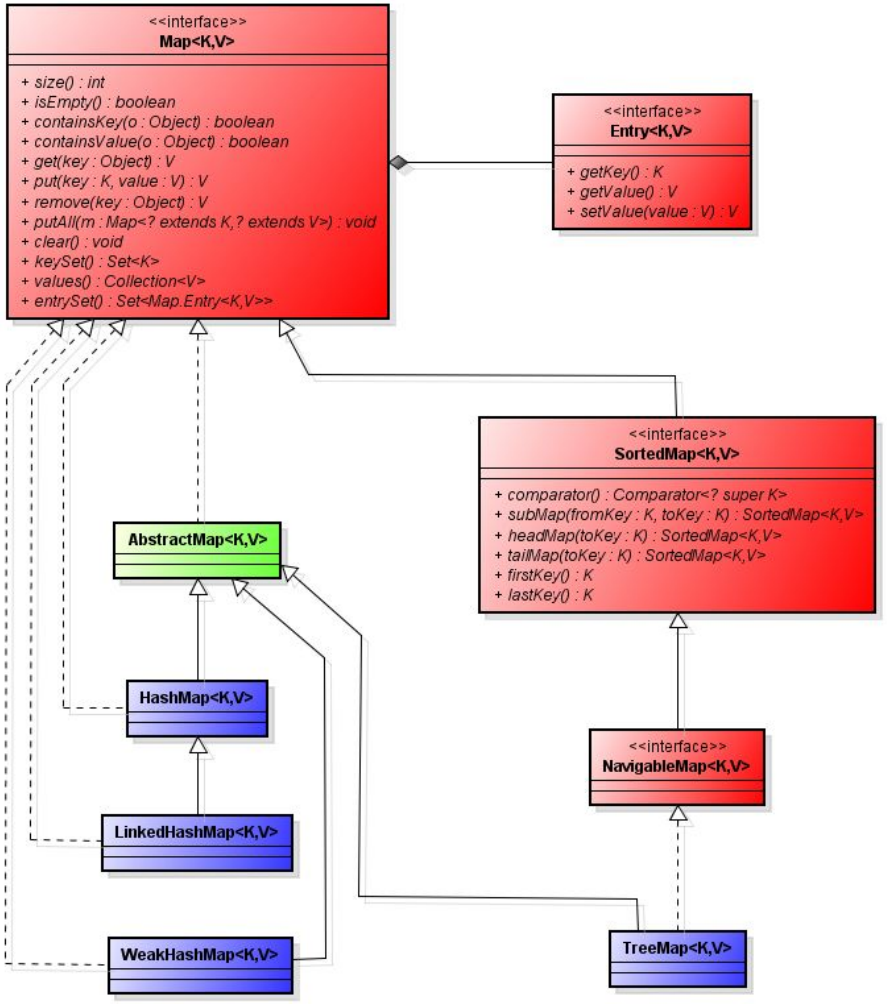
1. **HashSet** - коллекция, не позволяющая хранить одинаковые объекты(как и любой Set). Использует для хранения хэш-таблицу.
2. **LinkedHashSet** - поддерживает связный список элементов набора в том порядке, в котором они вставлялись.
3. **TreeSet** - коллекция, которая хранит свои элементы в виде упорядоченного по значениям дерева. Использует сбалансированное бинарное красно-черное дерево для

интерфейс Set

Интерфейс предназначен для работы с множествами (т.е. в коллекции не может быть два одинаковых элемента).

- ***comparator***() - возвращает объект, используемый для сравнения элементов, или null если используется натуральный порядок элементов;
- ***first***() - первый (наименьший) элемент множества;
- ***last***() - последний (наивысший) элемент;
- ***subSet***(E fromElement, E toElement) - возвращает подмножество элементов из диапазона [fromElement; toElement);
- ***headSet***(E toElement) - возвращает множество элементов меньших чем указанный элемент;
- ***tailSet***(E fromElement) - возвращает часть множества из элементов, больших или равных чем указанный элемент.

Map



1. **HashMap** — основан на хэш-таблицах, реализует интерфейс Map (Ключи и значения могут быть любых типов, в том числе и null. Данная реализация не дает гарантий относительно порядка элементов с течением времени.

2. **LinkedHashMap -** создает коллекцию, которая для хранения элементов
применяет дерево. Объекты сохраняются в отсортированном порядке по возрастанию. Время доступа и извлечения элементов достаточно мало.

3. **WeakHashMap** - коллекция, использующая слабые ссылки для ключей (а не значений).

Интерфейс Map

Map хранит пары "ключ-значение". Каждое значение можно найти по его ключу. Например, в таблице могут находиться записи с информацией о сотрудниках, где ключами являются идентификационные номера сотрудников, а значениями — объекты Employee.

- **Vget(KeyK)** – Возвращает объект, соответствующий указанному ключу или значение null, если карта не содержит указанный ключ. Ключ может быть равен null.
- **Vput(KeyK, ValueV)** – Добавляет ключ и значение к карте. Если такой ключ уже имеется, то новый объект заменяет предыдущий, связанный с этим ключом. Этот метод возвращает предыдущее значение объекта или значение null, если ключ не содержался в карте ранее. Ключ может быть равен null, но значение должно быть отлично от null.
- **voidputAll(Map<? extendsK, ? extendsV> entries)** – Добавляет все элементы заданной карты к текущей.
- **booleancontainsKey(Objectkey)** – Возвращает значение true, если в карте имеется указанный ключ.
- **booleancontainsValue(Objectvalue)** – Возвращает значение true, если в карте имеется указанное значение.
- **Set<Map.Entry<K, V>> entrySet()** – Возвращает представление карты в виде множества объектов Map.Entry, т.е. пар "ключ-значение". Из этого представления можно удалять элементы, при этом они удаляются и из карты, но добавлять их нельзя.
- **Set<K> keySet()** – Возвращает представление карты в виде множества всех ключей. Из этого представления можно удалять элементы, при этом ключи и соответствующие им значения автоматически удаляются из карты, но добавлять новые элементы нельзя.
- **Collection<V> values()** – Возвращает представление карты в виде множества всех значений. Из этого представления можно удалять элементы, при этом значения и соответствующие им ключи автоматически удаляются из карты, но добавлять новые элементы нельзя.

Быстродействие операций

	Временная сложность							
	Среднее				*Худшее*			
	Индекс	Поиск	Вставка	Удаление	Индекс	Поиск	Вставка	Удаление
ArrayList	O(1)	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(n)
Vector	O(1)	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(n)
LinkedList	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)
Hashtable	n/a	O(1)	O(1)	O(1)	n/a	O(n)	O(n)	O(n)
HashMap	n/a	O(1)	O(1)	O(1)	n/a	O(n)	O(n)	O(n)
LinkedHashMap	n/a	O(1)	O(1)	O(1)	n/a	O(n)	O(n)	O(n)
TreeMap	n/a	O(log(n))	O(log(n))	O(log(n))	n/a	O(log(n))	O(log(n))	O(log(n))
HashSet	n/a	O(1)	O(1)	O(1)	n/a	O(n)	O(n)	O(n)
LinkedHashSet	n/a	O(1)	O(1)	O(1)	n/a	O(n)	O(n)	O(n)
TreeSet	n/a	O(log(n))	O(log(n))	O(log(n))	n/a	O(log(n))	O(log(n))	O(log(n))

ArrayList<String> list = new ArrayList<String>();

Спасибо за внимание!