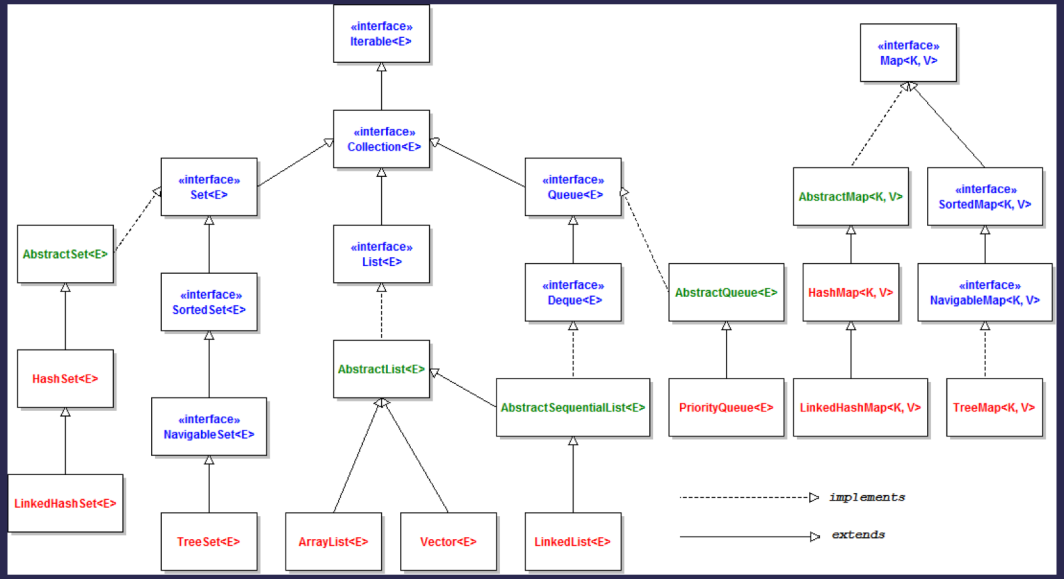
Курс Java-разработчик

Коллекции

Collection


«interface»
Iterable<E>
«interface»
Collection<E>
«interface»
Set<E>
«interface»
List<E>
«interface»
Queue<E>
«interface»
Map<K, V>
AbstractSet<E>
«interface»
SortedSet<E>
AbstractList<E>
«interface»
Deque<E>
AbstractQueue<E>
AbstractMap<K, V>
«interface»
SortedMap<K, V>
HashMap<K, V>
«interface»
NavigableMap<K, V>
HashSet<E>
«interface»
NavigableSet<E>
AbstractSequentialList<E>
PriorityQueue<E>
LinkedHashMap<K, V>
TreeMap<K, V>
LinkedHashSet<E>
TreeSet<E>
ArrayList<E>
Vector<E>
LinkedList<E>
implements
extends

Map

Соотносит уникальные ключи со значениями. Ключ — это объект, который вы используете для последующего извлечения данных. Задавая ключ и значение, вы можете помещать значения в объект карты. Такие коллекции облегчают поиск элемента, если нам известен ключ - уникальный идентификатор объекта.

Основные методы Map<K,V>

- boolean containsKey(Object k)
- boolean containsValue(Object v)
- Set<Map.Entry<K, V>> entrySet()
- V get(Object k)
- V getOrDefault(Object k, V defaultValue)
- V put(K k, V v)
- V putIfAbsent(K k, V v)
- Set<K> keySet()
- Collection<V> values()
- void putAll(Map<? extends K, ? extends V> map)
- V remove(Object k)

Основные методы SortedMap<K,V>

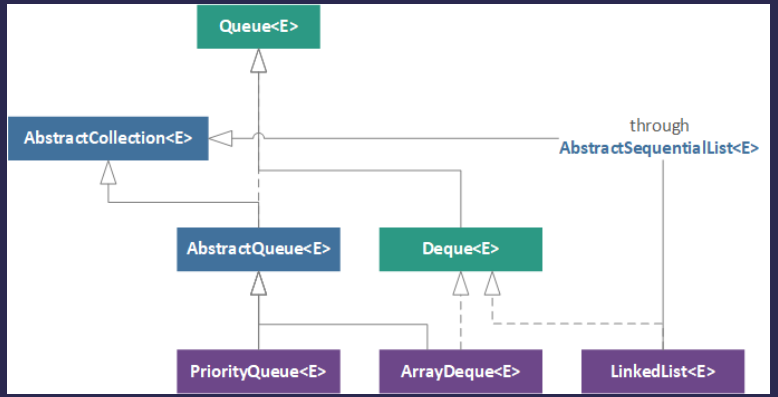
- K firstKey()
- K lastKey()

Реализации **Map**

- **HashMap** — «карта» основана на хэш-таблицах. Ключи и значения могут быть любых типов, в том числе и null.
- **LinkedHashMap** - расширяет класс **HashMap**. Связный список элементов в карте, расположенных в том порядке, в котором они вставлялись. Это позволяет организовать перебор карты в порядке вставки.
- **TreeMap** - «карта», основанная на дереве (красно-черное дерево). Объекты сохраняются в отсортированном порядке по возрастанию.

Queue

Это коллекция, предназначенная для хранения элементов в порядке, нужном для их обработки (FIFO - "First in first out"). Это стандартная модель однонаправленной очереди.



Основные методы **Queue<E>**

- E element() throws NoSuchElementException
- boolean offer(E obj)
- E peek()
- E poll()
- E remove() throws NoSuchElementException

Основные методы **Deque<E>**

- void addFirst(E obj)
- E getFirst() throws NoSuchElementException
- boolean offerFirst(E obj)
- E peekFirst()
- E pollFirst()
- E pop() throws NoSuchElementException
- void push(E element)
- E removeFirst() throws NoSuchElementException

Аналогично есть методы для последнего элемента

Реализации **Queue**

- **PriorityQueue** - очередь упорядочивает элементы либо по их натуральному порядку
- **ArrayDeque** – очередь, основанная на массиве, позволяющая извлекать/добавлять элементы с обеих сторон.

Домашнее задание

- Создать вольеры (не один!). Вольеров должно быть ограниченное количество. Если добавляется животное в несуществующий вольер - должно отработать исключение.