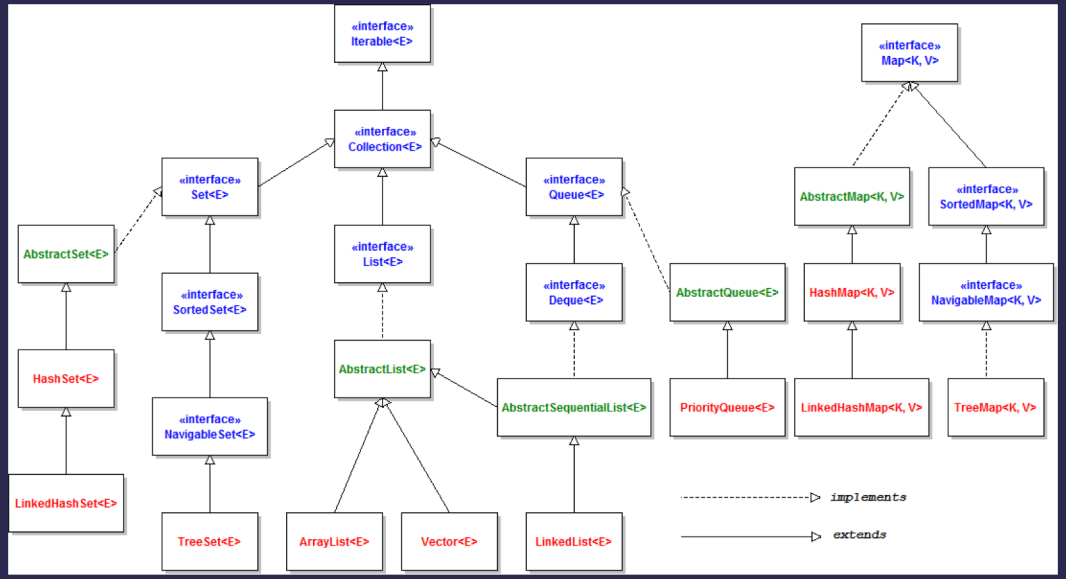
Курс Java-разработчик

Коллекции

Collection

Контейнерный класс – это класс, объекты которого выступают в роли хранилищ других объектов. Коллекция является неупорядоченной.



Основные методы **Collection<E>**

- boolean add (E item)
- boolean addAll (Collection<? extends E> col)
- void clear ()
- boolean contains (Object item)
- boolean isEmpty ()
- boolean remove (Object item)
- boolean removeAll (Collection<?> col)
- int size ()
- Object[ ] toArray ()

List

Представляет собой неупорядоченную коллекцию, в которой допустимы дублирующие значения. Иногда их называют последовательностями (sequence). Элементы такой коллекции пронумерованы, начиная от нуля, к ним можно обратиться по индексу.

Основные методы List<E>

- void add(int index, E obj)
- boolean addAll(int index, Collection<? extends E> col)
- E get(int index)
- int indexOf(Object obj)
- int lastIndexOf(Object obj)
- * static <E> List<E> of(E… elem)
- E remove(int index)
- E set(int index, E obj)

* - введен в JDK 9.

Реализации **List**

- **ArrayList** – список на массиве. Длина автоматически увеличивается при добавлении новых элементов. Если при добавлении элемента, оказывается, что массив полностью заполнен, будет создан новый массив размером (n * 3) / 2 + 1, в него будут помещены все элементы из старого массива + новый, добавляемый элемент.
- **LinkedList** – двусвязный список. Это структура данных, состоящая из узлов, каждый из которых содержит как собственно данные, так и две ссылки («связки») на следующий и предыдущий узел списка.

Set

Описывает коллекцию, не содержащую повторяющихся элементов. Это соответствует математическому понятию множества (set)

Основные методы **Set<E>**

Такие же, как и Collection ☺

Реализации **Set**

- **HashSet** – хэш-таблица. Хеш-таблица представляет такую структуру данных, в которой все объекты имеют уникальный ключ или хеш-код. Данный ключ позволяет уникально идентифицировать объект в таблице.
- **LinkedHashSet** – поддерживает связный список элементов набора в том порядке, в котором они вставлялись.
- **TreeSet** – коллекция, которая хранит свои элементы в виде упорядоченного по значениям дерева (красно-черного дерева).

Домашнее задание

- Создать вольер. Должна быть возможность добавлять/удалять животных. Если добавляем одинаковых животных в варьер - отрабатывает исключение. Если добавляем животное в заполненный вольер - отрабатывает исключение.

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.

Домашнее задание

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