Интерфейсы

Цели: ознакомиться с коллекциями в Java

План урока:

- List Interface
- Set Interface
- Map Interface

Collections

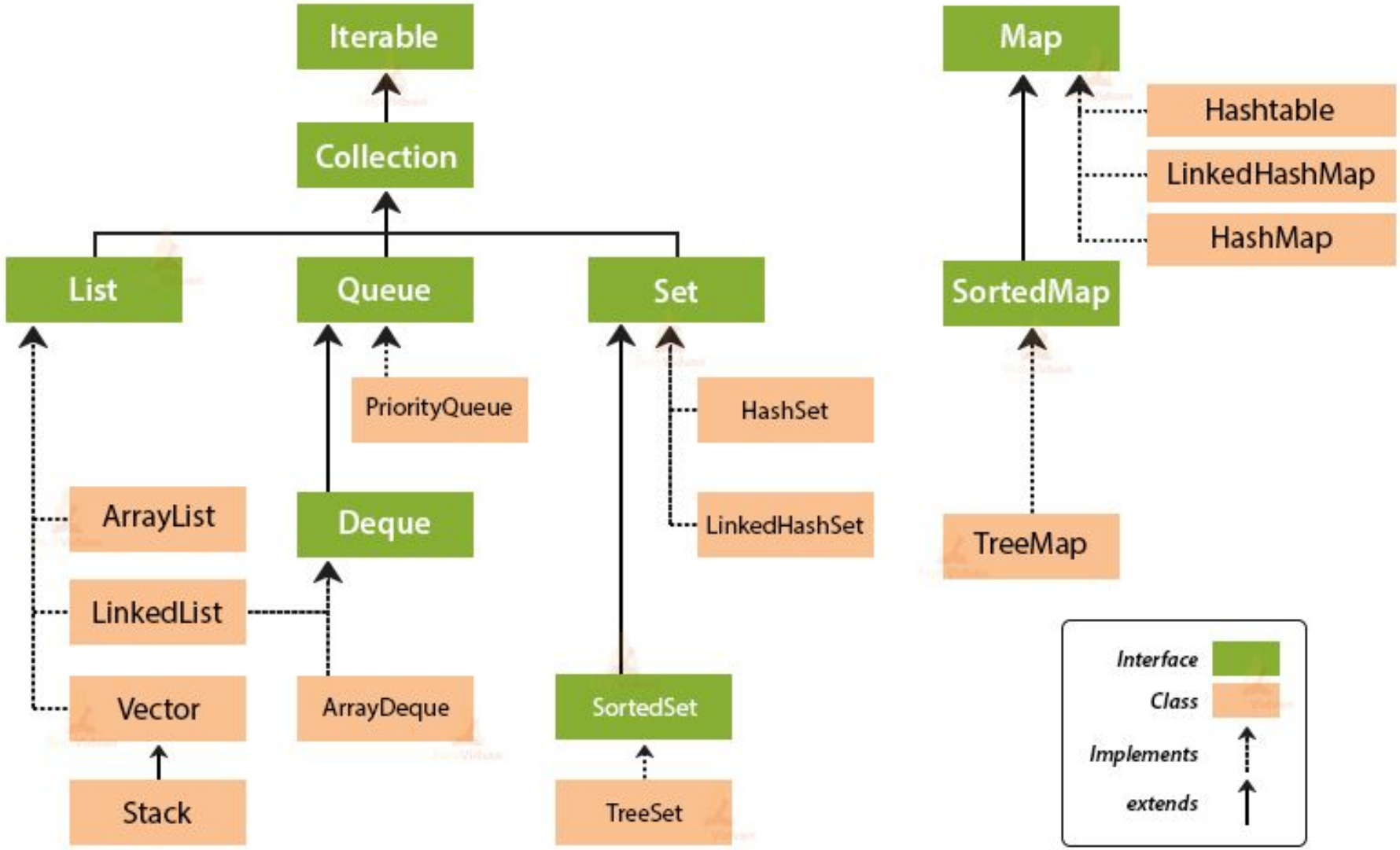
Java collections framework — это набор связанных классов и интерфейсов, реализующих широко используемые структуры данных — коллекции.

Для хранения наборов данных в Java предназначены массивы. Однако их не всегда удобно использовать, прежде всего потому, что они имеют фиксированную длину. Эту проблему в Java решают коллекции. Однако суть не только в гибких по размеру наборах объектов, но в и том, что классы коллекций реализуют различные алгоритмы и структуры данных, например, такие как стек, очередь, дерево и ряд других.

Интерфейс Collection является базовым для всех коллекций, определяя основной функционал

Интерфейс Collection является обобщенным и расширяет интерфейс Iterable, поэтому все объекты коллекций можно перебирать в цикле по типу for-each.

Collection Framework Hierarchy in Java



List: ArrayList

ArrayList - автоматически расширяемый массив. Иными словами - усовершенствованный массив с большими возможностями

- **ArrayList()** - Пустой конструктор с начальной емкостью внутреннего массива = 10.
- **ArrayList(int initialCapacity)** - В качестве параметра конструктора выступает значения начального размера внутреннего массива.
- **ArrayList.add(E e)** - Добавляет новый элемент в конец списка. Возвращает boolean-значение (*true* — успех, *false* — не добавлено):
- **add(int index, E element)** - Добавляет элемент element в позицию index. При добавлении происходит сдвиг всех элементов справа от указанного индекса на 1 позицию вправо:
- **addAll(Collection <? extends E> collection)** - Добавление всех элементов коллекции collection в список в порядке их расположения в collection.
- **addAll(int index, Collection <? extends E> collection)** - Добавление всех элементов collection в список начиная с индекса index. При этом все элементы сдвинутся вправо на количество элементов в списке collection:
- **clear()** - Удаление всех элементов из списка.
- **clone()** - Возвращает объект-копию массива:
- **contains(Object o)** - Проверка наличие объекта в списке, возвращает boolean-значение.
- **ensureCapacity(int minCapacity)** - Увеличивает размер внутреннего массива, чтобы в него поместилось количество элементов, переданных в minCapacity.
- **get(int index)** - Возвращает элемент, который расположен в указанной позиции списка.

  indexOf(Object o) - Метод возвращает индекс первого вхождения элемента в списке. Если элемента не существует в списке, метод вернет -1.

List: ArrayList

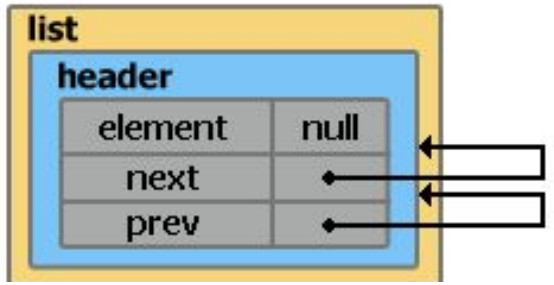
- **isEmpty()** - Метод возвращает *true*, если список пустой, *false* в обратном случае.
  Если в списке содержатся только элементы null, метод вернет *false*. Иными словами, null элементы также учитываются этим методом.
- **iterator()** - Возвращает итератор для списка для последующего использования в цикле или при любой другой обработке.
  lastIndexOf(Object o) - Функционал метода похож на indexOf(), отличие в том, что возвращается индекс последнего элемента в списке.
  Если элемент не найден, также возвращает -1.
- **remove(int index)** - Удаление элемента в указанной позиции индекса. После удаления сдвигает все элементы влево для заполнения освободившегося пространства.
  remove(Object o) - Метод удаляет из списка переданный элемент o. Если элемент присутствует в списке, он удаляется, а все элементы смещаются влево. Если элемент существует в списке и успешно удален, метод возвращает true, в обратном случае — *false*.
- **removeAll(Collection<?> c)** - Если необходимо удалить несколько элементов, не стоит делать это в цикле по условию: гораздо удобнее и безопаснее воспользоваться методом removeAll(). Он принимает коллекцию элементов, которая будет удалена из списка.
- **set(int index, E element)** - Замена элемента в указанной позиции index на переданный element.
- **size()** - Лучший способ (практически единственный) для того, чтобы узнать размер массива.
- **sort(Comparator<? super E> c)** - Сортировка списка по заданному правилу. Правило сортировки представляет собой

List: LinkedList

LinkedList - Двусвязный список

В LinkedList элементы фактически представляют собой звенья одной цепи. У каждого элемента помимо тех данных, которые он хранит, имеется ссылка на предыдущий и следующий элемент. По этим ссылкам можно переходить от одного элемента к другому.

- LinkedList() - создает пустой список
- LinkedList(Collection<? extends E> col) - создает список, в который добавляет все элементы коллекции col
- **addFirst()**- добавляет элемент в начало списка
- **addLast()** - добавляет элемент в конец списка
- **removeFirst()** - удаляет первый элемент из начала списка
- **removeLast()** - удаляет последний элемент из конца списка
- **getFirst()** - получает первый элемент
- **getLast()** - получает последний элемент



Set: TreeSet

Объекты хранятся в отсортированном и возрастающем порядке.

- **add**(Object o) - Добавляет указанный элемент к этому набору, если он еще не присутствует.
- **addAll**(Collection c) - Добавляет все элементы в указанной коллекции к этому набору.
- **clear()** - Удаляет все элементы из этого набора.
- **clone()** -Возвращает мелкую копию этого экземпляра TreeSet.
- **comparator()** - Возвращает компаратор, используемый для порядка этого отсортированного набора, или null (нуль), если этот набор деревьев использует свои элементы естественного упорядочения.
- **contains(Object o)** - Возвращает true, если этот набор содержит указанный элемент.
- **first()** - Возвращает первый (самый низкий) элемент, находящийся в этом отсортированном наборе.
- **headSet(Object toElement)** - Возвращает представление (вид) части этого набора, элементы которого строго меньше, чем toElement.
- **isEmpty()** - Возвращает true, если этот набор не содержит элементов.
- **iterator()** - Возвращает итератор над элементами этого набора.
- **last()** - Возвращает последний (самый высокий) элемент, находящийся в этом отсортированном наборе.
- **remove(Object o)** - Удаляет указанный элемент из этого набора, если он присутствует.
- **size()** - Возвращает количество элементов в этом наборе (его мощность).
- **subSet(Object fromElement, Object toElement)** - Возвращает представление (вид) части этого набора, элементы которого варьируются от fromElement, включительно, до toElement, исключительно.
- **SortedSet tailSet(Object fromElement)** - Возвращает представление (вид) части этого набора, элементы которого больше или равны fromElement.

Set: HashSet

Создает коллекцию, которая использует хеш-таблицу для хранения.

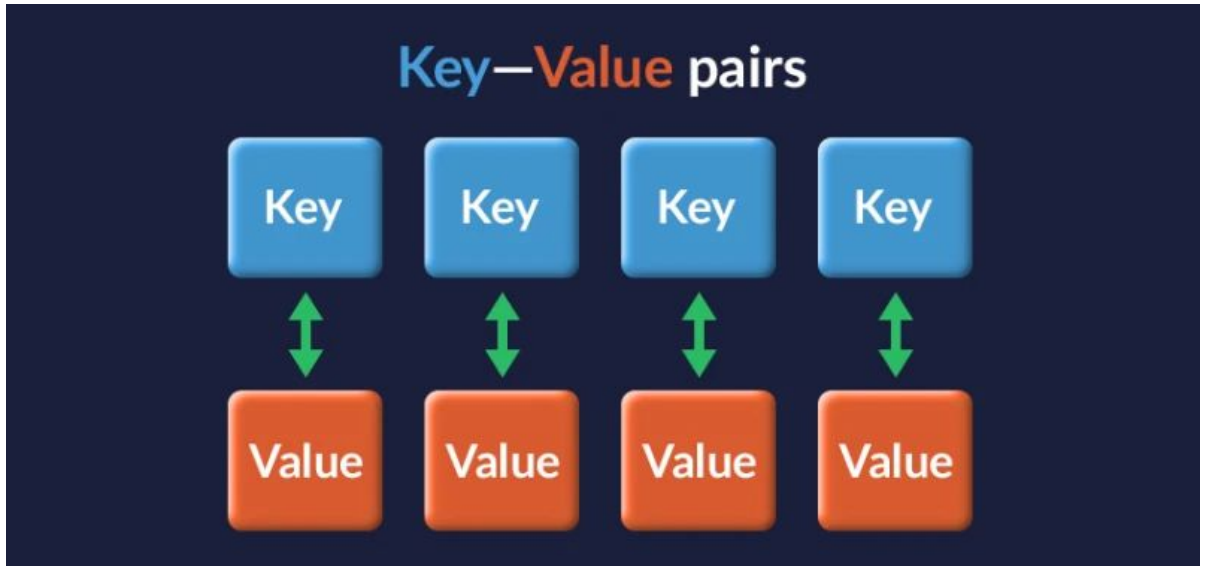
Хэш-таблица хранит информацию с помощью механизма, называемого хешированием. В хэшировании информационный контент ключа используется для определения уникального значения, называемого его хэш-кодом.

Хэш-код затем используется как индекс, в котором хранятся данные, связанные с ключом. Преобразование ключа в его хэш-код выполняется автоматически.

- **add(Object o)** -Добавляет указанный элемент к этому набору, если он еще не присутствует.
- **clear()** - Удаляет все элементы из этого набора.
- **clone()** - Возвращает мелкую копию этого экземпляра HashSet: сами элементы не клонируются.
- **contains(Object o)** - Возвращает true, если этот набор содержит указанный элемент.
- **isEmpty()** - Возвращает true, если этот набор не содержит элементов.
- **iterator()** - Возвращает итератор по элементам этого набора.
- **remove(Object o)** - Удаляет указанный элемент из этого набора, если он присутствует.
- **size()** - Возвращает количество элементов в этом наборе (его количество элементов).

Map: HashMap

*HashMap — основан на хэш-таблицах, реализует интерфейс Map (что подразумевает хранение данных в виде пар ключ/значение). Ключи и значения могут быть любых типов, в том числе и null. Данная реализация не дает гарантий относительно порядка элементов с течением времени.*ользуется как индекс, в котором хранятся данные, связанные с ключом. Преобразование ключа в его хэш-код выполняется автоматически.



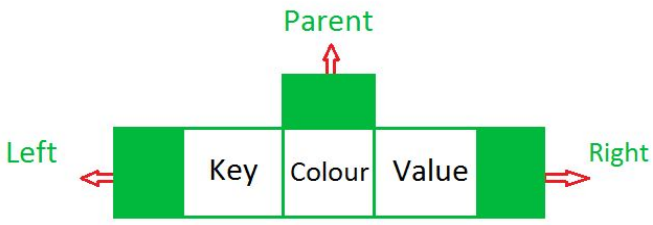
Map: HashMap

- **clear()** - Удаляет все соответствия с этого Map.
- **clone()** - Возвращает мелкую копию этого экземпляра HashMap: сами ключи и значения не клонируются.
- **containsKey(Object key)**- Возвращает true, если этот Map содержит отображение для указанного ключа.
- **containsValue(Object value)** - Возвращает true, если этот Map отображает одну или несколько клавиш в указанное значение.
- **entrySet()** - Возвращает представление (вид) коллекции отображений, содержащихся в этом Map.
- **get(Object key)** - Возвращает значение, для которого указанный ключ отображается в этой хэш-карте идентификатора, или null (нуль), если Map не содержит отображения для этого ключа.
- **isEmpty()** - Возвращает true, если этот Map не содержит отображений значений ключа.
- **keySet()** - Возвращает заданное представление (вид) ключей, содержащихся на этом Map.
- **put(Object key, Object value)** - Связывает указанное значение с указанным ключом на этом Map.
- **putAll(Map m)**- Копирует все отображения с указанного Map на этот Map. Эти отображения заменят любые отображения, которые это отображение имело для любого из ключей, находящихся в настоящее время на указанном Map.
- **remove(Object key)** - Удаляет отображение для этого ключа с этого Map, если присутствует.
- **size()** - Возвращает количество ключ-значение отображений на этом Map.
- **values()** - Возвращает представление (вид) коллекции значений, содержащихся на этой карте.

Map: TreeMap

Создает коллекцию, которая для хранения элементов применяет **дерево**. Следует отметить, что, в отличие от HashMap, карта деревьев гарантирует, что ее элементы будут отсортированы в порядке возрастания ключа.

- **clear()** - Удаляет все отображения из этой TreeMap.
- **clone()** - Возвращает мелкую копию этого экземпляра TreeMap.
- **comparator()** - Возвращает компаратор, используемый для порядка этого Map, или null (нуль), если этот Map использует естественный порядок своих ключей.
- **containsKey(Object key)** - Возвращает true, если этот Map содержит отображение для указанного ключа.



Map: TreeMap

- **containsValue(Object value)** - Возвращает true, если этот Map отображает одну или несколько клавиш в указанное значение.
- **entrySet()**- Возвращает заданный вид отображений, содержащихся в этом Map.
- **firstKey()** - Возвращает первый (самый низкий) ключ на этом сортированном Map.
- **get(Object key)** - Возвращает значение, на которое этот Map отображает указанный ключ.
- **headMap(Object toKey)** - Возвращает представление (вид) части этого Map, ключи которой строго меньше, чем toKey.
- **keySet()** - Возвращает Set вид ключей, содержащихся в этом Map.
- **lastKey()** - Возвращает последний (самый высокий) ключ в настоящее время на этом отсортированном Map.
- **put(Object key, Object value)** - Связывает указанное значение с указанным ключом на этом Map.
- **putAll(Map map)** - Копирует все отображения с указанного Map на этот Map.
- **remove(Object key)** - Удаляет отображение этого ключа из этого TreeMap, если оно присутствует.
- **size()** - Возвращает количество отображений ключ-значение на этом Map.
- **subMap(Object fromKey, Object toKey)** - Возвращает представление (вид) части этого Map, ключи которого варьируется от fromKey, включительно, до toKey, исключительно.
- **tailMap(Object fromKey)** - Возвращает представление (вид) части этого Map, ключи которого больше или равны fromKey.
- **values()** - Возвращает представление (вид) коллекции значений, содержащихся на этом Map.

Вопросы?