|  |  |  |
| --- | --- | --- |
|  | Основная функциональность | Примеры типичного использования |
| Set | Фукциональность интефейса Collection.\* | Используется, когда нужно не допускать дублирования элементов, при этом их порядок не важен.  Set<Letter> alphabet = new TreeSet<>(); |
| List | Фукциональность интефейса Collection.\*  void add(int index, E element) - добавление элемента в конкретную позицию.  E get(int index) - возвращает элемент по индексу.  int indexOf(Object o) - возвращает индекс первого соответствующего элемента(-1, если не найден).  int lastIndexOf(Object o) - то же только для последнего найденного элемента.  E set(int index, E e) - заменяет элемент по индексу, возвращает старый. | Реализации данного интерфейса дают возможность хранить повторяющиеся элементы. Элементы храняться в конкретных позициях, определяемых целочисленным индесом.  List<String> wordsInText = new ArrayList<>(); |
| Queue | Фукциональность интефейса Collection.\*  E element() - возвращает следующий элемент или бросает исключение, если очередь пуста.  boolean offer(E e) - добавляет элемент в конец очереди.  E remove() - удаляет следующий элемент  void push(E e) - добавляет элемент в начало очереди E poll() - возвращает и удаляет следующий элемент(null, если пуста)  E peek() - возвращает следующий элемент  E pop() - возвращает и удаляет следующий элемент(exception, если пуста)  Для LIFO принято использовать push, poll, peek.  Для FIFO offer, poll, peek. | Элемнты добавляются и извлекаются в особом порядке, по принципу FIFO(first-in, first-out), LIFO(first-in, first-out) в зависимости от реализации интерфейса.  Queue<Task> tasks = new PriorityQueue<> |
| Map | HashMap - хранит элементы в хэш-таблице.  LinkedHashMap - сохраняет порядок добавления элементов.  TreeMap - хранит ключи в отсортированной древовидной структуре.  Hashtable - как HashMap, только стар.  isEmpty() - проверка на пустоту  size() - количество пар ключ-значение  clear() - очистка  V get(Object key) - получение значения по ключу  V put(Object key, V value) - добавляет пару, возвращает предыдущее значение или null.  V remove(Object key) - удаляет и возвращает значение по ключу  boolean containsKey(Object key) - проверяет есть ли ключ  boolean containsValue(Object key) - проверяет есть ли значение  Set<K> keySet() - возвращает Set ключей  Collection<V> values - возвращает Collection значений | Используется, чтобы хранить пары ключ-значение.  Map<Integer, String> students = new HashMap<>();  ключ - student id  значение - имя студента |

\*Фукциональность интефейса Collection.

Добавление нового элемента в коллекцию, удаление элемента, проверка на пустоту, получение   
размера коллекции, очистка, а так же проверка элемента на принадлежность коллекии.

Методы add(), remove(), isEmpty(), size(), clear(), contains() соответственно.