##### **Задание 4. Применение коллекций**

Заполните таблицу.

|  |  |  |
| --- | --- | --- |
|  | Основная функциональность | Примеры типичного использования |
| Set | Множество - коллекция без повторяющихся элементов. Элементы хранятся в произвольном порядке, а не в порядке добавления.  **boolean add(E e)** – добавить элемент в коллекцию.  **boolean contains(Object o) – Вернёт true, если элемент присутствует в коллекции.**  **boolean remove(Object o) – если удаление прошло успешно вернёт true.**  Iterator<E> iterator() – вернёт итератор коллекции.  int size() – вернёт число элементов.  boolean isEmpty() – вернёт true, если коллекция пуста.  void clear() – удаляет все элементы из коллекции. | HashSet — основная реализация множества  EnumSet — множество из enums  BitSet\* — множество битов  LinkedHashSet — сохраняет порядок вставки  TreeSet — отсортированные set  ConcurrentSkipListSet — отсортированный многопоточный set  CopyOnWriteArraySet — редкие обновления, частые чтения |
| List | Лист – коллекция, элементы которой хранятся в порядке добавления, можно вставлять новые элементы в любое место коллекции. Элементы могут повторяться.  **boolean add(E e)** – добавить элемент в коллекцию.  **void add(int index, E e )** – добавить элемент в коллекцию на позицию с номером index.  E get(int index) – вернет элемент по индексу.  **boolean remove(int index) – если удаление прошло успешно вернёт true.**  **boolean remove(Object o) – если удаление прошло успешно вернёт true.**  Iterator<E> iterator() – вернёт итератор коллекции.  int size() – вернёт число элементов.  boolean isEmpty() – вернёт true, если коллекция пуста.  void clear() – удаляет все элементы из коллекции. | ArrayList — основной список, основан на массиве  LinkedList — полезен лишь в некоторых редких случаях  Vector — устарел  CopyOnWriteArrayList — редкие обновления, частые чтения |
| Queue | Очередь – коллекция, позволяющая реализовывать принципы FIFO/LIFO,элементы хранятся строго в порядке добавления, добавление в середину невозможно. Элементы могут повторяться.  **boolean add**(E e) – добавить элемент в коллекцию.  **E element()** – достаёт элемент из очереди не удаляя его.  **boolean offer(E e) – добавляет элемент, если это возможно и возвращает true.**  **E peek() – достаёт из очереди головной элемент, не удаляя его.**  E poll()**– достаёт из очереди головной элемент, удаляя его.**  E remove() **– достаёт из очереди головной элемент, удаляя его.** | ArrayDeque — основная реализация, основан на массиве  Stack — устарел  PriorityQueue — отсортированная очередь |
| Map | Отображение – коллекция для хранения пар [ключ - значение]. Все ключи должны быть уникальны, а значения могут повторяться. Пары хранятся в случайном порядке и доступны в виде множества типа Map.Entry.  void clear() – удаляет все элементы из коллекции.  boolean containsKey(Object key) – вернёт true, если коллекция содержит ключ key.  boolean containsValue(Object val) – вернёт true, если коллекция содержит значение val.  Set<Map.Entry<K,V>> entrySet() – вернёт множество значений типа Map.Entry<K,V>.  V get(Object key) – вернёт значение по ключу.  boolean isEmpty() – вернет true, если коллекция пустая.  Set<K> keySet() – вернет множество ключей.  V put(K key, V val) – добавит в коллекцию пару [key – val].  V remove(Object key) **– удаляет пару по ключу, возвращая значение.**  int size() – вернёт число элементов.  **Collection<V> values() – вернёт коллекцию значений.** | HashMap — основная реализация  EnumMap — enum в качестве ключей  Hashtable — устарел  IdentityHashMap — ключи сравниваются с помощью ==  LinkedHashMap — сохраняет порядок вставки  TreeMap — сортированные ключи  WeakHashMap — слабые ссылки, полезно для кэшей  ConcurrentHashMap — основная многопоточная реализация  ConcurrentSkipListMap — отсортированная многопоточная реализация |