**1. List Functions:**

A List in Java allows duplicate elements and maintains insertion order. Common implementations include ArrayList and LinkedList.

* **add(E element)**: Adds an element to the list.
* **add(int index, E element)**: Inserts an element at the specified index.
* **get(int index)**: Retrieves the element at the specified position.
* **set(int index, E element)**: Replaces the element at the specified position with the given element.
* **remove(int index)**: Removes the element at the specified index.
* **remove(Object o)**: Removes the first occurrence of the specified element.
* **size()**: Returns the number of elements in the list.
* **contains(Object o)**: Returns true if the list contains the specified element.
* **indexOf(Object o)**: Returns the index of the first occurrence of the specified element, or -1 if not found.
* **lastIndexOf(Object o)**: Returns the index of the last occurrence of the specified element.
* **isEmpty()**: Returns true if the list is empty.
* **clear()**: Removes all elements from the list.
* **subList(int fromIndex, int toIndex)**: Returns a view of the portion of the list between fromIndex (inclusive) and toIndex (exclusive).

**2. Set Functions:**

A Set in Java does not allow duplicate elements and typically does not maintain insertion order. Common implementations include HashSet, LinkedHashSet, and TreeSet.

* **add(E element)**: Adds the specified element to the set if it is not already present.
* **remove(Object o)**: Removes the specified element from the set.
* **size()**: Returns the number of elements in the set.
* **contains(Object o)**: Returns true if the set contains the specified element.
* **isEmpty()**: Returns true if the set contains no elements.
* **clear()**: Removes all elements from the set.
* **iterator()**: Returns an iterator over the elements in the set.

**3. Queue Functions:**

A Queue in Java is a collection used to hold multiple elements prior to processing. It follows FIFO (First In First Out) ordering. Common implementations include LinkedList and PriorityQueue.

* **offer(E e)**: Inserts the specified element into the queue (returns false if it can't be added).
* **add(E e)**: Inserts the specified element into the queue (throws an exception if it can't be added).
* **poll()**: Retrieves and removes the head of the queue, or returns null if the queue is empty.
* **remove()**: Retrieves and removes the head of the queue, throwing an exception if the queue is empty.
* **peek()**: Retrieves, but does not remove, the head of the queue, or returns null if the queue is empty.
* **element()**: Retrieves, but does not remove, the head of the queue, throwing an exception if the queue is empty.
* **isEmpty()**: Returns true if the queue contains no elements.
* **size()**: Returns the number of elements in the queue.

**4. Map Functions:**

A Map in Java is a collection that maps keys to values. It does n ot allow duplicate keys, but each key can map to exactly one value. Common implementations include HashMap, LinkedHashMap, and TreeMap.

* **put(K key, V value)**: Associates the specified value with the specified key.
* **putAll(Map<? extends K,? extends V> m)**: Copies all mappings from the specified map to this map.
* **putIfAbsent(K key, V value)**: Adds the specified key-value pair if the key is not already associated with a value.
* **get(Object key)**: Returns the value associated with the specified key, or null if no mapping exists.
* **remove(Object key)**: Removes the mapping for the specified key, if it exists.
* **replace(K key, V value)**: Replaces the value for the specified key, only if it is already mapped.
* **replace(K key, V oldValue, V newValue)**: Replaces the entry for a key only if it is currently mapped to a specific value.
* **containsKey(Object key)**: Returns true if the map contains a mapping for the specified key.
* **containsValue(Object value)**: Returns true if the map maps one or more keys to the specified value.
* **size()**: Returns the number of key-value mappings in the map.
* **isEmpty()**: Returns true if the map contains no key-value mappings.
* **clear()**: Removes all mappings from the map.
* **keySet()**: Returns a Set view of the keys contained in the map.
* **values()**: Returns a Collection view of the values contained in the map.
* **entrySet()**: Returns a Set view of the key-value mappings contained in the map.