**Q.3.Data Structure-List,Set And Map**

**List:**

The List interface extends **Collection** and declares the behavior of a collection that stores a sequence of elements.

* Elements can be inserted or accessed by their position in the list, using a zero-based index.
* A list may contain duplicate elements.
* In addition to the methods defined by **Collection**, List defines some of its own.
* Several of the list methods will throw an UnsupportedOperationException if the collection cannot be modified, and a ClassCastException is generated when one object is incompatible with another.

|  |  |
| --- | --- |
| 1 | **void add(int index, Object obj)**  Inserts obj into the invoking list at the index passed in the index. Any pre-existing elements at or beyond the point of insertion are shifted up. Thus, no elements are overwritten. |
| 2 | **boolean addAll(int index, Collection c)**  Inserts all elements of **c** into the invoking list at the index passed in the index. Any pre-existing elements at or beyond the point of insertion are shifted up. Thus, no elements are overwritten. Returns true if the invoking list changes and returns false otherwise. |
| 3 | **Object get(int index)**  Returns the object stored at the specified index within the invoking collection. |
| 4 | **int indexOf(Object obj)**  Returns the index of the first instance of obj in the invoking list. If obj is not an element of the list, .1 is returned. |
| 5 | **int lastIndexOf(Object obj)**  Returns the index of the last instance of obj in the invoking list. If obj is not an element of the list, .1 is returned. |
| 6 | **ListIterator listIterator( )**  Returns an iterator to the start of the invoking list. |
| 7 | **ListIterator listIterator(int index)**  Returns an iterator to the invoking list that begins at the specified index. |
| 8 | **Object remove(int index)**  Removes the element at position index from the invoking list and returns the deleted element. The resulting list is compacted. That is, the indexes of subsequent elements are decremented by one. |
| 9 | **Object set(int index, Object obj)**  Assigns obj to the location specified by index within the invoking list. |
| 10 | **List subList(int start, int end)**  Returns a list that includes elements from start to end.1 in the invoking list. Elements in the returned list are also referenced by the invoking object. |

**Set Interface:**

A Set is a Collection that cannot contain duplicate elements. It models the mathematical set abstraction.

The Set interface contains only methods inherited from Collection and adds the restriction that duplicate elements are prohibited.

Set also adds a stronger contract on the behavior of the equals and hashCode operations, allowing Set instances to be compared meaningfully even if their implementation types differ.

The methods declared by Set are summarized in the following table −

|  |  |
| --- | --- |
| 1 | **add( )**  Adds an object to the collection. |
| 2 | **clear( )**  Removes all objects from the collection. |
| 3 | **contains( )**  Returns true if a specified object is an element within the collection. |
| 4 | **isEmpty( )**  Returns true if the collection has no elements. |
| 5 | **iterator( )**  Returns an Iterator object for the collection, which may be used to retrieve an object. |
| 6 | **remove( )**  Removes a specified object from the collection. |
| 7 | **size( )**  Returns the number of elements in the collection. |

**Map Interface:**

The Map interface maps unique keys to values. A key is an object that you use to retrieve a value at a later date.

Given a key and a value, you can store the value in a Map object. After the value is stored, you can retrieve it by using its key.

Several methods throw a NoSuchElementException when no items exist in the invoking map.

A ClassCastException is thrown when an object is incompatible with the elements in a map.

A NullPointerException is thrown if an attempt is made to use a null object and null is not allowed in the map.

An UnsupportedOperationException is thrown when an attempt is made to change an unmodifiable map.

**Methods:-**

|  |  |
| --- | --- |
| 1 | **void clear( )**  Removes all key/value pairs from the invoking map. |
| 2 | **boolean containsKey(Object k)**  Returns true if the invoking map contains **k** as a key. Otherwise, returns false. |
| 3 | **boolean containsValue(Object v)**  Returns true if the map contains **v** as a value. Otherwise, returns false. |
| 4 | **Set entrySet( )**  Returns a Set that contains the entries in the map. The set contains objects of type Map.Entry. This method provides a set-view of the invoking map. |
| 5 | **boolean equals(Object obj)**  Returns true if obj is a Map and contains the same entries. Otherwise, returns false. |
| 6 | **Object get(Object k)**  Returns the value associated with the key **k**. |
| 7 | **int hashCode( )**  Returns the hash code for the invoking map. |
| 8 | **boolean isEmpty( )**  Returns true if the invoking map is empty. Otherwise, returns false. |
| 9 | **Set keySet( )**  Returns a Set that contains the keys in the invoking map. This method provides a set-view of the keys in the invoking map. |
| 10 | **Object put(Object k, Object v)**  Puts an entry in the invoking map, overwriting any previous value associated with the key. The key and value are k and v, respectively. Returns null if the key did not already exist. Otherwise, the previous value linked to the key is returned. |
| 11 | **void putAll(Map m)**  Puts all the entries from **m** into this map. |
| 12 | **Object remove(Object k)**  Removes the entry whose key equals **k**. |
| 13 | **int size( )**  Returns the number of key/value pairs in the map. |
| 14 | **Collection values( )**  Returns a collection containing the values in the map. This method provides a collection-view of the values in the map. |