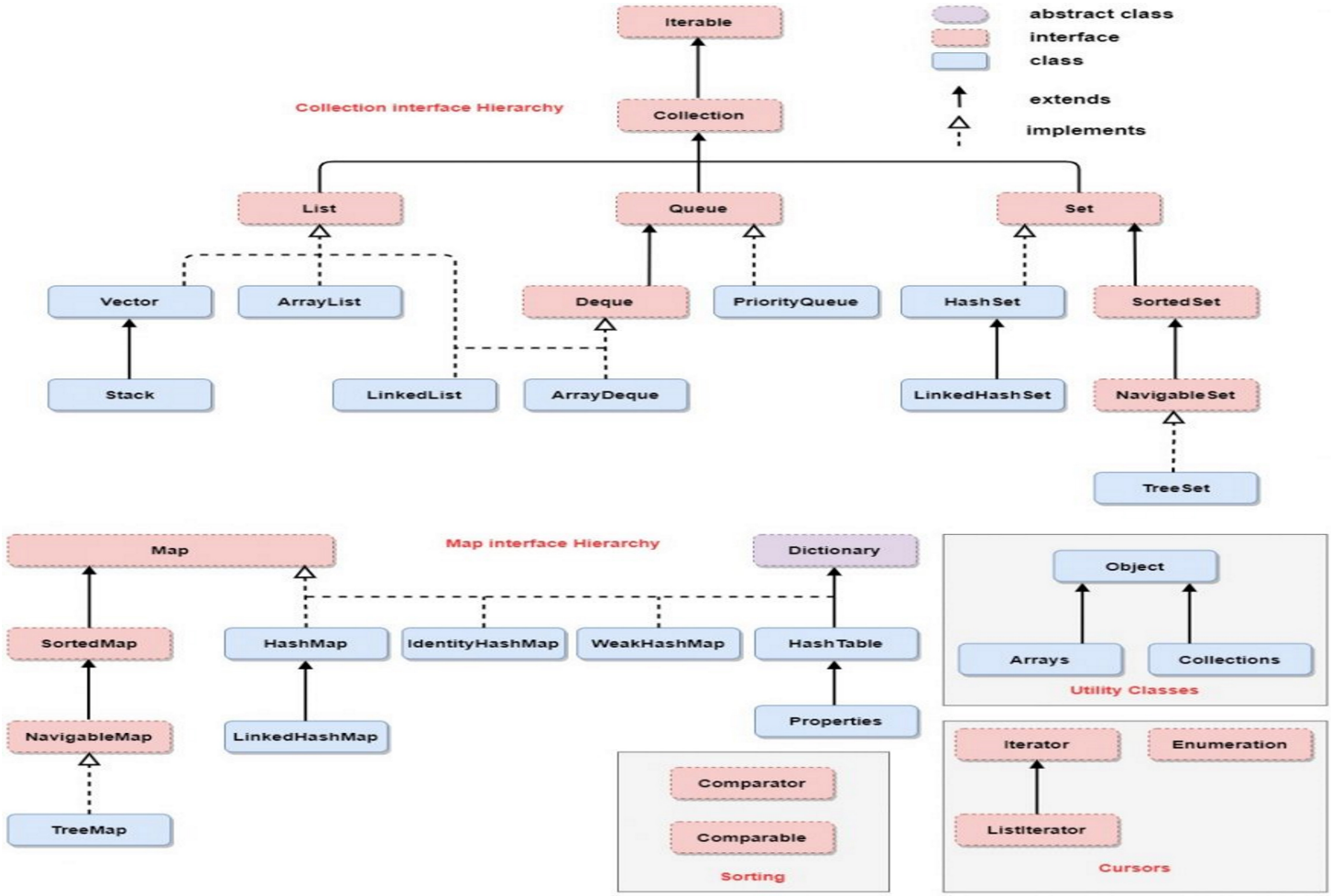
Collection Cheat Sheet

10 September 2023 23:31



Collection

To represent group of object as a single entity.

1	public boolean add(E e)	It is used to insert an element in this collection.
2	public boolean addAll(Collection<? extends E> c)	It is used to insert the specified collection elements in the invoking collection.
3	public boolean remove(Object element)	It is used to delete an element from the collection.
4	public boolean removeAll(Collection<?> c)	It is used to delete all the elements of the specified collection from the invoking collection.
5	default boolean removeIf(Predicate<? super E> filter)	It is used to delete all the elements of the collection that satisfy the specified predicate.
6	public boolean retainAll(Collection<?> c)	It is used to delete all the elements of invoking collection except the specified collection.
7	public int size()	It returns the total number of elements in the collection.
8	public void clear()	It removes the total number of elements from the collection.
9	public boolean contains(Object element)	It is used to search an element.
10	public boolean containsAll(Collection<?> c)	It is used to search the specified collection in the collection.
11	public Iterator iterator()	It returns an iterator.
12	public Object[] toArray()	It converts collection into array.
13	public <T> T[] toArray(T[] a)	It converts collection into array. Here, the runtime type of the returned array is that of the specified array.
14	public boolean isEmpty()	It checks if collection is empty.
15	default Stream<E> parallelStream()	It returns a possibly parallel Stream with the collection as its source.
16	default Stream<E> stream()	It returns a sequential Stream with the collection as its source.
17	default Spliterator<E> spliterator()	It generates a Spliterator over the specified elements in the collection.
18	public boolean equals(Object element)	It matches two collections.
19	public int hashCode()	It returns the hash code number of the collection.

List

Duplicates allowed & Insertion Order is preserved (by index)

void add(int index, E element)	It is used to insert the specified element at the specified position in a list.
boolean addAll(int index, Collection<? extends E> c)	It is used to append all the elements in the specified collection, starting at the specified position of the list.
boolean equals(Object o)	It is used to compare the specified object with the elements of a list.
E get(int index)	It is used to fetch the element from the particular position of the list.
int lastIndexOf(Object o)	It is used to return the index in this list of the last occurrence of the specified element, or -1 if the list does not contain this element.
int indexOf(Object o)	It is used to return the index in this list of the first occurrence of the specified element, or -1 if the List does not contain this element.
E remove(int index)	It is used to remove the element present at the specified position in the list.
void replaceAll(UnaryOperator<E> operator)	It is used to replace all the elements from the list with the specified element.
E set(int index, E element)	It is used to replace the specified element in the list, present at the specified position.
void sort(Comparator<? super E> c)	It is used to sort the elements of the list on the basis of specified comparator.
List<E> subList(int fromIndex, int toIndex)	It is used to fetch all the elements lies within the given range.

List contain 4 classes
ArrayList(1.2)
LinkedList (1.2)
VectorList(1.0)
Stack(1.0)

ArrayList

(For Retrievable(Search/Access) operation best)

ArrayList al = new ArrayList(); //default capacity 10 (Newcapacity = currentcapacity*(3/2)+1)
ArrayList al = new ArrayList(int cap);
ArrayList al = new ArrayList(Collection c);

Resizable or Growable array
Insertion order is preserved
Duplicate object are allowed
Heterogeneous objects are allowed
Null insertion is possible
It implement Serializable & Clonable + RandomAccess (vector also) Interface
toString() overridden //[obj1,obj2....]
Method are not Synchronized (Vector case its Synchronized)
We can use collection class SynchronizedList [public static List SynchronizedList(List l)] //Same for Map & Set
Eg: List l = Collections.SynchronizedList(al);

LinkedList

(For Insertion/Deletion operation best)

LinkedList al = new LinkedList();
LinkedList al = new LinkedList(Collection c);

Insertion order is preserved
Duplicate object are allowed
Heterogeneous objects are allowed
Null insertion is possible
It implement Serializable & Clonable Interface
Used LinkedList to implement Stack & Queue

void addFirst(Object o)
void addLast(Object o)
void removeFirst()
void removeLast()
Object getFirst()
Object getLast()

*Take care in set or add [Set will replace whereas add will add the element and shift forward the current element]

VECTOR

(For Retrievable(Search/Access) operation best)

Vector al = new Vector(); //default capacity 10 (Newcapacity = currentcapacity*2)
Vector al = new Vector(int cap);
Vector al = new Vector(int cap, int inccap);
Vector al = new Vector(Collection c);

Resizable or Growable array
Insertion order is preserved
Duplicate object are allowed
Heterogeneous objects are allowed
Null insertion is possible
It implement Serializable & Clonable + RandomAccess (vector also) Interface
toString() overridden //[obj1,obj2....]
Method are Synchronized (Array case its not Synchronized)

addElement()	It is used to append the specified component to the end of this vector. It increases the vector size by one.
capacity()	It is used to get the current capacity of this vector.
clone()	It returns a clone of this vector.
copyInto()	It is used to copy the components of the vector into the specified array.
elementAt()	It is used to get the component at the specified index.
elements()	It returns an enumeration of the components of a vector.
ensureCapacity()	It is used to increase the capacity of the vector which is in use, if necessary. It ensures that the vector can hold at least the number of components specified by the minimum capacity argument.
firstElement()	It is used to get the first component of the vector.
forEach()	It is used to perform the given action for each element of the Iterable until all elements have been processed or the action throws an exception.
get()	It is used to get an element at the specified position in the vector.
hashCode()	It is used to get the hash code value of a vector.
indexOf()	It is used to get the index of the first occurrence of the specified element in the vector. It returns -1 if the vector does not contain the element.
insertElementAt()	It is used to insert the specified object as a component in the given vector at the specified index.
isEmpty()	It is used to check if this vector has no components.
iterator()	It is used to get an iterator over the elements in the list in proper sequence.
lastElement()	It is used to get the last component of the vector.
lastIndexOf()	It is used to get the index of the last occurrence of the specified element in the vector. It returns -1 if the vector does not contain the element.
removeAllElements()	It is used to remove all elements from the vector and set the size of the vector to zero.
removeElement()	It is used to remove the first (lowest-indexed) occurrence of the argument from the vector.
removeElementAt()	It is used to delete the component at the specified index.
removeIf()	It is used to remove all of the elements of the collection that satisfy the given predicate.
removeRange()	It is used to delete all of the elements from the vector whose index is between fromIndex, inclusive and toIndex, exclusive.
replaceAll()	It is used to replace each element of the list with the result of applying the operator to that element.
retainAll()	It is used to retain only that element in the vector which is contained in the specified collection.
setElementAt()	It is used to set the component at the specified index of the vector to the specified object.
setSize()	It is used to set the size of the given vector.
size()	It is used to get the number of components in the given vector.
sort()	It is used to sort the list according to the order induced by the specified Comparator.
spliterator()	It is used to create a late-binding and fail-fast Spliterator over the elements in the list.
subList()	It is used to get a view of the portion of the list between fromIndex, inclusive, and toIndex, exclusive.
toArray()	It is used to get an array containing all of the elements in this vector in correct order.
toString()	It is used to get a string representation of the vector.
trimToSize()	It is used to trim the capacity of the vector to the vector's current size.

Stack

It is child class of vector and contain only one constructor

Stack s = new Stack();

empty()	boolean	The method checks the stack is empty or not.
push(E item)	E	The method pushes (insert) an element onto the top of the stack.
pop()	E	The method removes an element from the top of the stack and returns the same element as the value of that function.
peek()	E	The method looks at the top element of the stack without removing it.
search(Object o)	int	The method searches the specified object and returns the position of the object. (offset value return from top to down(1, 2,3...) if not present return -1.