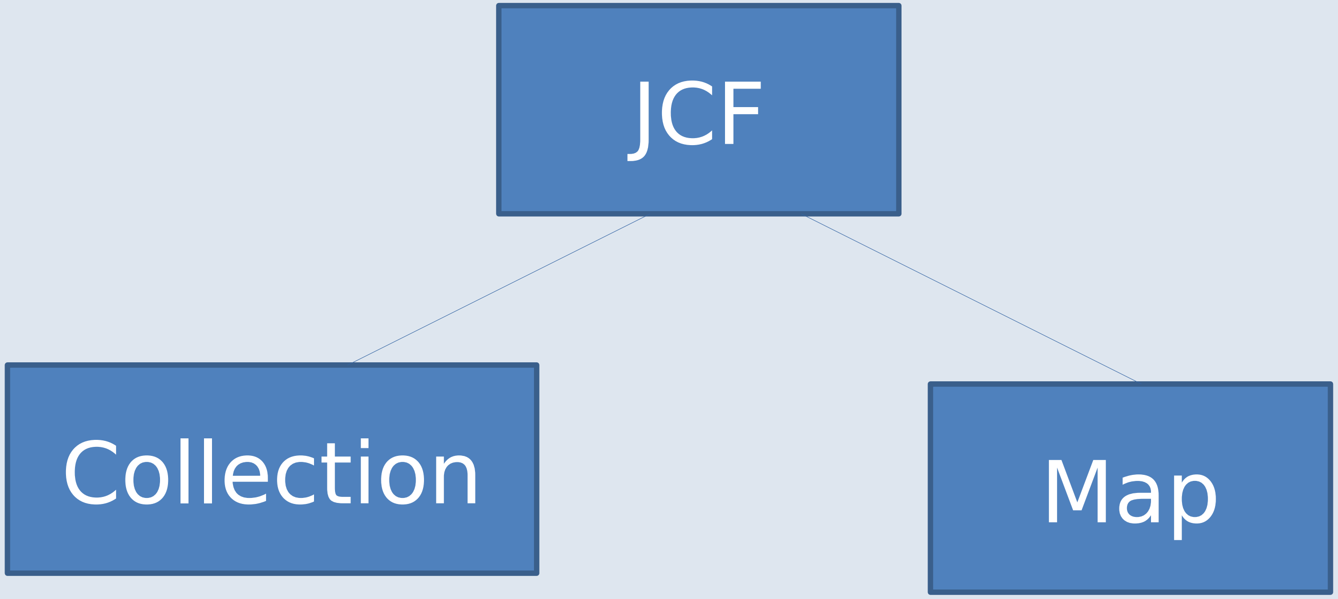
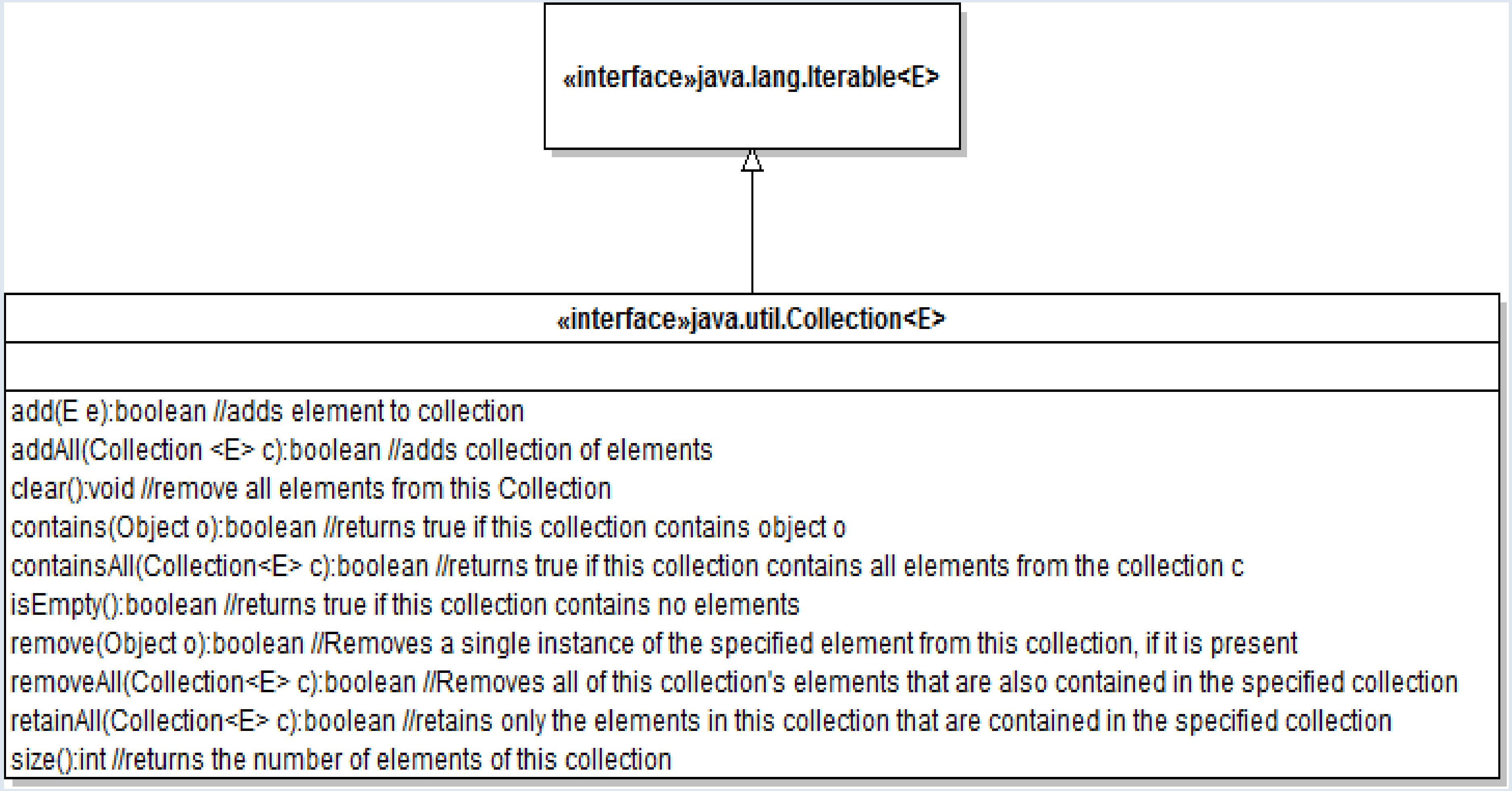
Java Collections Framework (JCF)

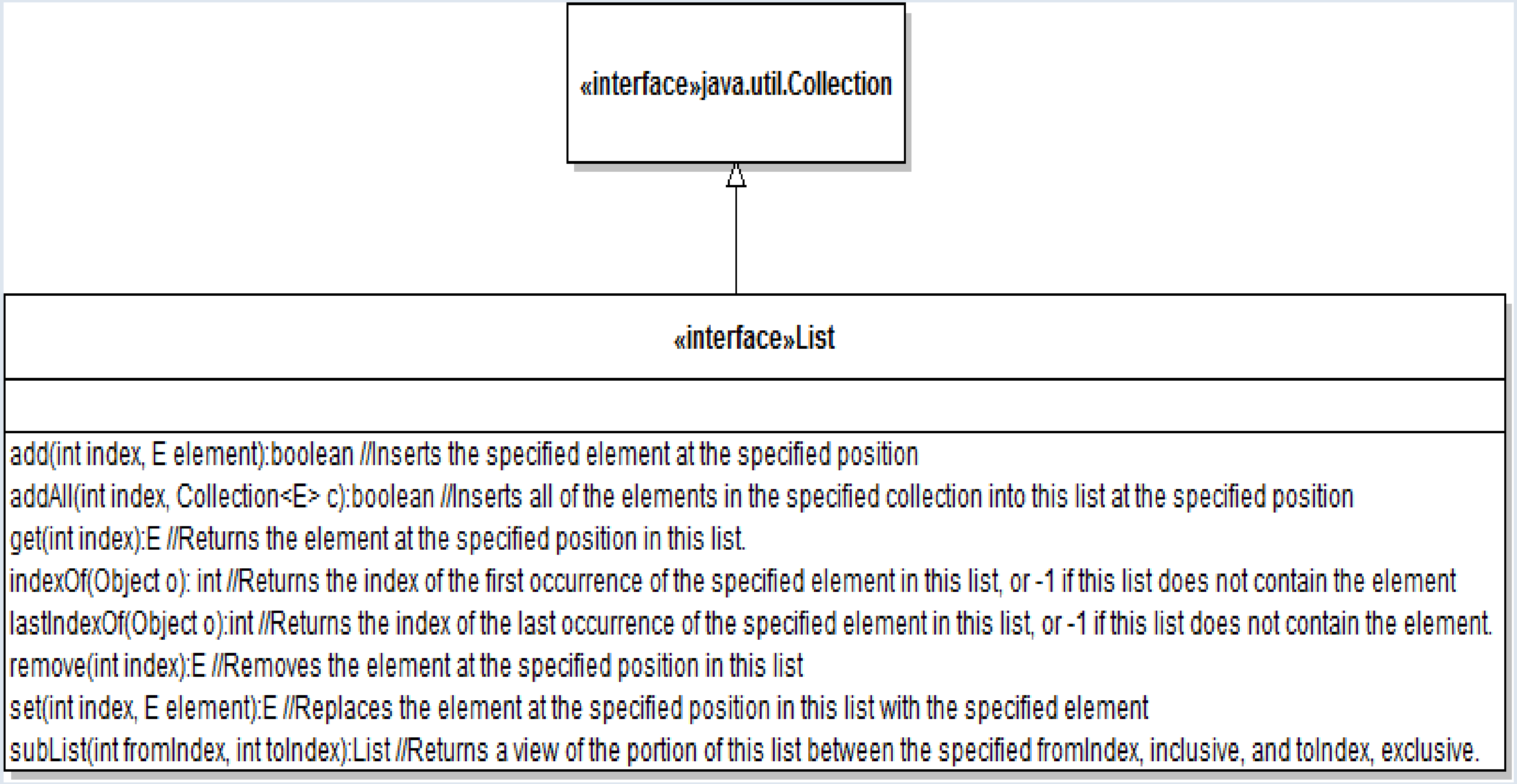


Collection


«interface»java.lang.Iterable<E>
«interface»java.util.Collection<E>

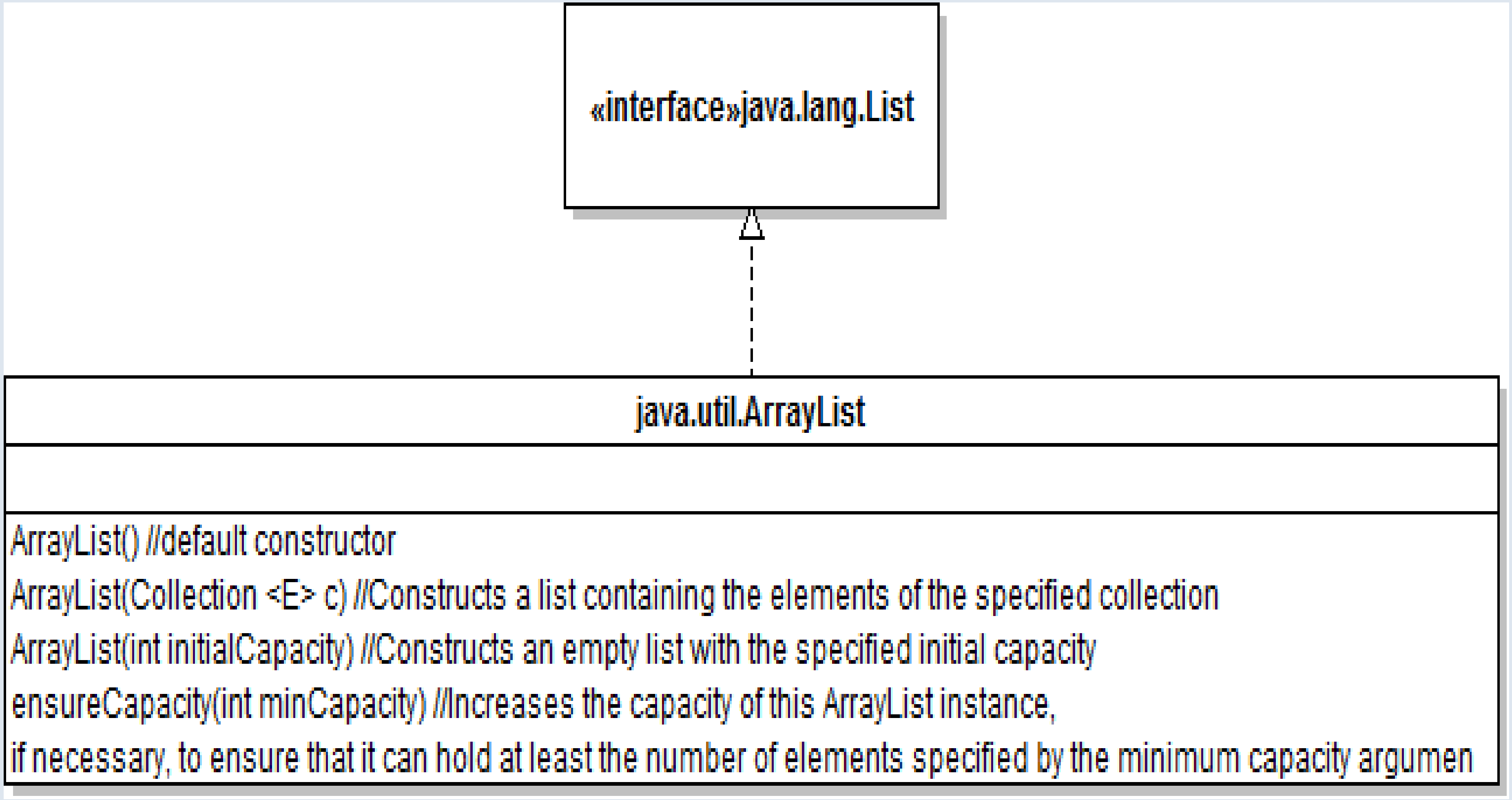
add(E e):boolean //adds element to collection
addAll(Collection <E> c):boolean //adds collection of elements
clear():void //remove all elements from this Collection
contains(Object o):boolean //returns true if this collection contains object o
containsAll(Collection<E> c):boolean //returns true if this collection contains all elements from the collection c
isEmpty():boolean //returns true if this collection contains no elements
remove(Object o):boolean //Removes a single instance of the specified element from this collection, if it is present
removeAll(Collection<E> c):boolean //Removes all of this collection's elements that are also contained in the specified collection
retainAll(Collection<E> c):boolean //retains only the elements in this collection that are contained in the specified collection
size():int //returns the number of elements of this collection

List



add(int index, E element):boolean //Inserts the specified element at the specified position
addAll(int index, Collection<E> c):boolean //Inserts all of the elements in the specified collection into this list at the specified position
get(int index):E //Returns the element at the specified position in this list.
indexOf(Object o): int //Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element
lastIndexOf(Object o):int //Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.
remove(int index):E //Removes the element at the specified position in this list
set(int index, E element):E //Replaces the element at the specified position in this list with the specified element
subList(int fromIndex, int toIndex):List //Returns a view of the portion of this list between the specified fromIndex, inclusive, and toIndex, exclusive.

ArrayList



ArrayList() //default constructor

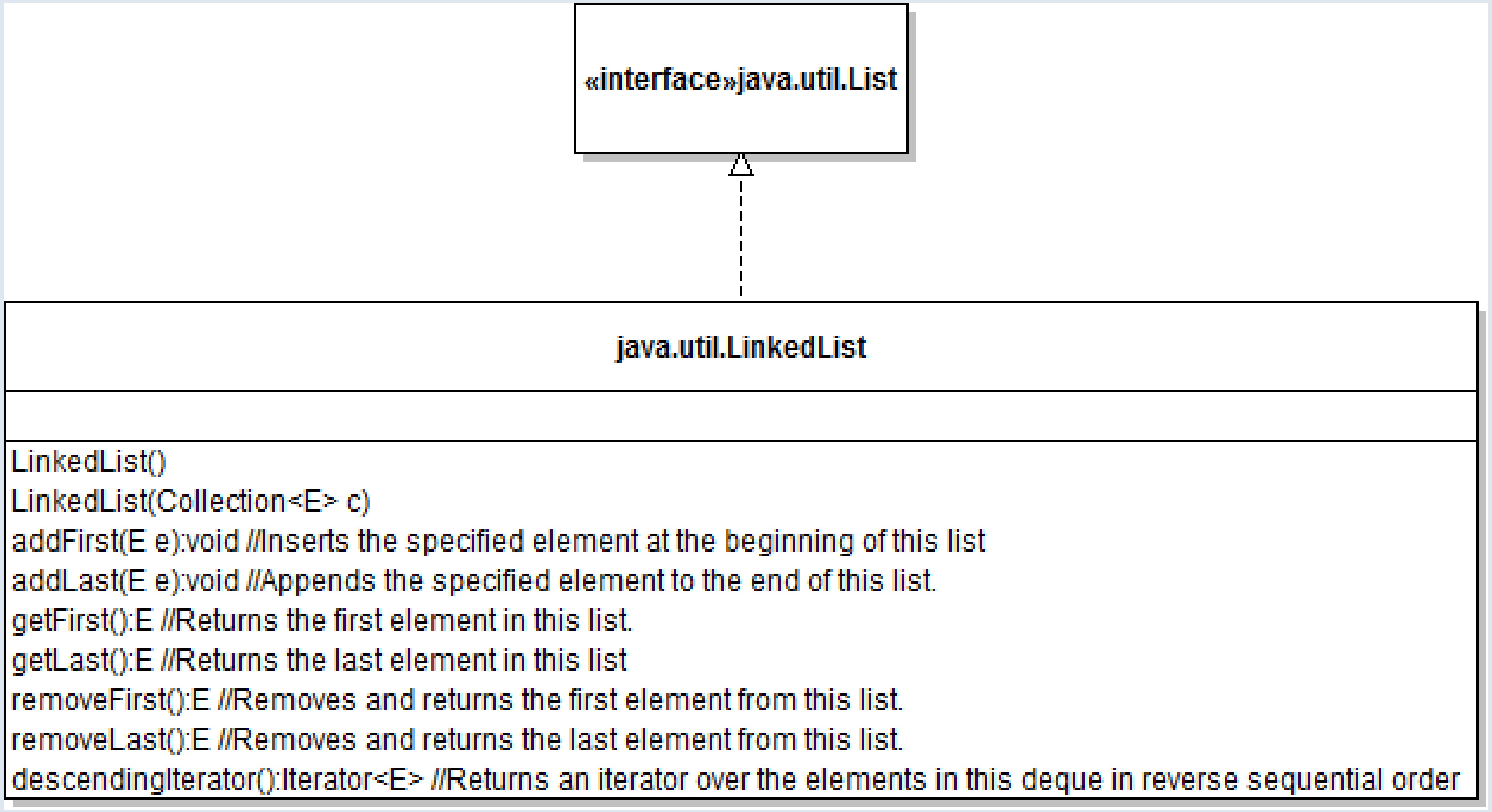
ArrayList(Collection <E> c) //Constructs a list containing the elements of the specified collection

ArrayList(int initialCapacity) //Constructs an empty list with the specified initial capacity

ensureCapacity(int minCapacity) //Increases the capacity of this ArrayList instance,

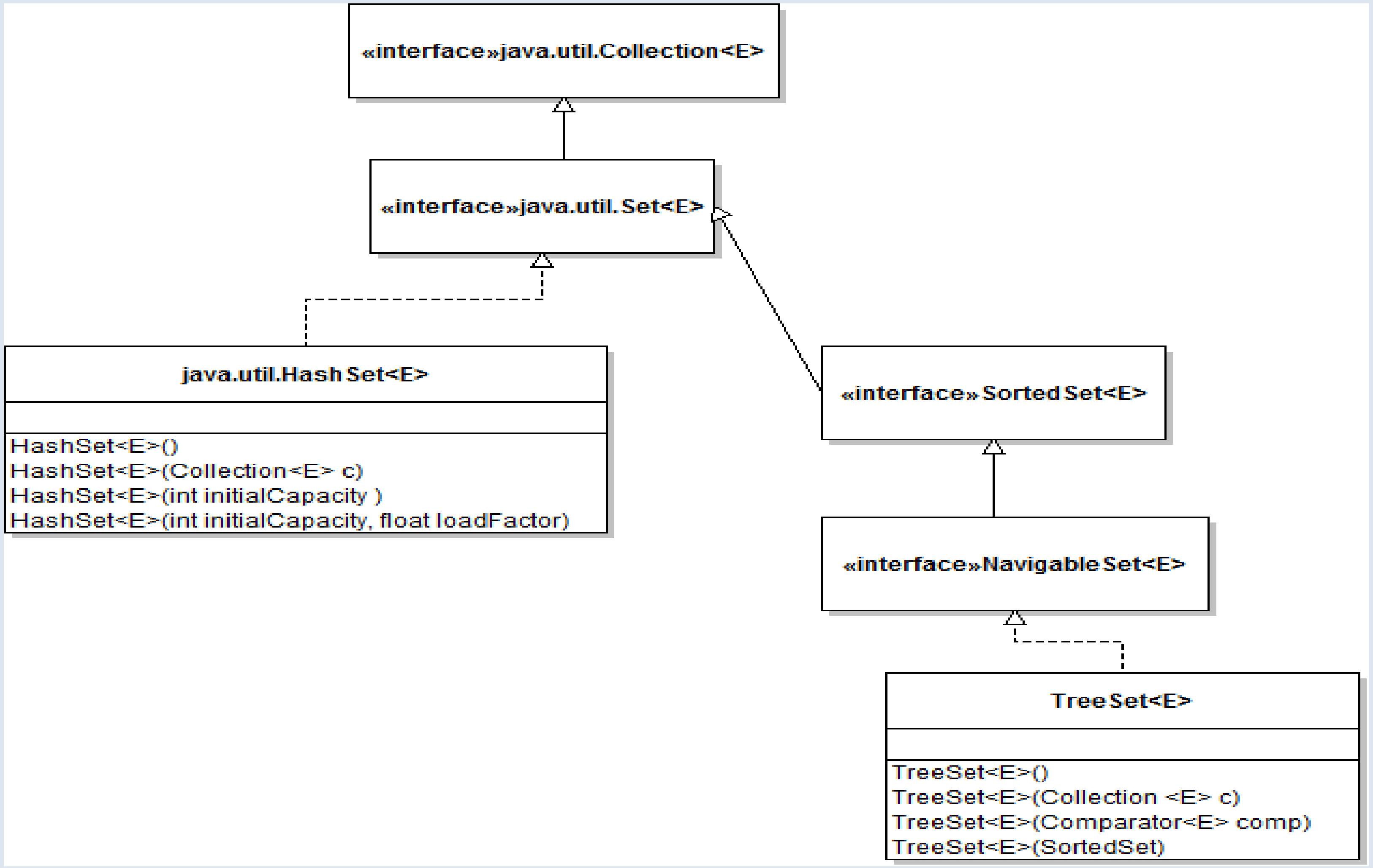
if necessary, to ensure that it can hold at least the number of elements specified by the minimum capacity argumen

LinkedList



LinkedList()
LinkedList(Collection<E> c)
addFirst(E e):void //Inserts the specified element at the beginning of this list
addLast(E e):void //Appends the specified element to the end of this list.
getFirst():E //Returns the first element in this list.
getLast():E //Returns the last element in this list
removeFirst():E //Removes and returns the first element from this list.
removeLast():E //Removes and returns the last element from this list.
descendingIterator():Iterator<E> //Returns an iterator over the elements in this deque in reverse sequential order

Set


«interface»java.util.Collection<E>
«interface»java.util.Set<E>
java.util.HashSet<E>
HashSet<E>()
HashSet<E>(Collection<E> c)
HashSet<E>(int initialCapacity)
HashSet<E>(int initialCapacity, float loadFactor)
«interface» SortedSet<E>
«interface»NavigableSet<E>
TreeSet<E>
TreeSet<E>()
TreeSet<E>(Collection <E> c)
TreeSet<E>(Comparator<E> comp)
TreeSet<E>(SortedSet)

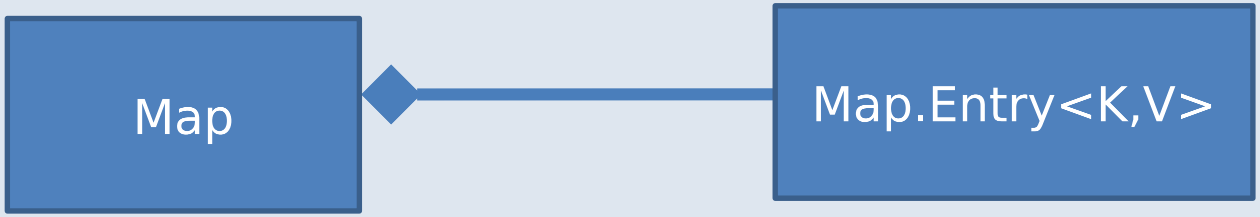
SortedSet

Comparator<? super **E**>	**comparator**() Returns the comparator used to order the elements in this set, or null if this set uses the **natural ordering** of its elements.
E	**first**() Returns the first (lowest) element currently in this set.
SortedSet<**E**>	**headSet**(**E** toElement) Returns a view of the portion of this set whose elements are strictly less than toElement.
E	**last**() Returns the last (highest) element currently in this set.
SortedSet<**E**>	**subSet**(**E** fromElement, **E** toElement) Returns a view of the portion of this set whose elements range from fromElement, inclusive, to toElement, exclusive.
SortedSet<**E**>	**tailSet**(**E** fromElement) Returns a view of the portion of this set whose elements are greater than or equal to fromElement.

NavigableSet

E	**ceiling**(**E** e) Returns the least element in this set greater than or equal to the given element, or null if there is no such element.
Iterator<**E**>	**descendingIterator**() Returns an iterator over the elements in this set, in descending order.
NavigableSet<**E**>	**descendingSet**() Returns a reverse order view of the elements contained in this set.
E	**floor**(**E** e) Returns the greatest element in this set less than or equal to the given element, or null if there is no such element.
NavigableSet<**E**>	**headSet**(**E** toElement, boolean inclusive) Returns a view of the portion of this set whose elements are less than (or equal to, if inclusive is true) toElement.
E	**higher**(**E** e) Returns the least element in this set strictly greater than the given element, or null if there is no such element.
E	**lower**(**E** e) Returns the greatest element in this set strictly less than the given element, or null if there is no such element.
E	**pollFirst**() Retrieves and removes the first (lowest) element, or returns null if this set is empty.
E	**pollLast**() Retrieves and removes the last (highest) element, or returns null if this set is empty.
NavigableSet<**E**>	**subSet**(**E** fromElement, boolean fromInclusive, **E** toElement, boolean toInclusive) Returns a view of the portion of this set whose elements range from fromElement to toElement.
NavigableSet<**E**>	**tailSet**(**E** fromElement, boolean inclusive) Returns a view of the portion of this set whose elements are greater than (or equal to, if inclusive is true) fromElement.

Map



- Mapping keys to values.
  - Map<K, V> K-type of key, V-type of value
- No duplicate keys
- It models the mathematical *function* abstraction. *V=F(K)*
- *Basic operation*
  - put, get, remove, containsKey, containsValue, size, empty
- Bulk operations
  - putAll , clear
- Collection views
  - keySet, entrySet, values

Map

void	**clear**() Removes all of the mappings from this map
boolean	**containsKey**(**Object** key) Returns true if this map contains a mapping for the specified key.
boolean	**containsValue**(**Object** value) Returns true if this map maps one or more keys to the specified value.
Set<**Map.Entry**<**K,V**>>	**entrySet**() Returns a **Set** view of the mappings contained in this map.
V	**get**(**Object** key) Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
boolean	**isEmpty**() Returns true if this map contains no key-value mappings.
Set<**K**>	**keySet**() Returns a **Set** view of the keys contained in this map.
V	**put**(**K** key, **V** value) Associates the specified value with the specified key in this map (optional operation).
void	**putAll**(**Map**<K,V> m) Copies all of the mappings from the specified map to this map
V	**remove**(**Object** key) Removes the mapping for a key from this map if it is present).
int	**size**() Returns the number of key-value mappings in this map.
Collection<**V**>	**values**() Returns a **Collection** view of the values contained in this map.

Map.Entry<K,V>

boolean	**equals**(**Object** o) Compares the specified object with this entry for equality.
K	**getKey**() Returns the key corresponding to this entry.
V	**getValue**() Returns the value corresponding to this entry.
int	**hashCode**() Returns the hash code value for this map entry.
V	**setValue**(**V** value) Replaces the value corresponding to this entry with the specified value (optional operation).

HashMap

HashMap() Constructs an empty HashMap with the default initial capacity (16) and the default load factor (0.75).
HashMap(int initialCapacity) Constructs an empty HashMap with the specified initial capacity and the default load factor (0.75).
HashMap(int initialCapacity, float loadFactor) Constructs an empty HashMap with the specified initial capacity and load factor.
HashMap(**Map**<**K**, **V**> m) Constructs a new HashMap with the same mappings as the specified Map.

TreeMap

TreeMap() Constructs a new, empty tree map, using the natural ordering of its keys.
TreeMap(**Comparator**< **K**> comparator) Constructs a new, empty tree map, ordered according to the given comparator.
TreeMap(**Map**<K, **V**> m) Constructs a new tree map containing the same mappings as the given map, ordered according to the *natural ordering* of its keys.
TreeMap(**SortedMap**<**K**, **V**> m) Constructs a new tree map containing the same mappings and using the same ordering as the specified sorted map.

TreeMap

SortedMap<**K**,**V**>	**headMap**(**K** toKey) Returns a view of the portion of this map whose keys are strictly less than toKey.
NavigableMap<**K**,**V**>	**headMap**(**K** toKey, boolean inclusive) Returns a view of the portion of this map whose keys are less than (or equal to, if inclusive is true) toKey.
NavigableMap<**K**,**V**>	**subMap**(**K** fromKey, boolean fromInclusive, **K** toKey, boolean toInclusive) Returns a view of the portion of this map whose keys range from fromKey to toKey.
SortedMap<**K**,**V**>	**subMap**(**K** fromKey, **K** toKey) Returns a view of the portion of this map whose keys range from fromKey, inclusive, to toKey, exclusive.
SortedMap<**K**,**V**>	**tailMap**(**K** fromKey) Returns a view of the portion of this map whose keys are greater than or equal to fromKey.
NavigableMap<**K**,**V**>	**tailMap**(**K** fromKey, boolean inclusive) Returns a view of the portion of this map whose keys are greater than (or equal to, if inclusive is true) fromKey.