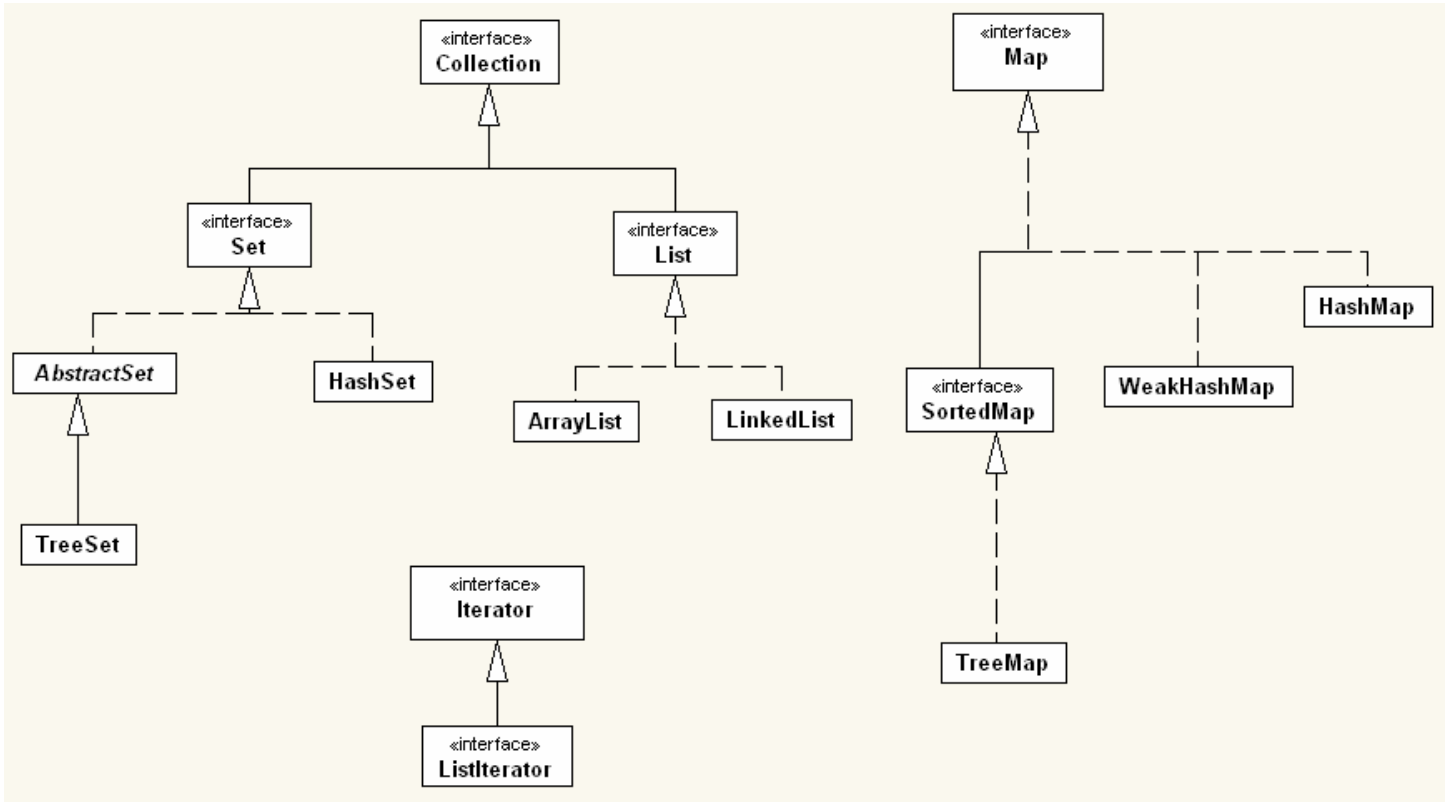
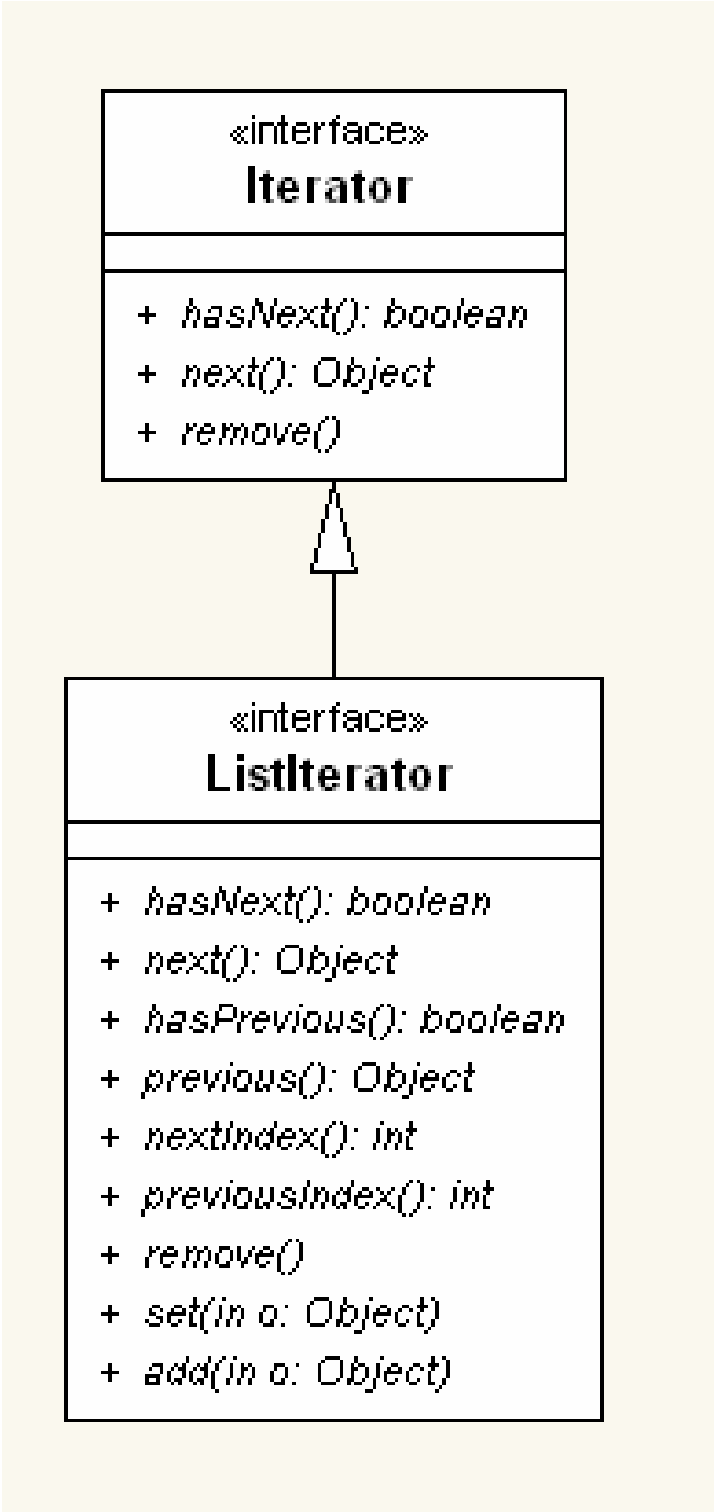

«interface»
Collection
«interface»
Map
«interface»
Set
«interface»
List
HashMap
AbstractSet
HashSet
ArrayList
LinkedList
«interface»
SortedMap
WeakHashMap
TreeSet
«interface»
Iterator
TreeMap
«interface»
ListIterator


«interface»
Collection
+ size(): int
+ isEmpty(): boolean
+ contains(in o: Object): boolean
+ iterator(): Iterator
+ toArray(): Object[]
+ toArray(in a: Object): Object[]
+ add(in o: Object): boolean
+ remove(in o: Object): boolean
+ containsAll(in c: Collection): boolean
+ addAll(in c: Collection): boolean
+ removeAll(in c: Collection): boolean
+ retainAll(in c: Collection): boolean
+ clear()
+ equals(in o: Object): boolean
+ hashCode(): int

Method Summary

boolean	**add**(Object o) Ensures that this collection contains the specified element (optional operation).
boolean	**addAll**(Collection c) Adds all of the elements in the specified collection to this collection (optional operation).
void	**clear**() Removes all of the elements from this collection (optional operation).
boolean	**contains**(Object o) Returns `true` if this collection contains the specified element.
boolean	**containsAll**(Collection c) Returns `true` if this collection contains all of the elements in the specified collection.
boolean	**equals**(Object o) Compares the specified object with this collection for equality.
int	**hashCode**() Returns the hash code value for this collection.
boolean	**isEmpty**() Returns `true` if this collection contains no elements.
Iterator	**iterator**() Returns an iterator over the elements in this collection.
boolean	**remove**(Object o) Removes a single instance of the specified element from this collection, if it is present (optional operation).
boolean	**removeAll**(Collection c) Removes all this collection's elements that are also contained in the specified collection (optional operation).
boolean	**retainAll**(Collection c) Retains only the elements in this collection that are contained in the specified collection (optional operation).
int	**size**() Returns the number of elements in this collection.
Object[]	**toArray**() Returns an array containing all of the elements in this collection.
Object[]	**toArray**(Object[] a) Returns an array containing all of the elements in this collection; the runtime type of the returned array is that of the specified array.


«interface»
Iterator
+ hasNext(): boolean
+ next(): Object
+ remove()
«interface»
ListIterator
+ hasNext(): boolean
+ next(): Object
+ hasPrevious(): boolean
+ previous(): Object
+ nextIndex(): int
+ previousIndex(): int
+ remove()
+ set(in o: Object)
+ add(in o: Object)

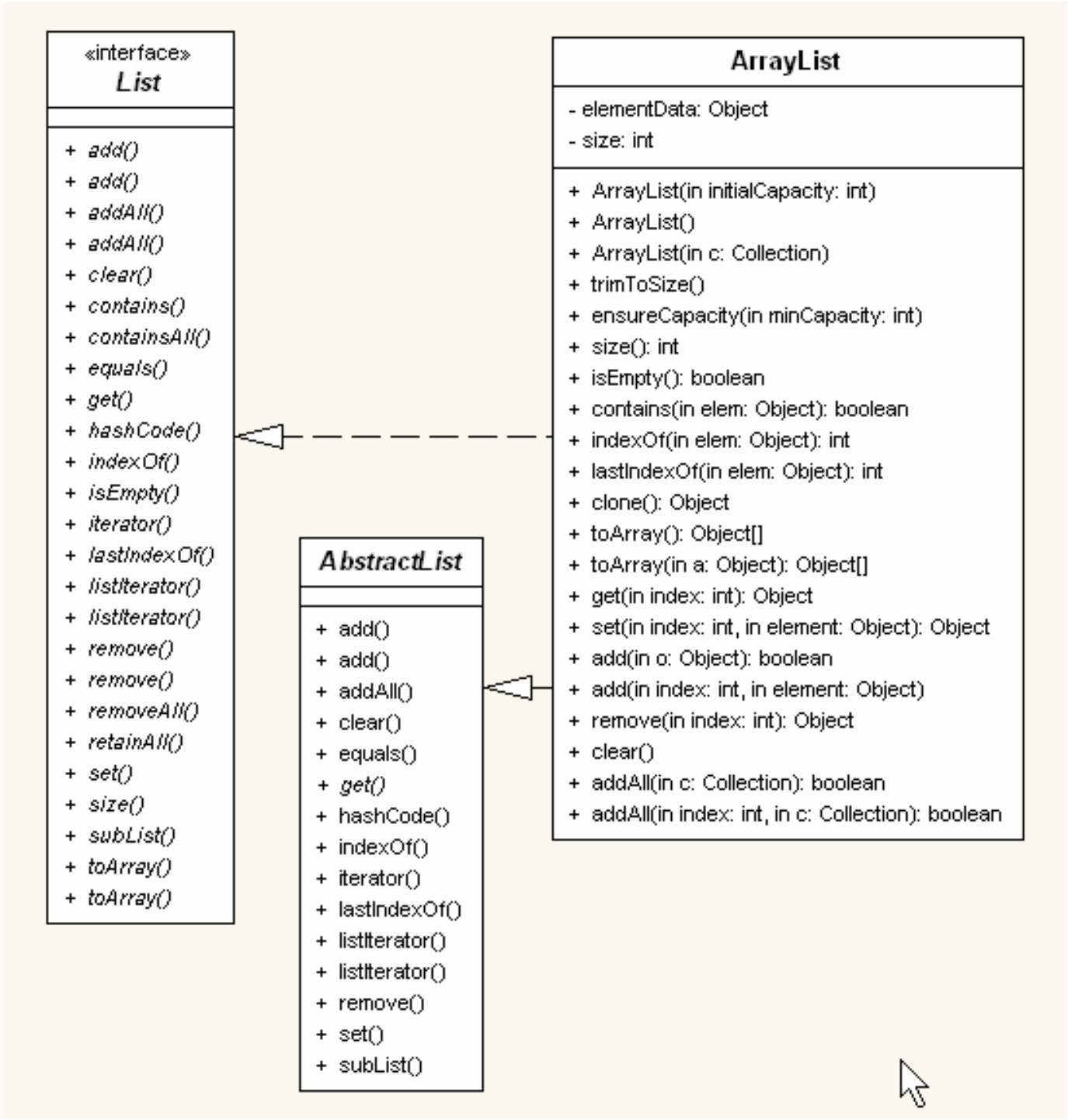
Method Summary

boolean	**hasNext**() Returns `true` if the iteration has more elements.
Object	**next**() Returns the next element in the iteration.
void	**remove**() Removes from the underlying collection the last element returned by the iterator (optional operation).

Method Summary

void	**add**(Object o) Inserts the specified element into the list (optional operation).
boolean	**hasNext**() Returns `true` if this list iterator has more elements when traversing the list in the forward direction.
boolean	**hasPrevious**() Returns `true` if this list iterator has more elements when traversing the list in the reverse direction.
Object	**next**() Returns the next element in the list.
int	**nextIndex**() Returns the index of the element that would be returned by a subsequent call to `next`.
Object	**previous**() Returns the previous element in the list.
int	**previousIndex**() Returns the index of the element that would be returned by a subsequent call to `previous`.
void	**remove**() Removes from the list the last element that was returned by `next` or `previous` (optional operation).
void	**set**(Object o) Replaces the last element returned by `next` or `previous` with the specified element (optional operation).

La clase ArrayList



Constructor Summary

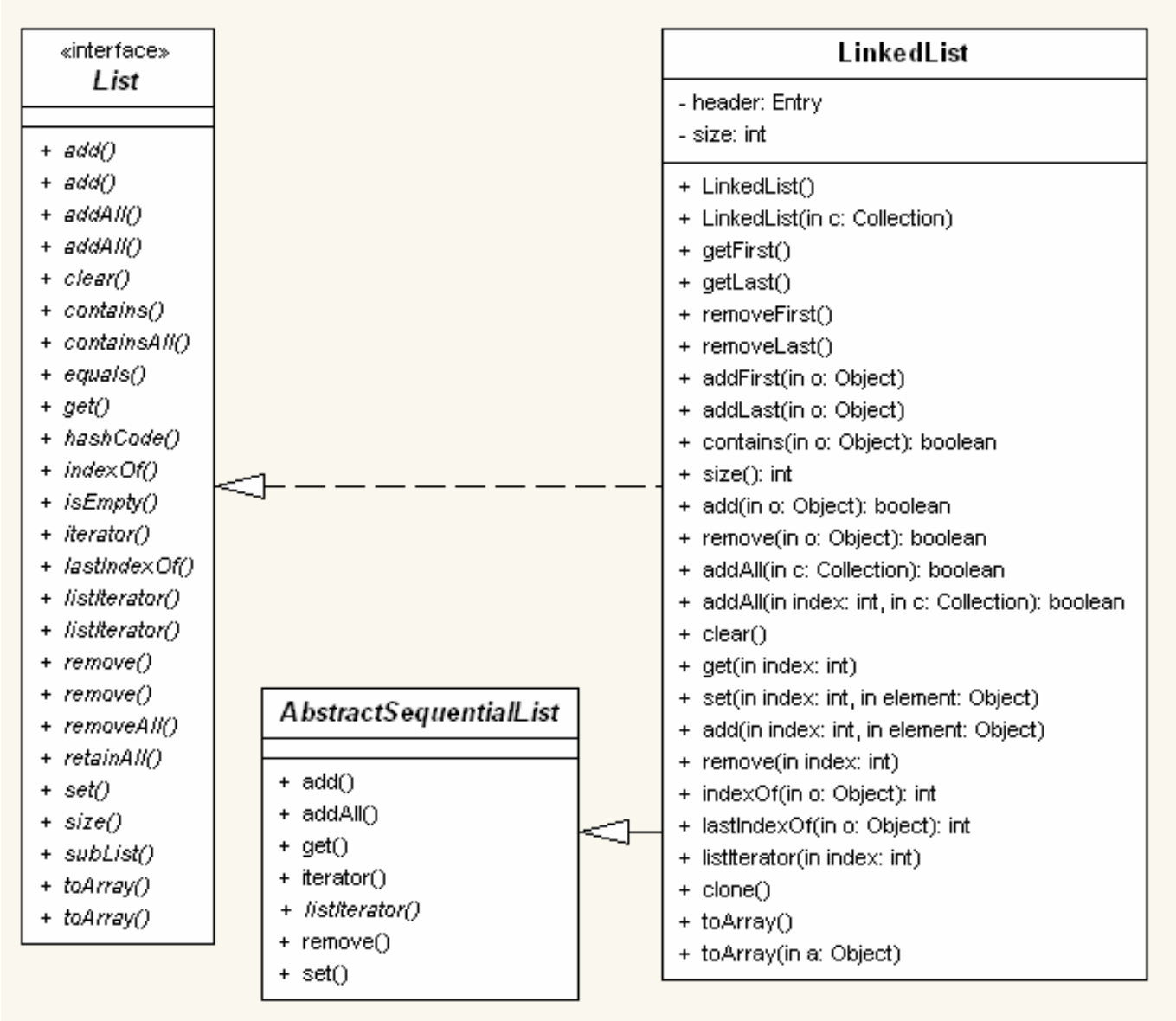
ArrayList() Constructs an empty list with an initial capacity of ten.
ArrayList(Collection c) Constructs a list containing the elements of the specified collection, in the order they are returned by the collection's iterator.
ArrayList(int initialCapacity) Constructs an empty list with the specified initial capacity.

Method Summary

void	**add**(int index, Object element) Inserts the specified element at the specified position in this list.
boolean	**add**(Object o) Appends the specified element to the end of this list.
boolean	**addAll**(Collection c) Appends all of the elements in the specified Collection to the end of this list, in the order that they are returned by the specified Collection's Iterator.
boolean	**addAll**(int index, Collection c) Inserts all of the elements in the specified Collection into this list, starting at the specified position.
void	**clear**() Removes all of the elements from this list.
Object	**clone**() Returns a shallow copy of this ArrayList instance.
boolean	**contains**(Object elem) Returns true if this list contains the specified element.
void	**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 argument.
Object	**get**(int index) Returns the element at the specified position in this list.
int	**indexOf**(Object elem) Searches for the first occurence of the given argument, testing for equality using the equals method.

boolean	**isEmpty**() Tests if this list has no elements.
int	**lastIndexOf**(Object elem) Returns the index of the last occurrence of the specified object in this list.
Object	**remove**(int index) Removes the element at the specified position in this list.
protected void	**removeRange**(int fromIndex, int toIndex) Removes from this List all of the elements whose index is between fromIndex, inclusive and toIndex, exclusive.
Object	**set**(int index, Object element) Replaces the element at the specified position in this list with the specified element.
int	**size**() Returns the number of elements in this list.
Object[]	**toArray**() Returns an array containing all of the elements in this list in the correct order.
Object[]	**toArray**(Object[] a) Returns an array containing all of the elements in this list in the correct order; the runtime type of the returned array is that of the specified array.
void	**trimToSize**() Trims the capacity of this ArrayList instance to be the list's current size.

La clase LinkedList



Constructor Summary

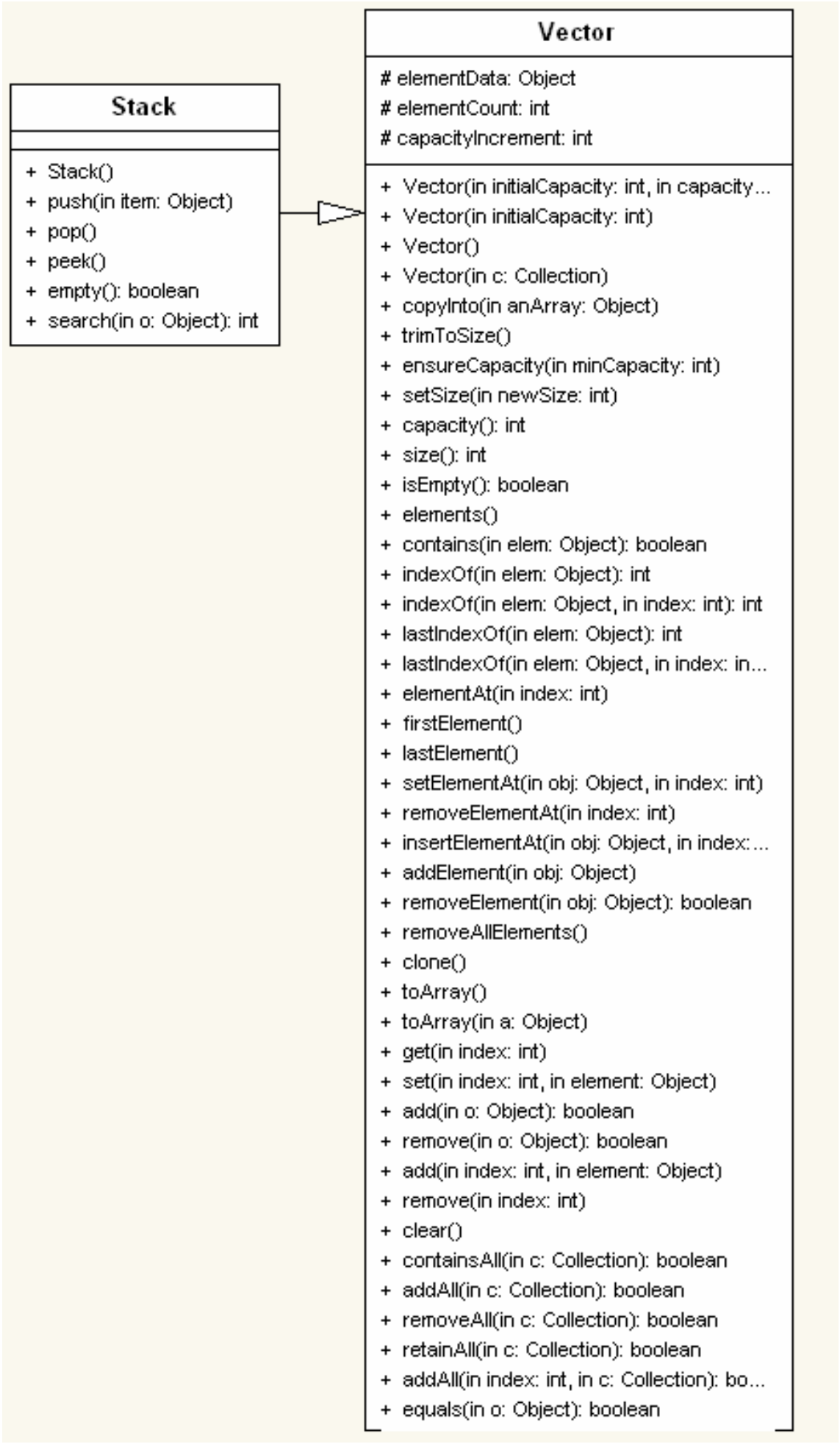
LinkedList() Constructs an empty list.
LinkedList(Collection c) Constructs a list containing the elements of the specified collection, in the order they are returned by the collection's iterator.

Method Summary

void	**add**(int index, Object element) Inserts the specified element at the specified position in this list.
boolean	**add**(Object o) Appends the specified element to the end of this list.
boolean	**addAll**(Collection c) Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's iterator.
boolean	**addAll**(int index, Collection c) Inserts all of the elements in the specified collection into this list, starting at the specified position.
void	**addFirst**(Object o) Inserts the given element at the beginning of this list.
void	**addLast**(Object o) Appends the given element to the end of this list.
void	**clear**() Removes all of the elements from this list.
Object	**clone**() Returns a shallow copy of this LinkedList.
boolean	**contains**(Object o) Returns true if this list contains the specified element.
Object	**get**(int index) Returns the element at the specified position in this list.
Object	**getFirst**() Returns the first element in this list.

Object	**getLast**() Returns the last element in this list.
int	**indexOf**(Object o) Returns the index in this list of the first occurrence of the specified element, or -1 if the List does not contain this element.
int	**lastIndexOf**(Object o) Returns the index in this list of the last occurrence of the specified element, or -1 if the list does not contain this element.
ListIterator	**listIterator**(int index) Returns a list-iterator of the elements in this list (in proper sequence), starting at the specified position in the list.
Object	**remove**(int index) Removes the element at the specified position in this list.
boolean	**remove**(Object o) Removes the first occurrence of the specified element in this list.
Object	**removeFirst**() Removes and returns the first element from this list.
Object	**removeLast**() Removes and returns the last element from this list.
Object	**set**(int index, Object element) Replaces the element at the specified position in this list with the specified element.
int	**size**() Returns the number of elements in this list.
Object[]	**toArray**() Returns an array containing all of the elements in this list in the correct order.
Object[]	**toArray**(Object[] a) Returns an array containing all of the elements in this list in the correct order; the runtime type of the returned array is that of the specified array.

Las clases Stack y Vector



Constructor Summary

Stack() Creates an empty Stack.

Method Summary

boolean	**empty**() Tests if this stack is empty.
Object	**peek**() Looks at the object at the top of this stack without removing it from the stack.
Object	**pop**() Removes the object at the top of this stack and returns that object as the value of this function.
Object	**push**(Object item) Pushes an item onto the top of this stack.
int	**search**(Object o) Returns the 1-based position where an object is on this stack.

Constructor Summary

Vector() Constructs an empty vector so that its internal data array has size 10 and its standard capacity increment is zero.
Vector(Collection c) Constructs a vector containing the elements of the specified collection, in the order they are returned by the collection's iterator.
Vector(int initialCapacity) Constructs an empty vector with the specified initial capacity and with its capacity increment equal to zero.
Vector(int initialCapacity, int capacityIncrement) Constructs an empty vector with the specified initial capacity and capacity increment.

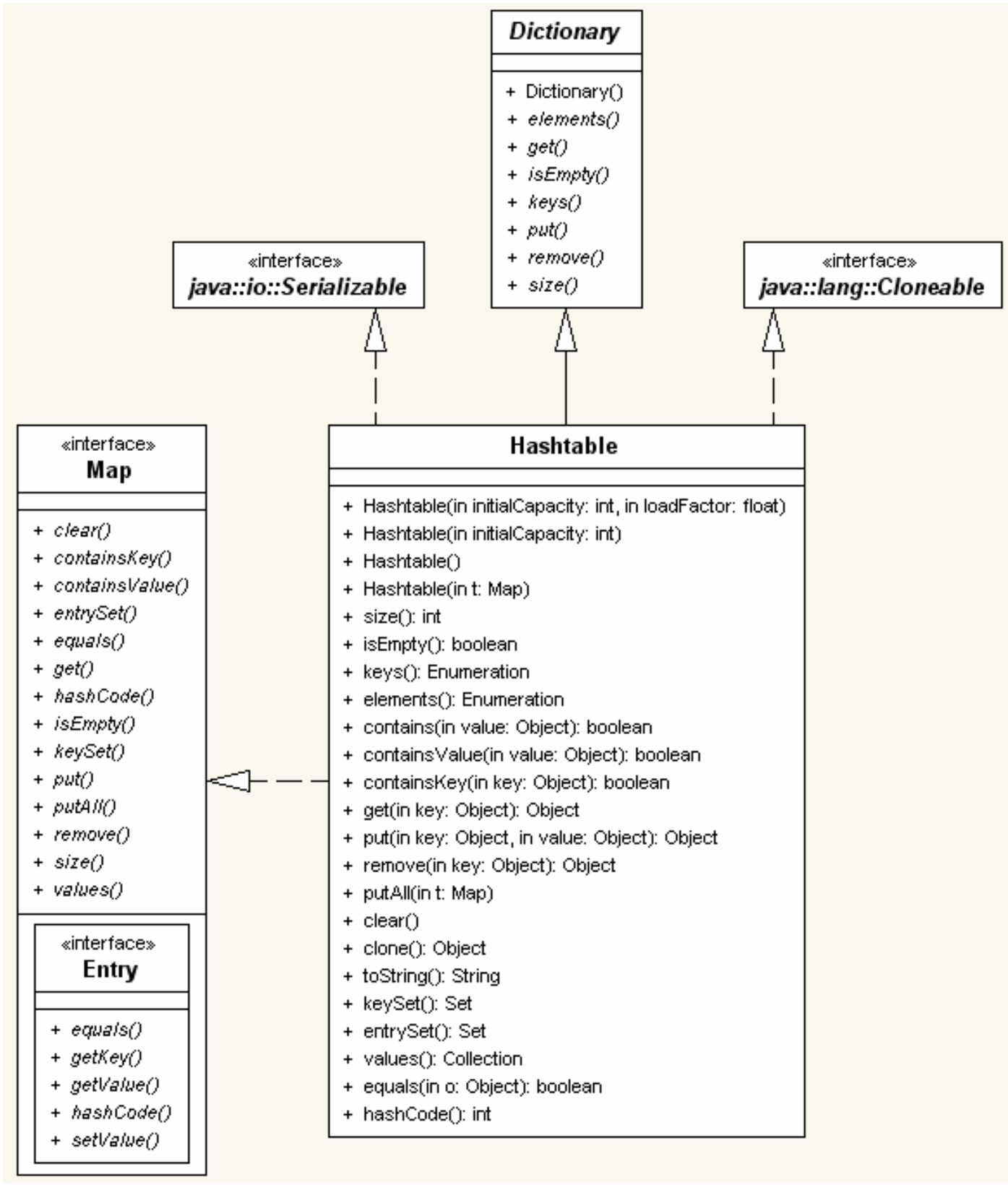
Method Summary

void	**add**(int index, Object element) Inserts the specified element at the specified position in this Vector.
boolean	**add**(Object o) Appends the specified element to the end of this Vector.
boolean	**addAll**(Collection c) Appends all of the elements in the specified Collection to the end of this Vector, in the order that they are returned by the specified Collection's Iterator.
boolean	**addAll**(int index, Collection c) Inserts all of the elements in in the specified Collection into this Vector at the specified position.
void	**addElement**(Object obj) Adds the specified component to the end of this vector, increasing its size by one.
int	**capacity**() Returns the current capacity of this vector.
void	**clear**() Removes all of the elements from this Vector.
Object	**clone**() Returns a clone of this vector.

Object	**clone**() Returns a clone of this vector.
boolean	**contains**(Object elem) Tests if the specified object is a component in this vector.
boolean	**containsAll**(Collection c) Returns true if this Vector contains all of the elements in the specified Collection.
void	**copyInto**(Object[] anArray) Copies the components of this vector into the specified array.
Object	**elementAt**(int index) Returns the component at the specified index.
Enumeration	**elements**() Returns an enumeration of the components of this vector.
void	**ensureCapacity**(int minCapacity) Increases the capacity of this vector, if necessary, to ensure that it can hold at least the number of components specified by the minimum capacity argument.
boolean	**equals**(Object o) Compares the specified Object with this Vector for equality.
Object	**firstElement**() Returns the first component (the item at index 0) of this vector.
Object	**get**(int index) Returns the element at the specified position in this Vector.
int	**hashCode**() Returns the hash code value for this Vector.
int	**indexOf**(Object elem) Searches for the first occurence of the given argument, testing for equality using the equals method.
int	**indexOf**(Object elem, int index) Searches for the first occurence of the given argument, beginning the search at index, and testing for equality using the equals method.
void	**insertElementAt**(Object obj, int index) Inserts the specified object as a component in this vector at the specified index.

boolean	**isEmpty**() Tests if this vector has no components.
Object	**lastElement**() Returns the last component of the vector.
int	**lastIndexOf**(Object elem) Returns the index of the last occurrence of the specified object in this vector.
int	**lastIndexOf**(Object elem, int index) Searches backwards for the specified object, starting from the specified index, and returns an index to it.
Object	**remove**(int index) Removes the element at the specified position in this Vector.
boolean	**remove**(Object o) Removes the first occurrence of the specified element in this Vector If the Vector does not contain the element, it is unchanged.
boolean	**removeAll**(Collection c) Removes from this Vector all of its elements that are contained in the specified Collection.
void	**removeAllElements**() Removes all components from this vector and sets its size to zero.

boolean	**removeElement**(Object obj) Removes the first (lowest-indexed) occurrence of the argument from this vector.
void	**removeElementAt**(int index) Deletes the component at the specified index.
protected void	**removeRange**(int fromIndex, int toIndex) Removes from this List all of the elements whose index is between fromIndex, inclusive and toIndex, exclusive.
boolean	**retainAll**(Collection c) Retains only the elements in this Vector that are contained in the specified Collection.
Object	**set**(int index, Object element) Replaces the element at the specified position in this Vector with the specified element.
void	**setElementAt**(Object obj, int index) Sets the component at the specified index of this vector to be the specified object.
void	**setSize**(int newSize) Sets the size of this vector.
int	**size**() Returns the number of components in this vector.
List	**subList**(int fromIndex, int toIndex) Returns a view of the portion of this List between fromIndex, inclusive, and toIndex, exclusive.
Object[]	**toArray**() Returns an array containing all of the elements in this Vector in the correct order.
Object[]	**toArray**(Object[] a) Returns an array containing all of the elements in this Vector in the correct order; the runtime type of the returned array is that of the specified array.
String	**toString**() Returns a string representation of this Vector, containing the String representation of each element.
void	**trimToSize**() Trims the capacity of this vector to be the vector's current size.


Dictionary
+ Dictionary()
+ elements()
+ get()
+ isEmpty()
+ keys()
+ put()
+ remove()
+ size()
«interface»
java::io::Serializable
«interface»
java::lang::Cloneable
«interface»
Map
+ clear()
+ containsKey()
+ containsValue()
+ entrySet()
+ equals()
+ get()
+ hashCode()
+ isEmpty()
+ keySet()
+ put()
+ putAll()
+ remove()
+ size()
+ values()
«interface»
Entry
+ equals()
+ getKey()
+ getValue()
+ hashCode()
+ setValue()
Hashtable
+ Hashtable(in initialCapacity: int, in loadFactor: float)
+ Hashtable(in initialCapacity: int)
+ Hashtable()
+ Hashtable(in t: Map)
+ size(): int
+ isEmpty(): boolean
+ keys(): Enumeration
+ elements(): Enumeration
+ contains(in value: Object): boolean
+ containsValue(in value: Object): boolean
+ containsKey(in key: Object): boolean
+ get(in key: Object): Object
+ put(in key: Object, in value: Object): Object
+ remove(in key: Object): Object
+ putAll(in t: Map)
+ clear()
+ clone(): Object
+ toString(): String
+ keySet(): Set
+ entrySet(): Set
+ values(): Collection
+ equals(in o: Object): boolean
+ hashCode(): int

Constructor Summary

Hashtable() Constructs a new, empty hashtable with a default initial capacity (11) and load factor, which is 0.75.
Hashtable(int initialCapacity) Constructs a new, empty hashtable with the specified initial capacity and default load factor, which is 0.75.
Hashtable(int initialCapacity, float loadFactor) Constructs a new, empty hashtable with the specified initial capacity and the specified load factor.
Hashtable(Map t) Constructs a new hashtable with the same mappings as the given Map.

Method Summary

void	**clear**() Clears this hashtable so that it contains no keys.
Object	**clone**() Creates a shallow copy of this hashtable.
boolean	**contains**(Object value) Tests if some key maps into the specified value in this hashtable.
boolean	**containsKey**(Object key) Tests if the specified object is a key in this hashtable.
boolean	**containsValue**(Object value) Returns true if this Hashtable maps one or more keys to this value.
Enumeration	**elements**() Returns an enumeration of the values in this hashtable.
Set	**entrySet**() Returns a Set view of the entries contained in this Hashtable.
boolean	**equals**(Object o) Compares the specified Object with this Map for equality, as per the definition in the Map interface.
Object	**get**(Object key) Returns the value to which the specified key is mapped in this hashtable.
int	**hashCode**() Returns the hash code value for this Map as per the definition in the Map interface.

boolean	**isEmpty**() Tests if this hashtable maps no keys to values.
Enumeration	**keys**() Returns an enumeration of the keys in this hashtable.
Set	**keySet**() Returns a Set view of the keys contained in this Hashtable.
Object	**put**(Object key, Object value) Maps the specified key to the specified value in this hashtable.
void	**putAll**(Map t) Copies all of the mappings from the specified Map to this Hashtable These mappings will replace any mappings that this Hashtable had for any of the keys currently in the specified Map.
protected void	**rehash**() Increases the capacity of and internally reorganizes this hashtable, in order to accommodate and access its entries more efficiently.
Object	**remove**(Object key) Removes the key (and its corresponding value) from this hashtable.
int	**size**() Returns the number of keys in this hashtable.
String	**toString**() Returns a string representation of this Hashtable object in the form of a set of entries, enclosed in braces and separated by the ASCII characters ", " (comma and space).
Collection	**values**() Returns a Collection view of the values contained in this Hashtable.