```
// spécification du type List<E>
// E est le type des éléments de la liste

public interface List<E> {
  // Opérations élémentaires

  // Returns the number of elements in this list.
  int           size();

  // Returns true if this list contains no elements.
  boolean       isEmpty();

  // Returns true if this list contains the specified element.
  boolean       contains(E element);

  // Appends the specified element to the end of this list.
  boolean       add(E element);

  // Removes the first occurrence of the specified element from this list, if
  // it is present.
  boolean       remove(E element);

  // opérations "de masse"

  // Returns true if this list contains all of the elements of the specified
  // collection.
  boolean       containsAll(Collection<E> 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(Collection<E> c);

  // Removes from this list all of its elements that are contained in the
  // specified collection.
  boolean       removeAll(Collection<E> c);

  // Retains only the elements in this list that are contained in the
  // specified collection.
  boolean       retainAll(Collection<E> c);

  // Removes all of the elements from this list.
  void          clear();

  // itération

  // Returns an iterator over the elements in this list.
  Iterator<E>           iterator();

  // Returns a list iterator over the elements in this list.
  ListIterator<E>       listIterator();

  // Returns a list iterator over the elements in this list, starting at the
  // specified position in the list.
  ListIterator<E>       listIterator(int index);                        //
```



```
  // autres opérations spécifiques liste

  // méthodes à accès direct

  // Returns the element at the specified position in this list.
  E             get(int index);

  // Replaces the element at the specified position in this list with the
  // specified element.
  E             set(int index, E element);

  // Returns the index of the first occurrence of the specified element in
  // this list, or -1 if this list does not contain the element.
  int           indexOf(E element);

  // Returns the index of the last occurrence of the specified element in this
  // list, or -1 if this list does not contain the element.
  int           lastIndexOf(E element);

  // Inserts the specified element at the specified position in this list.
  void          add(int index, E element);

  // Inserts all of the elements in the specified collection into this list at
  // the specified position.
  boolean       addAll(int index, Collection<E> c);

  // Removes the element at the specified position in this list.
  E             remove(int index);
}
```

```
// spécification du type Iterator<E>
// E est le type des éléments de la collection à parcourir

public interface Iterator<E> {
  // Returns true if the iteration has more elements.
  boolean       hasNext();

  // Returns the next element in the iteration.
  E     next();

  // Removes from the underlying collection the last element returned by this
  // iterator.
  void  remove();
}


// spécification du type ListIterator<E>
// E est le type des éléments de la collection à parcourir

public interface ListIterator<E> {

  // Returns true if this list iterator has more elements when traversing the
  // list in the forward direction.
  boolean       hasNext();

  // Returns the next element in the list and advances the cursor position.
  E     next();

  // Returns true if this list iterator has more elements when traversing the
  // list in the reverse direction.
  boolean       hasPrevious();

  // Returns the previous element in the list and moves the cursor position
  // backwards.
  E     previous();

  // modification de la liste

  // Inserts the specified element into the list.
  void  add(E e);

  // Replaces the last element returned by next() or previous() with the
  // specified element.
  void  set(E e);

  // Removes from the list the last element that was returned by next() or
  // previous().
  void  remove();

  // opérations avec indice

  // Returns the index of the element that would be returned by a subsequent
  // call to next().
  int   nextIndex();

  // Returns the index of the element that would be returned by a subsequent
  // call to previous().
  int   previousIndex();
}
```



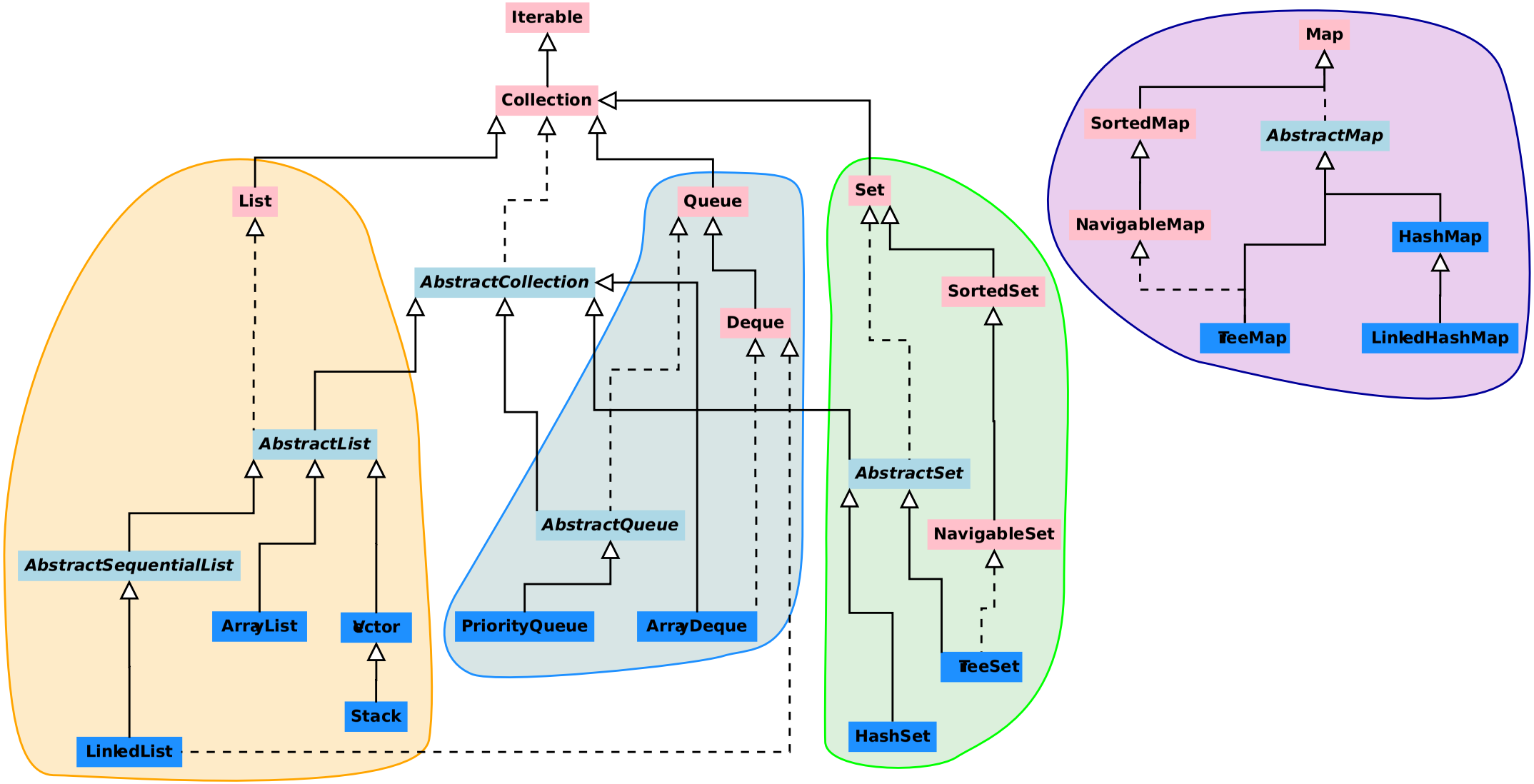
```
// spécification du type Set<E>
// E est le type des éléments de l'ensemble

public interface Set<E> {
  // Opérations élémentaires

  // Returns the number of elements in this set (its cardinality).
  int   size();

  // Returns true if this set contains no elements.
  boolean       isEmpty();

  // Returns true if this set contains the specified element.
  boolean       contains(E element);

  // Adds the specified element to this set if it is not already present
  boolean       add(E element);

  // Removes the specified element from this set if it is present
  boolean       remove(E element);

  // opérations "de masse"

  // Returns true if this set contains all of the elements of the specified
  // collection.
  boolean       containsAll(Collection<E> c);

  // Adds all of the elements in the specified collection to this set if
  // they're not already present.
  boolean       addAll(Collection<E> c);

  // Removes from this set all of its elements that are contained in the
  // specified collection.
  boolean       removeAll(Collection<E> c);

  // Retains only the elements in this set that are contained in the specified
  // collection.
  boolean       retainAll(Collection<E> c);

  // Removes all of the elements from this set (optional operation).
  void  clear();

  // itération

  // Returns an iterator over the elements in this set.
  Iterator<E>   iterator();
}
```

Dec 08, 10 9:14 **Queue.java** Page 1/1

```
// spécification du type Queue<E>
// E est le type des éléments de la file

public interface Queue<E> {
  // Opérations élémentaires

  // Returns the number of elements in this queue.
  int           size();

  // Returns true if this queue contains no elements.
  boolean       isEmpty();

  // Inserts the specified element into this queue if it is possible to do so
  // immediately without violating capacity restrictions.
  // Returns true if the element was added to this queue, else false
  boolean       offer(E e);

  // Retrieves, but does not remove, the head of this queue, or returns null
  // if this queue is empty.
  E     peek();

  // Retrieves and removes the head of this queue, or returns null if this
  // queue is empty.
  E     poll();

  // Removes all of the elements from this queue.
  void          clear();

  // itération

  // Returns an iterator over the elements in this queue.
  Iterator<E>           iterator();
}
```

Oct 18, 11 9:51 **Map.java,Map.Entry.java** Page 1/1

```
// spécification du type Map<K, V>
// K est le type des clés
// V est le type des valeurs

public interface Map <K, V> {
  // Associates the specified value with the specified key in this map
  // (optional operation).
  V     put(K key, V value);

  // Returns the value to which the specified key is mapped, or null if this
  // map contains no mapping for the key.
  V     get(K key);

  // Removes the mapping for a key from this map if it is present (optional
  // operation).
  V     remove(K key);

  // Returns true if this map contains a mapping for the specified key.
  boolean       containsKey(K key);

  // Returns true if this map maps one or more keys to the specified value.
  boolean       containsValue(V value);

  // Returns the number of key-value mappings in this map.
  int   size();

  // Returns true if this map contains no key-value mappings.
  boolean       isEmpty();

  // Removes all of the mappings from this map (optional operation).
  void  clear();

  // Returns a Set view of the keys contained in this map.
  Set<K>        keySet();

  // Returns a Collection view of the values contained in this map.
  Collection<V>         values();

  // Returns a Set view of the mappings contained in this map.
  Set<Map.Entry<K,V>>   entrySet();
}

public interface Map.Entry<K,V> {

  // Returns the key corresponding to this entry.
  K     getKey();

  // Returns the value corresponding to this entry.
  V     getValue();

  // Replaces the value corresponding to this entry with the specified value.
  V     setValue(V value);
}
```


Iterable
Collection
List
Queue
Set
Map
SortedMap
AbstractMap
NavigableMap
HashMap
AbstractCollection
Deque
SortedSet
TreeMap
LinkedHashMap
AbstractList
AbstractSet
AbstractQueue
NavigableSet
AbstractSequentialList
ArrayList
Vector
PriorityQueue
ArrayDeque
TreeSet
Stack
HashSet
LinkedList