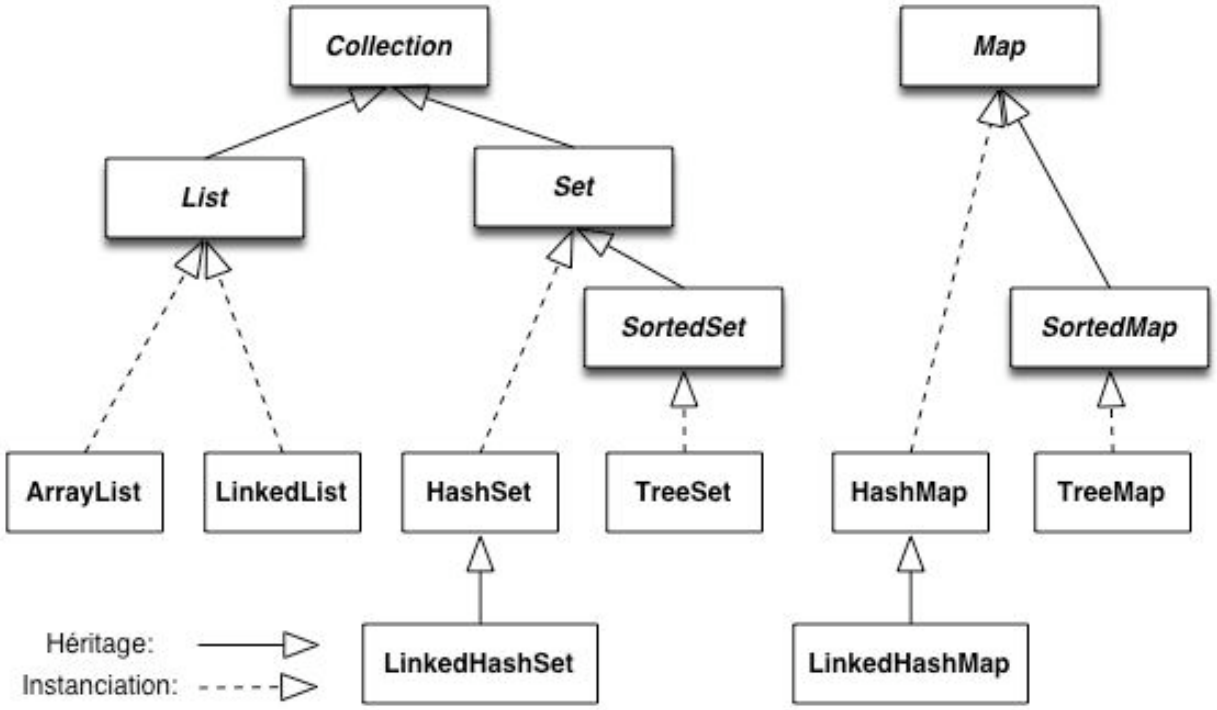
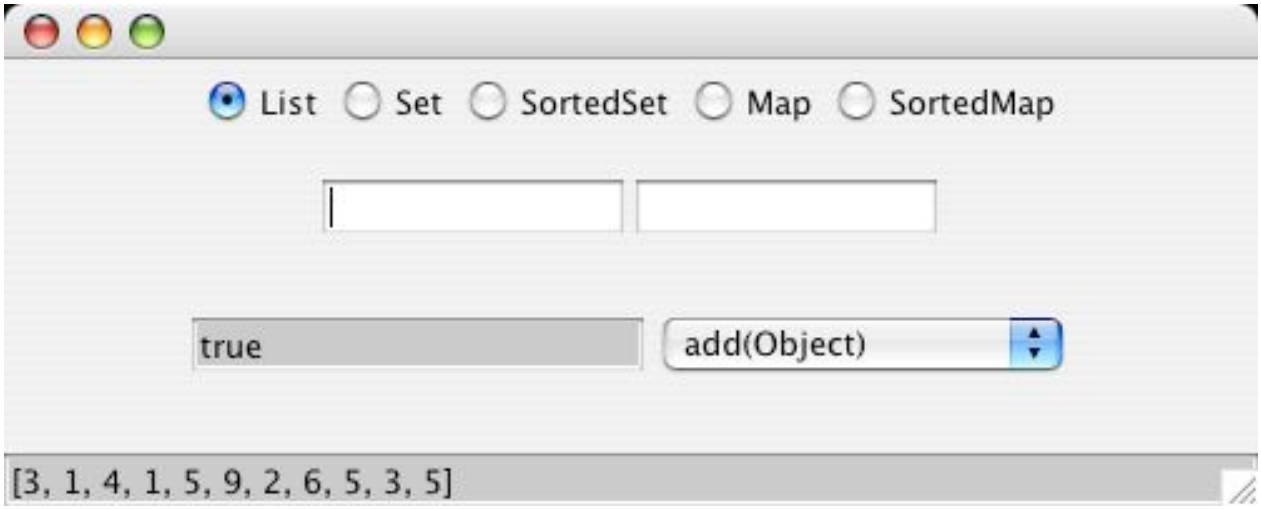
Collection Framework de Java




```
public interface Collection {
    // Opérations de base
    int size();
    boolean isEmpty();
    boolean contains(Object element);
    boolean add(Object element);    // Optional
    boolean remove(Object element); // Optional
    Iterator iterator();

    // Opérations de groupe
    boolean containsAll(Collection c);
    boolean addAll(Collection c);    // Optional
    boolean removeAll(Collection c); // Optional
    boolean retainAll(Collection c); // Optional
    void clear();                    // Optional

    // Transformations en tableaux
    Object[] toArray();
    Object[] toArray(Object a[]);
}
```

```
public interface Iterator {
    boolean hasNext();
    Object next();
    void remove();    // Optional
}

public interface ListIterator extends Iterator {
    boolean hasNext();
    Object next();

    boolean hasPrevious();
    Object previous();

    int nextIndex();
    int previousIndex();

    void remove();          // Optional
    void set(Object o);     // Optional
    void add(Object o);     // Optional
}

public interface Comparable {
    public int compareTo(Object o);
}

public interface Comparator {
    public int compare(Object o1, Object o2);
}
```

List

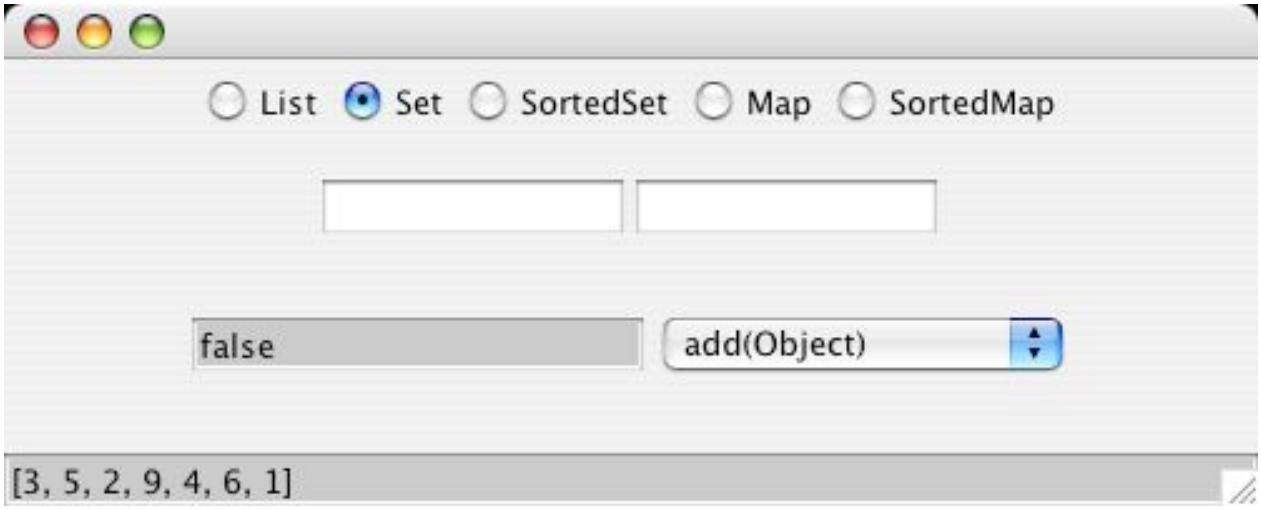



```
public interface List extends Collection {
    // accès par position
    Object get(int index);
    Object set(int index, Object element);   // Optional
    void add(int index, Object element);     // Optional
    Object remove(int index);                // Optional
    boolean addAll(int index, Collection c); // Optional

    // recherche
    int indexOf(Object o);
    int lastIndexOf(Object o);

    // Itération
    ListIterator listIterator();
    ListIterator listIterator(int index);

    // vue comme sous-liste
    List subList(int from, int to);
}
```

Création

```
List l = new ArrayList();
List l = new LinkedList();
```

Set

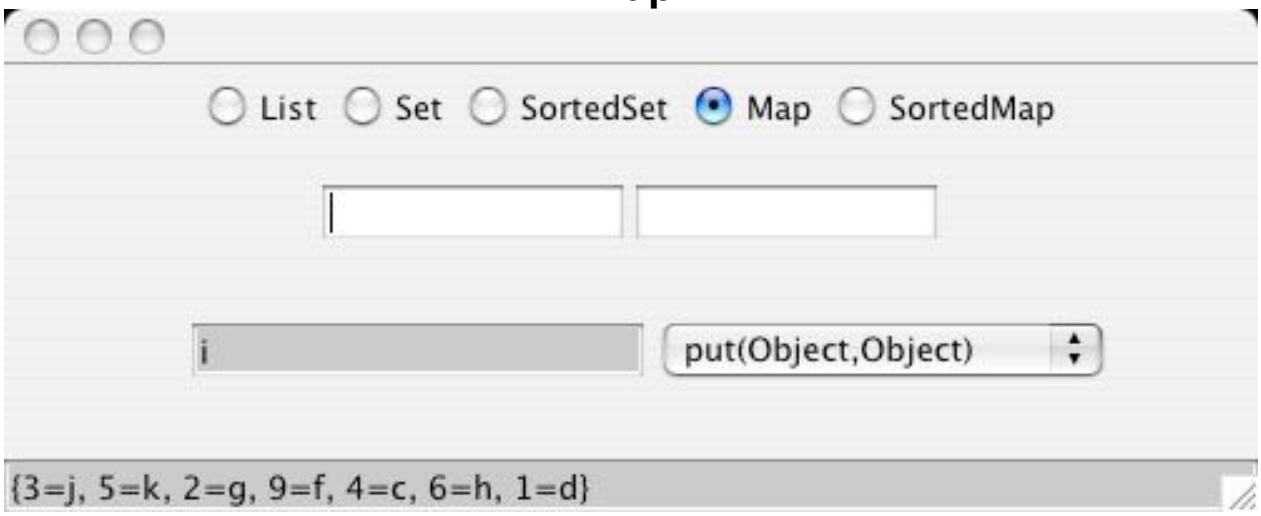



```
public interface Set extends Collection{
    // aucune nouvelle méthode
    // mais s'assure que tous les éléments sont distincts
}
```

Création

```
Set s = new HashSet();
Set s = new LinkedHashSet();
```

Map

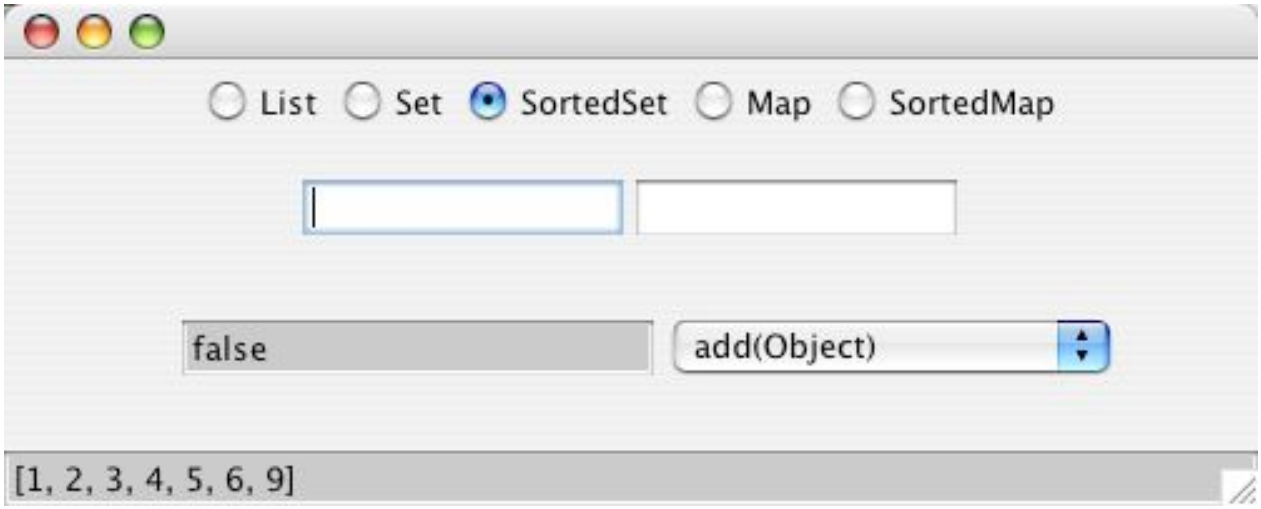



```
public interface Map {
    // Opérations de base
    Object put(Object key, Object value);
    Object get(Object key);
    Object remove(Object key);
    boolean containsKey(Object key);
    boolean containsValue(Object value);
    int size();
    boolean isEmpty();
//  Opérations de groupe
    void putAll(Map t);
    void clear();
//  Vues comme des Collections
    public Set keySet();
    public Collection values();
    public Set entrySet();
// Interface pour les entrées de la table (entrySet)
//      on la désigne par Map.Entry
    public interface Entry {
        Object getKey();
        Object getValue();
        Object setValue(Object value);
    }
}
```

Création

```
Map m = new HashMap();
Map m = new LinkedHashMap();
```

SortedSet




```
public interface SortedSet extends Set {
    // vue comme sous-ensemble
    SortedSet subSet(Object fromElement, Object toElement);
    SortedSet headSet(Object toElement);
    SortedSet tailSet(Object fromElement);

    // points extrêmes
    Object first();
    Object last();

    // accès au comparateur
    Comparator comparator();
}
```

Création

```
SortedSet ss = new TreeSet();
```

SortedMap


```
public interface SortedMap extends Map {

    // vue comme sous-table
    SortedMap subMap(Object fromKey, Object toKey);
    SortedMap headMap(Object toKey);
    SortedMap tailMap(Object fromKey);

    // accès aux clés extrêmes
    Object firstKey();
    Object lastKey();

    // accès au comparateur
    Comparator comparator();
}
```

Création

```
SortedMap sm = new TreeMap();
```

Collections

Contient des méthodes statiques qui opèrent et retournent des collections. On ne présente ici que les méthodes qui nous semblent les plus couramment utilisées.

```
public class Collections {
    // listes vides immuables
    public static final List EMPTY_LIST = {...}
    public static final List EMPTY_SET  = {...}
    public static final List EMPTY_MAP = {...}

    // Tri
    public static void sort(List list) {...}
    public static void sort(List list, Comparator c) { ... }

    // Recherche
    public static int binarySearch(List list, Object key){
    ...
    }
    public static int binarySearch(List list, Object key,
                                   Comparator c) { ... }

    public static Object min(Collection coll) { ... }
    public static Object min(Collection coll,
                             Comparator comp) { ... }

    public static Object max(Collection coll) { ... }
    public static Object max(Collection coll,
                             Comparator comp) { ... }
    // Remplissage
    public static void fill(List list, Object o) { ... }
    public static void copy (List dest, List src) { ... }
    // Brassage
    public static void reverse(List l) { ... }
    public static void shuffle(List list) { ... }
    public static void shuffle(List list, Random rnd){ ... }
    // Collection parcourable par énumération
    public static Enumeration enumeration(
                              final Collection c) { ... }
}
```

Arrays

Algorithmes génériques sur les tableaux dont les éléments sont les types prédéfinis (long, int, short, char, byte, float, double) ainsi que Object. Dans la description suivante, ces types sont notés *T*

```
public class Arrays {
    public static List asList(Object[] a) { ... }

    // Tri
    public static void sort( T[] a) { ... }
    public static void sort( T[] a, int fromIndex,
                                    int toIndex) { ... }

    public static void sort(Object[] a, Comparator c){ ... }
    public static void sort(Object[] a,
                            int fromIndex, int toIndex,
                            Comparator c) { ... }

    // Recherche
    public static int binarySearch( T[] a, T key) { ... }
    public static int binarySearch(Object[] a,
                                   Object key,
                                   Comparator c){ ... }
    // Test d'égalité de tous les éléments des tableaux
    public static boolean equals( T[] a, T[] a2) { ... }

    // Remplissage
    public static void fill( T[] a, T val) { ... }
    public static void fill( T[] a,
                             int fromIndex, int toIndex,
                             T val) { ... }
}
```