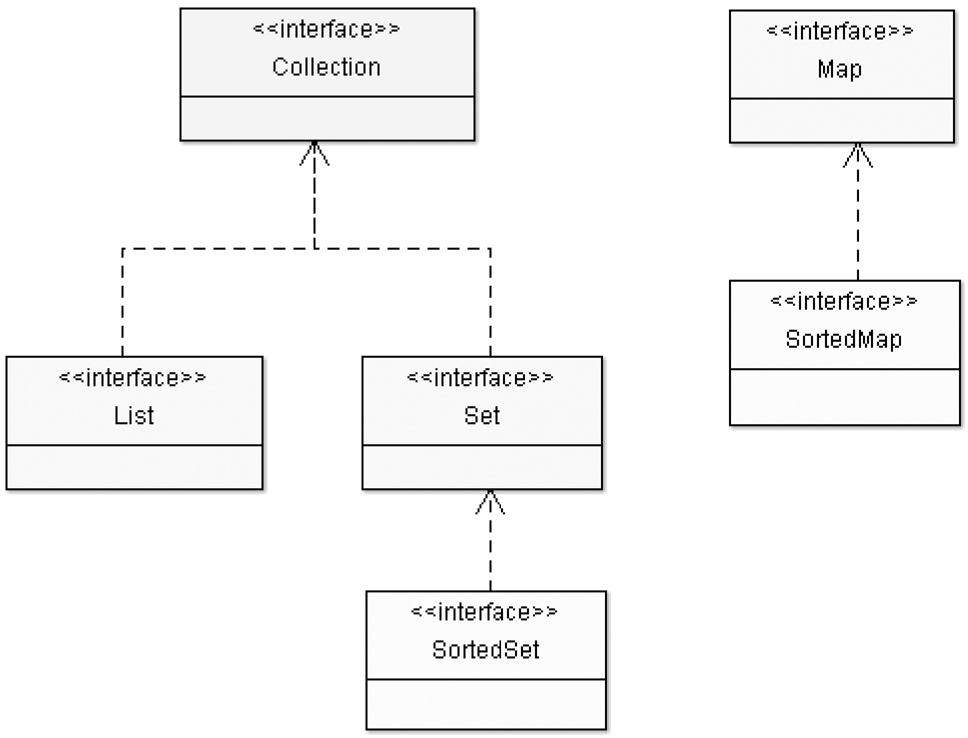

Interface Hierarchy

The ArrayList Object

The ArrayList Object is extremely practical. ArrayList does not need its size to be defined and can accept any type of data, including null. Below is a bit of code displaying the versatility of ArrayLists with regard to data being entered into them.

```
ArrayList<Object> al = new ArrayList<Object>();
al.add(12);
al.add("A string!");
al.add(12.20f);
al.add('d');

for(int i = 0; i < al.size(); i++){
      System.out.println("data at index " + i + " = " + al.get(i));
}

Methods :
```

add() adds an element;
get(int index) returns the element at the specified index;
remove(int index) removes the element at the specified index;
isEmpty() returns "True" is the object is empty;
removeAll() removes all the elements in the ArrayList;
contains(Object element) returns "True" if the passed element is contained in the ArrayList.

Map Objects

Collections of type map function with key – value pairs.

Hashtable,
HashMap,
TreeMap,
WeakHashMap…

The HashTable Object

```
Hashtable<Integer, String> ht = new Hashtable<Integer, String>();
ht.put(1, "spring");
ht.put(10, "summer");
ht.put(12, "autumn");
ht.put(45, "winter");

Enumeration<String> e = ht.elements();

while(e.hasMoreElements())
  System.out.println(e.nextElement());
```

Methods:

isEmpty() returns "True" if the object is empty;
contains(Object value) returns "true" if the value is contained in the Hashtable.
containsKey(Object key) returns "true" if the entered key is contained in the Hashtable.
put(Object key, Object value) adds a key-value pair to the collection;
elements() returns an enumeration of elements in the collection;
keys() returns the list of keys in the form of an enumerable.

Furthermore, you must know that a Hashtable object does not accept null values and is Thread Safe. That is to say, it is usable in multiple threads (this signifies that multiple elements of your program can be used simultaneously; we will return to this later) simultaneously without there being a risk of data conflicts.

The HashMap Object

This object differs very slightly from the Hashtable obejct:

it accepts the null value;
it is not Thread Safe;

Set Objects

The HashSet Object

```
HashSet hs = new HashSet();
hs.add("toto");
hs.add(12);
hs.add('d');

Iterator it = hs.iterator();
while(it.hasNext())
  System.out.println(it.next());

System.out.println("\nPath with an object array");
System.out.println("-----------------------------------");

Object[] obj = hs.toArray();
for(Object o : obj)
  System.out.println(o);
```

methods:

add() adds an element;
contains(Object value) returns “True” if the HashSet contains the entered value;
isEmpty() returns “True” if the HashSet is empty;
iterator() returns an object of type Iterator ;
remove(Object o) removes the object o from the collection;
toArray() returns an array of Objects.