Framework de Colecciones

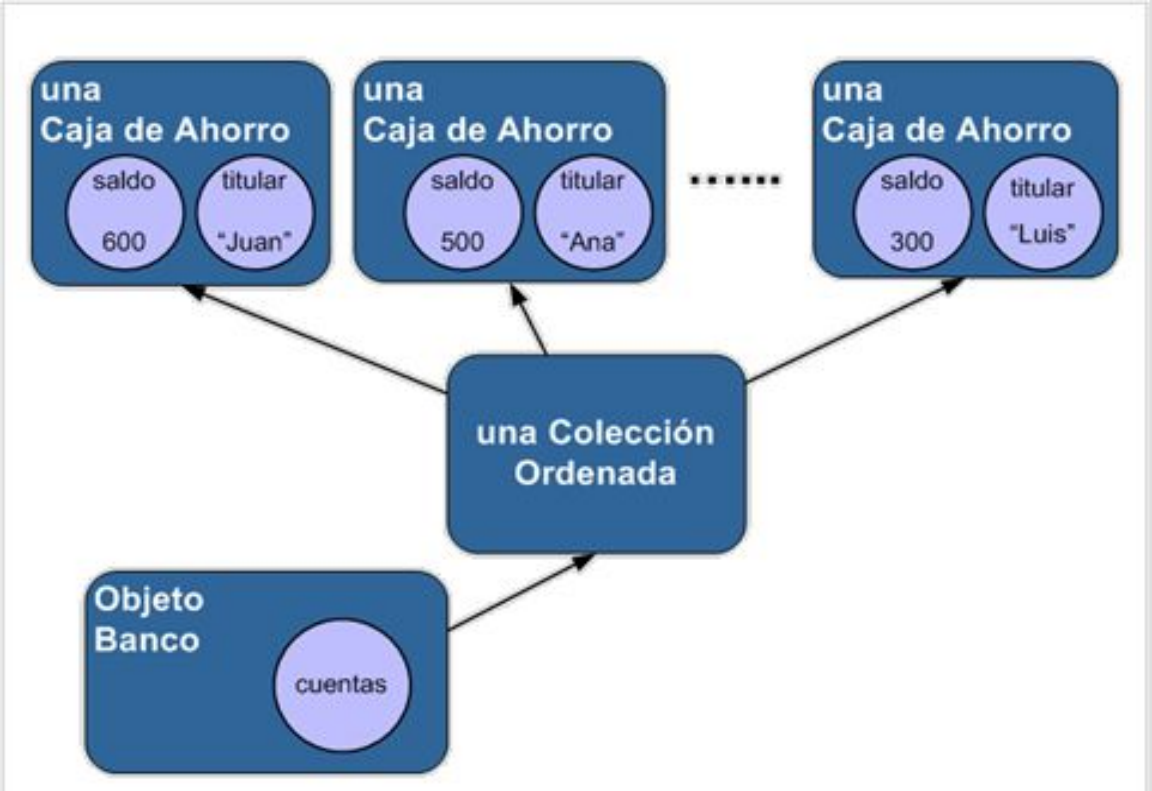
Seminario de Lenguajes

Temas

- Framework de colecciones
- Jerarquia de clases Collection/Map
- Interfaces/Implementaciones
  - Collections
    - Set
    - List
  - Map
- Mecanismos para recorrer una coleccion

Conocimiento múltiple

- Cuando deseamos conocer no sólo uno, sino varios objetos que compartan cierta naturaleza.
- Muchos objetos bajo un solo concepto



Que es una Colección?

- Una colección (también conocido como contenedor) es un objeto que contiene un grupo de objetos tratados como una sola unidad.

- Cualquier tipo de objetos se puede almacenar, recuperar y manipular como elementos de colecciones.

Aplication Programing Interface(API)

- Conjunto de clase e interfaces que facilitan el desarrollo de aplicaciones en la plataforma, brindando ciertas componentes que solucionan problemas específicos y de reiterada ocurrencia en distintos contextos.

- ¿Para que desarrollarlos una y otra vez?

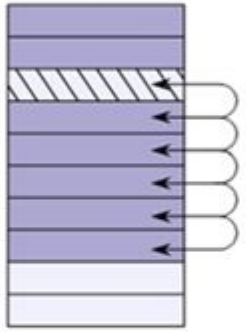
- Java provee un gran conjunto de elementos para utilizarlos en nuestro programas ej: Collection, Exception, AWT, Swing.

Tecnologías de almacenamiento

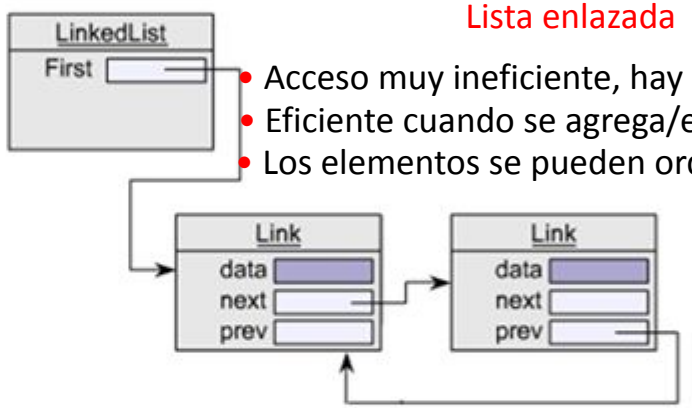
Existen cuatro tecnologías de almacenamiento básicas disponibles para almacenar objetos: arreglo, lista enlazada, árbol y tabla de hash.

Arreglo

- El acceso es muy eficiente.
- Es ineficiente cuando se agrega/elimina un elemento.
- Los elementos se pueden ordenar.

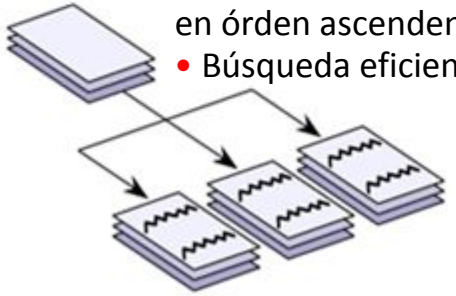


Lista enlazada



- Acceso muy ineficiente, hay que recorrer la lista.
- Eficiente cuando se agrega/elimina un elemento.
- Los elementos se pueden ordenar.

Arbol



- Almacenamiento de valores en órden ascendente.
- Búsqueda eficiente.

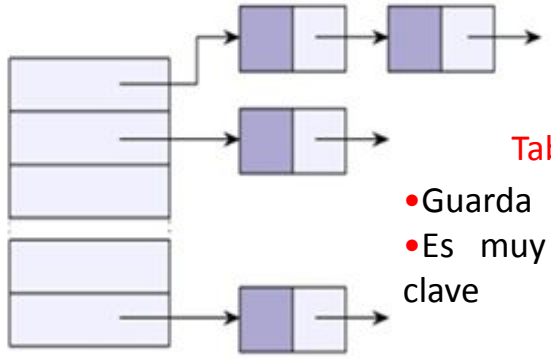


Tabla de hash

- Guarda duplas (clave, valor).
- Es muy rápido, accede por clave

Framework de Colecciones

- Operaciones basicas sobre una colección:

  - Agregar objetos a una colección
  - Quitar objetos de la colección
  - Encontrar un objeto o un grupo de objetos en una colección
  - Recuperar un objeto de una colección (sin quitarlo de la colección)
  - Iterar a través de una colección, viendo cada elemento, uno después de otro. -

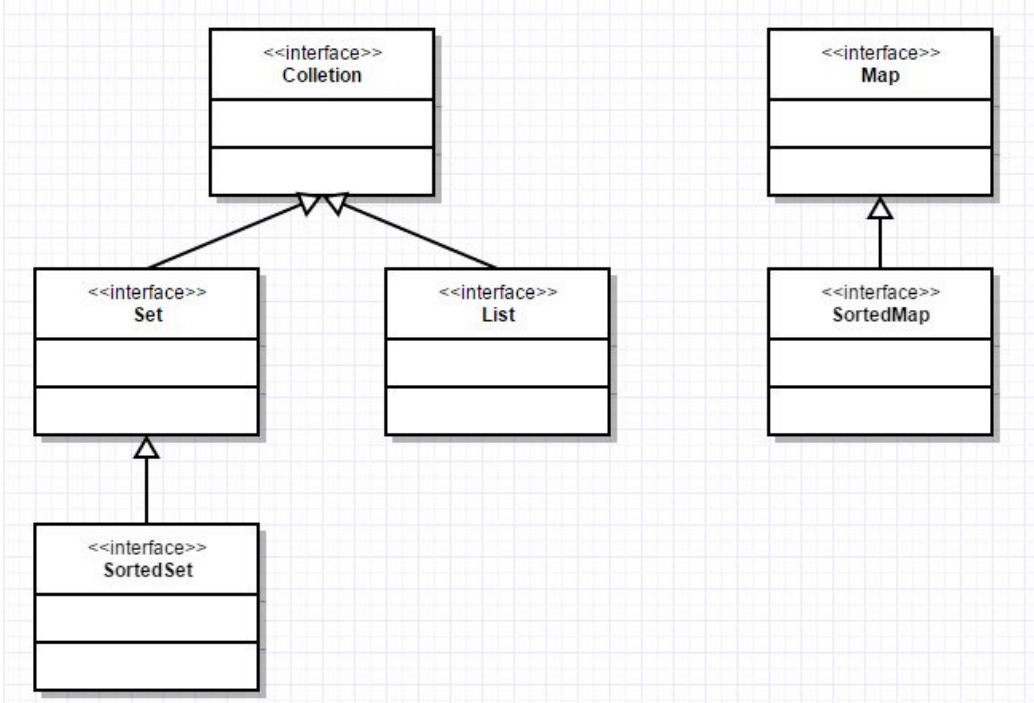
Framework de Colecciones

- Es una arquitectura unificada para administrar colecciones

- Partes principales del Framework:
  - Interfaces
    - Definen la funcionalidad común entre las colecciones
  - Implementaciones
    - Clases concretas las cuales proporcionan las estructuras de datos
  - Operaciones(algoritmos)
    - Métodos que realizan diversas operaciones sobre las colecciones.

Framework de Colecciones

- Compuesto por un conjunto de algoritmos, clases e interfaces que implementan estructuras de Datos.
- Están divididos en dos grupos (no hay relación entre ambos tipos).



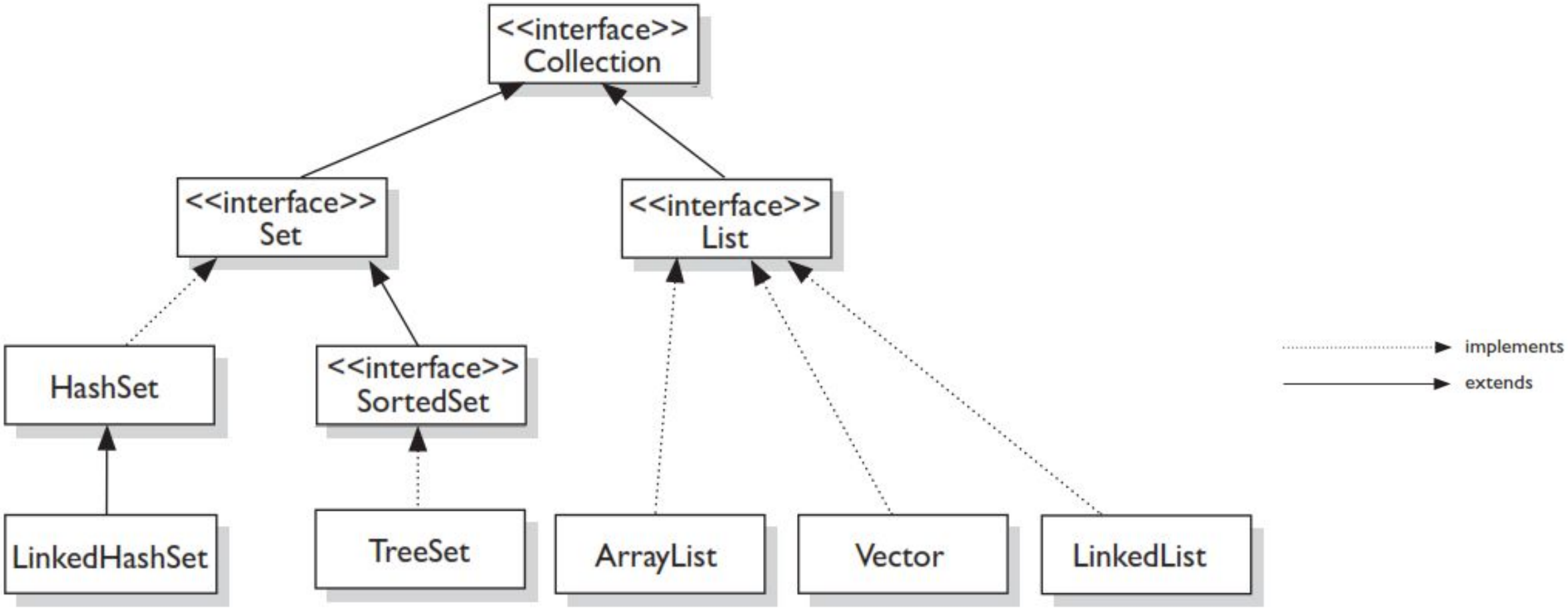
Framework de Colecciones - interfaces

Core Interface	*Descripción*
Collection	especifica el contrato que todas las colecciones deben implementar
Set	define la funcionalidad para un conjunto de elementos únicos
SortedSet	define la funcionalidad para un conjunto donde los elementos son ordenados
List	define la funcionalidad para una lista ordenada de elementos no únicos(repetidos)
Map	Se mapea una clave única (ID) a un valor especifico.
SortedMap	define la funcionalidad para un Map donde las claves son ordenadas

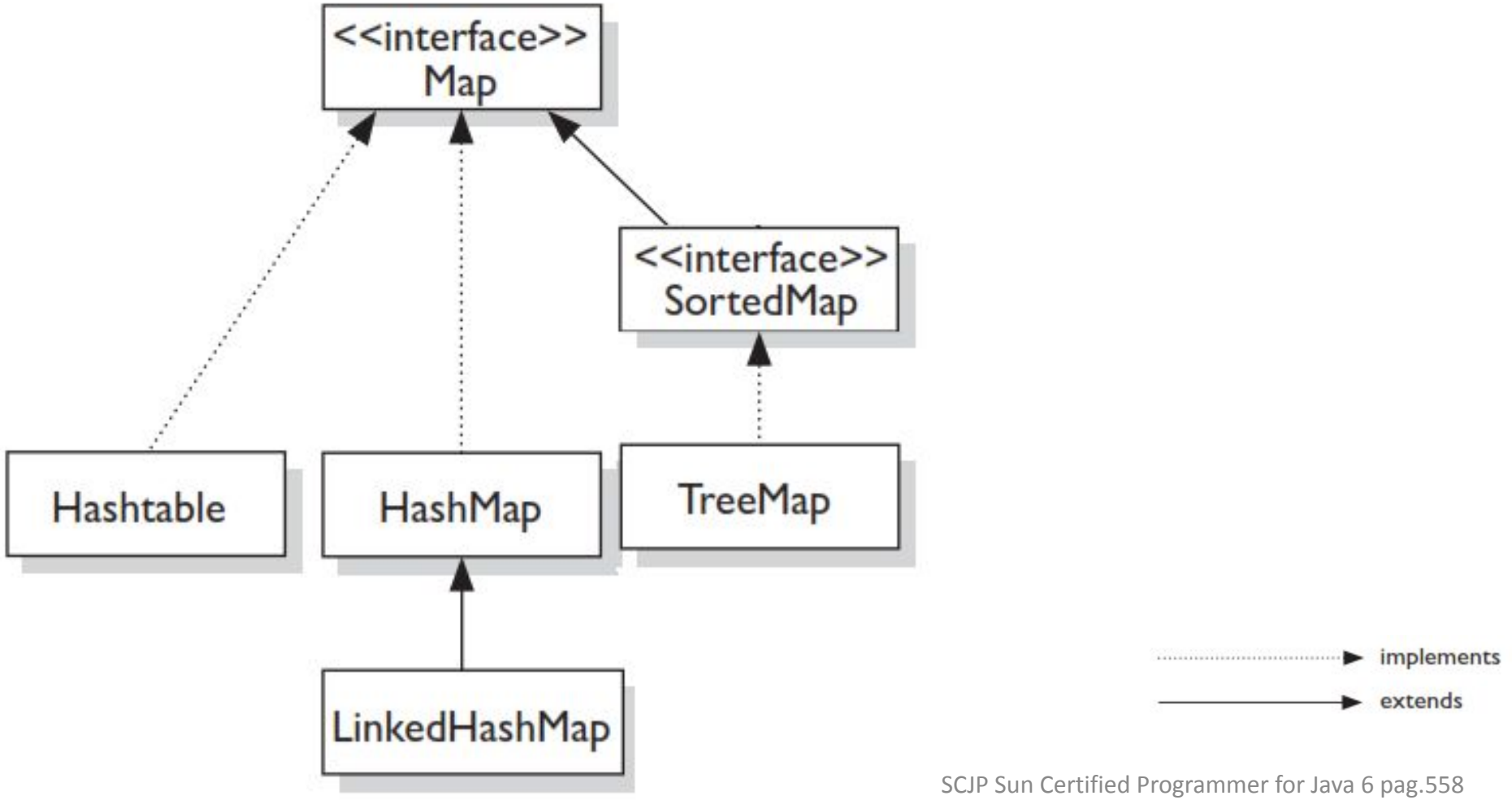
Framework de Colecciones - Implementaciones

Set	List	Map	Utilities
HashSet	**ArrayList**	**HashMap**	**Collections**
LinkedHashSet	**LinkedList**	**LinkedHashMap**	**Arrays**
TreeSet	**Vector**	**Hashtable**	
		TreeMap	

Jerarquía – Collection (List y Set)



Jerarquía - Map



SCJP Sun Certified Programmer for Java 6 pag.558

Características - Colecciones

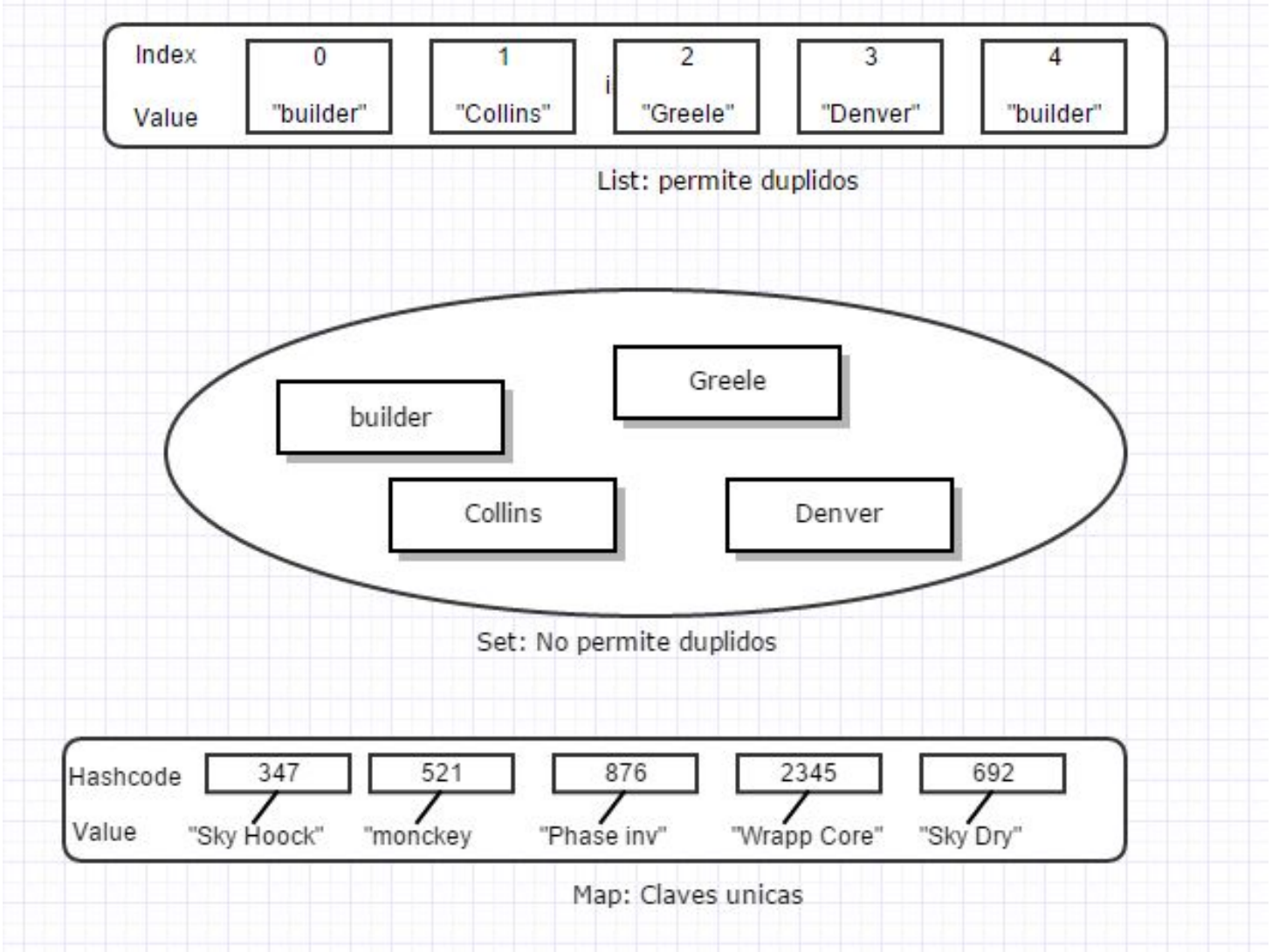
- **Ordered** (mantiene un orden)

- **Sorted** (orden natural)
  - El orden natural se da a través de la interface Comparable.

Collection classes

			Collection Properties			
Collection Class	**Storage Technologies**	**Collection Interface**	**Sorted**	**Ordered**	**Duplicates**	**Key**
LinkedList	Linked-List	List		x	x	
ArrayList	Array	List		x	x	
Vector	Array	List		x	x	
HashSet	HashTable	Set				
TreeSet	Tree	SortedSet	x			
HashMap	HashTable	Map				x
TreeMap	Tree	SortedMap	x			x
Hashtable/ Properties	HashTable	Map				x

Tipos de colecciones – List, Set y Map

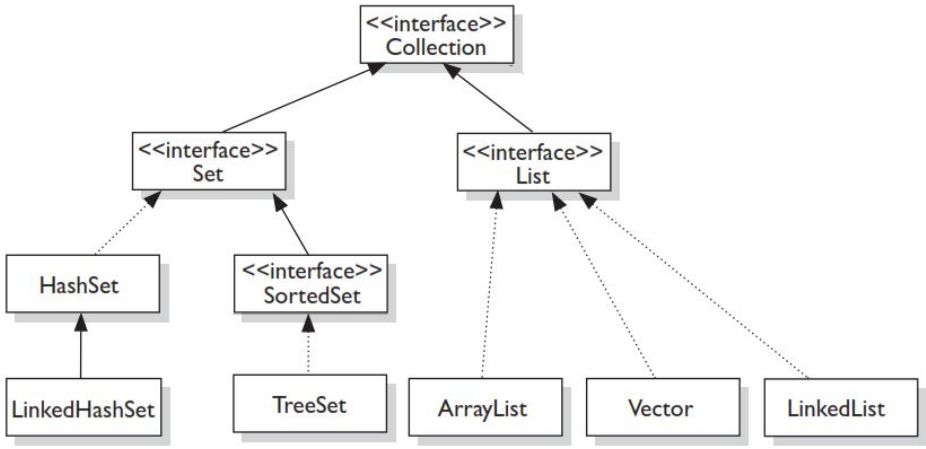


Interface List

•Esta interface está relacionada con el índice.

•Se diferencia del resto por el uso de índices tiene los siguientes métodos:

```
E get(int index)
int indexOf(E element)
int lastIndexOf(E element)
E remove(int indice)
E set(int index, E element)
boolean  add(int index, E element)
```



•Las tres implementaciones de List están ordenadas por posición o por el valor del índice.

•Puede contener objetos duplicados.

Interface List

Interface: List<E>

Implementaciones:

- ArrayList <E> : Soporta rápido acceso aleatorio
- LinkedList<E> : las inserciones y los borrados son más rápidos

```
import java.util.*;
public class ListTest{
   public static void main(String[] args){
      List<Integer> lista = new ArrayList<Integer>();
      lista.add(1);
      lista.add(2);
      lista.add(190);
      lista.add(2);
      lista.add(7);
      System.out.println(lista.get(0));
      System.out.println(lista.get(1));
      System.out.print(lista.toString());
   }
}
```

[1]

[2]

[1, 2, 190, 2, 7]

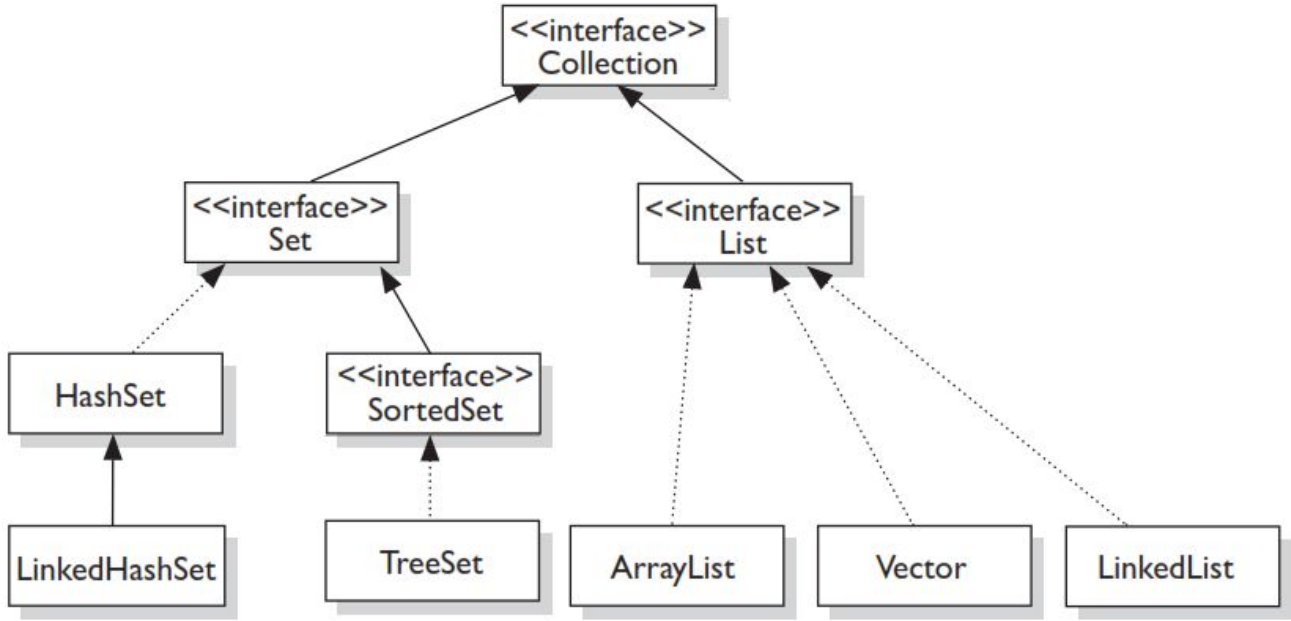
(1) Las listas brindan implementaciones de algortimos para ordenar elementos (sort), permutar en forma random (shuffle), reemplaza todas las ocurrencias de un elemento por otro (replaceAll), retorna la lista en orden inverso (reverse), etc.

(2) Las listas tienen operaciones de subrango que nos permiten trabajar con una parte de la lista desde una posición hasta otra posición inclusive:

```
list.subList(fromIndex, toIndex).clear();
int i = list.subList(fromIndex, toIndex).indexOf(o);
```

Interface Set

- Un Set no tiene orden y no permite duplicados.
- Se utiliza equals() para determinar si dos objetos son iguales.
- Tiene exactamente los mismos métodos que la interface Collection, pero agrega la restricción de no mantener duplicados



Interface Set

Interface: Set

Implementaciones:

- HashSet (mejor performance, almacena los datos en una tabla de hash)
- LinkedHashSet(versión ordenada de HashSet)
- TreeSet(ordenados de forma natural).

```
public  static void  main(String[]  args)  {
      Set<String> instrumentos= new HashSet<String>();
       instrumentos.add("Piano");
       instrumentos.add("Saxo");
       instrumentos.add("Violin");
       instrumentos.add("Flauta");
       instrumentos.add("Flauta");
       System.out.println(instrumentos.toString());
}
```

La salida es: [Violin, Piano, Saxo, Flauta]

(1) Es útil para crear colecciones sin duplicados desde una colección c con duplicados.

```
Set<String>  sinDup = new  TreeSet<String>(c);
```

(2) Soporta operaciones que modelan conjuntos:

```
s1.containsAll(s2)  - incluye
s1.addAll(s2)  - unión
s1.retainAll(s2)  - intersección
s1.removeAll(s2)  - diferencia
```

Cambiando únicamente la instanciación por un objeto TreeSet() Implementa SortedSet, obtenemos una colección ordenada:

Set<String> instrumentos= new TreeSet<String>(); [Flauta, Piano, Saxo, Violin]

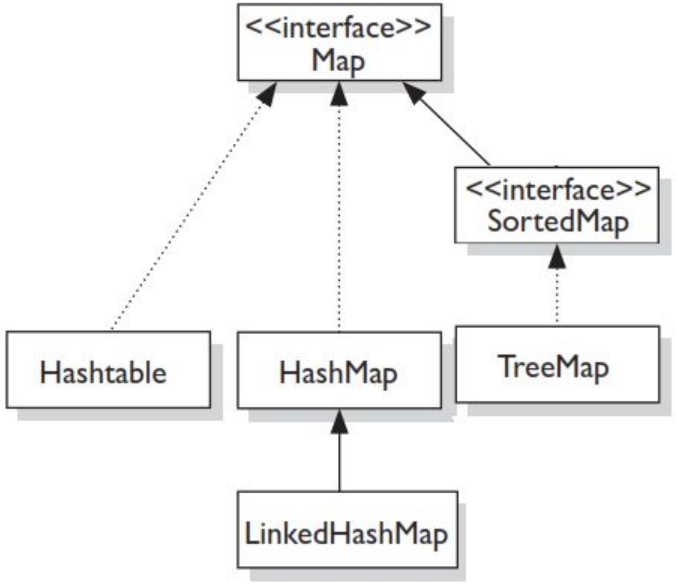
En este caso el compilador chequea que los objetos que se insertan (add()) sean Comparables!!

Interface Map

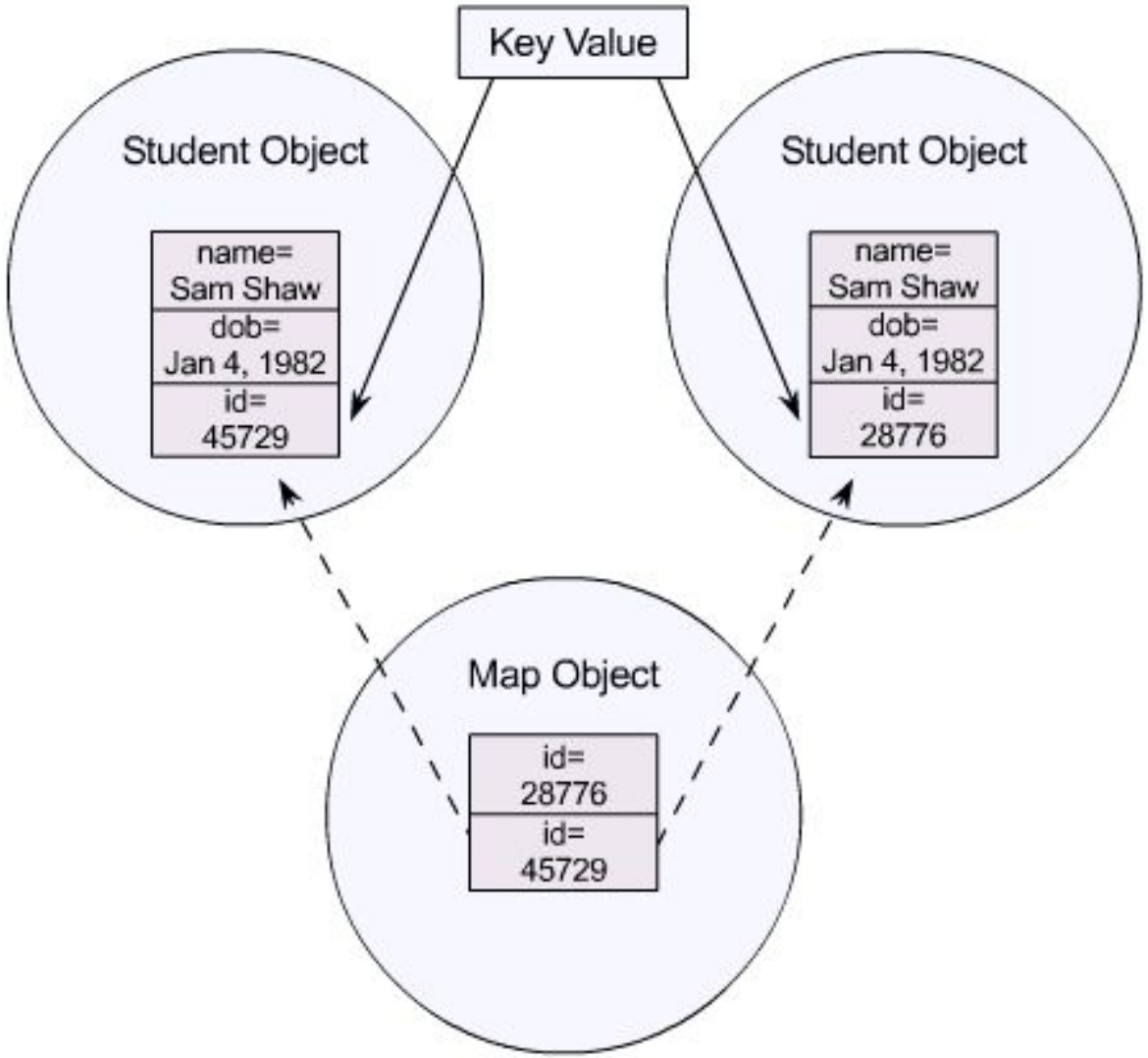
- Se mapea una clave única (ID) a un valor especifico. Clave y Valor, son Objetos.
- Las implementaciones de Map permiten buscar un valor dada la clave.
- El método hashCode() de la clase Object provee el identificador único del objeto (clave única).

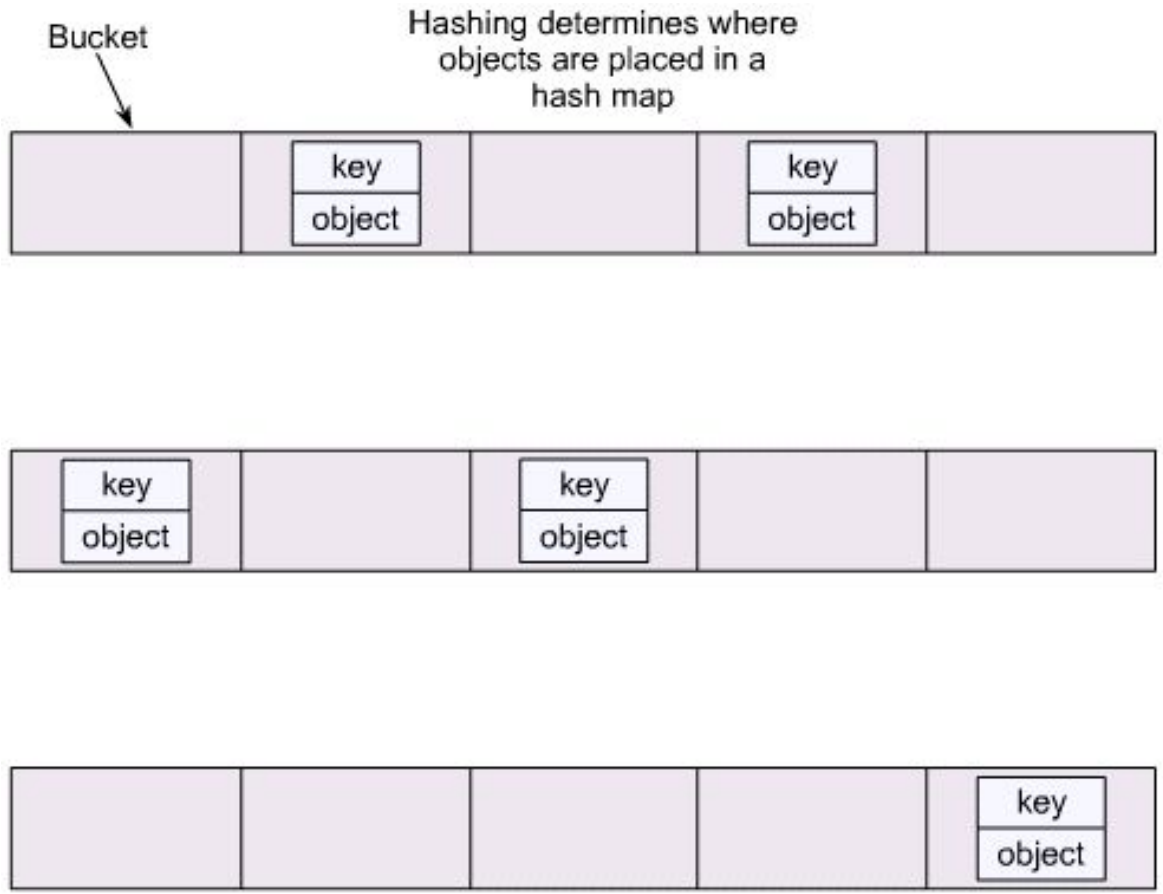
  Este método debe ser sobreescrito.
- Se basa en el método **equals** para saber si dos claves son iguales.
- Se diferencia del resto por el uso de claves :

```
V put (K Key, V Vaule)
V get(Object key)
V remove(Object key)
boolean containsKey(Object key)
boolean containsValue(Object key)
Set keySet()
Collection values()
```



Objetos Map


Key Value
Student Object
name=
Sam Shaw
dob=
Jan 4, 1982
id=
45729
Student Object
name=
Sam Shaw
dob=
Jan 4, 1982
id=
28776
Map Object
id=
28776
id=
45729


Bucket
Hashing determines where objects are placed in a hash map
key
object
key
object
key
object
key
object
key
object

Interface Map

Interface: Map<K,V>

Implementaciones:

- HashMap<K,V>: Utiliza el hashcode del elemento para localizarlo
- LinkedHashMap<K,V> : Mantiene el orden de inserción. Mas lento.
- TreeMap<K,V> : Garantiza que los elementos estarán en forma ascendente, según el orden natural

```
public  static void  main(String[]  args)  {
  Map<String, Integer>  numeros=new  HashMap<String, Integer>();
   numeros.put("uno",  new Integer(1));
   numeros.put("dos",  new Integer(2));
   numeros.put("tres",  new  Integer(3));
   System.out.println(numeros.toString())
   }
}
```

[tres=3, uno=1, dos=2]

Cambiando únicamente la instanciación por un objeto TreeMap() Implementa SortedMap() , obtenemos una colección ordenada:

```
Map<String, Integer> numeros=new TreeMap<String,Integer>();
```

{dos=2, tres=3, uno=1}

En este caso el compilador chequea que los objetos que se insertan (put()) sean Comparables!!

Mecanismos para recorrer colecciones

1) Usando la construcción: for/ for-each

2) Usando la interface Iterator/ListIterator

Estructura de control : For Each

For (ElementType elemento : Colección) {

// cuerpo del bucle

}

```
ArrayList<Integer> lista= new ArrayList<Integer>();
lista.add(1);
lista.add(2);
for  (Integer i: lista){

    System.out.println(i);
}
```

Interface Iterator

- Permite recorrer la colección de principio a fin

```
public void imprimir(Set<String> palabras) {
   Iterator<String> it = palabras.iterator();
      while (it.hasNext()) {
         String p = it.next();
         System.out.println(p);
      }
   }
```

Resumen

Class	Map	Set	List	Ordered	Sorted
`HashMap`	x			No	No
`HashTable`	x			No	No
`TreeMap`	x			Sorted	By *natural order* or custom comparison rules
`LinkedHashMap`	x			By insertion order or last access order	No
`HashSet`		x		No	No
`TreeSet`		x		Sorted	By *natural order* or custom comparison rules
`LinkedHashSet`		x		By insertion order	No
`ArrayList`			x	By index	No
`Vector`			x	By index	No
`LinkedList`			x	By index	No
`PriorityQueue`				Sorted	By to-do order

Resumen

TABLE 7-7 Key Methods in List, Set, and Map

Key Interface Methods	List	Set	Map	Descriptions
`boolean add(element)` `boolean add(index, element)`	X X	X		Add an element. For Lists, optionally add the element at an index point.

`boolean contains(object)`
`boolean containsKey(object key)`
`boolean containsValue(object value)`
	X	X	
			X
			X
				Search a collection for an object (or, optionally for Maps a key), return the result as a `boolean`.

`object get(index)` `object get(key)`	X		 X	Get an object from a collection, via an index or a key.
`int indexOf(object)`	X			Get the location of an object in a List.
`Iterator iterator()`	X	X		Get an Iterator for a List or a Set.
`Set keySet()`			X	Return a Set containing a Map's keys.
`put(key, value)`			X	Add a key/value pair to a Map.

`remove(index)`
`remove(object)`
`remove(key)`
	X
	X
		
		X
			
			X
				Remove an element via an index, or via the element's value, or via a key.

`int size()`	X	X	X	Return the number of elements in a collection.
`Object[] toArray()` `T[] toArray(T[])`	X	X		Return an array containing the elements of the collection.

Bibliografía

- Lesson: Interfaces - http://docs.oracle.com/javase/tutorial/collections/interfaces/index.html - *Acceso vía web*

- Thinking Java,, 4ta Edición, Bruce Eckel. Capítulo 11, Colecciones de Objetos. - *En la biblioteca y en la plataforma moodle*

- SCJP Sun Certified Programmer for Java 6 Study Guide , Exam (310-065) K. Sierra, B.Bates**.** Capítulo 7, Generics-Collections – *En la plataforma moodle*

- Como programar en Java, 9ta Edición, J.P. Deitel . Capítulo 19, Colecciones – *En la Biblioteca y en la plataforma moodle*

- Java a fondo, Ing. Pablo Augusto Sznajdleder, Capitulo 9 , Estructuras de datos dinámicas. –E*n la plataforma moodle*