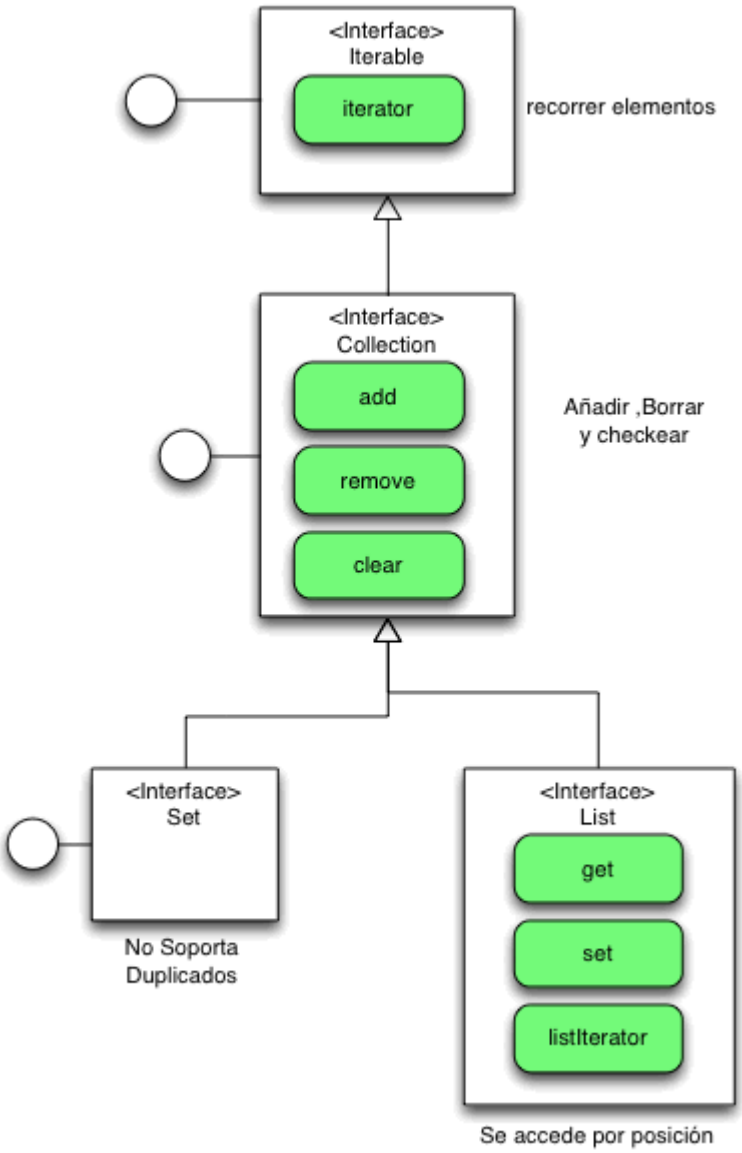
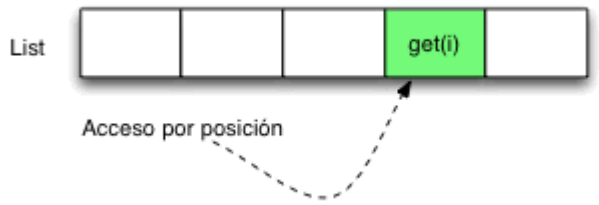
Todos usamos el framework de colecciones de Java para manejar conjuntos de objetos .Vamos a dedicar algunos artículos a abordar las colecciones más importantes. En este artículo vamos a introducir las Listas (List) y los Conjuntos (Set). Para ello vamos a ver en un primer lugar donde encajan ambos en la jerarquía de clases del framework de Colecciones.

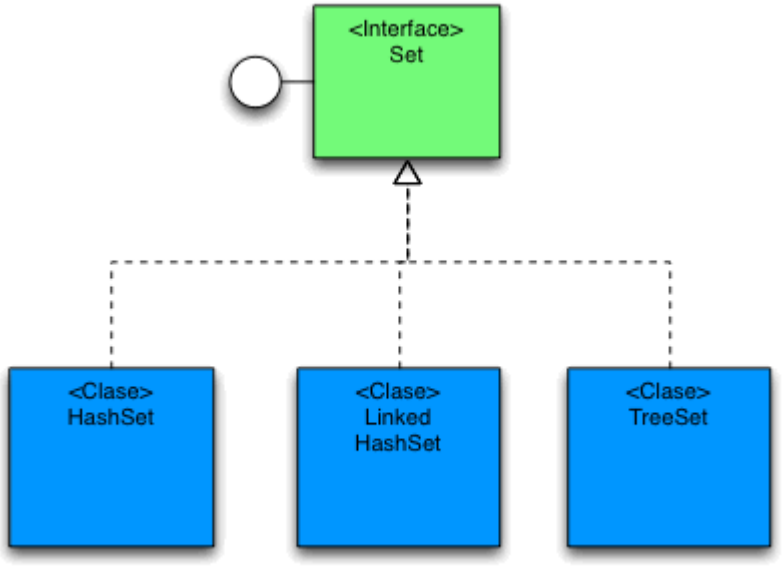


En el framework de colecciones el interface más general es Iterable el cual define que una colección se puede recorrer devolviendo para ello un Iterator. Una vez tenemos este interface aparece el interface Collection que define el concepto abstracto de “Colección” que hace referencia a un conjunto de elementos recorribles . Es en este interface en donde se añaden métodos como add() ,remove() y clear() para añadir o eliminar elementos de la colección. Set y List son interfaces hijos de Collection . Set practicamente no añade nada nuevo salvo que define que no podemos tener elementos repetidos. Por otro lado List si que añade métodos adicionales que nos permiten acceder por posición a cada elemento de la lista.



Java Collections e Implementaciones

Las clases que implementan los interfaces Set y List en el framework de Colecciones son las siguientes :



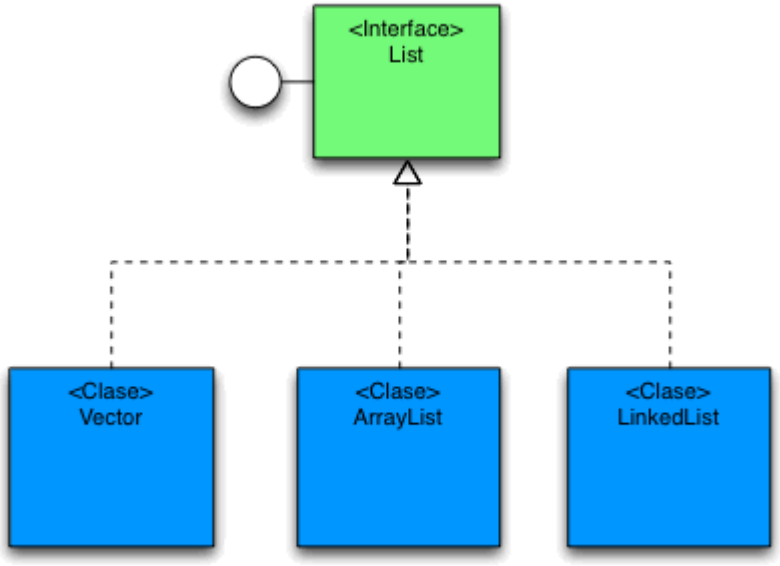
En el caso de los Sets existen tres implementaciones fundamentales:

HashSet: Define el concepto de conjunto (grupo de elementos no repetidos) a través de una estructura Hash . Es el conjunto más habitual.

TreeSet: Define el concepto de conjunto a través de una estructura de Arbol . Este conjunto se utiliza en casos en los cuales necesitamos un orden especifico de los elementos.

LinkedHashSet: Define el concepto de conjunto añadiendo una lista doblemente enlazada en la ecuación que nos asegura que los elementos siempre se recorren de la misma forma.

Acabamos de ver los Sets (Conjuntos) vamos a abordar ahora las Listas (List)



En este caso también existen tres implementaciones fundamentales :

Vector: Es la implementación más antigua y existe desde los inicios de Java. Hoy por hoy se usa muy poco porque se trata de una estructura de Lista a la que siempre se accede de forma sincronizada lo cual en muchos casos reduce el rendimiento.

ArrayList: Es la implementación más utilizada ya que no obliga a sincronizar el acceso.

LinkedList: Es una implementación a través de una lista doblemente enlazada lo cual nos permite un mayor rendimiento cuando insertamos o eliminamos elementos en medio de la lista.

Si aprendemos a manejar el framework de Colecciones muchas situaciones las podremos abordar de una forma sencilla. Imaginemonos que tenemos un Array de Strings:

```
String[] lista= { "hola","adios", "que","tal","estas","hola"};
```

Queremos eliminar los elementos repetidos y ordenar la lista alfabeticamente para mostrarla por pantalla. La operación puede ser tan sencilla como lo siguiente:

```
List<String> listaCadenas= Arrays.asList(lista);
 Set<String> conjunto= new TreeSet<String>(listaCadenas);
```

En este caso usamos la clase Arrays para convertir un Array en una lista . Hecho esto pasamos la lista al constructor del TreeSet que elimina elementos repetidos y la ordena. Por último la recorremos dicha lista .

```
for (String cadena : conjunto) {
 System.out.println(cadena);
 }
```

El resultado será el siguiente:

```
<terminated> Principal (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.
adios
estas
hola
que
tal
```

Ni siquiera hemos requerido de un bucle for para realizar las operaciones Java Collections

simplifica mucho la forma de trabajar si sabemos como abordar un problema.

Otros artículos relacionados : Java Generics , Java Generic y WildCards , Java Lambda