Colecciones | Concepto

Qué son las *Colecciones*?

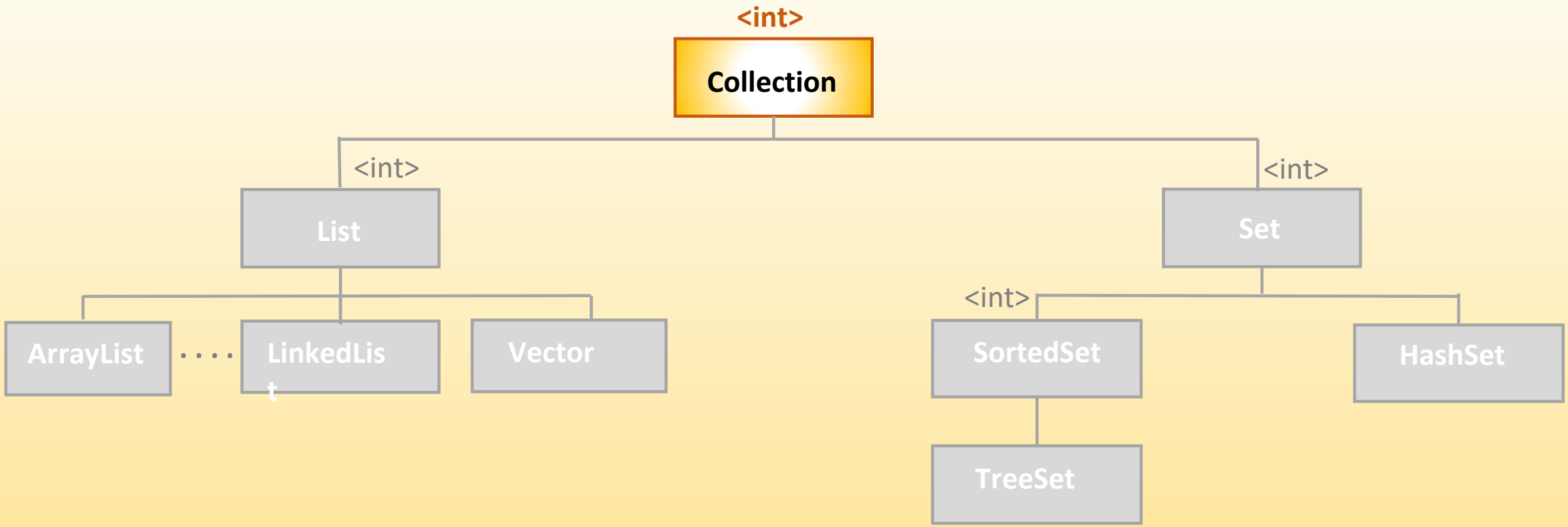
Son una evolución de los arreglos, nacen a partir de lo que se llama Java 2 (Java 1.4).

Las colecciones están dentro del paquete Java.útil. Se las conoce como ***Java Collections Framework.***

El framework consiste en una serie de interfaces básicas y varias implementaciones de referencia.

A diferencia de los Arreglos, las **Colecciones tienen tamaño dinámico**; es decir, crecen a medida que se le agregan más elementos.

Colecciones |Diagrama de flujo



Colecciones | Collection

El framework, de acuerdo al Diagrama de flujo previamente mostrado, se organiza en una jerarquía de la siguiente manera:

La interfaz ***Collection*** es una simple bolsa de elementos, ya que no almacena en ningún orden particular.

Define operaciones básicas:

`.add(Object o)` que sirve para agregar un elemento.

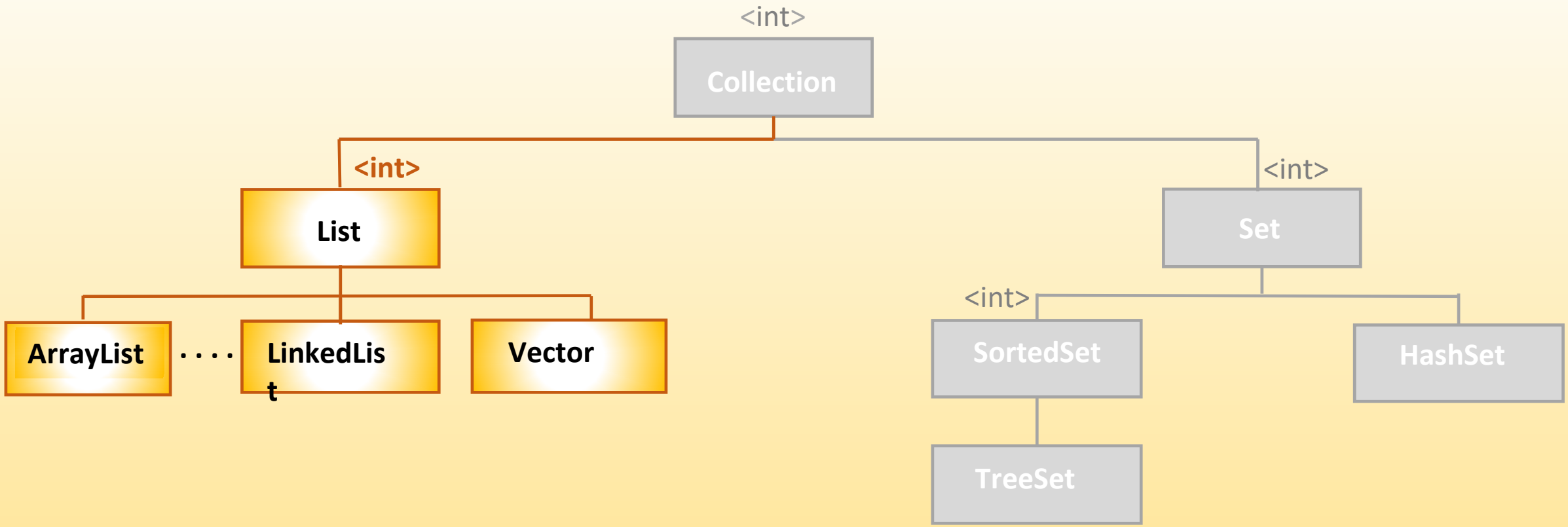
`.remove(Object o)` que sirve para quitar un elemento.

`.size()` que devuelve la cantidad de elementos (no el tamaño).

`.clear()` que limpia el contenido de la colección.

`.iterator()` que sirve para recorrer los elementos de **cualquier** colección.

Colecciones |Diagrama de flujo



Colecciones | List

Especifica Collection. Almacena los elementos de manera contigua y por esto tiene acceso secuencial a los elementos de forma similar a un arreglo. Agrega los métodos/operaciones:

- **`.add(Object o, int pos)`** que sirve para agregar un elemento en una posición determinada
- **`.remove(int pos)`** que sirve para quitar un elemento en una posición determinada
- **`.get(int pos)`** que sirve para obtener un elemento en una posición determinada

Debajo de ***List*** se desprenden las implementaciones más comunes son:

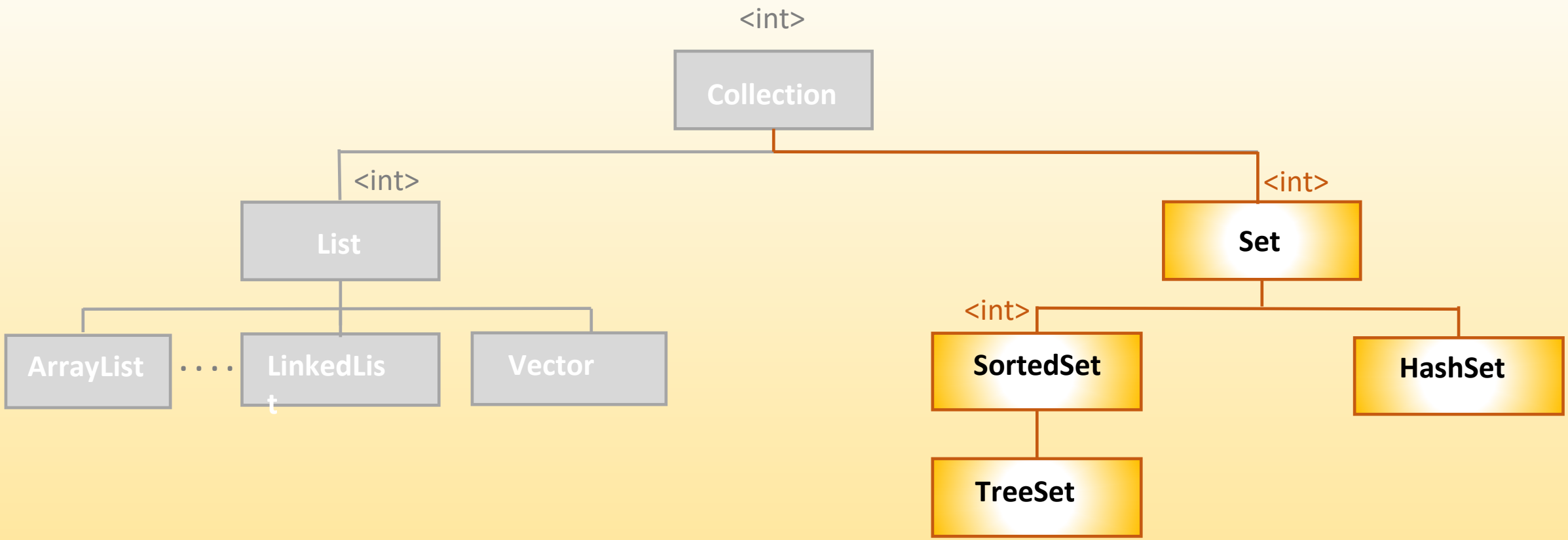
ArrayList como el nombre lo indica el soporte es un arreglo. No hay que preocuparse por manejarlo.

LinkedList cuyo soporte está basado en Nodos

Vector Igual que el LinkedList pero soporta accesos concurrentes.

Otras, como ser ***DoubleLinkedList***, ***FastList,*** *etc.*

Colecciones |Diagrama de flujo



Colecciones | Set

A la par de List se encuentra ***Set.***

Set especifica ***Collection*** en el sentido estricto de un conjunto, por lo tanto no admite duplicados, no admite valores nulos, y no dispone los elementos de manera contigua, por tanto no tiene acceso secuencial a los elementos y por ello no se pueden ordenar (sort).

Set a su vez tiene una hija que es ***SortedSet,*** que como su nombre lo indica **Sí** sirve para ordenar los elementos. La implementación más común de SortedSet es ***TreeSet***, que como su nombre lo indica su soporte es un árbol.

Otra implementación de *Set* es el ***HashSet***, que como su nombre lo indica su soporte es una tabla de Hash.

Colecciones | Recorrido

Al tener los elementos contiguos las listas se pueden recorrer de forma **similar a los arreglos *(for, while, do/while).***

Ejemplo con **for**:

```
for(int pos = 0; pos < miLista.size(); pos++) {
    System.out.println(miLista.get(pos));
 }
```

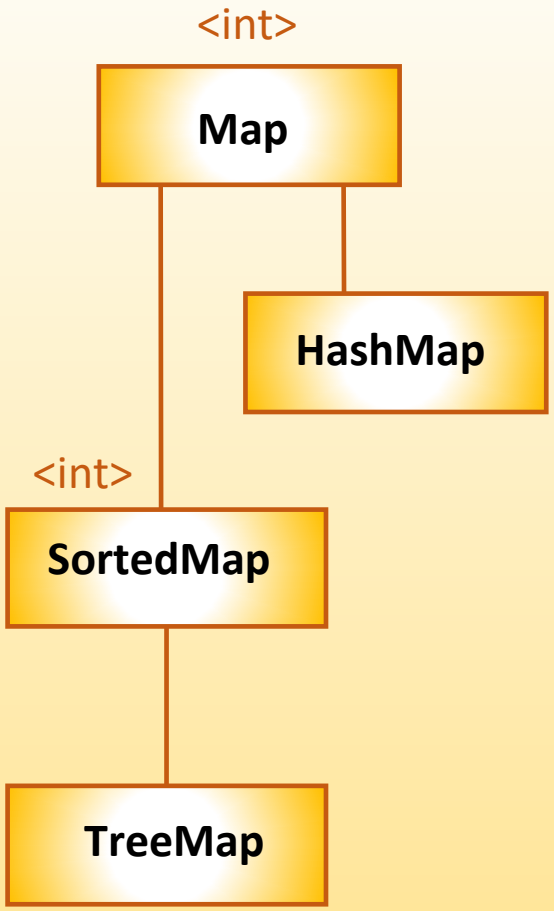
Las ***Collections*** en general y por tanto los ***Set*** (que dijimos que no tiene los elementos contiguos) se pueden recorrer con un ***Iterator.***

Ejemplo con **Iterator**:

```
for(Iterator i = miCollection.iterator();i.hasNext(); ) {
    System.out.println(i.next());
 }
```

Iterator es un patrón de diseño que sirve para recorrer elementos de una colección. Garantiza que se recorren todos los elementos una vez; no garantiza que cada vez que se recorran se haga en el mismo orden.

Colecciones |Diagrama de flujo



Colecciones | Mapas

Map: Los Mapas son conjuntos de ***tuplas*** (pares clave-valor). Es razonable pensar que las claves no se pueden repetir. Cada clave corresponde sólo a un valor.
En un ***Map*** los elementos no necesariamente están ordenados de forma contigua (depende de la implementación), por ello los elementos no están ordenados.

Las operaciones básicas de ***Map*** son:

`put(Object key,Object value)`

`get(Object key)` devuelve un Object válido.

`size()` da el tamaño del Map

`clear()` vacía el Map

`keySet()` que devuelve un Set (valores que no se pueden repetir) con todas las claves)

`.values()` que devuelve un List con todos los valores

Map tiene una hija que es ***SortedMap***; cuya implementación más común es ***TreeMap***, que como su nombre lo indica su soporte es un árbol. La implementación más común de Map es el ***HashMap***.