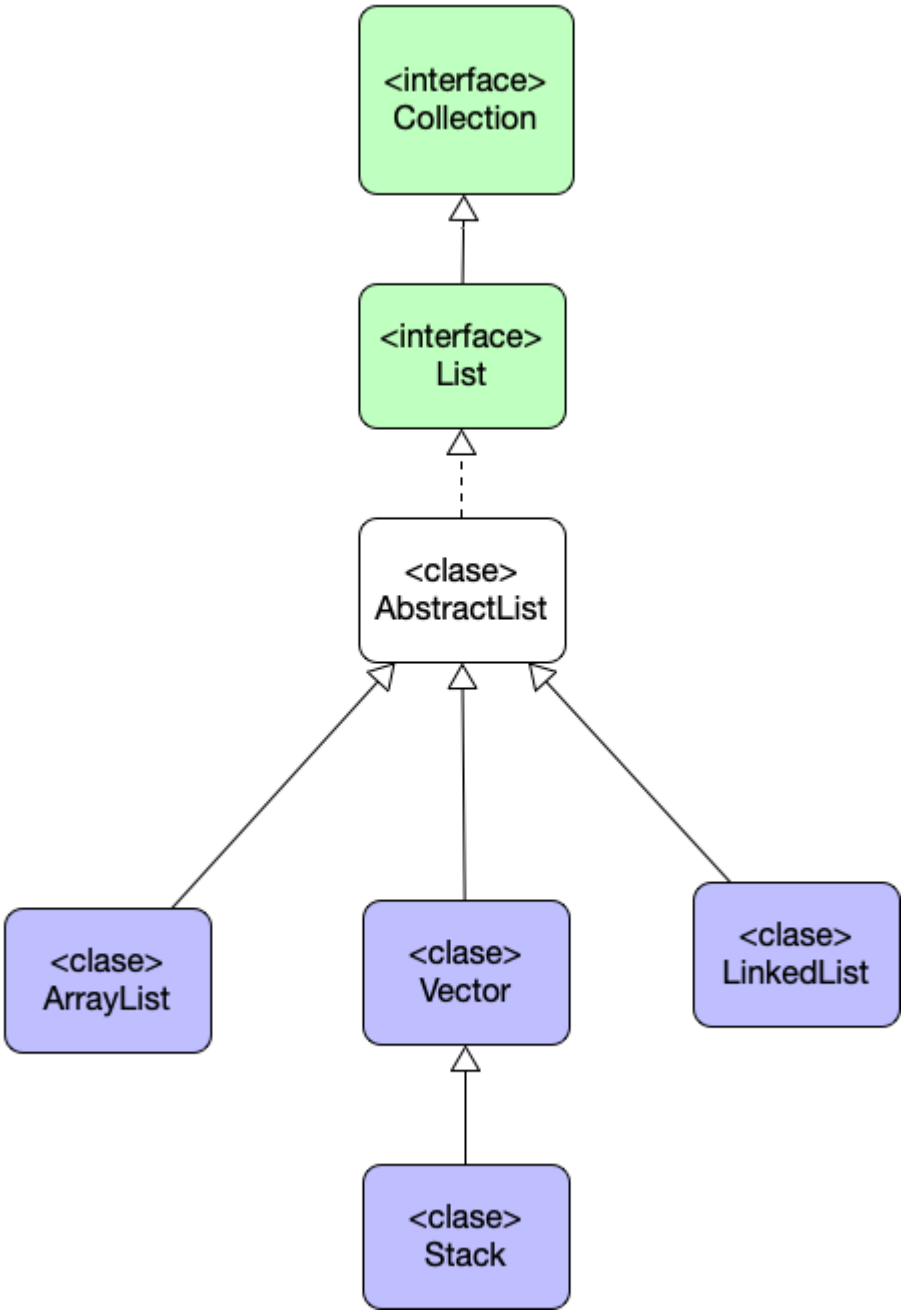
Tabla de Contenidos

- Java Collections Framework
- Java Set
- Java Queue
- Java Map
- Java 8 Collections y mejoras (Premium)
- Java Iterable
- Java Collections Framework y Homogeneidad
- Otros artículos relacionados

Java Collections Framework es el framework de Java que se encarga de gestionar todas los tipos de Colecciones que el lenguaje Java soporta . En sus primeras versiones tenía una estructura bastante sencilla pero con el paso del tiempo se han ido ampliando los elementos vamos a echar un vistazo a la collección entera:

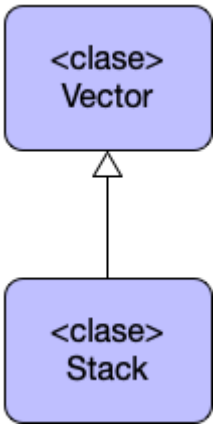
Java Collections Framework

El concepto más importante de Java a nivel de Java Collections Framework es quizás el concepto de Collection . Un collection es un conjunto de elementos sin un orden en concreto . En Java no es una clase sino un interface.



Por debajo de Collection esta el interface List que hace referencia a un grupo de elementos accesibles por posición . Este interface incluye el método get(i) para acceder a cualquier posición de la lista. Una vez tenemos claros los dos interfaces fundamentales . Nos encontramos con la clase AbstractList que es una clase abstracta que implementa parte de la funcionalidad para luego volver a heredar y encontrarnos con la clases instanciables o concretas:

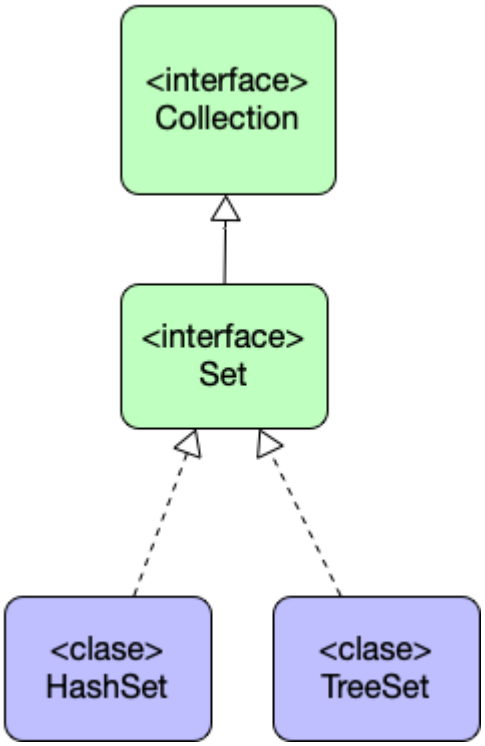
1. ArrayList : La clase más habitual que implementa al interface List y nos permite recorrer una lista estandar de elementos.
2. LinkedList: Una clase que añade métodos adicionales y permite disponer de una lista doblemente enlazada en la que las inserciones y modificaciones de la lista sean muy rápidas
3. Vector: Esta clase es legacy y se encarga de gestionar el concepto de lista de una forma Thread save . La clase existe desde Java 1.0
   La peculiaridad que tiene la clase Vector es que debajo de ella existe otra clase que sí que se usa y es la clase Stack que define el concepto de Pila.



Una vez que tenemos claro como funcionan la listas es momento de pasar a los conjuntos o Sets

Java Set

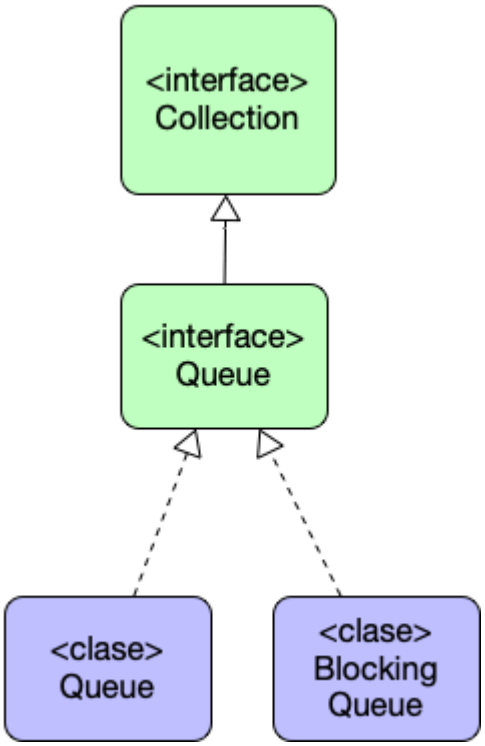
Set es otro de los interfaces fundamentales de Java . Un Set o conjunto define un conjunto de elementos que no contiene repetidos a nivel de Java existen varias implementaciones del interface Set



1. HashSet: Es la implementación de referencia del concepto de conjunto y permite añadir elementos no repetidos al grupo .
2. TreeSet: Es una implementación más compleja ya que mantiene los elementos del conjunto ordenados.

Java Queue

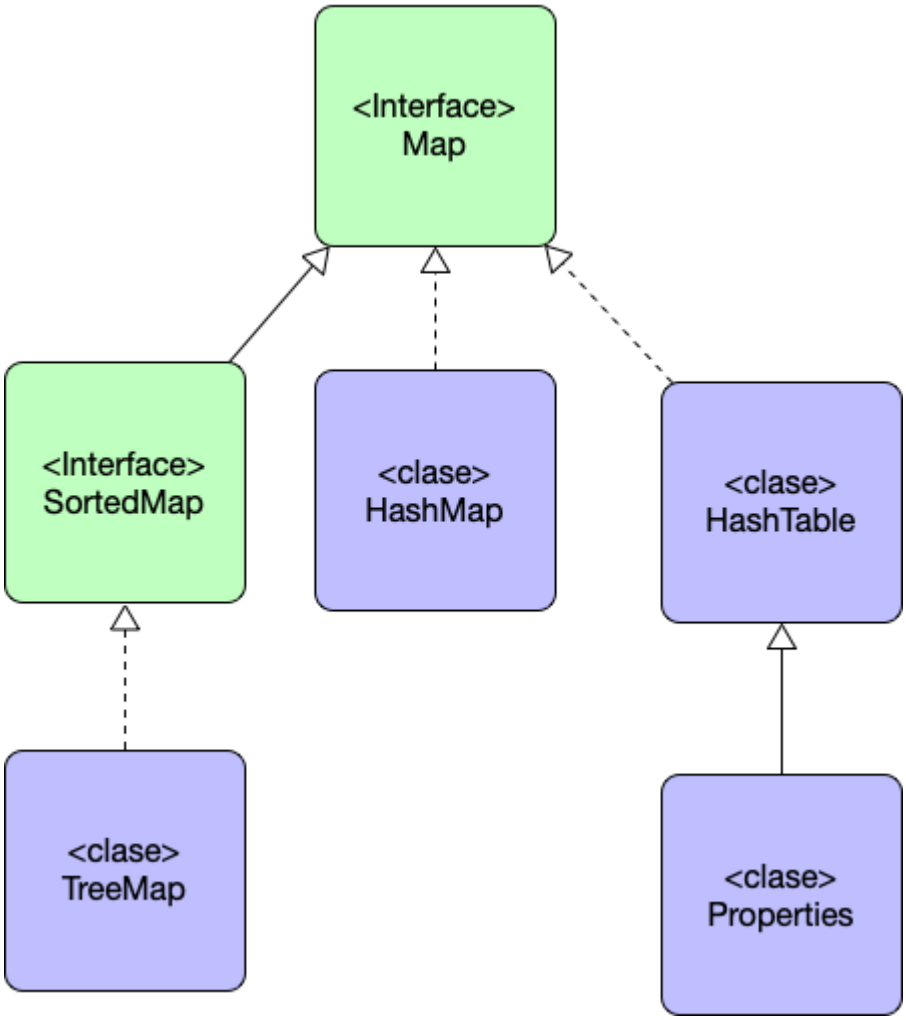
Otro de los conceptos fundamentales de Java el manejo de Colas de elementos en donde el primero que entra es el último que sale . Java define un interface Queue



1. DeQueue: Es la clase más habitual de Java para manejar colas
2. BlockingQueue: Es la clase de Java con la que nos encargamos de manejar una cola que necesita sincronización a nivel de programación concurrente.
   Acabamos de ver los tres interfaces que extienden de Collection es momento de abordar los interfaces Map o de diccionario que no están relacionados de forma directa con los anteriores pero que son imprescindibles para cualquier manejo del día a día.

Java Map

El concepto de Mapa o diccionario define los conceptos de clave y valor .Un diccionario siempre contiene una palabra y luego su definición o descripción . En un mapa en Java pasa algo similar disponemos de el tipo clave y luego de su valor. Vamos a ver las clases que están relacionadas con el concepto de diccionario.



En este caso existen 4 clases ligadas a los diccionarios:

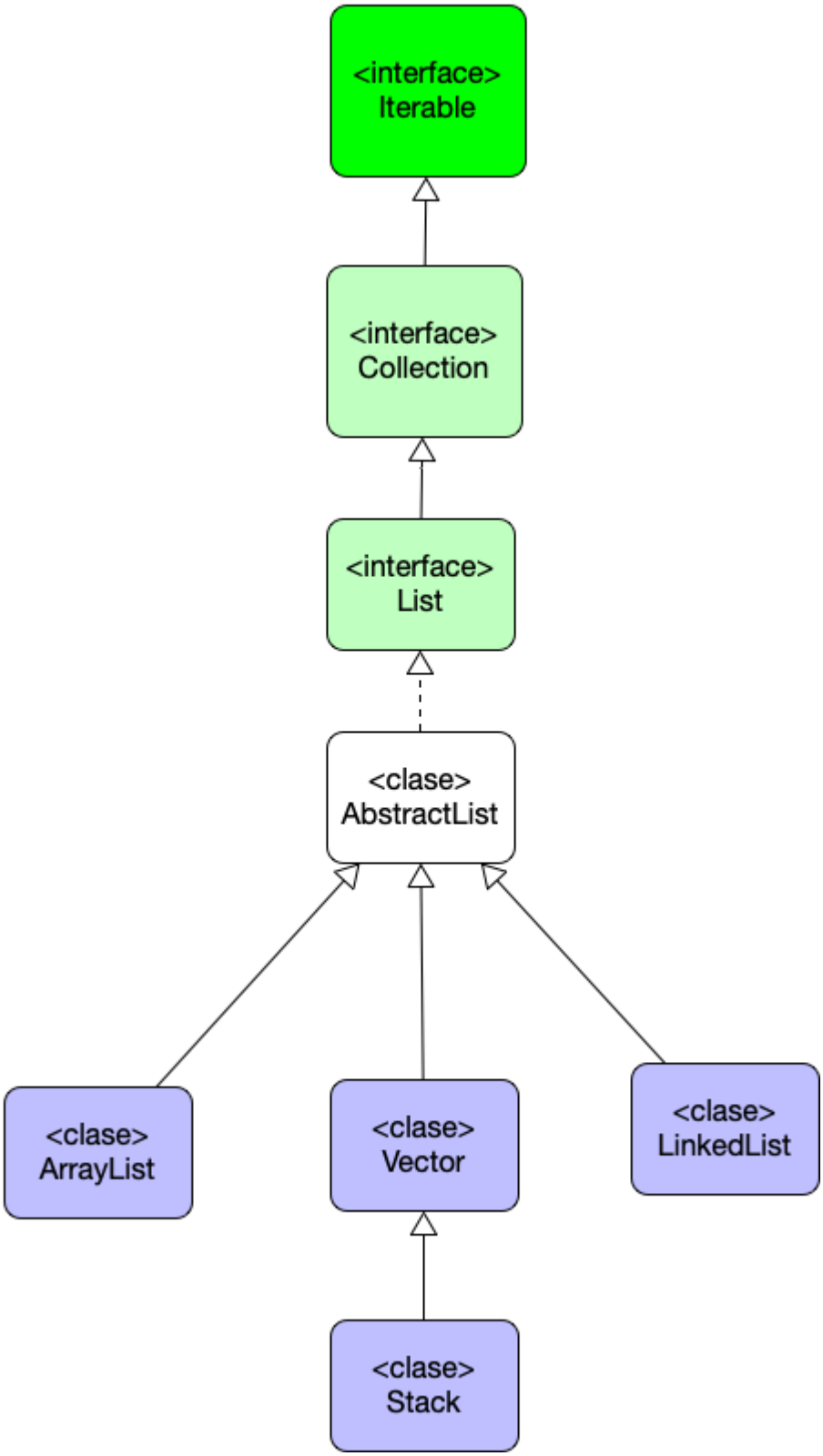
HashMap: La implementación por defecto y la que se usa más habitualmente

HashTable : La implementación legacy que existe desde Java 1.0 y esta sincronizada. De esta clase hereda la clase de Properties que es usada para cargar parejas de claves valor desde ficheros de texto.

TreeMap: Un diccionario ordenado que implementa no solo el interface Map sino el interface SortedMap que permite operaciones ligadas al ordenamiento.

Java 8 Collections y mejoras (Premium)

[ihc-hide-content ihc_mb_type=”show” ihc_mb_who=”4″ ihc_mb_template=”1″]¿Qué es lo que aporta el framework de colecciones a partir de Java 8 ? Quizás la reseña más importante es el añadir el interface Iterable por encima del interface collection . Esto puede parecer algo básico o algo casi ridículo ya que el interface Iterable añade poca funcionalidad a lo que ya tenemos conocido sin embargo es el que nos permite integrar de forma más natural la programación funcional dentro del lenguaje Java.


<interface>
Iterable
<interface>
Collection
<interface>
List
<clase>
AbstractList
<clase>
ArrayList
<clase>
Vector
<clase>
LinkedList
<clase>
Stack

Java Iterable

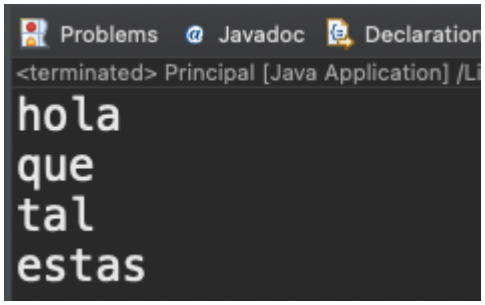
¿Qué métodos tiene el interface Iterable ? La realidad es que solo tres :

iterator: Este método devuelve un iterador y nos permite recorrer la colección de elementos que tengamos

forEach: El método más importante a nivel de colecciones ya que permite recorrer la colección como si se tratara de un Java Stream . A veces nos provoca equívocos pero su utilidad es grande.

```
package com.arquitecturajava;

import java.util.List;

public class Principal {

  public static void main(String[] args) {
    Iterable<String> lista= List.of("hola","que","tal","estas");
    lista.forEach(System.out::println);

  }

}
```

Este método recibe un Consumer es decir una clase que implementa el interface Consumer y procesa cada uno de los items para imprimirlos.



El último método que soporta este interface es

SplitIterator:Este método se utiliza a nivel de Java Streams para procesamiento en paralelo

Java Collections Framework y Homogeneidad

La creación del interface Iterable y del método forEach afecta a todo el API de collecciones de Java incluido a los Mapas o diccionarios ya que aunque estos nos se consideran colecciones como si mismos si beben de las modificaciones que se realizaron sobre las listas incluyendo un método forEach para recorrer los mapas.

```
package com.arquitecturajava;

import java.util.Map;

public class Principal2 {

  public static void main(String[] args) {
    Map<Integer,String> mapa= Map.of(1,"ordenador",2,"Tablet");
    mapa.forEach((clave, valor) -> System.out.println((clave + ":" +
valor)));
  }
}
```

Otros artículos relacionados

1. ¿Que es un Java Stream?
2. Java 8 Lambda y forEach (II)
3. ¿List vs ArrayList que es mejor?
4. Java Map

[/ihc-hide-content]