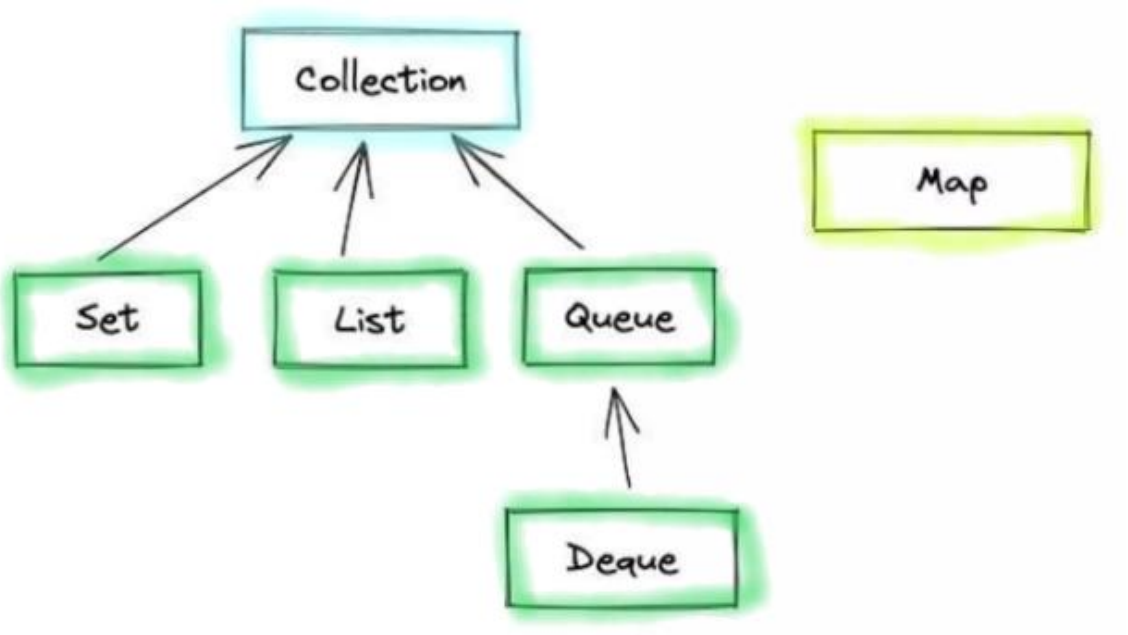
Collections



Collections es un FrameWork de interfaces que se implementó para poder proporcionar o implementar la manipulación de elementos o datos, sin preocuparse por los detalles de implementación, básicamente y a muy grandes rasgos los collections funcionan a base de for eachs, entre otras operaciones como add(ítem), size, isEmpy(), por poner algunos ejemplo.

Antes de Collection, hay una interface llamada Iterable y “collection” implementa de ella. La excepción a la regla es la interfaz Map, la cual se considera parte de las collections por el comportamiento que tiene, pero esta no implementa de “Iterable”.

compact1, compact2, compact3
java.lang

Interface Iterable<T>

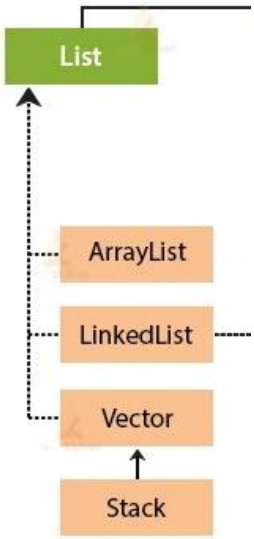
Type Parameters:

`T - the type of elements returned by the iterator`

All Known Subinterfaces:

`BeanContext, BeanContextServices, BlockingDeque<E>, BlockingQueue<E>, Collection<E>, Deque<E>, DirectoryStream<T>, List<E>, NavigableSet<E>, Path, Queue<E>, SecureDirectoryStream<T>, Set<E>, SortedSet<E>, TransferQueue<E>`

Desde la interfaz “Collection” tenemos 3 tipos de collections que de igual forma son interfaces:



List:

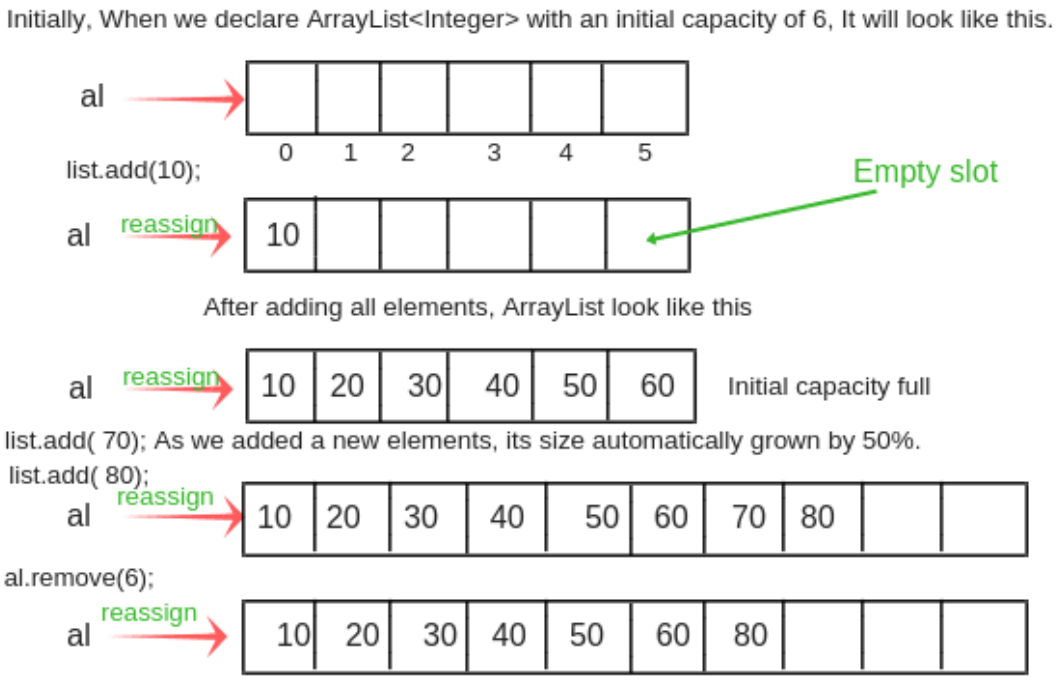
Son Collections que mantienen un orden, y mantienen el orden de inserción, permite null’s y duplicados, sus elementos pueden ser accedidos por índices

De acuerdo con la Documentación de Oracle:
“*El usuario de esta interfaz tiene un control preciso sobre en qué parte de la lista se inserta cada elemento. El usuario puede acceder a los elementos por su índice entero (posición en la lista) y buscar elementos en la lista... proporciona cuatro métodos para el acceso posicional (indexado) a los elementos de la lista. Las listas (como las matrices de Java) están basadas en cero. Tenga en cuenta que estas operaciones pueden ejecutarse en un tiempo proporcional al valor del índice para algunas implementaciones (la clase LinkedList, por ejemplo). Por lo tanto, la iteración sobre los elementos de una lista suele ser preferible a la indexación si la persona que llama no conoce la implementación.*”

Las clases más utilizadas y que veremos a continuación de la interfaz List son ArrayList y LinkedList:

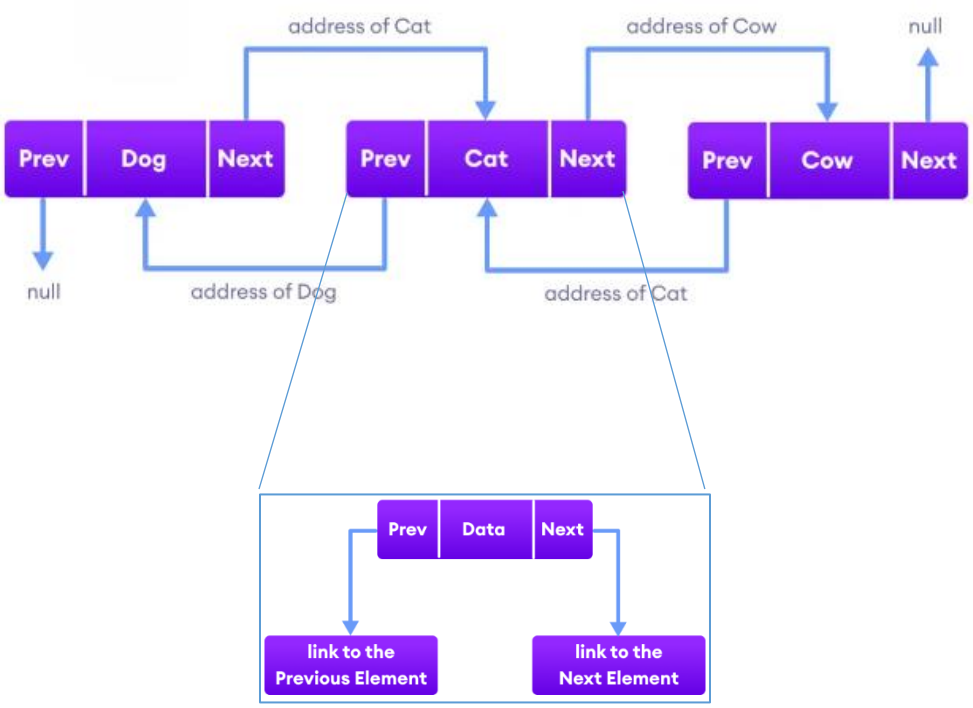
ArrayList:

Las elementos en las ArrayList son ordenados en la manera en la que fueron insertados, permite un acceso aleatorio debido a que tiene índice, y cuando se manipula, se recorre todo el array para poder hacer un cambio, lo cual se le conoce como manipulación lenta. Un arraylist puede guardar un arraylist dentro de este. A nivel de código funciona de la siguiente manera:



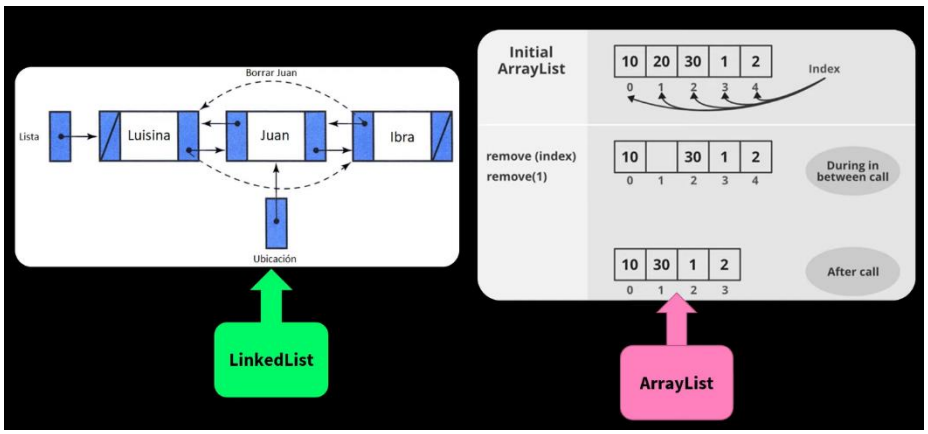
LinkedList:

Como su nombre lo indica, las linkedList son listas en las cuales sus elementos de en medio están enlazadas entre sí, por medio de punteros como lo muestra en la imagen, estos punteros apuntan hacia el elemento anterior o el elemento posterior, a excepción de la cabeza que en este caso es "Dog", el cual solo apunta hacia adelante, y la cola, que en este caso es "Cow" y solo apunta hacia atrás. A diferencia de los ArryList, estos no tienen índices.

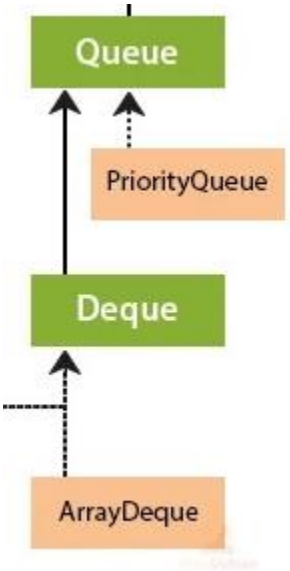


Una diferencia entre estas clases es la manera en la que ellas eliminan uno de sus elementos, si bien utilizan el mismo método .remove(), lo que pasa con ellas es particular para cada caso:

Para el caso del ArrayLIst (lado derecho de la imagen), se elimina el elemento seleccionado con .remove(1); y se elimina esa posición, posteriormente los elementos se recorren hacia la izquierda para ocupar ese espacio vacío.
A diferencia de lo que pasa en la LinkedList, cuando se elimina el registro, lo que pasa es que se reacomodan los punteros, si yo elimino a un elemento, se eliminan sus punteros, pero los registros de los elementos que quedan necesitan apuntar a alguien, por lo que apuntan a el subsecuente y al anterior respectivamente.



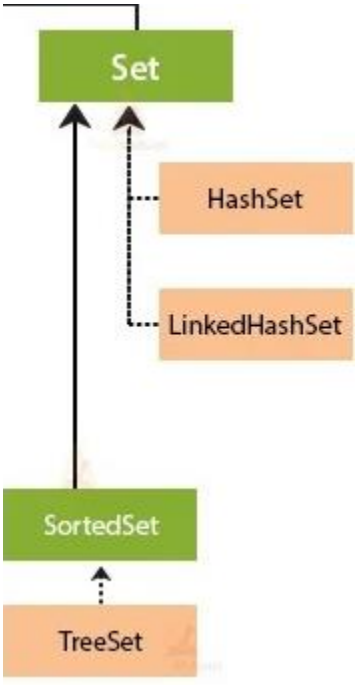
Queue



La interfaz Queue es como su nombre lo indica, tiene por clases elementos que se pueden asemejar a una fila, en ellas, se puede agregar elementos al final de la fila y sacar elementos de la parte delantera. Es muy útil cuando se necesita manejar una serie de tareas en el orden en que fueron agregadas.

La interfaz Queue ofrece métodos para agregar elementos (offer o add) al final de la cola, y métodos para eliminar y obtener elementos (poll o remove) desde el frente de la cola. También hay métodos para examinar el elemento en el frente sin eliminarlo (peek).

Set



La interfaz Set se trata de una colección de elementos únicos, lo que significa que no puede contener duplicados. Esto se logra utilizando los conceptos matemáticos de conjunto, donde cada elemento es único dentro del conjunto.

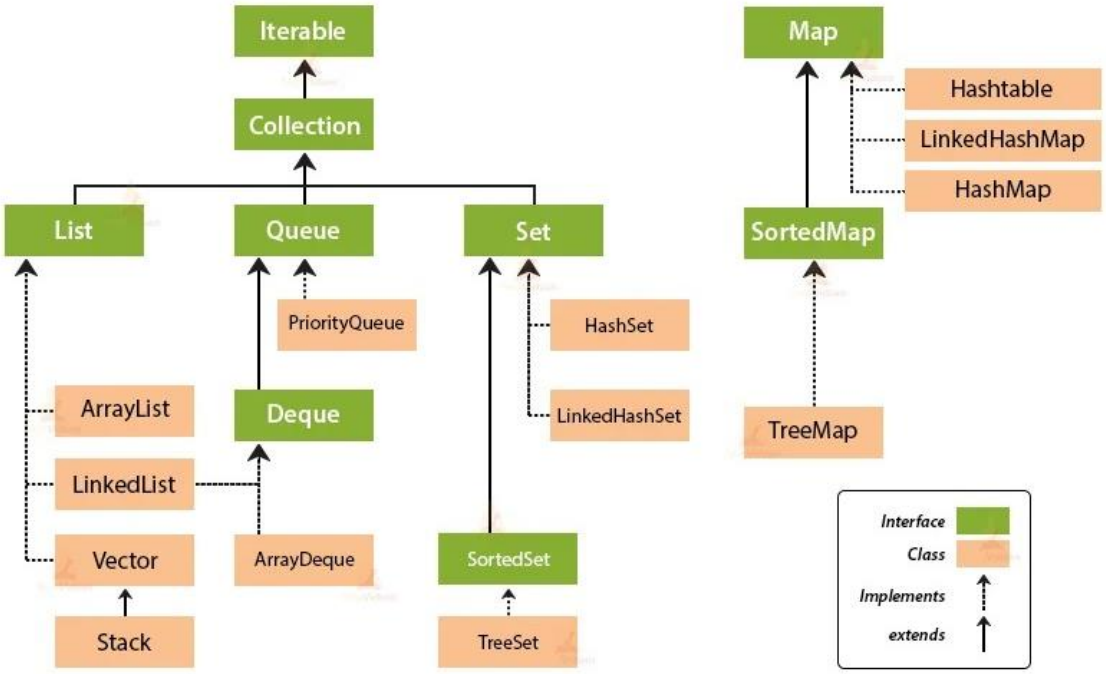
Las implementaciones de la interfaz Set no permiten elementos duplicados y proporcionan métodos para agregar (add) y eliminar (remove) elementos, así como también para verificar la existencia de un elemento (contains). No proporcionan un orden específico para los elementos, a diferencia de las listas. Algunas de sus clases mas utilizadas son las siguientes:

HashSet: Almacena los elementos en una tabla hash para un acceso rápido y eficiente.

TreeSet: Mantiene los elementos ordenados en orden natural o según un comparador personalizado.

LinkedHashSet: Mantiene el orden de inserción además de eliminar duplicados.

Este es el diagrama que ejemplifica las relaciones de implementación y herencia que tienen las diferentes Collections con sus clases más representativas.



Como se menciono arriba, la interfaz Map es considerada una collection, a pesar de no estar implementada a parir de dicha interfaz.

Map

Esta interfaz es muy popular debido a que nos permite almacenar objetos de clave valor, como si se tratara de tener una llave primaria. Es como un diccionario en el que puedes buscar un valor utilizando una clave única.

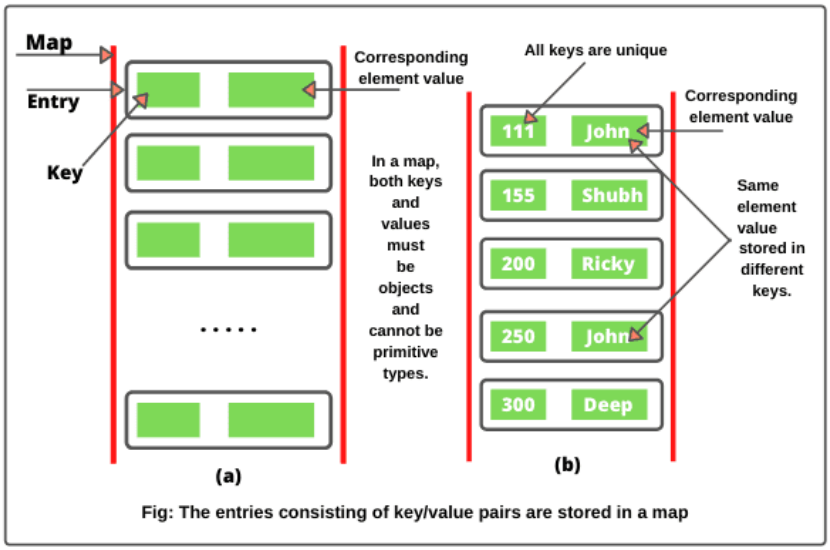


Fig: The entries consisting of key/value pairs are stored in a map