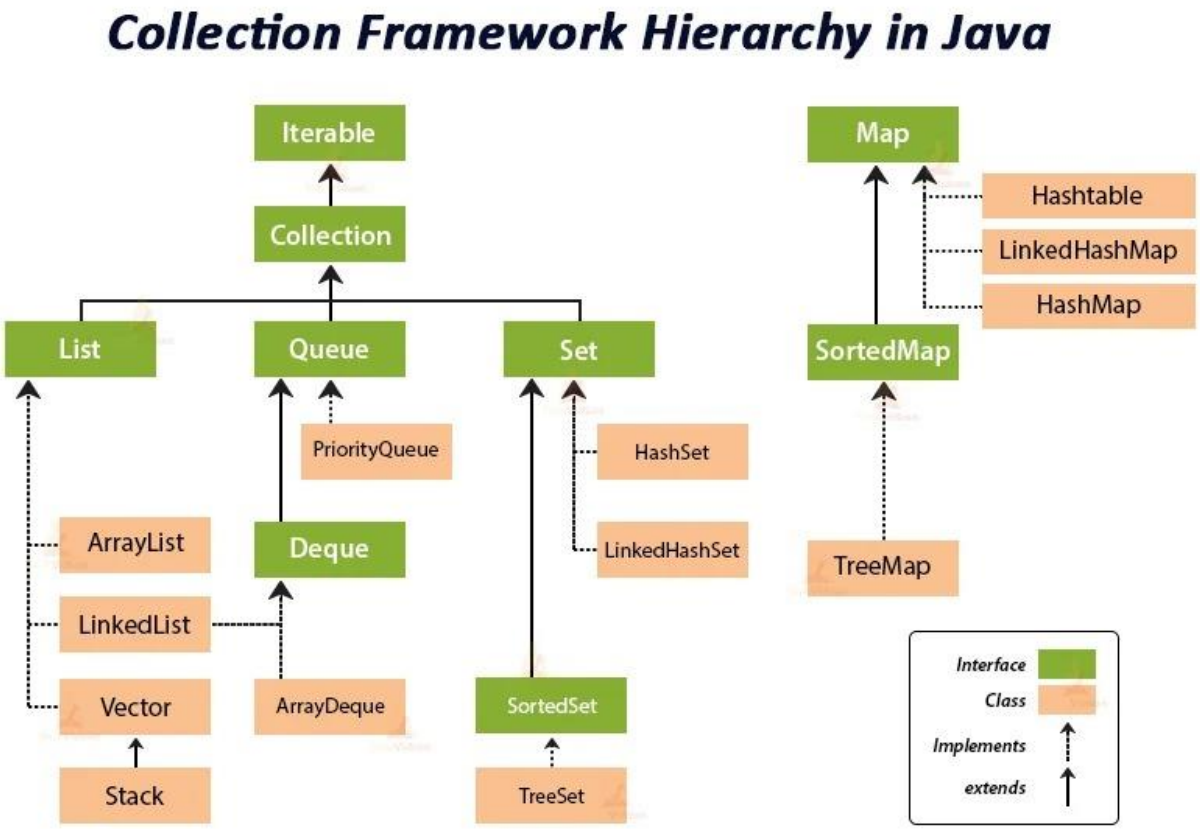
Collections en Java

Natalia Reyes Altamirano

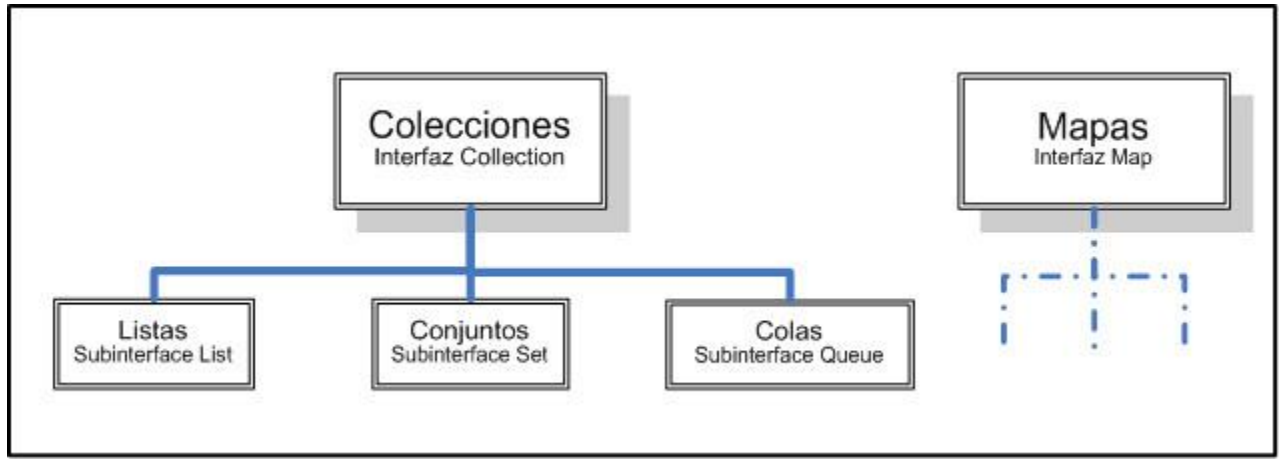
¿Qué son las collections?

- Son estructuras de datos que permiten almacenar y manipular grupos de objetos de manera eficiente.
- Permiten cambiar el tamaño de manera dinámica, proporcionan métodos para agregar, eliminar y buscar elementos, y ofrecen algoritmos y utilidades para manipular los datos de manera eficiente.
- Hacen que el código sea más legible, mantenible y reutilizable al proporcionar interfaces comunes para operar sobre los diferentes tipos de estructuras de datos.



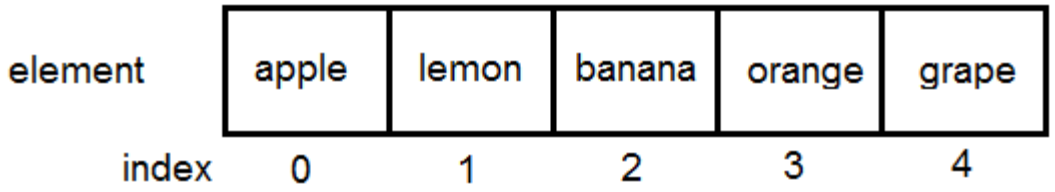
Interfaz Collection

- Define una colección de elementos, que es una agrupación de objetos.
- Es parte del paquete java.útil
- No define un orden específico para los elementos ni impone restricciones sobre el tipo de colección que debe ser.
- Es una interfaz genérica, lo que significa que puede ser parametrizada con un tipo específico para indicar el tipo de elementos que la colección contendrá.



List

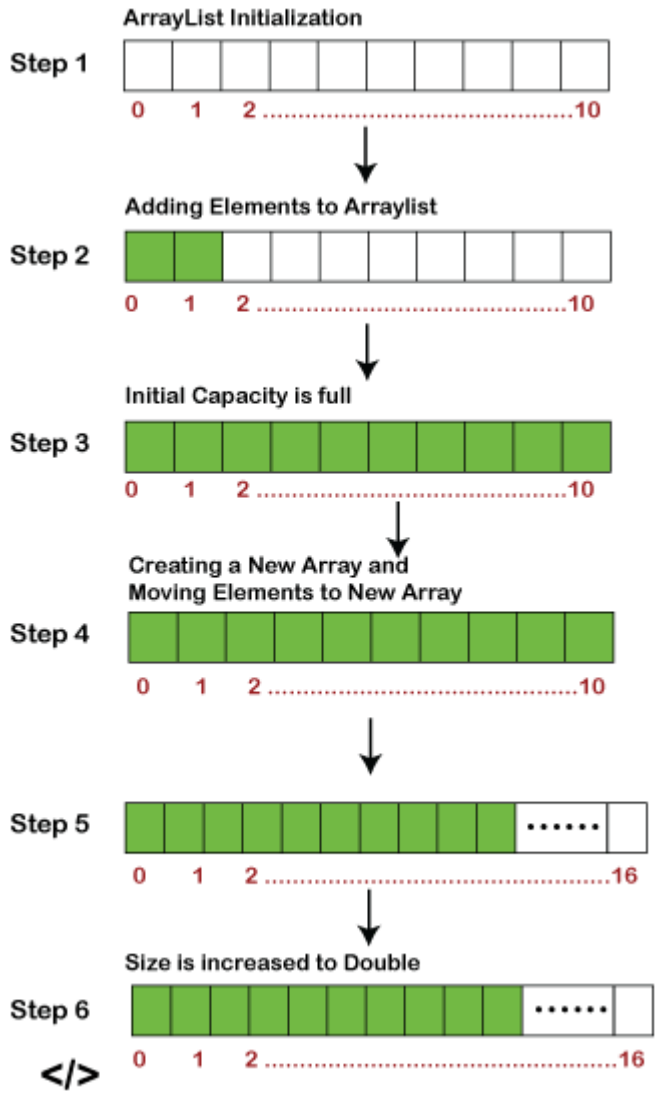
- Es una lista no ordenada
- Se mantiene el orden de los elementos pudiendo acceder a su contenido según la posición. Esta lista crece según las necesidades.



A List collection

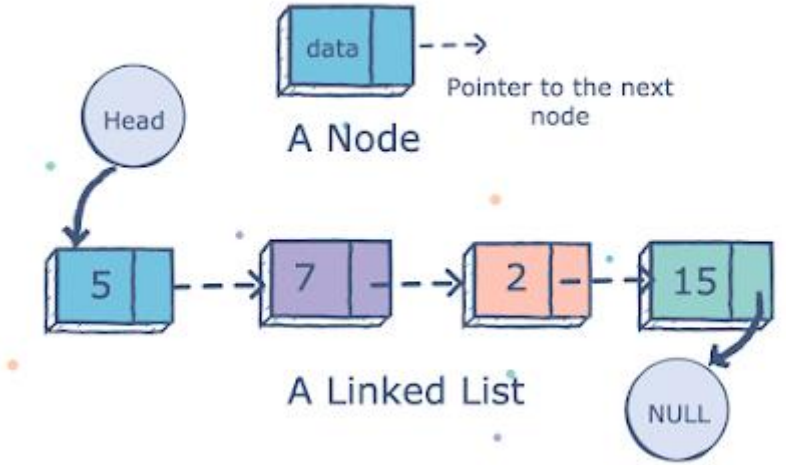
ArrayList

- Es una estructura de datos dinámica que proporciona un acceso rápido a elementos individuales y es adecuada para operaciones de inserción y eliminación ocasionales.
- No está preparada para trabajar con varios hilos



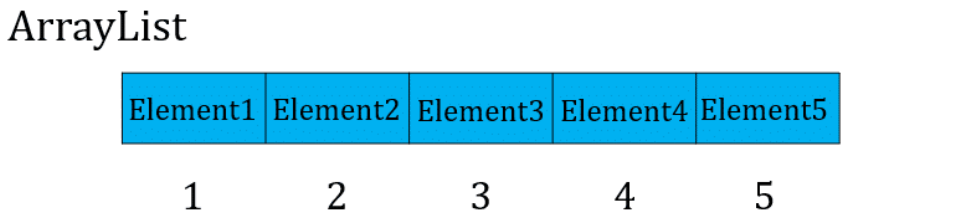
LinkedList

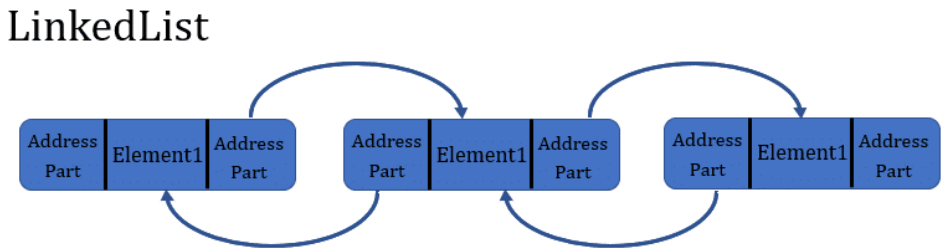
- Se implementa mediante una lista entrelazada, formada por nodos que apuntan al elemento siguiente y el elemento anterior.
- Esta implementación favorece las inserciones y el borrado, pero hacen muy lento el recorrido.



Diferencias ArrayList y LinkedList

- ArrayList es más adecuado para situaciones donde se requiere un acceso rápido a elementos individuales y donde las inserciones y eliminaciones ocurren principalmente al final de la lista.
- LinkedList es más adecuado cuando las inserciones y eliminaciones frecuentes se realizan en el inicio o el final de la lista y cuando no se necesita un acceso frecuente a elementos por su índice.



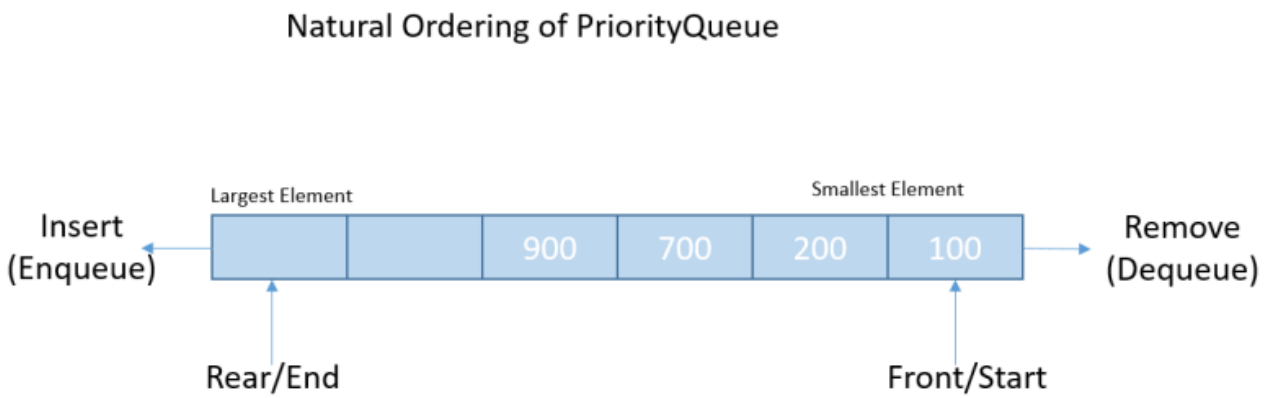


Diferencias ArrayList y LinkedList

ArrayList	LinkedList
ArrayList internally uses a dynamic array to store the elements	LinkedList internally uses a doubly linked list to store the elements
Manipulation with ArrayList is slow because it internally uses an array. If any element is removed from the array, all the bits are shifted in memory.	Manipulation with LinkedList is faster than ArrayList because it uses a doubly linked list, so no bit shifting is required in memory.
ArrayList consumes less memory than LinkedList	A LinkedList consumes more memory than an ArrayList because it also stores the next and previous references along with the data.
An ArrayList class can **act as a list** only because it implements List only.	LinkedList class can **act as a list and queue** both because it implements List and Deque interfaces.
ArrayList is better for storing and accessing data.	LinkedList is **better for manipulating** data.

PriorityQueue

- Es una clase que implementa una cola de prioridad.
- Representa una estructura de datos que organiza elementos en función de su prioridad. La prioridad se define mediante un comparador o el orden natural de los elementos.

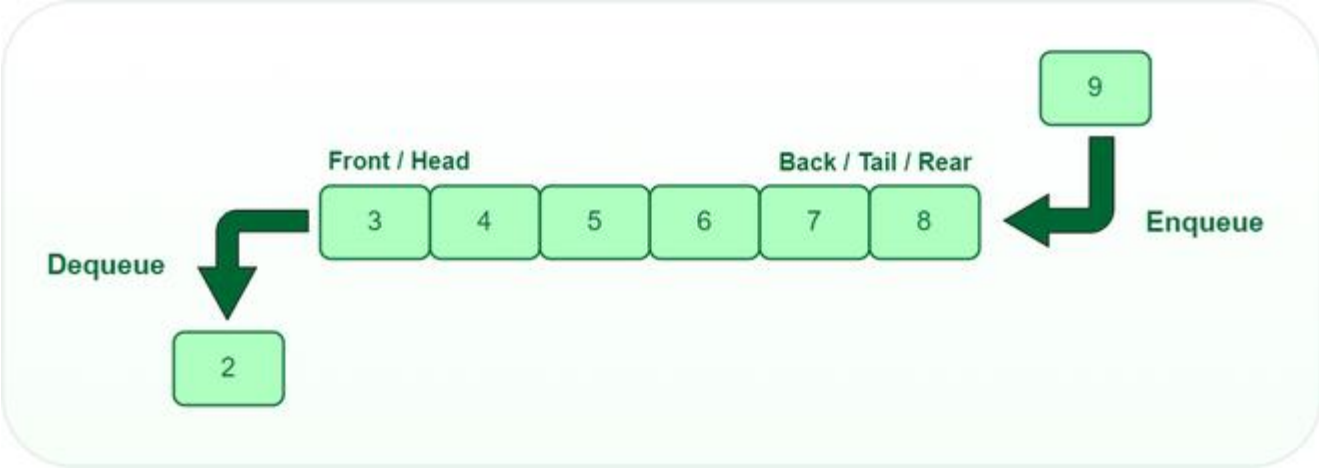


ArrayDeque

- Implementa una cola de doble final utilizando una matriz dinámica.
- Es decir, es una cola que permite la inserción y eliminación de elementos tanto por el frente como por la cola.

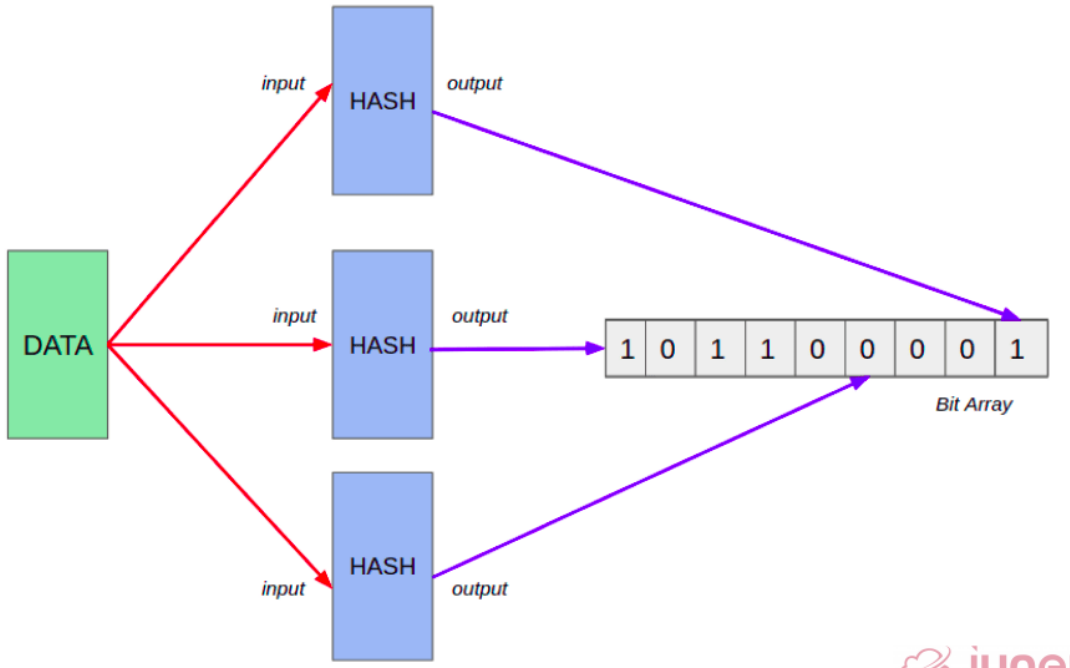
Diferencias ArrayDeque y PriorityQueue

- ArrayDeque es más adecuado cuando se necesita una cola de doble final con acceso rápido tanto al frente como a la cola.
- PriorityQueue es más adecuado cuando se necesita una cola de prioridad que ordene los elementos en función de su prioridad



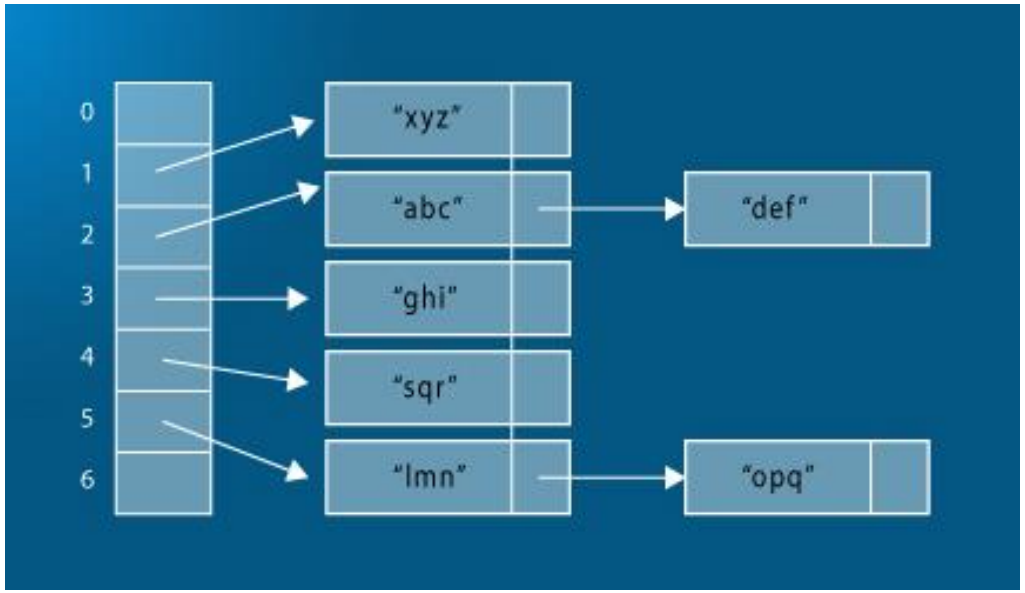
HashSet

- Implementa la interfaz Set y representa una colección de elementos únicos sin un orden específico.
- Utiliza una tabla hash para almacenar los elementos y proporciona un tiempo constante (O(1)) en promedio para operaciones básicas como agregar, eliminar y buscar elementos.



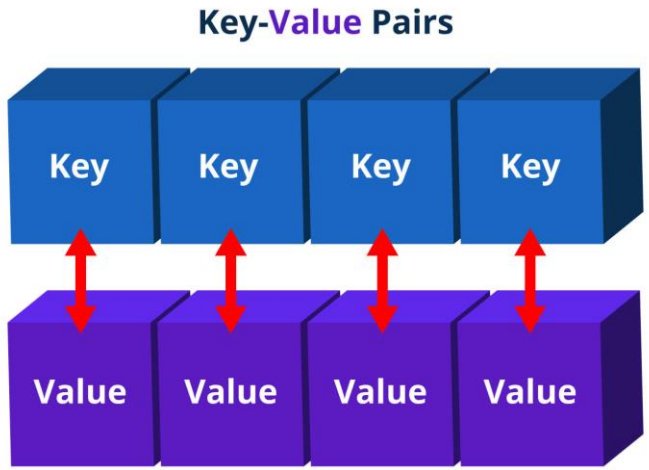
LinkedHashSet

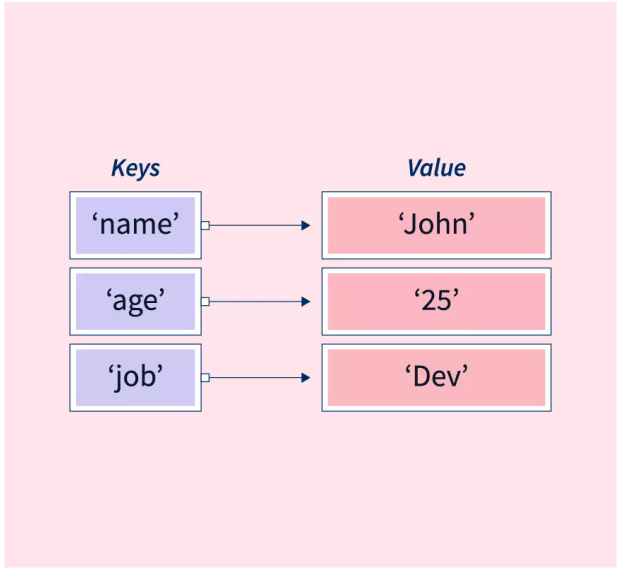
- Representa una colección de elementos únicos con un orden de inserción predecible.
- Mantiene el orden en el que los elementos fueron agregados al conjunto, lo que permite recorrerlos en el mismo orden en que se agregaron.



HashMap

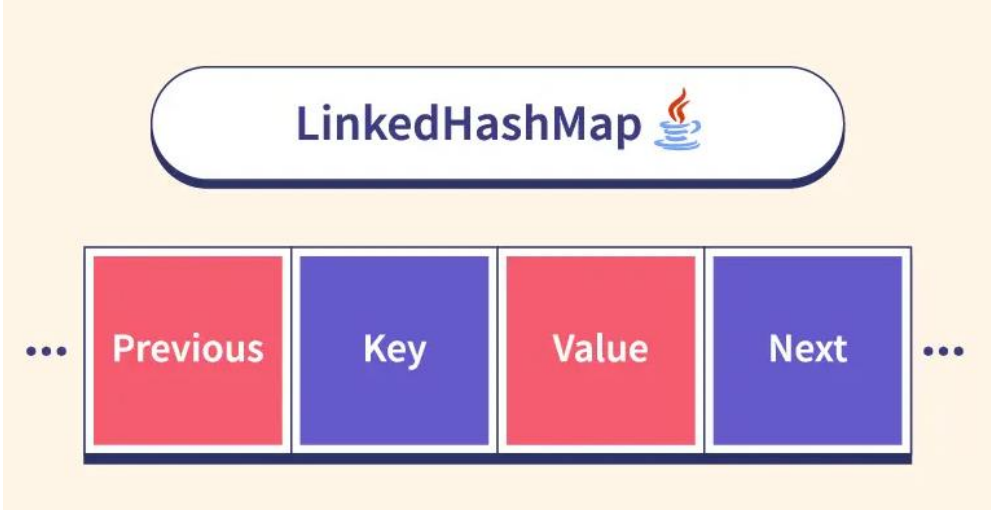
- Representa una implementación de mapa (diccionario) que permite almacenar pares clave-valor, donde cada clave debe ser única y se mapea a un solo valor.





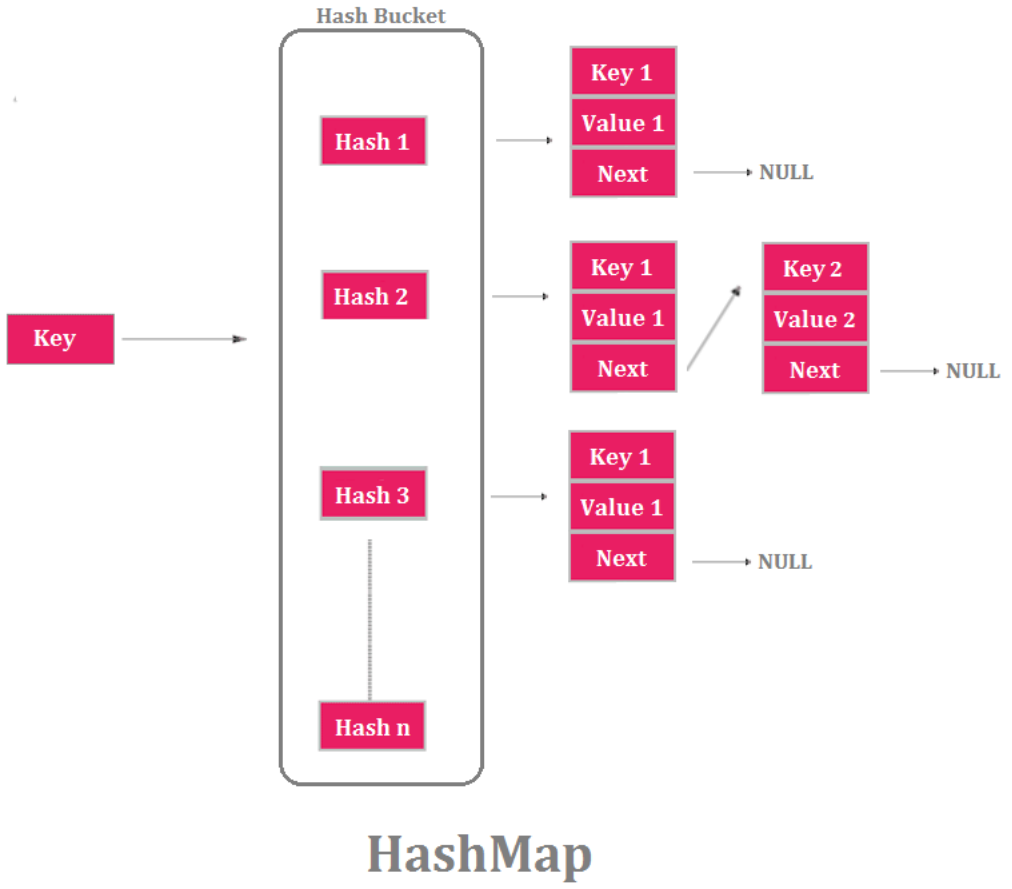
LinkedHashMap

- Representa una implementación de mapa que mantiene el orden de inserción de los elementos y permite acceso rápido a través de claves únicas.



Diferencias HashMap y LinkedHashMap

- HashMap es más adecuado cuando no se requiere un orden específico para los elementos y se busca un rendimiento óptimo en operaciones de búsqueda, inserción y eliminación.
- LinkedHashMap es más adecuado cuando se necesita mantener el orden de inserción de los elementos y se valora la iteración en el mismo orden en que se agregaron los elementos.



Diferencias HashMap y LinkedHashMap

HashMap	LinkedHashMap
HashMap doesn't maintain any order of keys or values.	LinkedHashMap maintains insertion order of keys, order in which keys are inserted in to LinkedHashMap.
HashMap requires less memory than LinkedHashMap	LinkedHashMap requires more memory than HashMap because LinkedHashMap maintains insertion order of keys.
Extends AbstractMap class	Extends HashMap class
A HashMap has a better performance than a LinkedHashMap	A LinkedHashMap has a less performance than a HashMap because a LinkedHashMap needs the expense of maintaining the doubly-linked list