Java Collections Framework y su estructura

```
{
  <Por="Guadalupe López"/>
 }
```

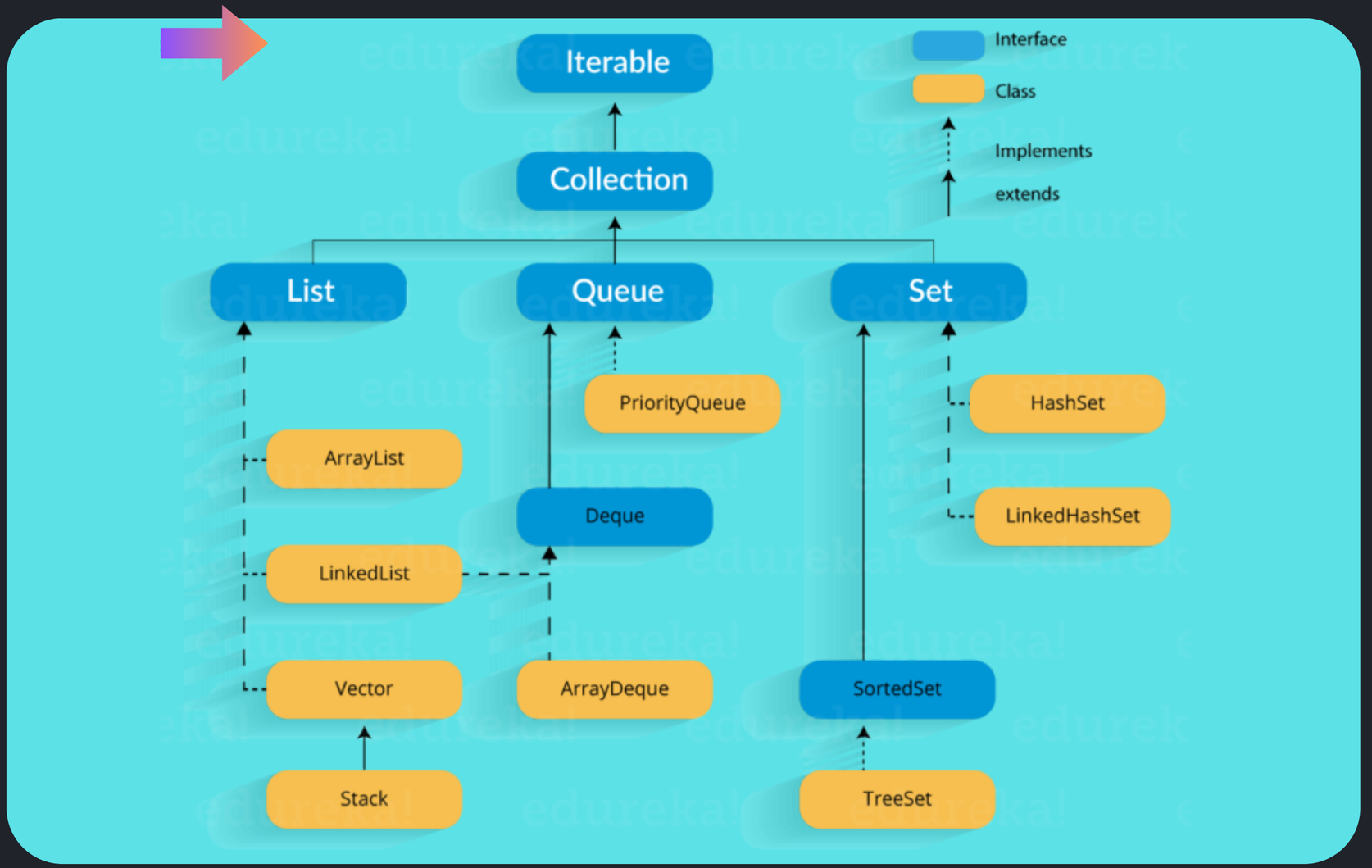
¿Qué es Java collection? {

una colección de Java es una colección de objetos representados como una sola unidad. La idea es proporcionar un conjunto de operaciones sobre los elementos agrupados lógicamente, como buscar, clasificar, etc.

Un collection es un conjunto de elementos sin un orden en concreto. En Java no es una clase sino un interface.

}

Estructura {


Iterable
Collection
List
Queue
Set
PriorityQueue
ArrayList
Deque
HashSet
LinkedHashSet
LinkedList
Vector
ArrayDeque
SortedSet
Stack
TreeSet
Interface
Class
Implements
extends

}

List

ArrayList y LinkedList { Ambas implementan la interfaz list y mantienen el orden de insercion. Ambas son clases no sincronizadas.

Diferencias

ArrayList	LinkedList
1) ArrayList utiliza internamente una **matriz dinámica** para almacenar los elementos.	LinkedList utiliza internamente una **lista doblemente enlazada** para almacenar los elementos.
2) La manipulación con ArrayList es **lenta** porque utiliza internamente una matriz. Si se elimina algún elemento de la matriz, todos los demás elementos se desplazan en la memoria.	La manipulación con LinkedList es **más rápida** que con ArrayList porque utiliza una lista doblemente enlazada, por lo que no se requiere un cambio de bits en la memoria.
3) Una clase ArrayList puede **actuar como una lista** solo porque implementa List solo.	La clase LinkedList puede **actuar como una lista y una cola** porque implementa las interfaces List y Deque.
4) ArrayList es **mejor para almacenar y acceder** a datos.	LinkedList es **mejor para manipular** datos.
5) La ubicación de memoria para los elementos de un ArrayList es contigua.	La ubicación de los elementos de una lista enlazada no es contagiosa.
6) Generalmente, cuando se inicializa una ArrayList, se asigna una capacidad predeterminada de 10 a la ArrayList.	No existe ningún caso de capacidad predeterminada en una LinkedList. En LinkedList, se crea una lista vacía cuando se inicializa LinkedList.
7) Para ser precisos, una ArrayList es una matriz de tamaño variable.	LinkedList implementa la lista doblemente enlazada de la interfaz de lista.

}

List

Vector { Es como la matriz dinámica que puede crecer o reducir su tamaño. A diferencia de la matriz, podemos almacenar n-número de elementos en ella ya que no hay límite de tamaño.
Se recomienda usar la clase Vector solo en la implementación segura para subprocesos.
Los iteradores devueltos por la clase Vector son a prueba de fallas . En caso de modificación simultánea, falla y lanza la ConcurrentModificationException.

ArrayList	Vector
1) ArrayList **no está sincronizado** .	El vector está **sincronizado** .
2) ArrayList **incrementa el 50 %** del tamaño actual de la matriz si el número de elementos supera su capacidad.	Los incrementos vectoriales **del 100 %** significan que se duplica el tamaño de la matriz si el número total de elementos supera su capacidad.
3) ArrayList **no es una clase heredada** . Se introduce en JDK 1.2.	Vector es una clase **heredada** .
4) ArrayList es **rápido** porque no está sincronizado.	Vector es **lento** porque está sincronizado, es decir, en un entorno de subprocesos múltiples, mantiene los otros subprocesos en estado ejecutable o no ejecutable hasta que el subproceso actual libera el bloqueo del objeto.
5) ArrayList usa la interfaz **Iterator** para recorrer los elementos.	Un vector puede usar la interfaz **de iterador** o la interfaz **de enumeración** para recorrer los elementos.

}

Deque

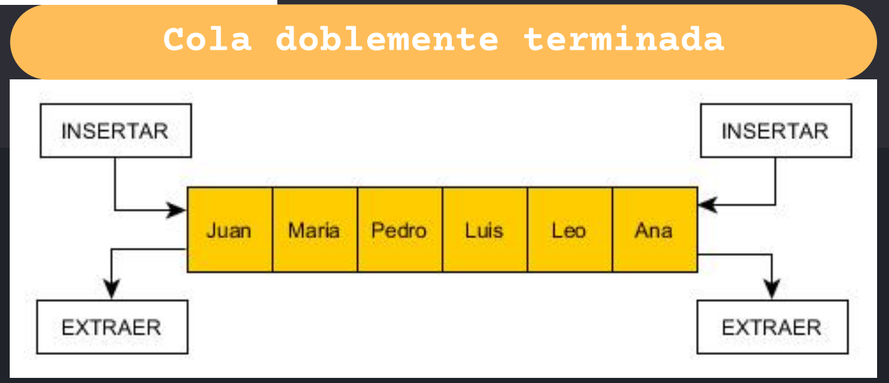
ArrayDeque {

Una cola doblemente terminada o deque es una estructura de datos lineal que permite insertar y eliminar elementos por ambos extremos, es decir implementa en una única estructura las funcionalidades de las pilas (estructuras LIFO) y las colas (estructuras FIFO), en otras palabras, estas estructuras podrían implementarse fácilmente con una deque. clase más robusta y completa que la clase Stack y tambien más rapida al ser usada como pila, más rápido que LinkedList cuando se utiliza como una cola . No tienen restricciones de capacidad, crecen según sea necesario, no admite elementos nulos.

Pilas (estructuras LIFO)

Colas (estructuras FIFO)

Cola doblemente terminada



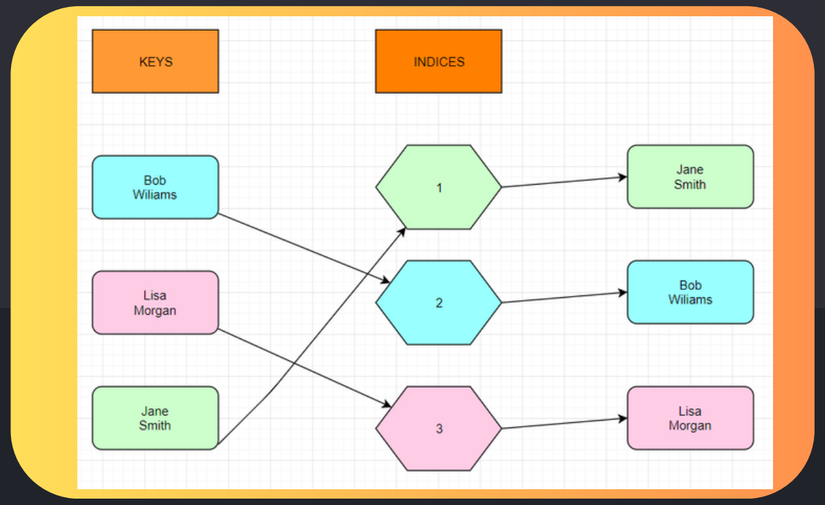
}

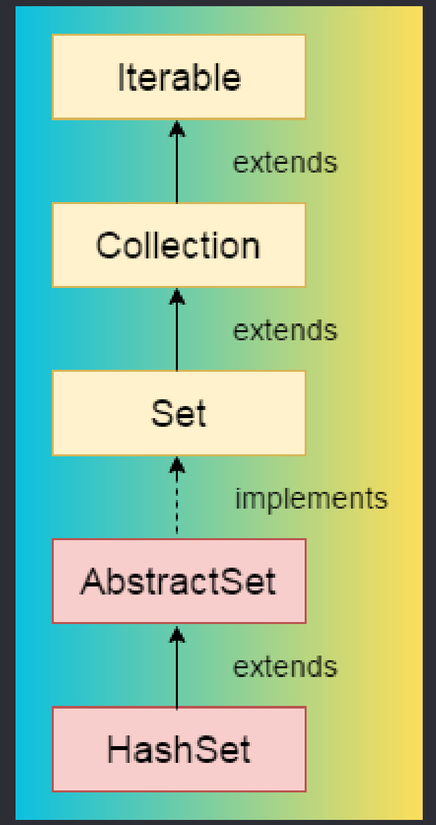
Set

HashSet {

Se usa para crear una colección que usa una tabla hash para el almacenamiento. Hereda la clase AbstractSet e implementa la interfaz Set.

- HashSet almacena los elementos usando un mecanismo llamado hashing.
- HashSet solo contiene elementos únicos.
- HashSet permite un valor nulo.
- La clase HashSet no está sincronizada.
- HashSet no mantiene el orden de inserción. Aquí, los elementos se insertan en función de su código hash.
- HashSet es el mejor enfoque para las operaciones de búsqueda.
- La capacidad predeterminada inicial de HashSet es 16 y el factor de carga es 0,75.





}

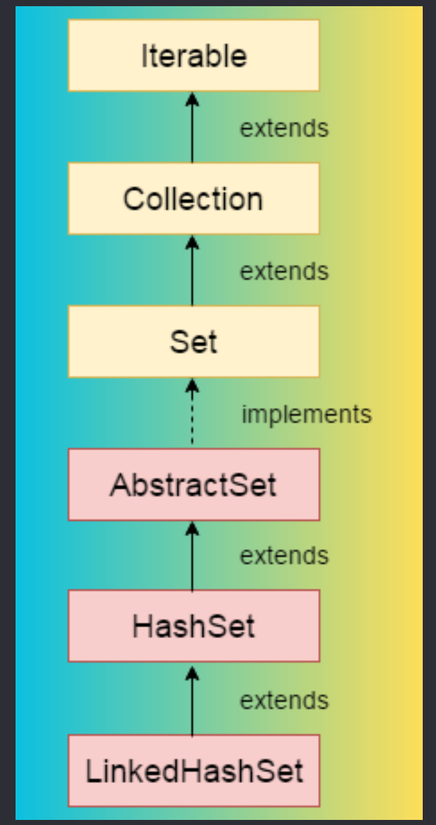
Set

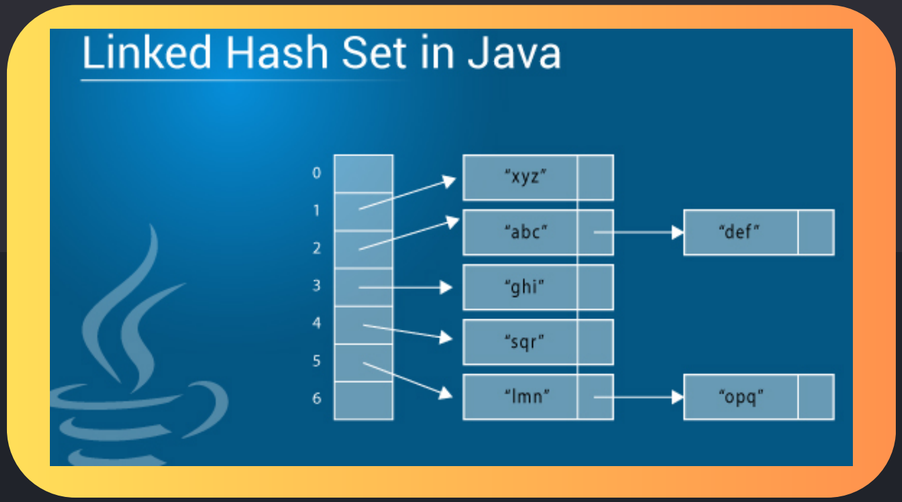
LinkedHashSet {

Los puntos importantes sobre la clase Java LinkedHashSet son:

- La clase Java LinkedHashSet contiene elementos únicos solo como HashSet.
- La clase Java LinkedHashSet proporciona todas las operaciones de conjuntos opcionales y permite elementos nulos.
- La clase Java LinkedHashSet no está sincronizada.
- La clase Java LinkedHashSet mantiene el orden de inserción.

La clase LinkedHashSet amplía la clase HashSet, que implementa la interfaz Set. La interfaz Set hereda las interfaces Collection e Iterable en orden jerárquico.





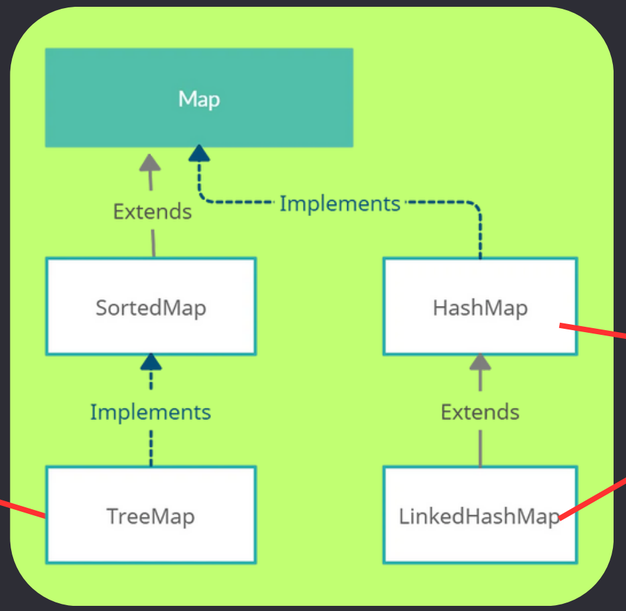
}

Interfaz Map

Map {

- Un mapa contiene valores sobre la base de la clave, es decir, par de clave y valor. Cada par de clave y valor se conoce como una entrada.
- Un mapa no permite claves duplicadas, pero puede tener valores duplicados.
- Un mapa es útil si tiene que buscar, actualizar o eliminar elementos en función de una clave.

Hay dos interfaces para implementar Map en Java: Map y SortedMap, y tres clases: HashMap, LinkedHashMap y TreeMap. La jerarquía de Java Map se da a continuación:



TreeMap no permite ninguna clave o valor nulo.

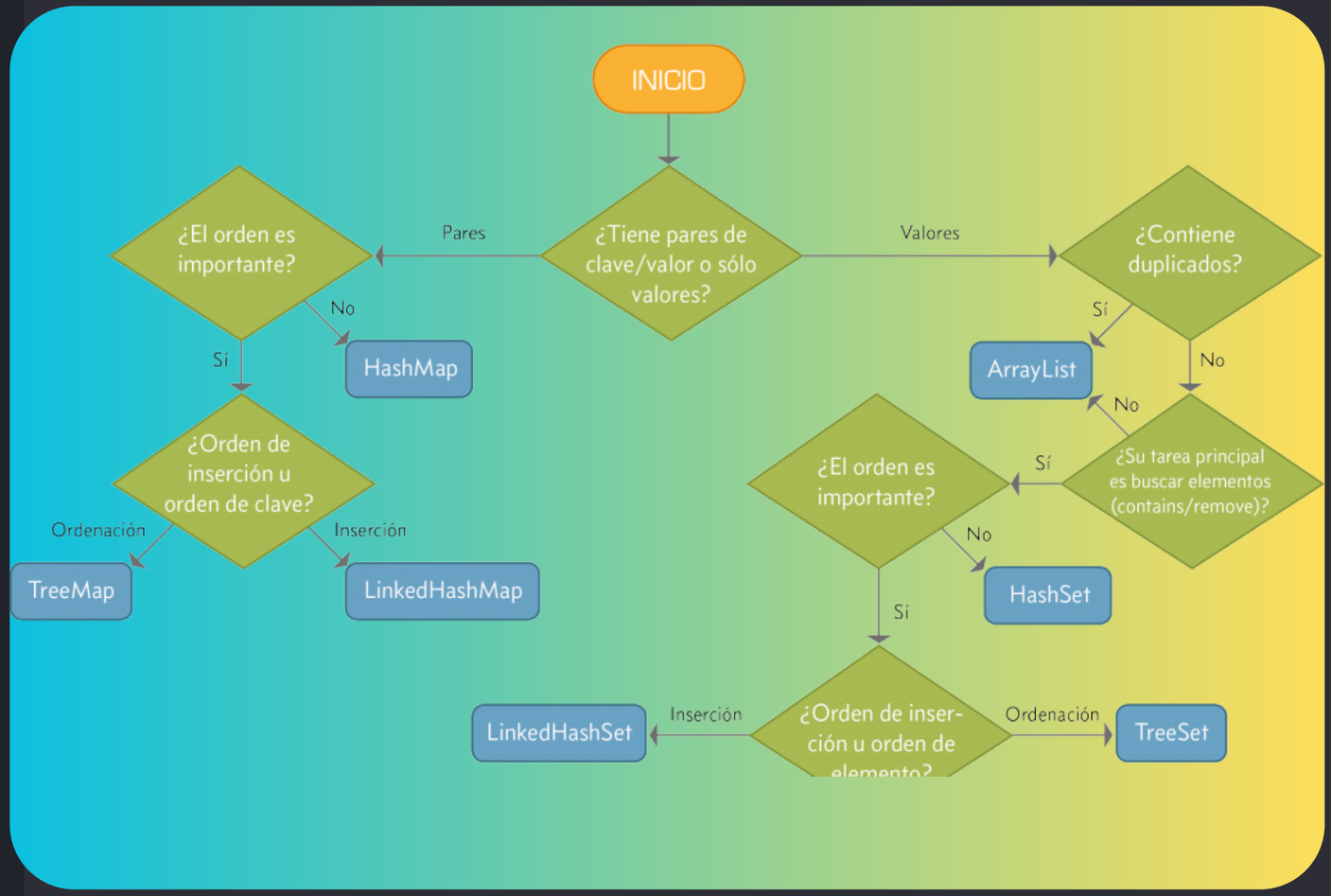
HashMap y LinkedHashMap permiten claves y valores nulos

}

Conclusión {

Java proporciona una serie de estructura muy variadas para almacenar datos. Estas estructuras, ofrecen diversas funcionalidades: ordenación de elementos, mejora de rendimiento, rango de operaciones… Es importante conocer cada una de ellas para saber cuál es la mejor situación para utilizarlas. Un buen uso de estas estructuras mejorará el rendimiento de nuestra aplicación.

Diagrama de decisión para el uso de collections JAVA



}

Bibliografía {

- Edureca,(2019).¿Qué es LinkedHashSet en Java?, recuperado de https://www.edureka.co/blog/linkedhashset-in-java/

- Java T point, recuperado de https://javatutorial.net/java-hashset-example/

- Jc Mouse (2017), ArrayDeque: Cola doblemente terminada, recuperado de, https://www.jc-mouse.net/java/arraydeque-cola-doblemente-terminada

}

Gracias {

<Por="Guadalupe López Velázquez"/>

}