**Java Collections Framework**

* **Set**

La interfaz define una colección que no puede contener elementos duplicados. Contiene únicamente los métodos heredados de *Collection* añadiendo la restricción de que los elementos duplicados están prohibidos.

Para comprobar si los elementos están o no duplicados, es necesario que estos tengan implementada de forma correcta. Los métodos *equals* y *hashCode*.

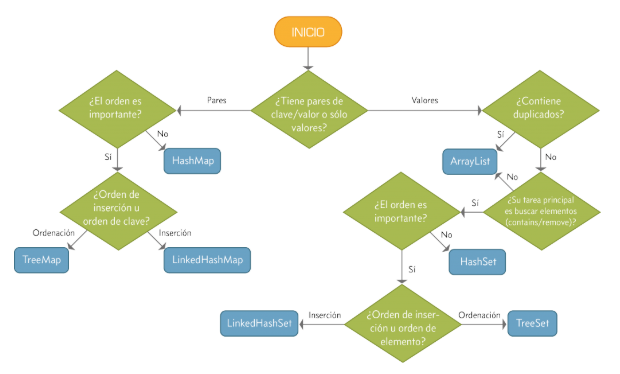
* **HashSet:** almacena los elementos en una tabla hash. Es la implementación con mejor rendimiento de todas pero no garantiza ningún orden a la hora de realizar iteraciones. Proporciona tiempos constantes en las operaciones básicas siempre y cuando la función hash disperse de forma correcta los elementos de la tabla hash. Es importante definir el tamaño inicial de la trbla ya que este tamaño marcará el rendimiento de la implementación.
* **TreeSet:** almacena los elementos ordenándolos en función de sus valores. Es más lento que HashSet. Los elementos almacenados deben implementar la interfaz *Comparable.* Garantiza, siempre, un rendimiento de log(N) en las operaciones básicas, debido a la estructura de árbol empleada para almacenar elementos.
* **LinkedHashSet:** almacena los elementos en función del orden de inserción. Es un poco más costosa que HashSet. (Vindel, 2015).
* **List**

La interfaz define una sucesión de elementos. A diferencia de Set, List sí admite elementos duplicados. A parte de los métodos heredados de *Collection*, añade métodos que permiten mejorar el acceso posicional a elementos, búsqueda de elementos, iteración sobre elementos y rango de operación.

* **ArrayList:** se basa en un array redimensionable que aumenta su tamaño según crece la colección de elementos. Tiene mejor rendimiento sobre la mayoría de situaciones.
* **LinkedList:** permite que mejore el rendimiento en ciertas ocasiones. Se basa en una lista doblemente enlazada de los elementos, teniendo cada uno de los elementos un puntero al anterior y al siguiente elemento. (Vindel, 2015).
* **Map**

Asocia claves a valores. No puede contener claves duplicadas y; cada una de dichas claves, solo puede tener asociado un valor como máximo.

* **HashMap:** almacena las claves en una tabla hash. Características similares a HashSet.
* **TreeMap:** almacena las claves ordenándolas en función de sus valores. Más lento que HashMap. Características similares a TreeSet.
* **LinkedHashMap:** almacena las claves en función del orden de inserción. (Vindel, 2015).



(Vindel, 2015).

**Clase de pila**

Se crean utilizando java.util.Stack

Push

Pop

LIFO

**LISP**

Es uno de los lenguajes de programación más simples que existen. Una vez conocidos los pocos conceptos que constituyen la base del lenguaje, el lector será capaz de interpretar prácticamente cualquier programa Lisp.

Celdas:

* Celdas literales. Los enteros, reales, cadenas y booleano. Sería igual con cualquier otro tipo de dato básico que queramos representar.
* Celdas de nombres. Un nombre es una cadena que identifica un valor. Cada una de estas celdas guarda la cadena con el nombre.
* Celdas de forma. Referencia a un objeto externo que es una forma aplicativa. Una forma puede ejecutarse si tenemos argumentos (lo que se denomina aplicar la forma a unos argumentos). Hay diversos tipos de formas que veremos más adelante. Estas celdas guardan una referencia a este objeto externo.

**Referencias**

Vindel, R. (2015). *Introducción a Colecciones en Java.* Obtenido de: https://www.adictosaltrabajo.com/2015/09/25/introduccion-a-colecciones-en-java/