|  |  |  |
| --- | --- | --- |
| Modificador y Tipo | Métodos | Explicación |
| static <T> boolean | addAll() | Sirve para insertar elementos en una colección cualquiera. |
| static <T> Queue<T> | asLifoQueue() | Retorna una Deque como Queue |
| static <T> int | binarySearch() | Usa la búsqueda binaria para encontrar un elemento en una colección. |
| static <E> Collection<E> | checkedCollection() | Retorna una vista segura dinámica de una collecion. |
| static <E> List<E> | checkedList() | Retorna una vista seguria dinámica de una lista. |
| static <K,V> Map<K,V> | checkedMap() | Retorna una vista segura dinámica de un Map. |
| static <K,V> NavigableMap<K,V> | checkedNavigableMap() | Retorna una vista dinámica segura de un Navigable Set |
| static <E> NavigableSet<E> | checkedNavigableSet() | Retorna una vista segura dinámica de un navigable set |
| static <E> Queue<E> | checkedQueue() | Retorna una vista segura de una queue |
| static <E>Set<E> | checkedSet() | Retorna una vista segura de un set especifico |
| static <K,V> SortedMap<K,V> | checkedSortedMap() | Retorna una vista segura de un sorted map |
| static <E> SortedSet<E> | checkedSortedSet() | Retorna una vista segura de un sorted set |

TreeMap

TreeMap es una implementacion de la interfaz Map. Map hereda aa interfaz SortedMap, que a su vez hereda la interfaz NavigableMap que finalmente hereda a TreeMap.

La estructura Map nos da la capcidad de guardar pares de datos, uno fungiendo como clave y otro como el valor en si que se guarda. El TreeMap sirve para guardar la pares de datos de forma ordenada en funcion de la clave, por ejemplo si esta fuera un int se podrian ordenar los datos de manera ascendente.

El TreeMap utiliza un arbol de ebano rojo como estructura de almacenamiento.

https://javarush.com/es/groups/posts/es.2584.caractersticas-de-treemap-en-java

https://jarroba.com/map-en-java-con-ejemplos/