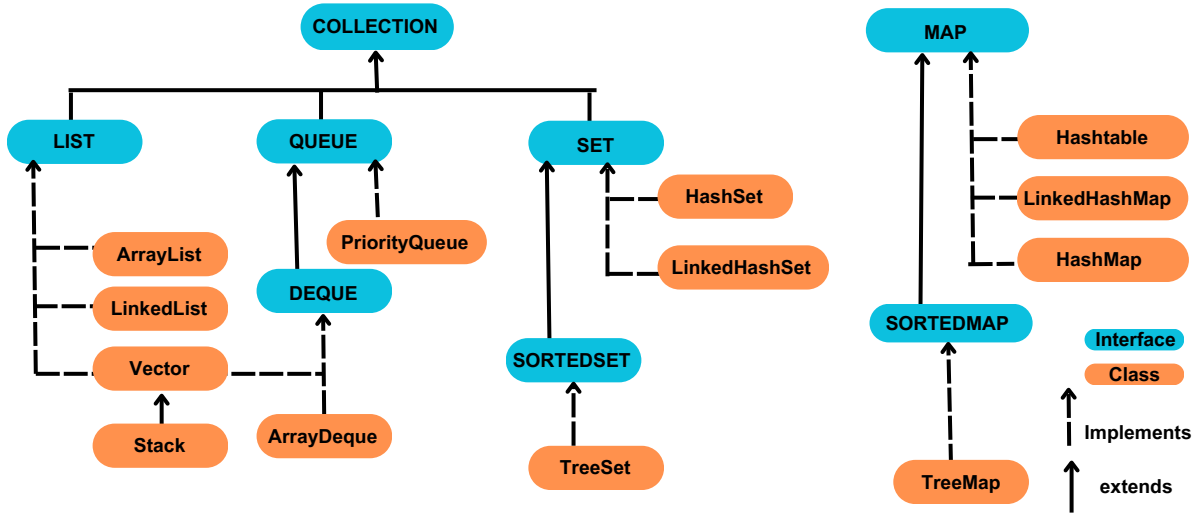
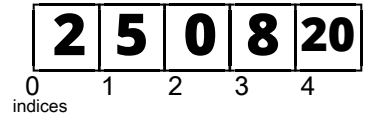
COLLECTION

ALMACENAR Y MANIPULAR GRUPOS DE OBJETOS
INTERFAZ COLLECTION ES LA RAÍZ DE LA JERARQUÍA DE COLECCIONES EN JAVA.



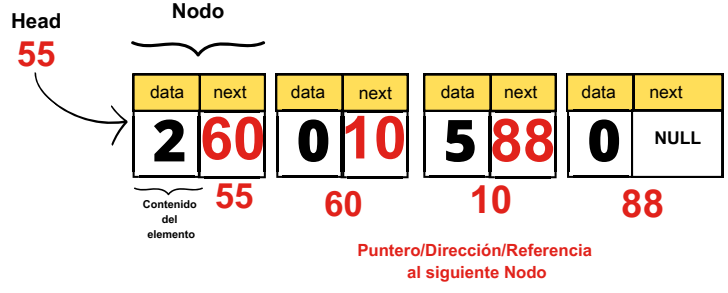
ArrayList

- Basada en un arreglo dinámico.
- Acceso y búsqueda a los elementos por índice.
- Inserciones y eliminaciones en el medio de la lista pueden ser costosas en términos de tiempo, ya que pueden requerir desplazamiento de elementos.
- Para escenarios donde se requiere un acceso rápido por índice y donde inserción y eliminación no son frecuentes en el medio de la lista.



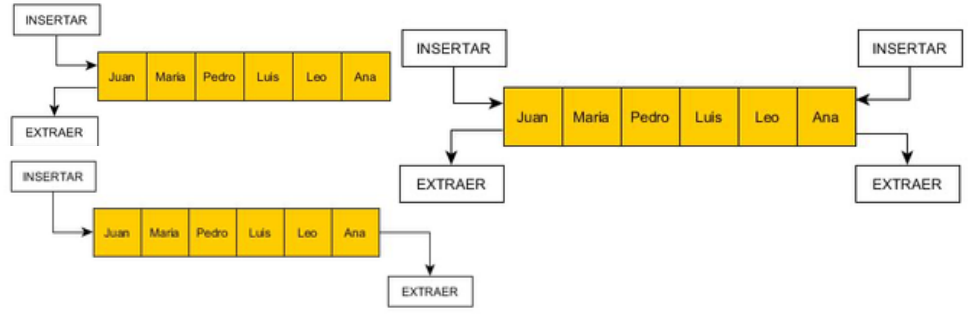
LinkedList

- Basada en una lista doblemente enlazada.
- Acceso y búsqueda lenta por índice, ya que debe recorrerse desde el inicio o el final de la lista para llegar al elemento deseado.
- Inserciones y eliminaciones en cualquier lugar de la lista más rápidas que ArrayList, ya que solo se requiere cambiar las referencias de los nodos enlazados.
- Para escenarios donde se requiere una alta frecuencia de inserciones o eliminaciones en cualquier posición de la lista, y el acceso por índice no es una prioridad.



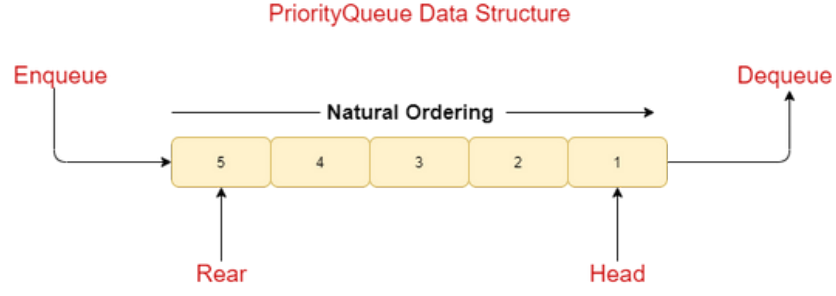
ArrayDeque

- "Double-ended Queue" (Cola de doble extremo). Proporciona una cola (FIFO - First-In-First-Out) y una pila (LIFO - Last-In-First-Out) en una sola estructura de datos.
- Más eficiente que LinkedList para operaciones de cola y pila.
- Permite inserción y eliminación rápida tanto al inicio como al final de la cola.
- No permite elementos nulos (null).



PriorityQueue

- Mantiene los elementos en orden natural o según un comparador proporcionado durante la creación de la cola. Los elementos se ordenan en función de su prioridad, y los elementos de mayor prioridad se acceden antes que los de menor prioridad.
- Permite acceso rápido al elemento de mayor prioridad (la cabeza de la cola).
- Se utiliza para implementar colas con prioridad en algoritmos como el algoritmo de Dijkstra y el algoritmo A*.

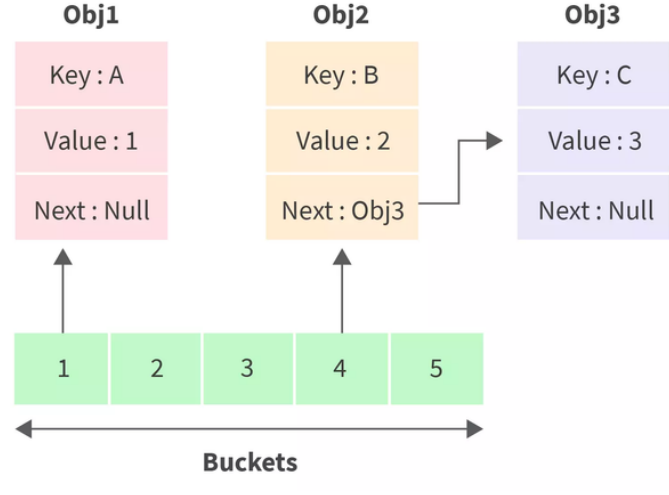


HashSet

- Utiliza una tabla hash para almacenar los elementos. No garantiza ningún orden específico para los elementos y puede cambiar el orden de los elementos con el tiempo. Se debe a que los elementos se almacenan en la tabla hash en función de sus valores hash y ubicación dentro de la tabla.
- Permite un acceso y búsqueda rápidos.
- No permite elementos duplicados.
- No garantiza un orden específico de los elementos.

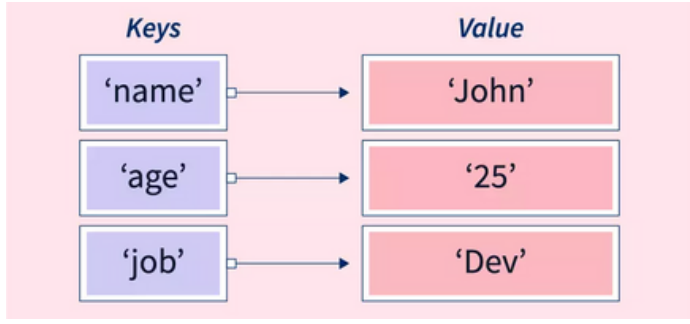
LinkedHashSet

- Mantiene el orden de inserción de los elementos. Combina la rapidez del acceso de HashSet con el orden de inserción de una lista enlazada.
- Permite un acceso y búsqueda rápidos.
- No permite elementos duplicados.
- Mantiene el orden de inserción de los elementos.



HashMap

- **Utiliza una tabla hash para almacenar los pares clave-valor. No garantiza un orden específico de las claves y puede cambiar el orden de las claves con el tiempo, ya que los elementos se almacenan en función de sus valores hash y ubicación dentro de la tabla.**
- **Permite un acceso rápido para recuperar elementos por clave.**
- **No garantiza un orden específico de las claves.**
- **Apropiado para situaciones donde el rendimiento de las operaciones de búsqueda es esencial y el orden de las claves no es relevante.**



TreeMap

- **mantiene las claves ordenadas en orden natural o en base a un comparador proporcionado. Las claves se organizan en un árbol rojo-negro, lo que garantiza que las claves se mantengan en orden.**
- **Tiene un rendimiento ligeramente más lento en comparación con HashMap debido al proceso de ordenamiento.**
- **Apropiado para situaciones donde se requiere un orden específico de las claves o una implementación de un diccionario con orden de clave.**