

# Colecciones





# Listas





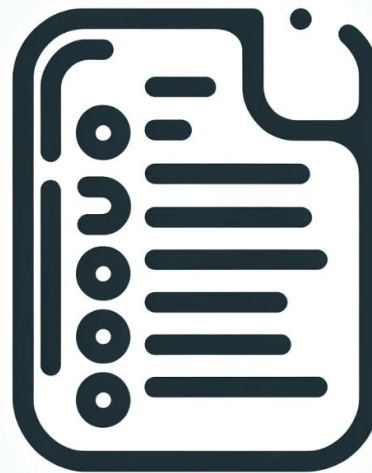
# Listas

**Definición:** Colección de datos **ordenada**.

Propiedades interesantes:

- Mantiene el **orden** de inserción.
- **Acceso directo** a elementos por el índice.
- Permite la inserción de valores **duplicados y nulos**.
- **Iterable**.

**ArrayList, Vector, Stack, LinkedList**



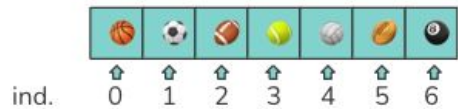


# Clase: ArrayList

**Definición:** Array de tamaño variable

Propiedades interesantes:

- Array de tamaño “dinámico” 🎉
- Rápidos en la obtención y modificación de datos
- Lento en la inserción o eliminación de datos





# ArrayList: Métodos

- **Añadir elementos**
  - void add(int posición, E elemento)
  - boolean add(E elemento)
  - boolean addAll(int posición, Collection<? extends E> c2)
- **Recuperar elemento en una posición**
  - E get(int posición)
- **Devuelve el (primer o último) índice de un elemento**
  - int indexOf(Object o)
  - int lastIndexOf(Object o)
- **Reemplaza elementos en una posición**
  - E set(int posición, E elemento)
- **Ordena los elementos de un array**
  - sort(Comparator<? super E> c)
- **Elimina los elementos situados en una posición**
  - E remove(int posición)

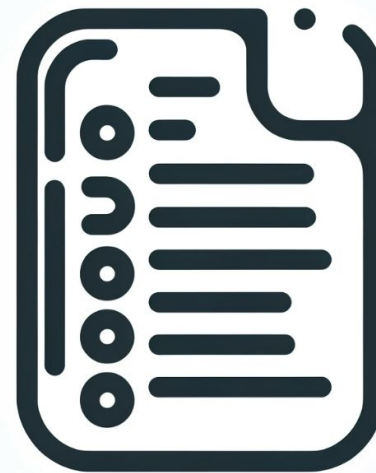
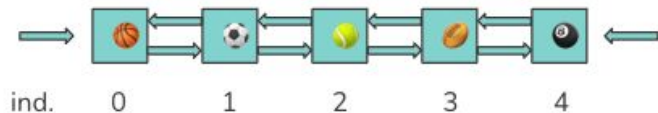


# LinkedList

**Definición:** Implementación de lista con nodos de doble enlace.

Propiedades interesantes:

- ~~Array~~ de tamaño dinámico 🎉🎉
- Rápidos en la inserción o eliminación de datos cabeza/cola
- Lentos en el acceso directo / modificación intermedios
- Puede actuar como una Cola/Pila





# LinkedList: Métodos

- **Inserta elementos**
  - `addFirst(E e)`
  - `addLast(E e)`
- **Devuelve elementos**
  - `E getFirst()`
  - `E getLast()`
- **Elimina elementos**
  - `E removeFirst()`
  - `E removeLast()`
- **Recorre descendientemente**
  - `Iterator<E> descendingIterator()`
- **Elimina la primera/última ocurrencia**
  - `boolean removeFirstOccurrence(Object o)`
  - `boolean removeLastOccurrence(Object o)`

# Sets





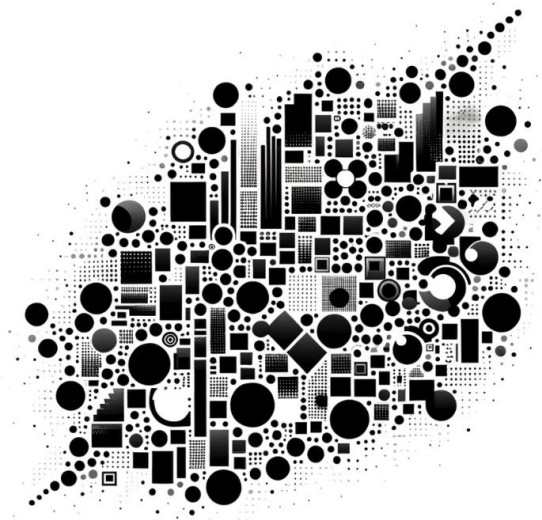
# Sets

**Definición:** Colección de datos **únicos**.

Propiedades interesantes:

- Mantiene la **unicidad**.
- **No ordenado** \*.
- Operaciones de conjuntos eficientes.
- **Iterable**.

HashSet, LinkedHashSet, TreeSet



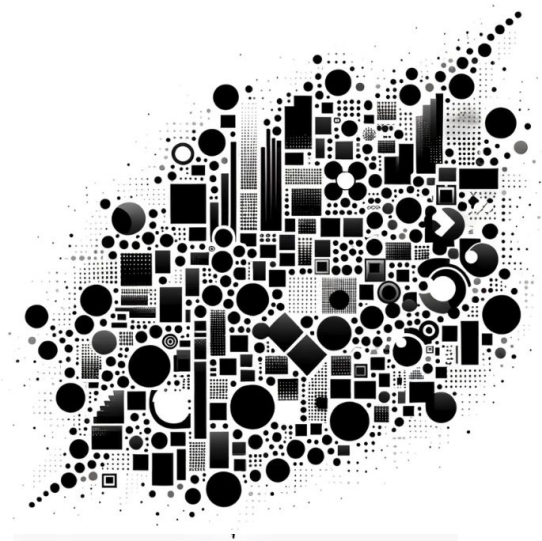
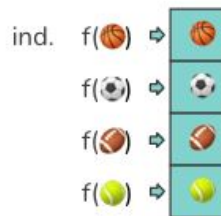


# Clase: HashSet

**Definición:** Utiliza una tabla Hash para almacenar elementos

Propiedades interesantes:

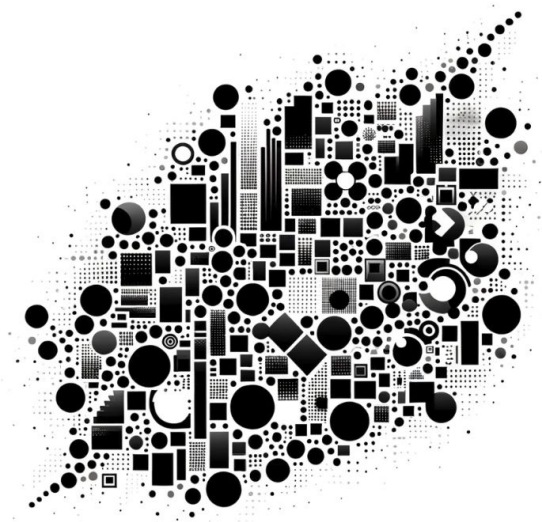
- Rápidos añadiendo, borrando y consultando.





## HashSet: Métodos

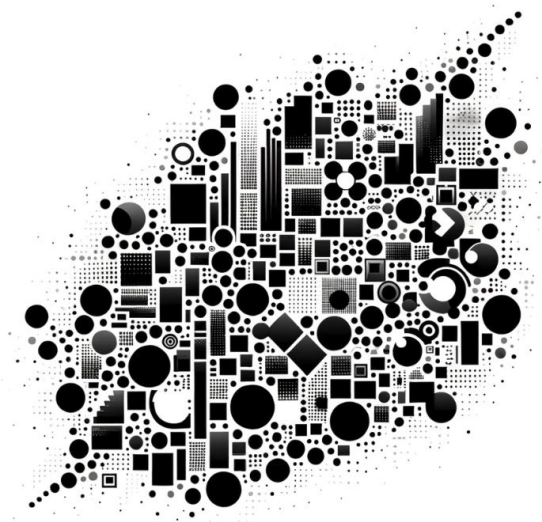
- **Inserta elementos**
  - `add(E e)`
- **Consulta existencia**
  - `boolean contains(E e)`
  - `boolean isEmpty()`
- **Elimina elementos**
  - `boolean remove(E e)`





# HashSet: Usos

- Eliminar duplicados
- Comprobar existencia en conjunto grande de datos
- Operaciones sobre conjuntos: unión, intersección...



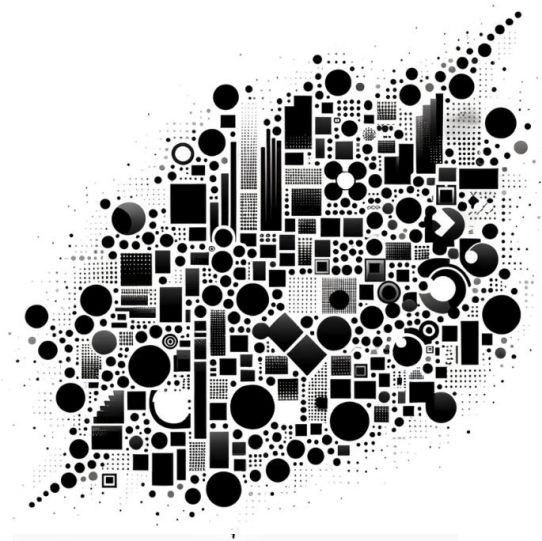
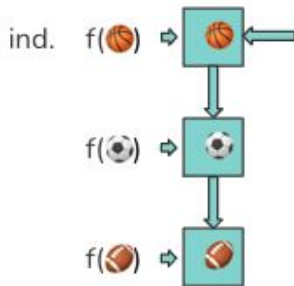


# Clase: LinkedHashSet

**Definición:** Utiliza una tabla Hash enlazada

Propiedades interesantes:


- Pérdida de rendimiento añadiendo, borrando y consultando.
- Garantiza el **orden** 🎉 🎉





## LinkedHashSet: Métodos

- **Inserta elementos**
  - add(E e)
- **Consulta existencia**
  - boolean contains(E e)
  - boolean isEmpty()
- **Elimina elementos**
  - boolean remove(E e)

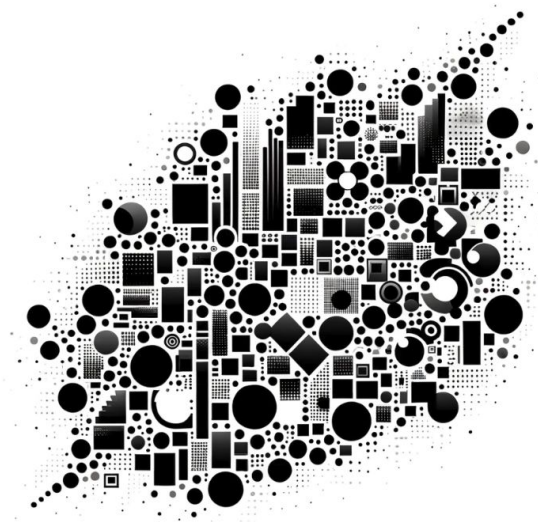


```
public class LinkedHashSet<E>  
extends HashSet<E>
```



# LinkedHashSet: Usos

- Eliminar duplicados y orden de inserción
- Cache LRU: Eliminar datos más antiguos
- Mantener histórico de eventos



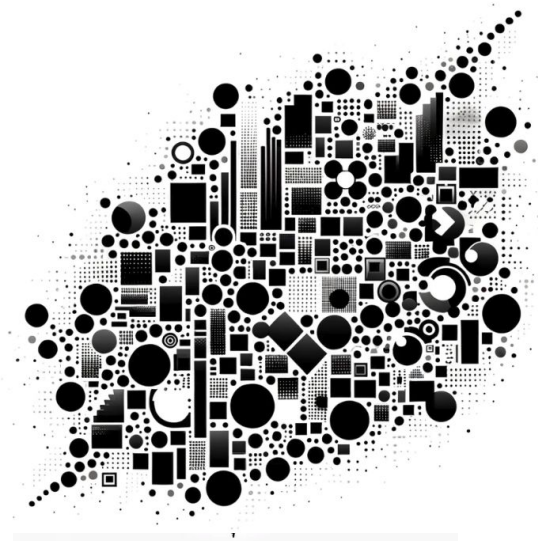
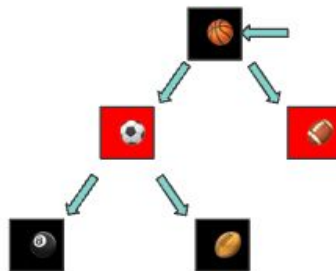


# Clase: TreeSet

**Definición:** Utiliza un árbol rojo-negro para almacenar elementos

Propiedades interesantes:

- El uso del árbol binario mejora el rendimiento búsqueda.
- Garantiza el **orden** 🎉 🎉
- No puede contener **null**

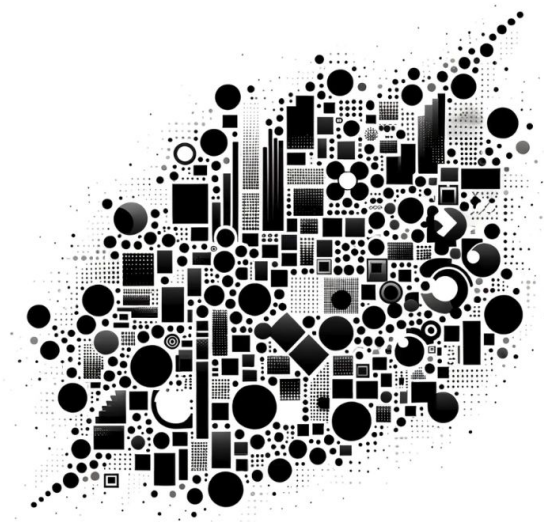






# TreeSet: Métodos

- **Inserta elementos**
  - `add(E e)`
- **Elimina elementos**
  - `remove(E e)`
- **Obtener elementos**
  - `first(E e)`
  - `last(E e)`
  - `subSet(E from, E to)`
- **Consulta existencia**
  - `boolean contains(E e)`
  - `boolean isEmpty()`
- **Elimina elementos**
  - `boolean remove(E e)`





# TreeSet: Usos

- Eliminar duplicados y orden customizable
- Búsqueda eficiente de elementos y rangos

