**TEMA 12:MÉTODOS DE ARRAYLIST,SET Y MAP**

Un Set en Java es una interfaz que define un conjunto de elementos únicos, sin un orden específico. Algunos de los métodos más comunes definidos en la interfaz Set son:

1. add(elemento): agrega el elemento especificado al conjunto si no está presente.
2. addAll(coleccion): agrega todos los elementos de la colección especificada al conjunto si no están presentes.
3. remove(elemento): elimina el elemento especificado del conjunto si está presente.
4. removeAll(coleccion): elimina todos los elementos de la colección especificada del conjunto.
5. clear(): elimina todos los elementos del conjunto, dejándolo vacío.
6. contains(elemento): devuelve true si el conjunto contiene el elemento especificado, o false en caso contrario.
7. containsAll(coleccion): devuelve true si el conjunto contiene todos los elementos de la colección especificada, o false en caso contrario.
8. isEmpty(): devuelve true si el conjunto no contiene elementos, o false en caso contrario.
9. size(): devuelve el número de elementos en el conjunto.
10. iterator(): devuelve un iterador sobre los elementos del conjunto.

Set<String> conjunto = new HashSet<>(); conjunto.add("elemento1"); conjunto.add("elemento2"); conjunto.add("elemento3");

Iterator<String> iterator = conjunto.iterator();

while (iterator.hasNext()) {

String elemento = iterator.next(); System.out.println(elemento);

}

Estos son solo algunos de los métodos más utilizados en la interfaz Set. Ten en cuenta que existen implementaciones específicas de Set, como HashSet, TreeSet y LinkedHashSet, que pueden proporcionar funcionalidades adicionales específicas de cada implementación.

**Buscar en un set**

**for(Socio s:club){**

**if(s.getNombre().equals(nombre)**

los métodos más comunes de la clase ArrayList en Java:

1. add(elemento): Agrega un elemento al final de la lista.
2. add(index, elemento): Inserta un elemento en la posición especificada.
3. get(index): Devuelve el elemento en la posición especificada.
4. set(index, elemento): Reemplaza el elemento en la posición especificada.
5. remove(index): Elimina el elemento en la posición especificada.
6. size(): Devuelve el número de elementos en la lista.
7. isEmpty(): Verifica si la lista está vacía.
8. contains(elemento): Verifica si la lista contiene el elemento especificado.
9. indexOf(elemento): Devuelve la posición del primer elemento igual al especificado.
10. lastIndexOf(elemento): Devuelve la posición del último elemento igual al especificado.
11. clear(): Elimina todos los elementos de la lista.
12. toArray(): Devuelve un arreglo que contiene todos los elementos de la lista.
13. addAll(coleccion): Agrega todos los elementos de una colección al final de la lista.
14. remove(elemento): Elimina la primera aparición del elemento especificado en la lista.
15. removeAll(coleccion): Elimina todos los elementos de la lista que están contenidos en la colección especificada.
16. retainAll(coleccion): Elimina todos los elementos de la lista que no están contenidos en la colección especificada.
17. subList(inicio, fin): Devuelve una vista parcial de la lista entre las posiciones de inicio (inclusive) y fin (exclusive).

**MÉTODOS DEL MAP**

En Java, la interfaz Map define una colección de pares clave-valor, donde cada clave es única y se utiliza para acceder a su correspondiente valor. Algunos de los métodos más comunes proporcionados por la interfaz Map son:

1. put(K key, V value): Inserta un par clave-valor en el mapa. Si la clave ya existe, se actualiza el valor asociado a esa clave.
2. get(Object key): Devuelve el valor asociado a la clave especificada, o null si la clave no está presente en el mapa.
3. remove(Object key): Elimina la entrada correspondiente a la clave especificada y devuelve el valor asociado a esa clave, o null si la clave no está presente.
4. containsKey(Object key): Verifica si el mapa contiene una clave específica. Devuelve true si la clave está presente, o false de lo contrario.
5. containsValue(Object value): Verifica si el mapa contiene un valor específico. Devuelve true si el valor está presente, o false de lo contrario.
6. size(): Devuelve el número de pares clave-valor en el mapa.
7. isEmpty(): Verifica si el mapa está vacío. Devuelve true si no hay elementos en el mapa, o false de lo contrario.
8. keySet(): Devuelve un conjunto con todas las claves presentes en el mapa.
9. values(): Devuelve una colección con todos los valores presentes en el mapa.
10. entrySet(): Devuelve un conjunto de pares clave-valor (Map.Entry) presentes en el mapa.-->sirve para recorrer el map y obtener la clave y valor de cada elemento

for (Map.Entry<String, Integer> entry : entries) { String clave = entry.getKey(); Integer valor = entry.getValue(); System.out.println("Clave: " + clave + ", Valor: " + valor); }

1. clear(): Elimina todos los pares clave-valor del mapa.

Estos son solo algunos de los métodos principales proporcionados por la interfaz Map. Ten en cuenta que hay otras implementaciones de Map en Java, como HashMap, TreeMap, LinkedHashMap, que pueden tener métodos adicionales o comportamientos específicos.