Metodos de Listas

**HashMap**

* put(K key, V value): Adds a key-value pair.
* get(Object key): Retrieves the value for a given key.
* remove(Object key): Removes a key-value pair.
* containsKey(Object key): Checks if a key exists.
* clear(): Removes all mappings.
* size(): Returns the number of entries.

**HashMap**

* put(K key, V value): Adds a key-value pair.
* get(Object key): Retrieves the value for a given key.
* remove(Object key): Removes a key-value pair.
* containsKey(Object key): Checks if a key exists.
* clear(): Removes all mappings.
* size(): Returns the number of entries.

**HashSet**

* add(E e): Agrega un elemento si no está presente
* clear(): Borra todos los elementos
* clone(): Crea una copia del HashSet
* contains(Object o): Verifica si un elemento está en el HashSet
* isEmpty(): Verifica si el HashSet está vacío
* iterator(): Devuelve un iterador para recorrer el HashSet
* remove(Object o): Elimina un elemento si está presente
* size(): Devuelve la cantidad de elementos
* toArray(): Convierte el HashSet en un array

**List**

public void showAllPlayersByPosition(){  
 List<Map.Entry<Integer, Player>> newListPlayers = new ArrayList<>(playersHashMap.entrySet()); newListPlayers.sort(Comparator.*comparing*(entry-> entry.getValue().getPosition()));  
  
 for (Map.Entry<Integer, Player> entry : newListPlayers) {  
 System.*out*.println(entry.getValue());  
 }  
  
}

public void showAllPlayers(){  
 List<Map.Entry<Integer, Player>> newListPlayers = new ArrayList<>(playersHashMap.entrySet());  
 newListPlayers.sort(Map.Entry.*comparingByKey*());  
 for (Map.Entry<Integer, Player> entry : newListPlayers) {  
 System.*out*.println(entry.getKey() + ":" + entry.getValue());  
 }  
  
}