| **Funcionalidad** | **Clase** | **Descripción** | **Ejemplo** |
| --- | --- | --- | --- |
| **length()** | String | Devuelve la longitud de la cadena. | int len = str.length(); |
| **charAt(int index)** | String | Devuelve el carácter en la posición especificada. | char ch = str.charAt(0); |
| **substring(int beginIndex)** | String | Devuelve una subcadena desde el índice especificado. | String sub = str.substring(5); |
| **substring(int beginIndex, int endIndex)** | String | Devuelve una subcadena entre los índices especificados. | String sub = str.substring(0, 5); |
| **indexOf(char c)** | String | Devuelve el índice de la primera aparición del carácter. | int index = str.indexOf('b'); |
| **lastIndexOf(char c)** | String | Devuelve el índice de la última aparición del carácter. | int lastIndex = str.lastIndexOf('b'); |
| **toUpperCase()** | String | Convierte la cadena a mayúsculas. | String upper = str.toUpperCase(); |
| **toLowerCase()** | String | Convierte la cadena a minúsculas. | String lower = str.toLowerCase(); |
| **trim()** | String | Elimina espacios en blanco al principio y al final. | String trimmed = str.trim(); |
| **split(String regex)** | String | Divide la cadena en un array usando un delimitador. | String[] parts = str.split(" "); |
| **replace(char oldChar, char newChar)** | String | Reemplaza todas las apariciones de un carácter por otro. | String replaced = str.replace('a', 'e'); |
| **length** | Array | Devuelve la longitud del array. | int len = array.length; |
| **Arrays.toString(array)** | Array | Convierte el array a una cadena para imprimirlo. | System.out.println(Arrays.toString(array)); |
| **Arrays.sort(array)** | Array | Ordena el array. | Arrays.sort(array); |
| **Arrays.binarySearch(array, key)** | Array | Busca un elemento en un array ordenado. | int index = Arrays.binarySearch(array, key); |
| **Arrays.copyOf(array, newLength)** | Array | Copia el array en uno nuevo de longitud especificada. | int[] newArray = Arrays.copyOf(array, newLength); |
| **size()** | List | Devuelve el número de elementos en la lista. | int size = list.size(); |
| **get(int index)** | List | Devuelve el elemento en la posición especificada. | E element = list.get(0); |
| **add(E element)** | List | Añade un elemento al final de la lista. | list.add(element); |
| **add(int index, E element)** | List | Inserta un elemento en la posición especificada. | list.add(1, element); |
| **remove(int index)** | List | Elimina el elemento en la posición especificada. | list.remove(0); |
| **contains(Object o)** | List | Verifica si la lista contiene un elemento. | boolean hasElement = list.contains(element); |
| **indexOf(Object o)** | List | Devuelve el índice de la primera aparición de un elemento. | int index = list.indexOf(element); |
| **toArray()** | List | Convierte la lista a un array. | Object[] array = list.toArray(); |
| **put(K key, V value)** | HashMap | Añade un par clave-valor al mapa. | map.put(key, value); |
| **get(Object key)** | HashMap | Devuelve el valor asociado con la clave. | V value = map.get(key); |
| **remove(Object key)** | HashMap | Elimina el par clave-valor asociado con la clave. | map.remove(key); |
| **containsKey(Object key)** | HashMap | Verifica si el mapa contiene la clave especificada. | boolean hasKey = map.containsKey(key); |
| **containsValue(Object value)** | HashMap | Verifica si el mapa contiene el valor especificado. | boolean hasValue = map.containsValue(value); |
| **size()** | HashMap | Devuelve el número de pares clave-valor en el mapa. | int size = map.size(); |
| **keySet()** | HashMap | Devuelve un Set de todas las claves en el mapa. | Set<K> keys = map.keySet(); |
| **values()** | HashMap | Devuelve una colección de todos los valores en el mapa. | Collection<V> values = map.values(); |
| **Iterator** | List, Map | Para recorrer listas y mapas. | Iterator<E> iterator = list.iterator(); while (iterator.hasNext()) { System.out.println(iterator.next()); } |