

9.D. Listas.

1. Listas (I).

¿En qué se diferencia una lista de un conjunto? Las listas son elementos de programación un poco más avanzados que los conjuntos. Su ventaja es que amplían el conjunto de operaciones de las colecciones añadiendo operaciones extra, veamos algunas de ellas:

- Las listas sí pueden almacenar duplicados, si no queremos duplicados, hay que verificar manualmente que el elemento no esté en la lista antes de su inserción.
- Acceso posicional. Podemos acceder a un elemento indicando su posición en la lista.
- Búsqueda. Es posible buscar elementos en la lista y obtener su posición. En los conjuntos, al ser colecciones sin aportar nada nuevo, solo se podía comprobar si un conjunto contenía o no un elemento de la lista, retornando verdadero o falso. Las listas mejoran este aspecto.
- Extracción de sublistas. Es posible obtener una lista que contenga solo una parte de los elementos de forma muy sencilla.

En Java, para las listas se dispone de una interfaz llamada `java.util.List`, y dos implementaciones (`java.util.LinkedList` y `java.util.ArrayList`), con diferencias significativas entre ellas. Los métodos de la interfaz `List`, que obviamente estarán en todas las implementaciones, y que permiten las operaciones anteriores son:

- `E get(int index)`. El método `get` permite obtener un elemento partiendo de su posición (`index`).
- `E set(int index, E element)`. El método `set` permite cambiar el elemento almacenado en una posición de la lista (`index`), por otro (`element`).
- `void add(int index, E element)`. Se añade otra versión del método `add`, en la cual se puede insertar un elemento (`element`) en la lista en una posición concreta (`index`), desplazando los existentes.
- `E remove(int index)`. Se añade otra versión del método `remove`, esta versión permite eliminar un elemento indicando su posición en la lista.
- `boolean addAll(int index, Collection<? extends E> c)`. Se añade otra versión del método `addAll`, que permite insertar una colección pasada por parámetro en una posición de la lista, desplazando el resto de elementos.
- `int indexOf(Object o)`. El método `indexOf` permite conocer la posición (índice) de un elemento, si dicho elemento no está en la lista retornará -1.
- `int lastIndexOf(Object o)`. El método `lastIndexOf` nos permite obtener la última ocurrencia del objeto en la lista (dado que la lista sí puede almacenar duplicados).
- `List<E> subList(int from, int to)`. El método `subList` genera una sublista (una vista parcial de la lista) con los elementos comprendidos entre la posición inicial (incluida) y la posición final (no incluida).

Ten en cuenta que los elementos de una lista empiezan a numerarse por 0. Es decir, que el primer elemento de la lista es el 0. Ten en cuenta también que `List` es una interfaz genérica, por lo que `<E>` corresponde con el tipo base usado como parámetro genérico al crear la lista.

Autoevaluación

Si `M` es una lista de números enteros, se puede poner `"M.add(M.size(),3);"`.

- ☐ Verdadero.
- ☐ Falso.