9.D. Listas.

2. Listas (II).

Y, ¿cómo se usan las listas? Pues para usar una lista haremos uso de sus implementaciones `LinkedList` y `ArrayList`. Veamos un ejemplo de su uso y después obtendrás respuesta a esta pregunta.

Supongo que intuirás como se usan, pero nunca viene mal un ejemplo sencillo, que nos aclare las ideas. El siguiente ejemplo muestra cómo usar un `LinkedList` pero valdría también para `ArrayList` (no olvides importar las clases `java.util.LinkedList` y `java.util.ArrayList` según sea necesario). En este ejemplo se usan los métodos de acceso posicional a la lista:

```
LinkedList<Integer> t=new LinkedList<Integer>(); // Declaración y creación del LinkedList de enteros.

t.add(1); // Añade un elemento al final de la lista.

t.add(3); // Añade otro elemento al final de la lista.

t.add(1,2); // Añade en la posición 1 el elemento 2.

t.add(t.get(1)+t.get(2)); // Suma los valores contenidos en la posición 1 y 2, y lo agrega al final.

t.remove(0); // Elimina el primer elementos de la lista.

for (Integer i: t) System.out.println("Elemento:" + i); // Muestra la lista.
```

En el ejemplo anterior, se realizan muchas operaciones, ¿cuál será el contenido de la lista al final? Pues será 2, 3 y 5. En el ejemplo cabe destacar el uso del bucle for-each, recuerda que se puede usar en cualquier colección.

Veamos otro ejemplo, esta vez con `ArrayList`, de cómo obtener la posición de un elemento en la lista:

```
ArrayList<Integer> al=new ArrayList<Integer>(); // Declaración y creación del ArrayList de enteros.

al.add(10); al.add(11); // Añadimos dos elementos a la lista.

al.set(al.indexOf(11), 12); // Sustituimos el 11 por el 12, primero lo buscamos y luego lo reemplazamos.
```

En el ejemplo anterior, se emplea tanto el método `indexOf` para obtener la posición de un elemento, como el método `set` para reemplazar el valor en una posición, una combinación muy habitual. El ejemplo anterior generará un `ArrayList` que contendrá dos números, el 10 y el 12. Veamos ahora un ejemplo algo más difícil:

```
al.addAll(0, t.subList(1, t.size()));
```

Este ejemplo es especial porque usa sublistas. Se usa el método `size` para obtener el tamaño de la lista. Después el método `subList` para extraer una sublista de la lista (que incluía en origen los números 2, 3 y 5), desde la posición 1 hasta el final de la lista (lo cual dejaría fuera al primer elemento). Y por último, se usa el método `addAll` para añadir todos los elementos de la sublista al `ArrayList` anterior.

Debes saber que las operaciones aplicadas a una sublista repercuten sobre la lista original. Por ejemplo, si ejecutamos el método `clear` sobre una sublista, se borrarán todos los elementos de la sublista, pero también se borrarán dichos elementos de la lista original:

```
al.subList(0, 2).clear();
```

Lo mismo ocurre al añadir un elemento, se añade en la sublista y en la lista original.

Debes conocer

Las listas enlazadas son un elemento muy recurrido y su funcionamiento interno es complejo. Te recomendamos el siguiente artículo de la wikipedia para profundizar un poco más en las listas enlazadas y los diferentes tipos que hay.

Listas enlazadas.

Autoevaluación

Completa con el número que falta.

Dado el siguiente código:

```
LinkedList<Integer> t=new LinkedList<Integer>();
```

```
        t.add(t.size()+1); t.add(t.size()+1);  Integer suma = t.get(0) + t.get(1);
```

El valor de la variable suma después de ejecutarlo es 3 Resolver