



Colecciones de tipo Lista



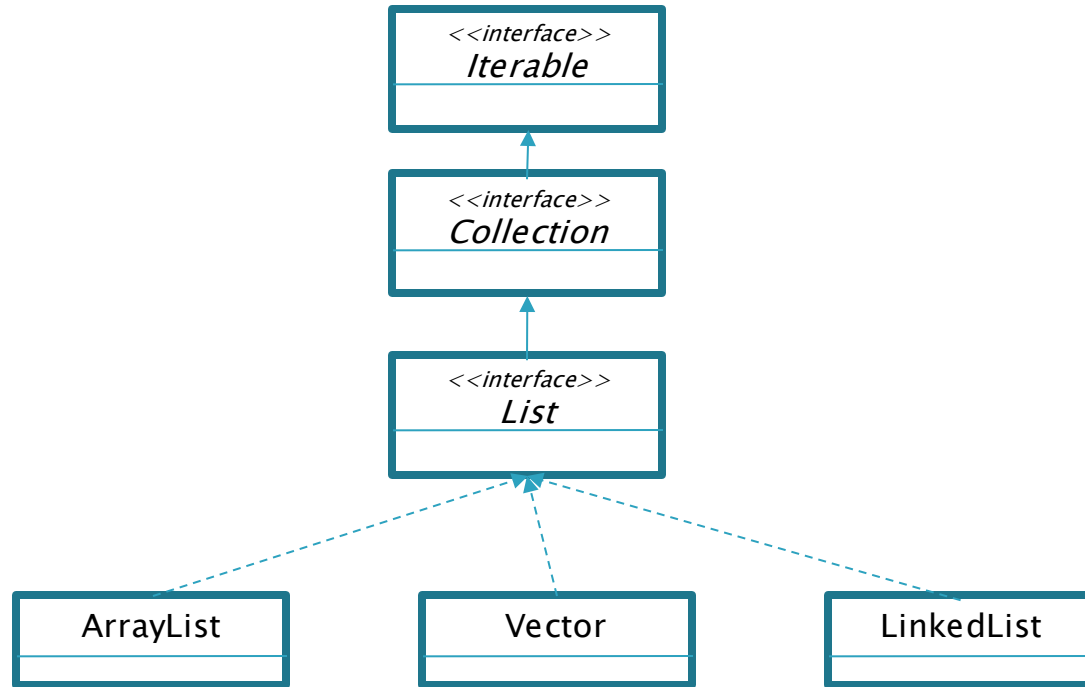
Introducción

- Una colección es una agrupación de objetos sin tamaño fijo
 - Se puede añadir y eliminar objetos de una colección dinámicamente
 - Para gestionar colecciones disponemos de clases e interfaces específicas en `java.util`
 - Tipos:
 - Listas
 - Tablas
 - Conjuntos
- 

Listas

- Cada elemento tiene una posición asociada a partir del orden de llegada, siendo 0 la posición del primero
 - Las listas implementan la interfaz List, que a su vez implementa Collection.
 - Son colecciones de tipo genérico (preparadas para admitir cualquier objeto Java)
 - La principal clase de colección es ArrayList.
- 

Classes e interfaces de Listas



Creación de listas

➤ Como instancias de ArrayList:

```
List<Integer> enteros=new ArrayList<>();
```

➤ A partir del método asList de Arrays:

```
List<Integer> enteros=Arrays.asList(6,2,4,10,21);
```

Lista de tamaño FIJO, no admite inserción ni eliminación

➤ Mediante método de factoría de List:

```
List<Integer> enteros=List.of(40,29,11,28);
```

➤ Mediante el método copyOf de List:

```
List<Integer> copia=List.copyOf(enteros);
```

INMUTABLES, no admiten la eliminación, modificación e inserción de elementos, ni valores null

Principales métodos para listas

➤ **boolean add(T dato).** Añade el dato a la colección y lo coloca al final de la misma. T representa el tipo indicado al crear el objeto de colección:

```
ArrayList<String> nombres=new ArrayList<>();  
nombres.add("Maria"); //elemento en posición 0  
nombres.add("Angel"); //elemento en posición 1
```

➤ **boolean add(int pos,T dato).** Añade el elemento en la posición indicada, desplazando hacia adelante los que se encuentren en dicha posición

```
nombres.add("Luis", 1); //desplaza a Angel a la posición 2
```

Principales métodos para listas II

➤ `T set(int pos, T dato)`. Sustituye el elemento existente en la posición indicada por el nuevo dato suministrado como parámetro. Devuelve el elemento sustituido

`nombres.set(0, "Laura"); // sustituye María por Laura`

➤ `int size()`. Devuelve el total de elementos de la colección

➤ `T get(int pos)`. Devuelve el elemento que ocupa la posición indicada. Si la posición es menor que 0 o mayor o igual que *size()*, se producirá una excepción

Revisión conceptos

Indica que se mostrará al ejecutar el siguiente código

```
List<Integer> nums=List.of(11,22,31,10);  
nums.set(0,nums.get(1));  
System.out.println(nums.get(2)-nums.get(0));
```

Respuesta

Se producirá una excepción en la segunda instrucción, ya que se trata de una lista inmutable y no se puede modificar su contenido

Principales métodos para listas III

- `T remove (int pos)`. Elimina el elemento que ocupa la posición indicada y lo devuelve. Si la posición es menor que 0 o mayor o igual que `size()`, se producirá una excepción
- `boolean remove(Object ob)`. Elimina el elemento indicado en caso de que exista. Devuelve `true` si dicho elemento estaba presente en la colección. Si hay más de un elemento, elimina la primera ocurrencia

Recorrido de una Lista

➤ Se puede recorrer con un bucle for estándar:

```
//muestra el contenido de la lista de nombres
for(int i=0;i<nombres.size();i++){
    System.out.println(nombres.get(i));
}
```

➤ También mediante un for each:

```
//misma función que el bloque anterior
for(String s:nombres){
    System.out.println(s);
}
```

➤ Método forEach (se estudia en lambdas)

Revisión conceptos



Indica que se mostrará al ejecutar el siguiente código

```
List<Integer> nums=new ArrayList(List.of(6,11,22,31,10,5,7)); //1
for(int k=0;k<nums.size();k++){
    if(nums.get(k)%2==0){
        nums.remove(k); //2
    }
}
System.out.println(nums);
```

- a. Error de compilación en línea 1
- b. Excepción en línea 2
- c. [11, 31, 5, 7]

Respuesta

La respuesta correcta es la C. La primera instrucción compila bien, es posible crear un ArrayList a partir de un Collection. El ArrayList es mutable, por lo que se puede modificar su contenido y la línea 2 es correcta.