

ESTRUCTURAS DE DATOS

NOTAS SOBRE JAVA

Contenedores lineales

Manuel Montenegro Montes

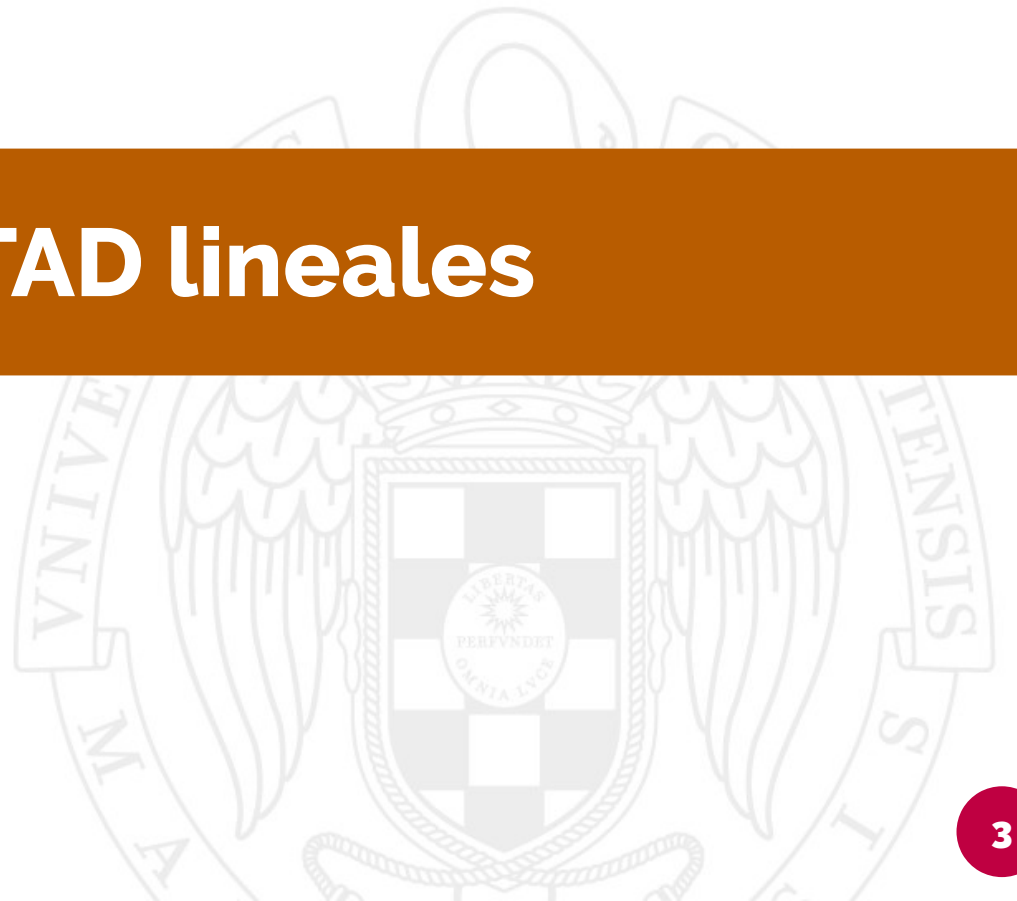
Departamento de Sistemas Informáticos y Computación
Facultad de Informática – Universidad Complutense de Madrid

Librería estándar de Java

- Proporciona un gran número de clases con implementaciones de los TADs más comunes, y algoritmos para su manipulación.
- Estas clases suelen estar en el paquete `java.util`.



Clases de TAD lineales



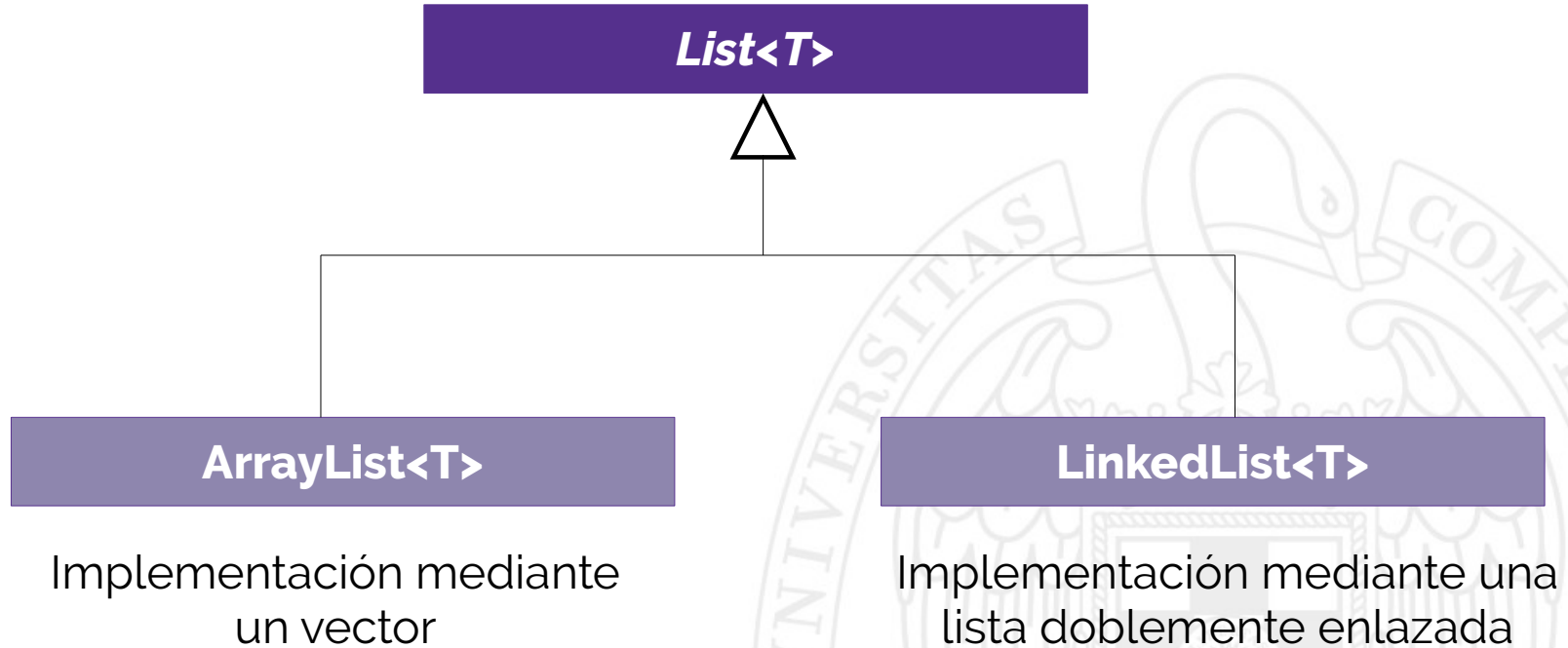
La interfaz *List*

- Define las operaciones del TAD Lista.

List<T>

```
+ add(T)
+ add(pos, T)
+ remove(pos)
+ contains(T): bool
+ get(pos): T
+ set(pos, T)
+ size(): int
+ subList(pos1, pos2): List<T>
+ toArray(T[]): T[]
...
```

Implementaciones de List<T>



Ejemplo

```
List<Integer> l = new ArrayList<>();  
for (int i = 0; i < 10; i++) {  
    l.add(i * 3);  
}  
System.out.println(l);
```

[0, 3, 6, 9, 12, 15, 18, 21, 24, 27]

Otros TADs lineales

Stack<T>

- + empty(): bool
- + peek(): T
- + pop(): T
- + push(T): bool
- ...

Queue<T>

- + add(T)
- + element(): T
- + peek(): T
- + poll(): T
- ...

Deque<T>

ArrayDeque<T>

Iteradores

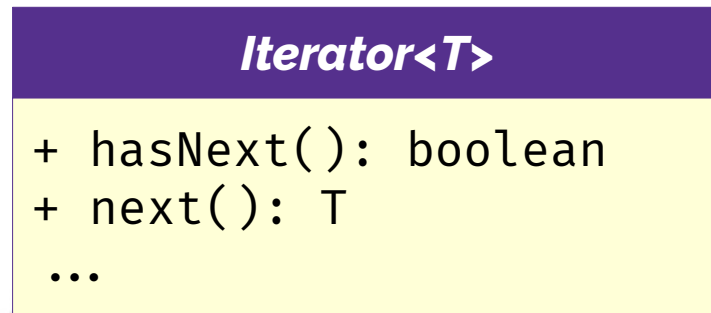
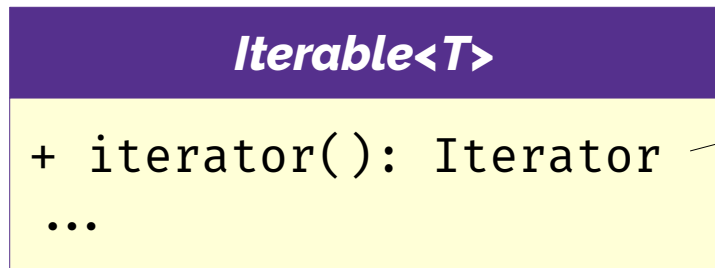


Iteradores

- Es posible obtener un iterador a partir de cualquier clase que implemente la interfaz `Iterable`.
 - `List`
 - `Stack`
 - `Queue`
 - `etc.`



Iteradores



Ejemplo

```
List<Integer> l = ... ;  
  
... // Insertar elementos en l
```

```
int suma = 0;  
Iterator<Integer> it = l.iterator();  
while (it.hasNext()) {  
    suma += it.next();  
}  
System.out.println(suma);
```



Sintaxis alternativa

```
List<Integer> l = ...;
```

```
... // Insertar elementos en l
```

```
int suma = 0;
```

```
Iterator<Integer> it = l.iterator();
```

```
while (it.hasNext()) {
```

```
    suma += it.next();
```

```
}
```

```
System.out.println(suma);
```

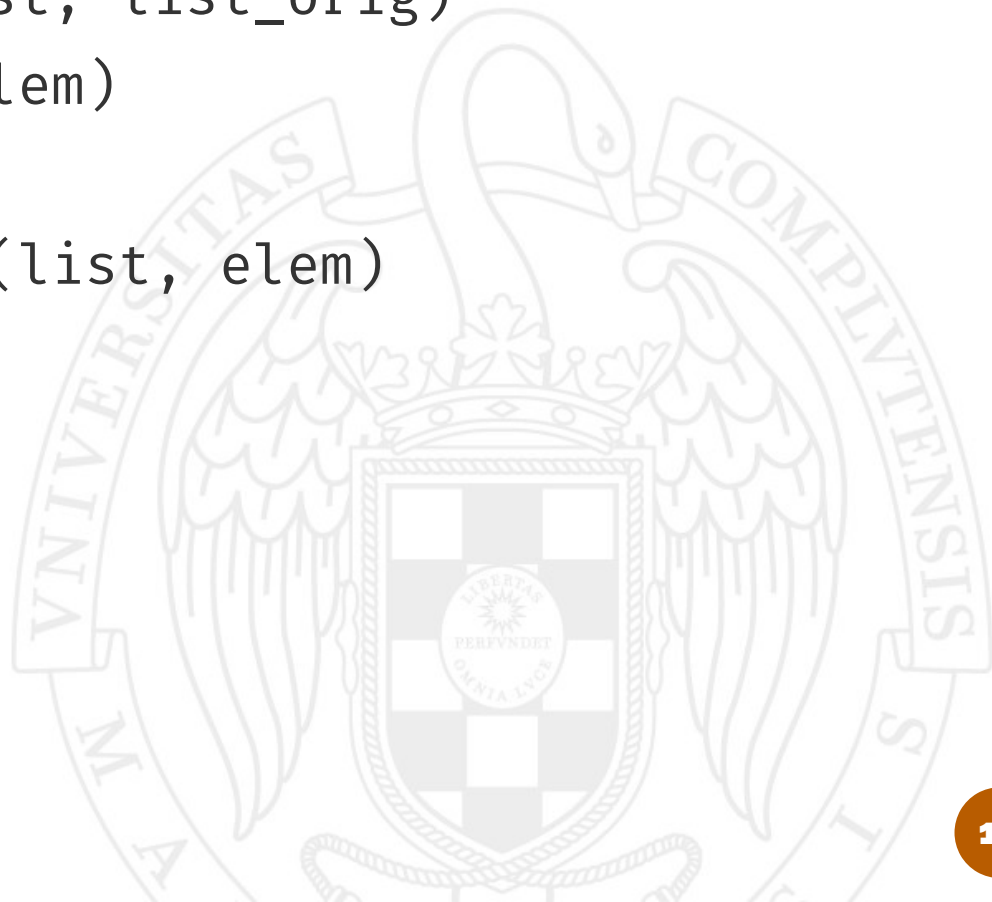
= { for (Integer x: l) {
 suma += x;
}

Funciones de utilidad



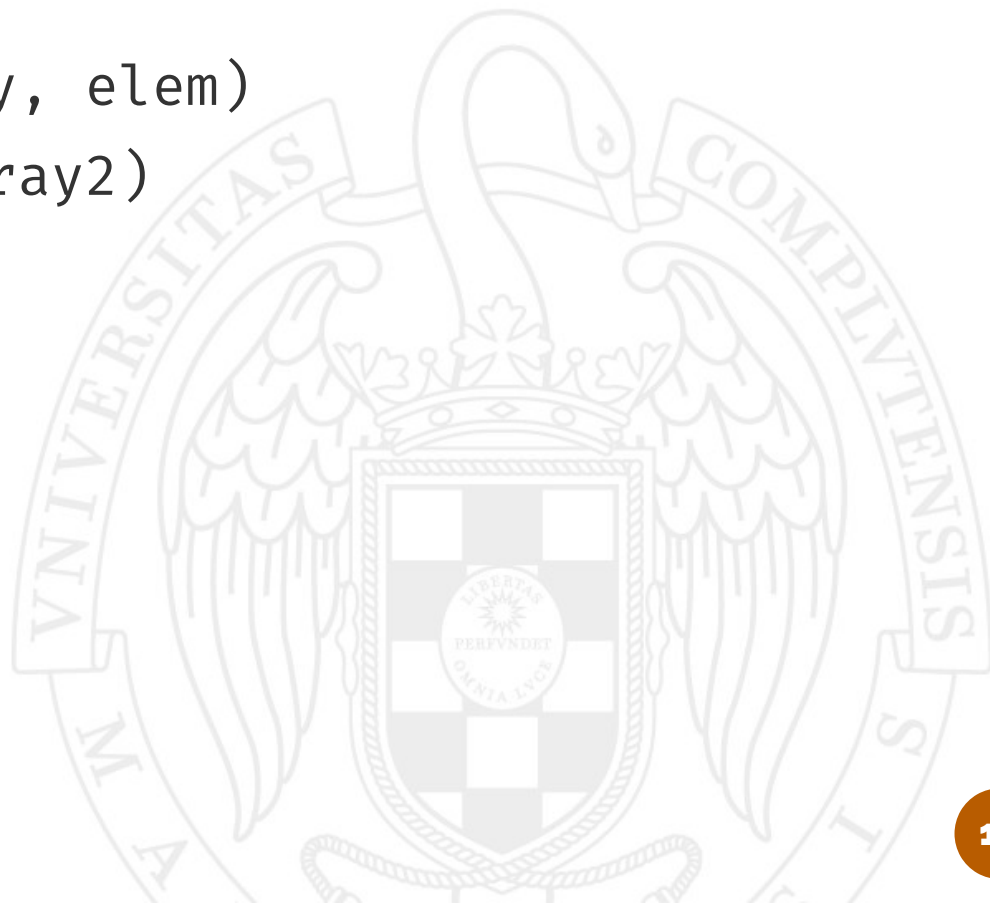
La clase Collections

- Contiene varios métodos estáticos que trabajan con listas.
 - `Collections.copy(list_dest, list_orig)`
 - `Collections.fill(list, elem)`
 - `Collections.max(list)`
 - `Collections.binarySearch(list, elem)`
 - `Collections.sort(list)`



La clase Arrays

- Utilidades similares, pero para arrays en lugar de listas.
 - `Arrays.asList(elems)`
 - `Arrays.binarySearch(array, elem)`
 - `Arrays.equals(array1, array2)`
 - `Arrays.sort(array)`
 - `Arrays.toString(array)`



Ejemplo

```
List<String> l = Arrays.asList("Ricardo", "Adrián", "Lucía", "Clara");  
Collections.sort(l);  
System.out.println(l);
```

[Adrián, Clara, Lucía, Ricardo]