METRICA

CONSULTING

Genéricos y colecciones

ÍNDICE

Genéricos

01

02 NECESIDAD

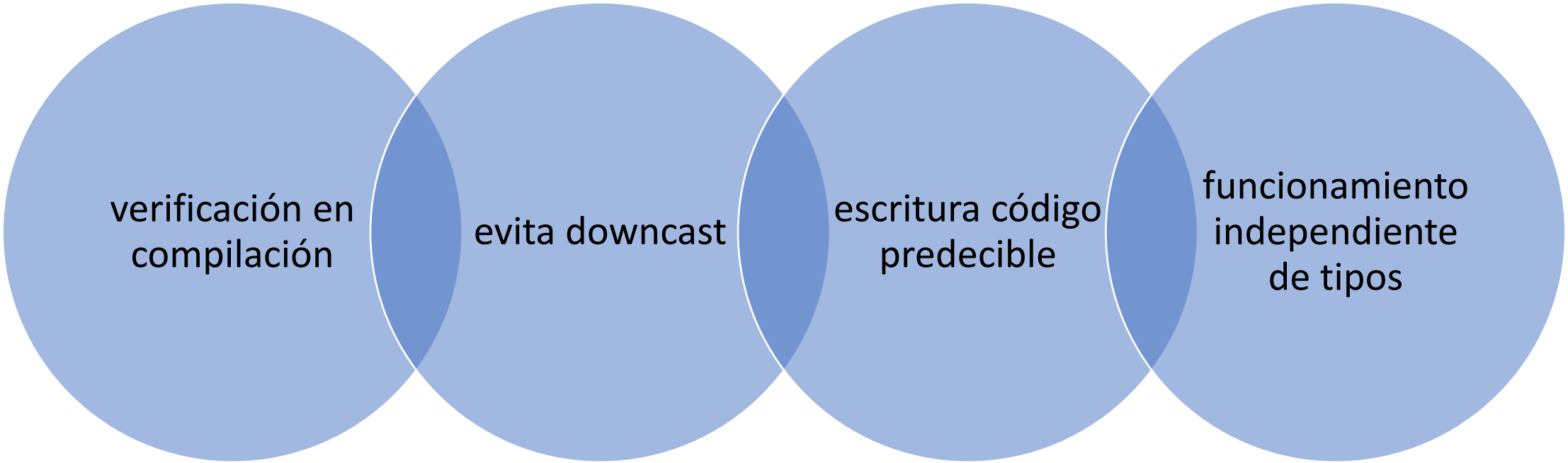
Vector

(Object)

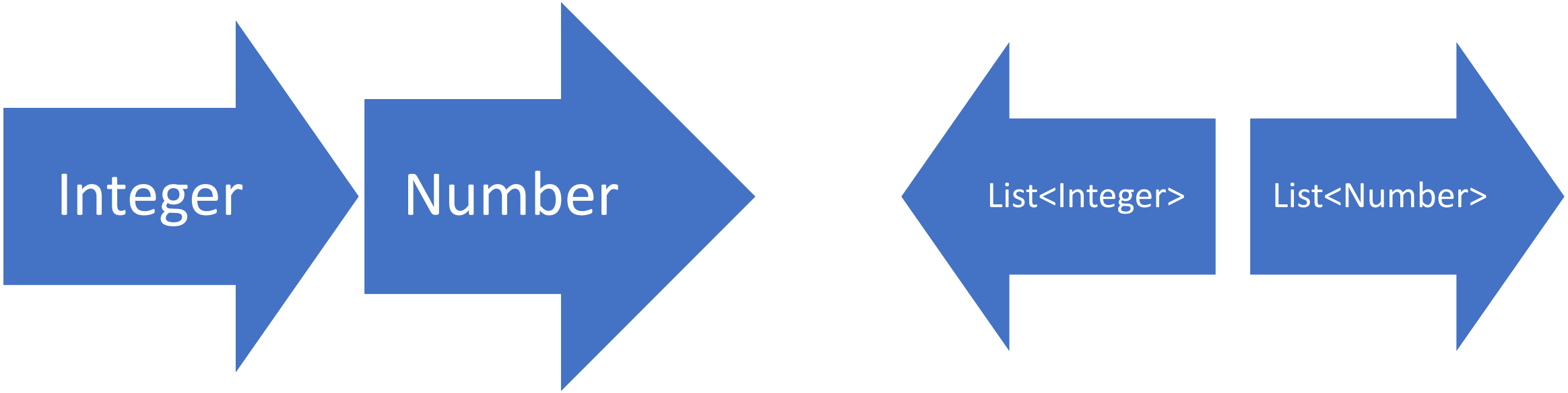
Encapsulación

- Reutilización
- Ocultación de miembros
- Sistemización

03 VENTAJAS


verificación en compilación
evita downcast
escritura código predecible
funcionamiento independiente de tipos

04 PROBLEMAS CON LA HERENCIA



05 LIMITANTES Y COMODINES

- <T extends **Number**>
- <T extends **Comparable<T>, List<String>>**
- <? extends **Comparable<String>>**

06

USO

Comparable<T>

- compareTo(T otro): int

Comparator<T>

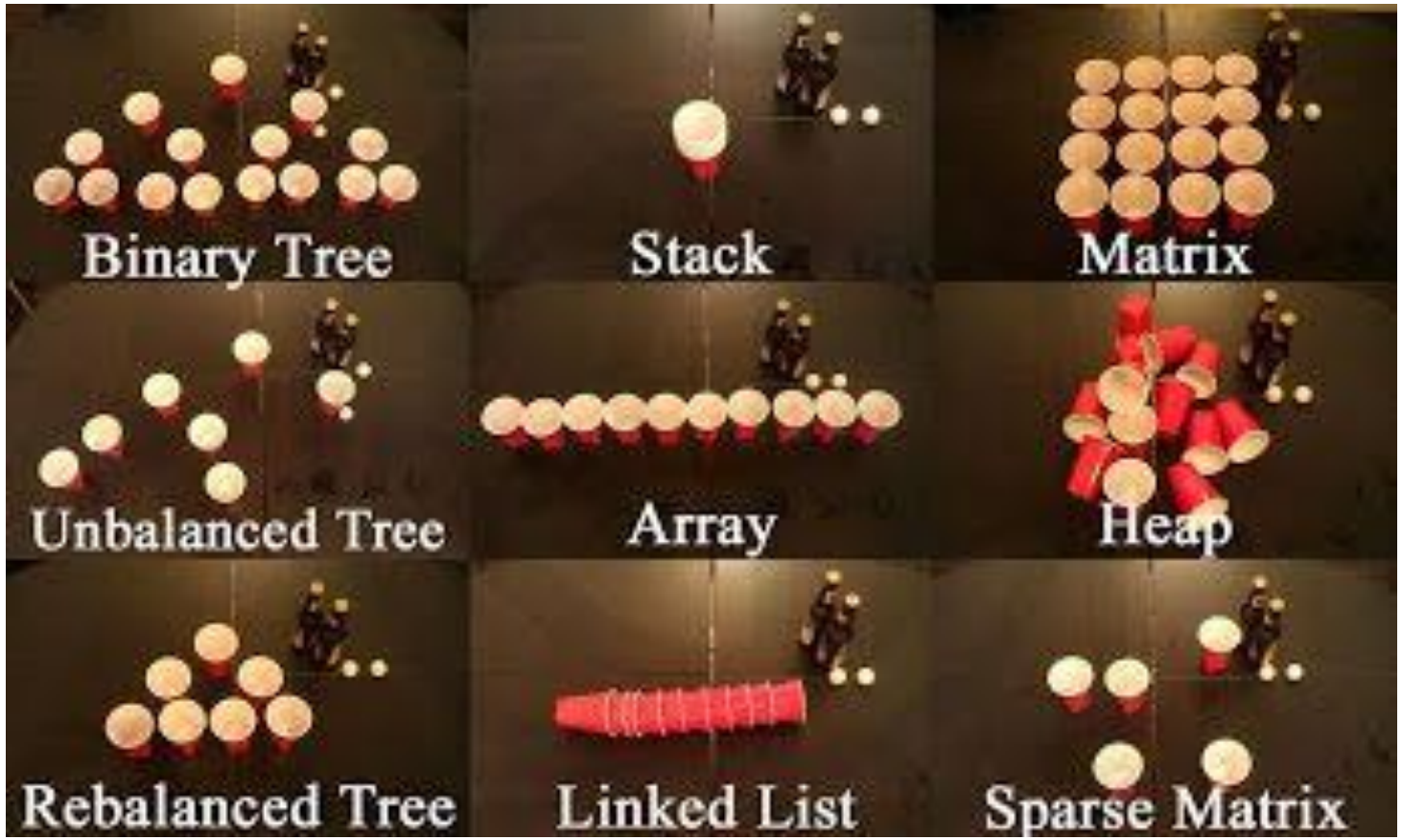
- compare(T o1, T o2): int

Iterator<T>

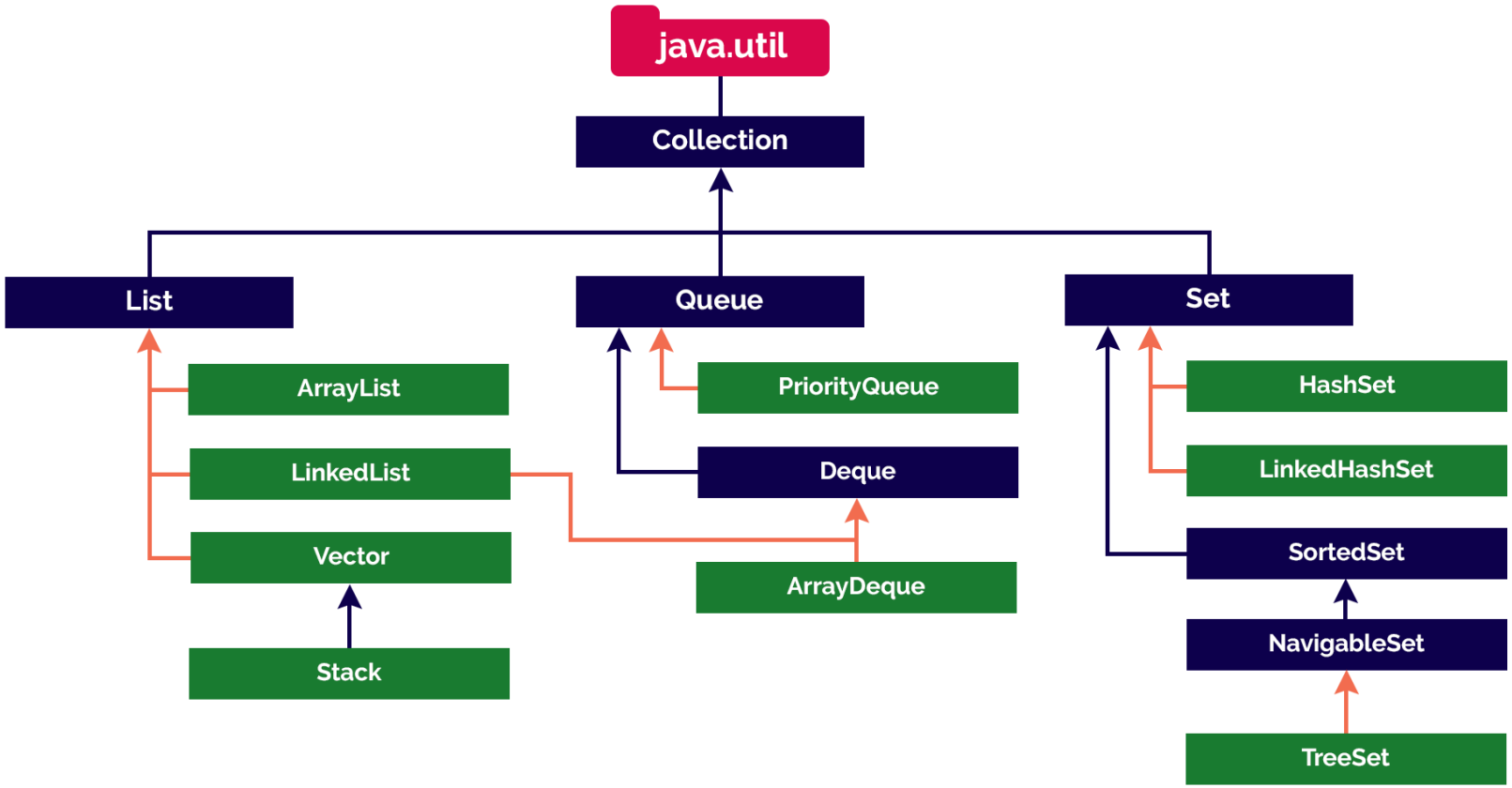
- next(): T
- hasNext(): boolean

COLLECTION

07



08 Collection<E>

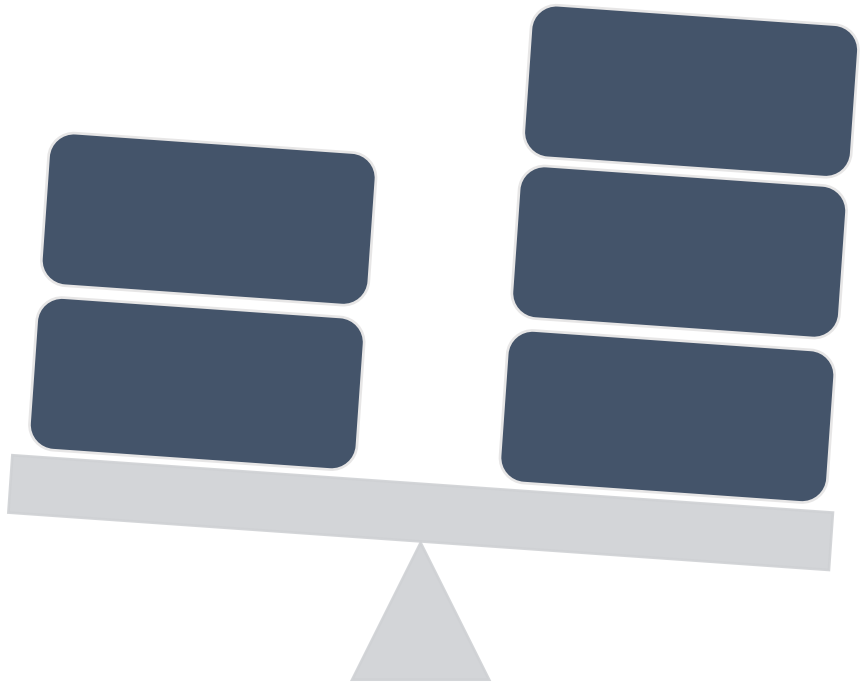


09

List<E>

ArrayList<T>

LinkedList<T>

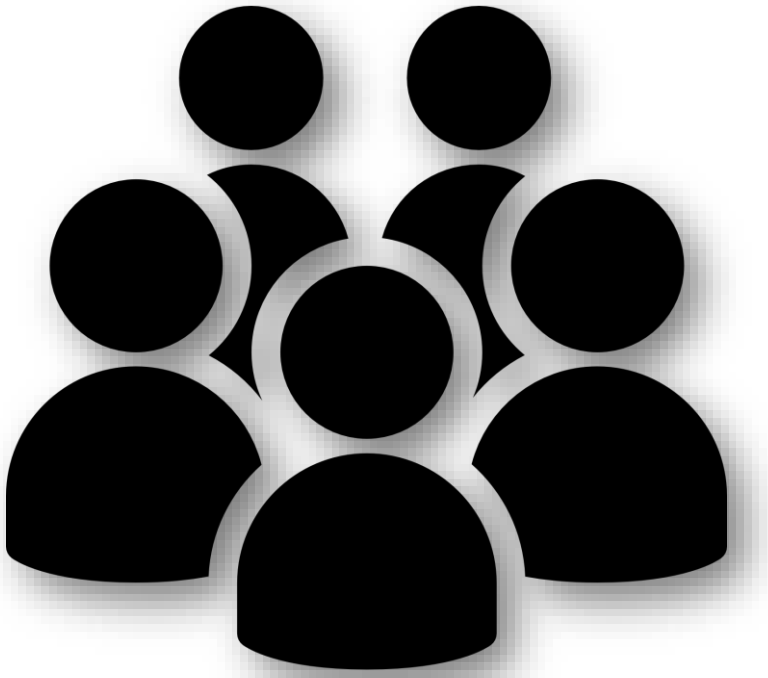


10

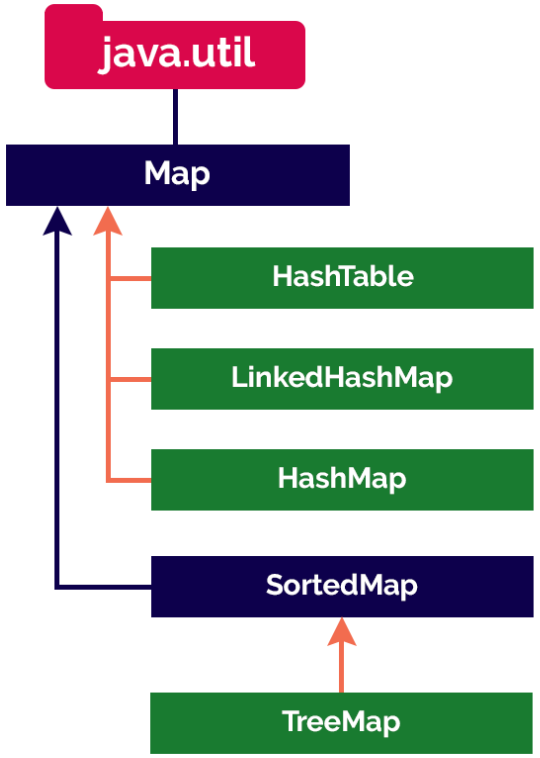
Queue<E>

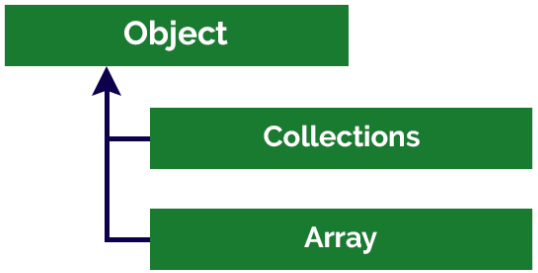
10

Set<E>



11 Map<K,V>

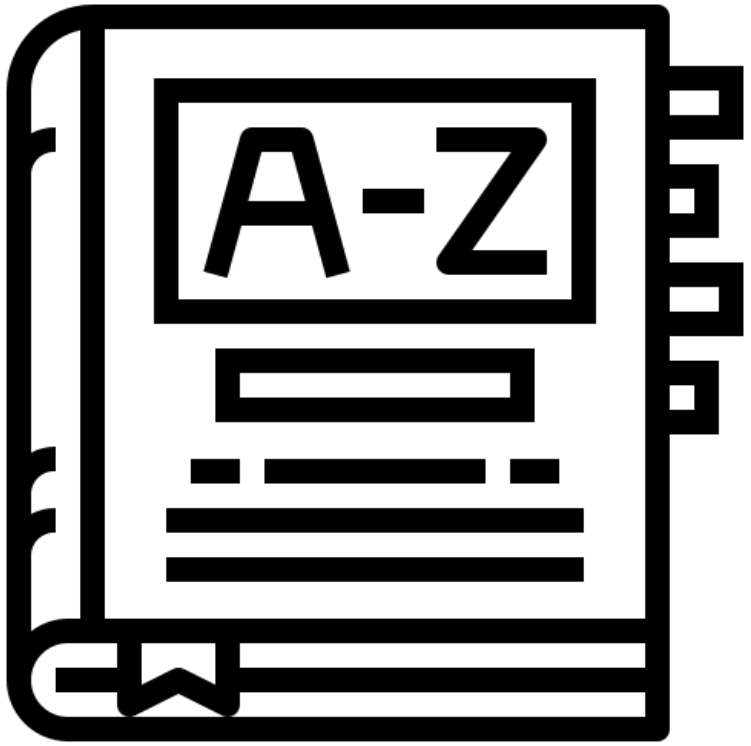

java.util
Map
HashTable
LinkedHashMap
HashMap
SortedMap
TreeMap


Object
Collections
Array


Package
Interfaces
Classes

12

USO DE DICCIONARIOS



- acceder a objetos por un id
- contar elementos no acotados
- agrupar elementos por campo
- traducciones y trasposiciones
- tablas de doble entrada

	talla P	talla M	talla G
camisetas	79	127	83
pantalones	42	73	54

Map<K1 , Map<K2,V> >

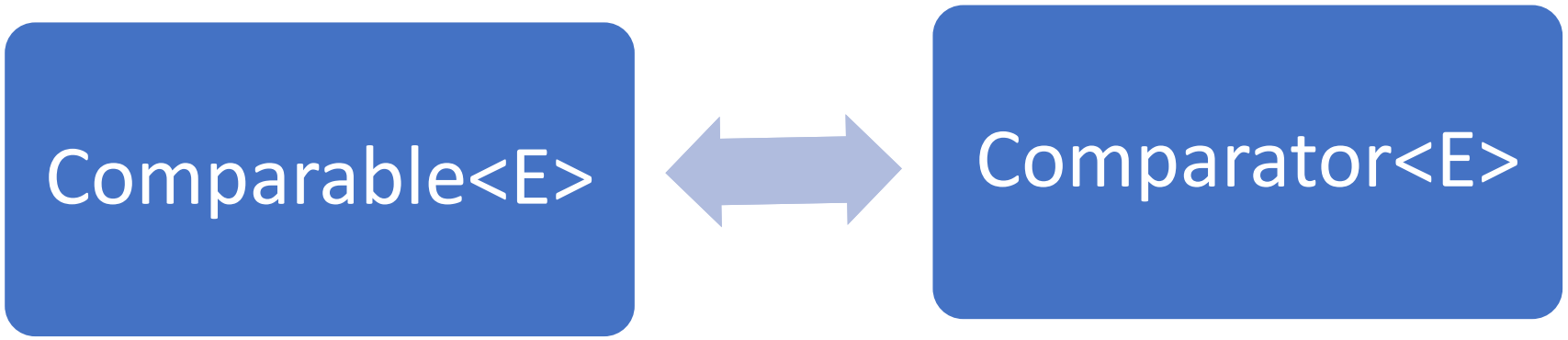
13 ESTRUCTURAS ORDENADAS

TreeMap<K,V>

TreeSet<E>

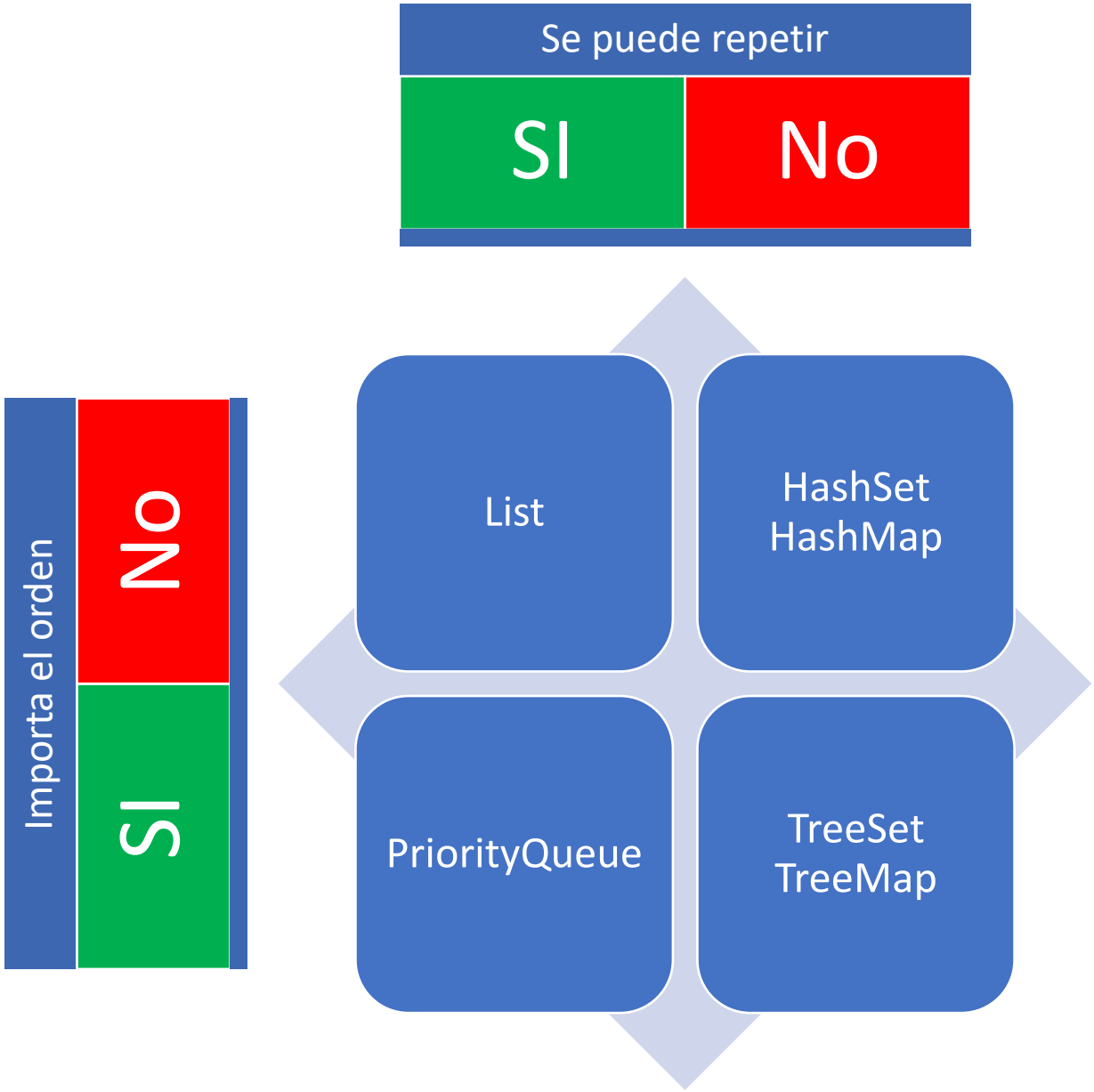
PriorityQueue<E>

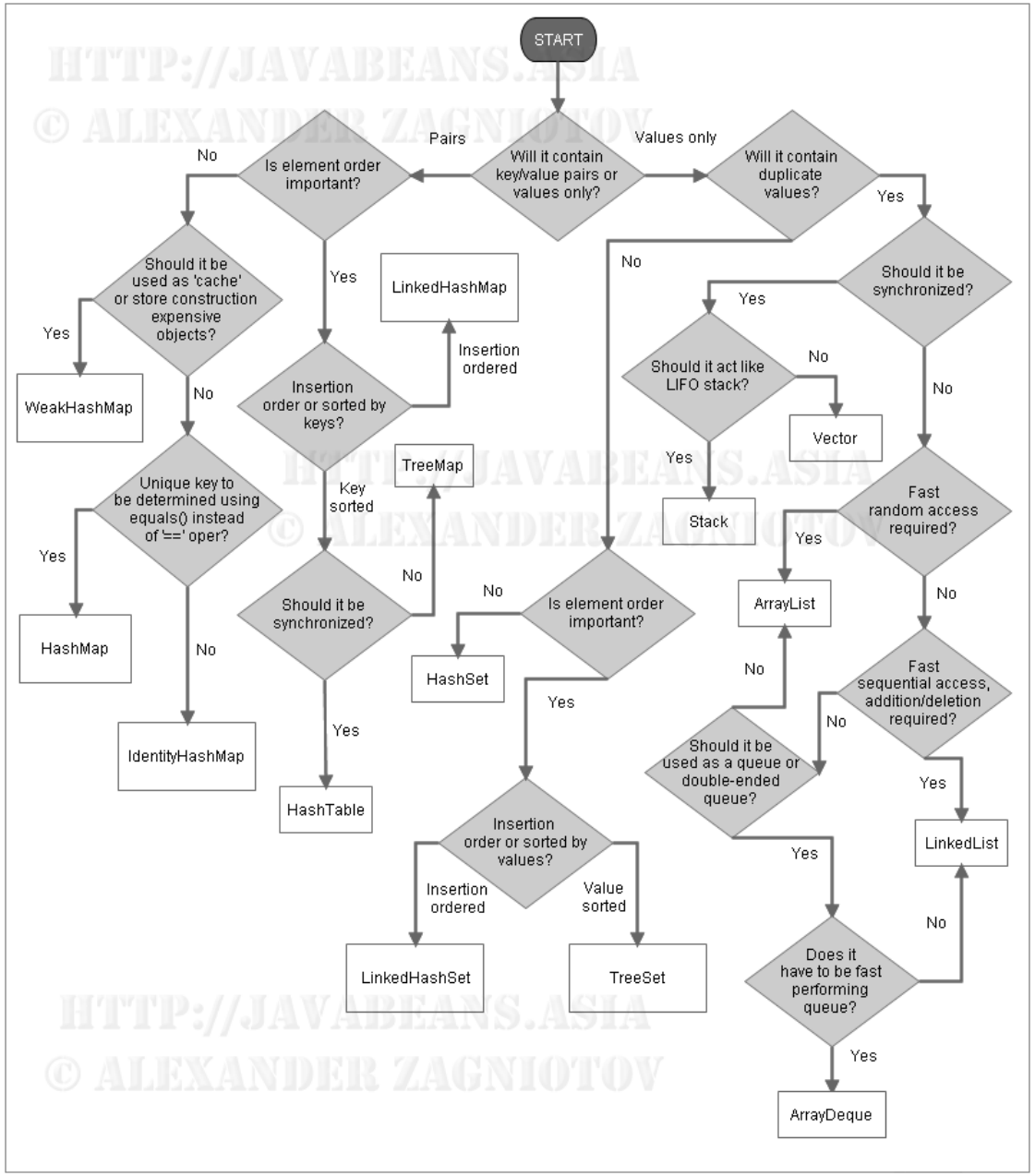
14 ESTRUCTURAS ORDENADAS



15

ELECCIÓN ESTRUCTURA




START
Will it contain key/value pairs or values only?
Pairs
Values only
Is element order important?
No
Yes
Will it contain duplicate values?
Yes
No
Should it be used as 'cache' or store construction expensive objects?
Yes
No
WeakHashMap
Insertion order or sorted by keys?
Yes
LinkedHashMap
Insertion ordered
Key sorted
TreeMap
Unique key to be determined using equals() instead of '==' oper?
Yes
No
HashMap
IdentityHashMap
Should it be synchronized?
No
Yes
HashTable
Is element order important?
No
Yes
HashSet
Insertion order or sorted by values?
Insertion ordered
Value sorted
LinkedHashSet
TreeSet
Should it be synchronized?
Yes
No
Should it act like LIFO stack?
No
Yes
Vector
Stack
Fast random access required?
Yes
No
ArrayList
Fast sequential access, addition/deletion required?
No
Yes
LinkedList
Should it be used as a queue or double-ended queue?
No
Yes
Does it have to be fast performing queue?
No
Yes
ArrayDeque

16 java.util.Arrays

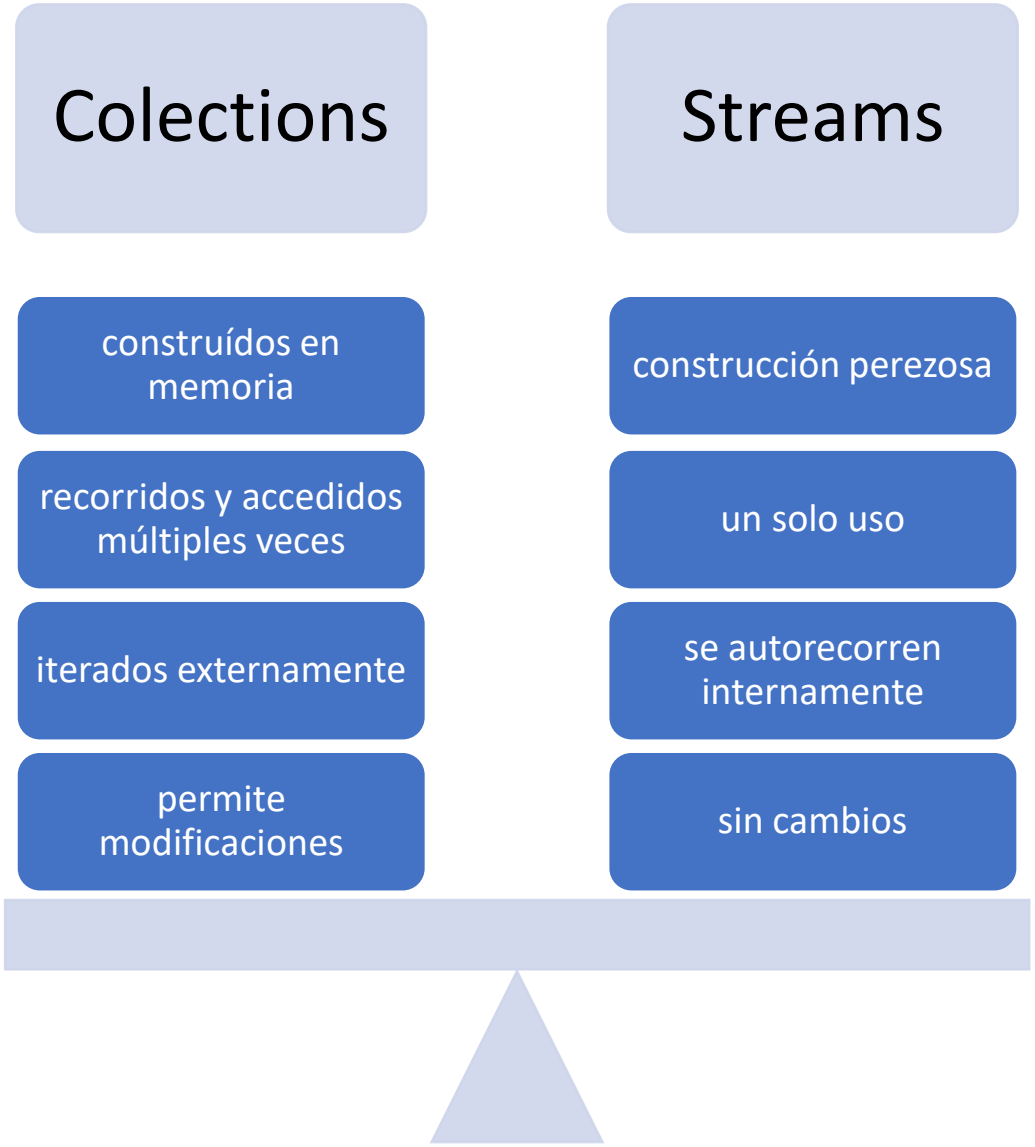
- asList(T …): List<T>
- compare(T[ ], T[ ]): int
- copyOf(T[ ]): T[ ]
- fill(T[ ], T)
- binarySearch(T[ ], T,Comparator<T>): int
- parallelSort(T[ ])
- sort(T[ ], Comparator<T>)
- stream(T[ ]): Stream<T>
- toString(T[ ]):String

16 java.util.Collections

disjoint(Collection<?>, Collection<?>): boolean
max(Collection<T>, Comparator<T>): T
replaceAll(List<T>, T, T): boolean
fill(List<T>, T)
binarySearch(List<T>, T, Comparator<T>): int
sort(List<T>, Comparator<T>)

17

Collection vs Stream



A partir de un texto, contar cuantas veces se repite cada palabra

Dadas dos ciudades, devolver la distancia que las separa

- Madrid
- Barcelona
- Cádiz
- Valencia
- Coruña

Obtener un número aleatorio. Mediante preguntas al usuario devolver si el número proporcionado es mayor, menor, igual o ya consultado hasta que se averigüe el inicial.

Dada una lista de Productos, imprimirla ordenada por precio, cantidad o nombre (a elegir por el usuario)

ETRICA

CONSULTING