TIPOS DE COLECCIONES

COLECCIONES

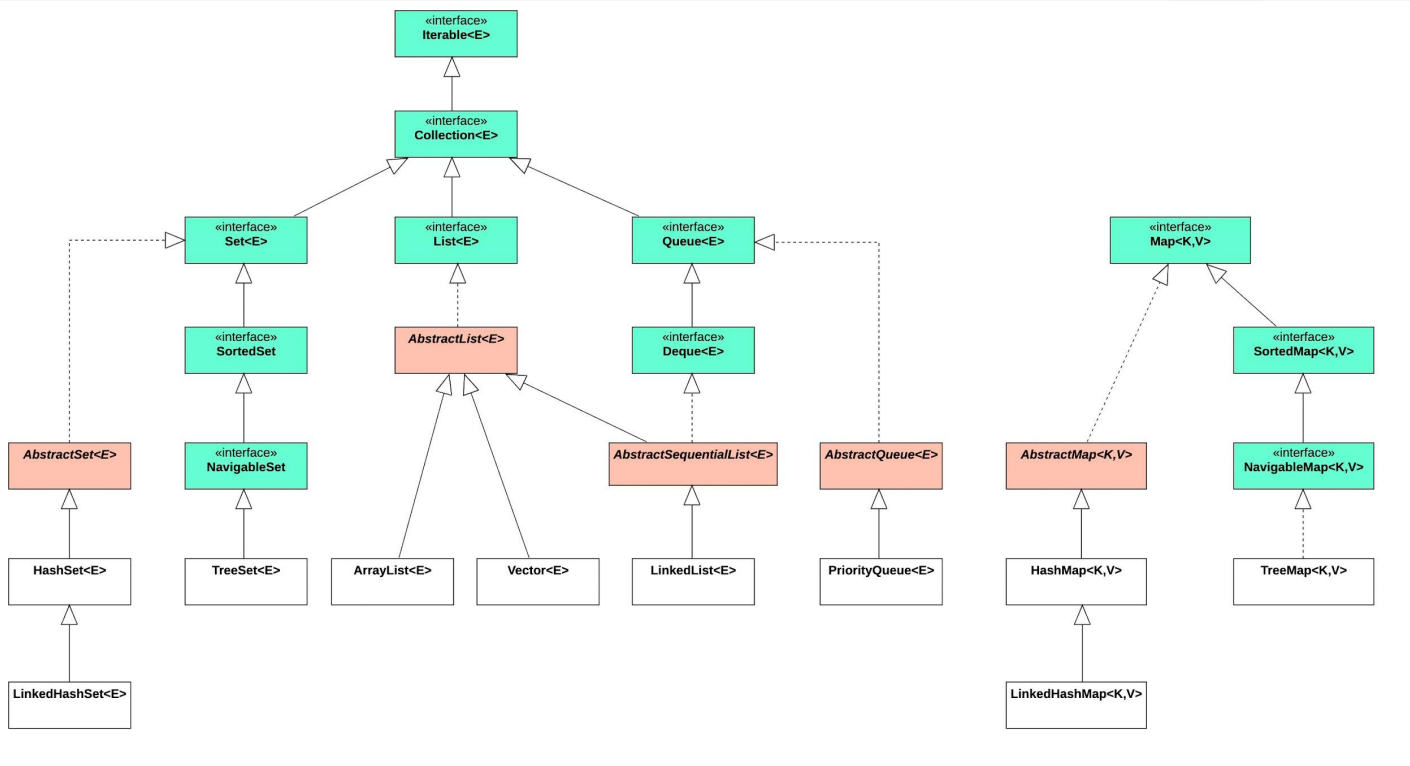
COLECCIONES

- Contenedores de elementos de un tipo con uso de genéricos.
- Disponibles desde Java 2 (1.2) (paquete java.util.)
- Ventajas sobre el uso de arrays
  - Menor esfuerzo de programación.
  - Aumento de calidad del código, velocidad de ejecución, ...
  - Mejora la interoperabilidad con librerías de terceros
  - ...

FRAMEWORK COLECCIONES JAVA

- **Interfaces**: tipos de datos, independientes de su representación
- **Implementaciones**: concreciones de los diferentes tipos
- **Algoritmos**: métodos para buscar, ordenar, clasificar. Funcionalidades reutilizables.

TIPOS DE COLECCIONES



ALGUNAS INTERFACES CLAVE

- *Iterable<E>*
- *Collection<E>*
- *List<E>*
- *Set<E>*
- *Map<K,V>*

ITERABLE<E>

- Iterador: patrón de diseño que nos permite *atravesar* un contenedor de elementos y acceder a los mismos.
- Métodos más usuales: *hasNext()*, *next()*, *remove()*
- Permite usar el bucle *for-each*

```
// Obtenemos el iterable por la vía que corresponda
Iterable<String> unIterable = obtenerIterable();

// Lo podemos recorrer usando un bucle for-each
for (String s: unIterable) {
    System.out.println(s);
}
```

COLLECTION<E>

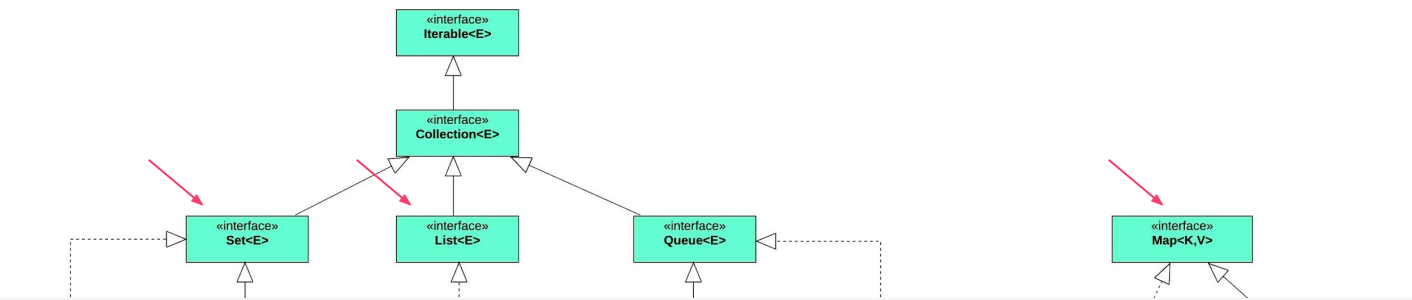
- Extiende a *Iterable<E>*
- Raíz de la mayoría de colecciones de Java (salvo *Map*)
- Base para interfaces de colecciones más concretas:
  - Que permiten o no nulos
  - Que permiten o no repetidos
  - Que están o no ordenadas
- Ideal para crear *algoritmos lo más genéricos posibles*.

MÉTODOS DE COLLECTION<E>

- Tamaño: *size* y *isEmpty*
- Comprobación: *contains*
- Añadir y eliminar: *add* y *remove*
- Iterar: *iterator*
- Operaciones bulk: *containsAll*, *addAll*, *removeAll*, *removeIf*, *retainAll*, *clear*
- Transformar en array: *toArray*
- Streams: *stream*, *parallelStream*

TIPOS DE COLECCIONES

- *List* : permite duplicados, se pueden ordenar, estructuras especiales (pilas, colas)
- *Set* : no permite duplicados, con o sin orden
- *Map* : pares clave, valor.



ELECCIÓN DEL TIPO