CREACIÓN Y USO DE LIST, SET Y MAP

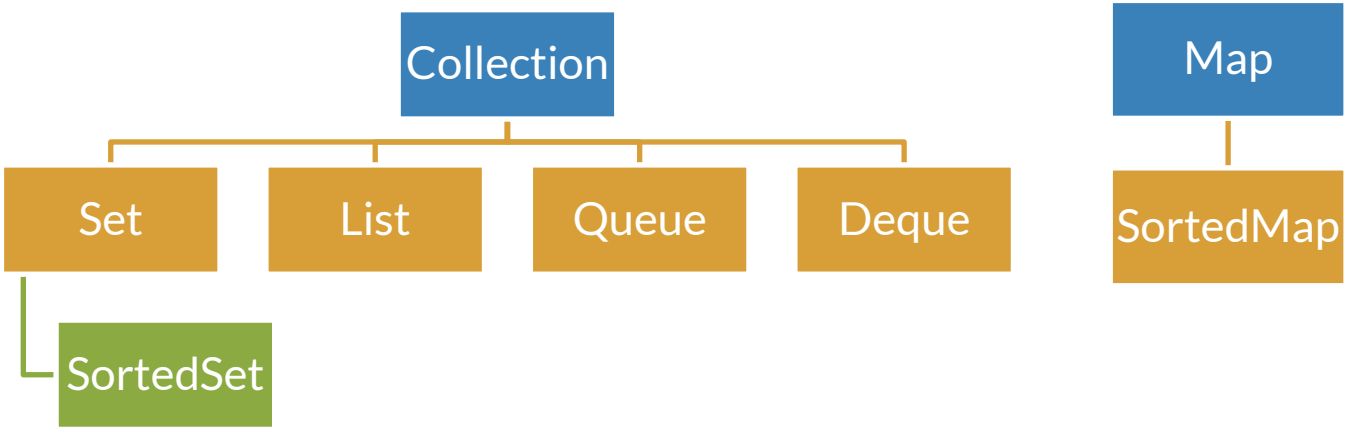
API COLLECTIONS

Desde Java SE 2 se ofrece el tratamiento de colecciones. Actualmente tiene

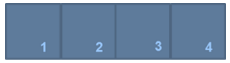
- **Interfaces**: tipos de datos
- **Implementaciones**: concreciones de los diferentes interfaces.
- **Algoritmos**: para realizar operaciones como ordenación, búsqueda, …

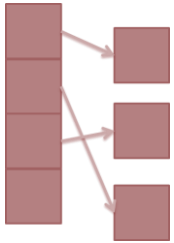
Todas las colecciones están definidas como **genéricas**.

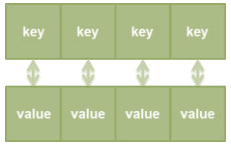
TIPOS DE COLECCIONES



TIPOS DE COLECCIONES







List

- Lineal
- Posibilidad de orden.
- Con repetidos

Set

- No soporta duplicados.
- Posibilidad de orden de elementos

Map

- Estructura clave, valor.
- Posibilidad de orden de elementos

1.

COLECCIONES LINEALES (LIST)

INTERFAZ LIST

- Los elementos tienen posición
- Permite duplicados
- También permite búsqueda e iteraciones
- Las implementaciones más conocidas son **ArrayList** y **LinkedList**.
- Si no sabemos cual escoger, utilizaremos siempre ArrayList.

CONSTRUCCIÓN DE UN LIST

A partir de Java 1.5

- Inclusión de los genéricos
- Permiten parametrizar el tipo

```
List<String> cars = new ArrayList<String>();
```

A partir de Java 1.7

- Operador *diamond*
- Nos ahorra indicar dos veces el tipo

```
List<String> cars = new ArrayList<>();
```

OPERACIONES CON LIST

Nombre	Uso
add	Añade un elemento al final la lista
addAll	Añade todos los elementos de la colección pasada como argumento
clear	Elimina todos los elementos de la lista.
contains	Comprueba si un elemento está o no en la lista
get	Devuelve el elemento de la posición especificada de la lista
isEmpty	Verifica si la lista está vacía
remove	Elimina un elemento de la lista
size	Devuelve el número de elementos de la lista
toArray	Devuelve la lista como un array

2.

COLECCIONES SIN REPETIDOS (SET)

INTERFAZ SET

- No puede contener repetidos.
- Propone tres implementaciones: **HashSet**, **TreeSet** y **LinkedHastSet**.
- ***HashSet*** es la más eficiente, pero no nos asegura nada sobre el orden.
- ***TreeSet*** utiliza un árbol Red-Black, ordena según el valor.
- ***LinkedHashSet*** es un HashSet ordenado por orden de inserción.

OPERACIONES CON SET

Nombre	Uso
add	Añade un elemento al conjunto, si aun no está contenido
addAll	Añade todos los elementos de la colección pasada como argumento si es que aun no están presentes.
clear	Elimina todos los elementos del conjunto.
contains	Comprueba si un elemento está o no en el conjunto
isEmpty	Verifica si el conjunto está vacío
remove	Elimina un elemento del conjunto
size	Devuelve el número de elementos de la lista
toArray	Devuelve la lista como un array

3.

COLECCIONES CALVE, VALOR

INTERFAZ MAP

- No es un subtipo de Collection (List y Set sí que lo son).
- Cada elemento tiene estructura clave, valor.
- La clave sirve para acceder directamente al valor.
- Las implementaciones son ***HashMap***, ***TreeMap*** y ***LinkedHashMap***. Las consideraciones son análogas a Set.

OPERACIONES CON MAP

Nombre	Uso
clear	Elimina todos los elementos del *diccionario*.
containsKey	Comprueba si una clave está presente en el *diccionario*.
containsValue	Comprueba si un valor está presente en el *diccionario*.
get	Devuelve el valor asociado a una clave.
isEmpty	Verifica si el conjunto está vacío
keySet	Devuelve un Set con todas las claves.
put	Permite insertar un par clave, valor
remove	Elimina un elemento del conjunto
size	Devuelve el número de elementos de la lista
values	Devuelve un Collection con los valores