I.I Colecciones

Acceso a Datos

Alejandro Roig Aguilar
alejandro.roig@iesalvarofalomir.org

IES Álvaro Falomir
Curso 2023-2024

Concepto de colección y tipos

Las colecciones son estructuras dinámicas de datos: su tamaño varía a medida que se añaden o quitan elementos en tiempo de ejecución.

Entre los tipos principales de colecciones encontramos:

- Listas. Colecciones indexadas, donde cada elemento ocupa una posición referenciada por un índice numérico.
- Mapas. Colecciones de pares clave-valor, donde cada valor está asociado a una clave que lo identifica. También llamados tablas hash o diccionarios.
- Otros: pilas, colas, conjuntos, árboles, grafos, etc.

Listas

Estructuralmente, los elementos de una lista no tiene porqué estar contiguos en memoria, como sí ocurre con los ArrayList de Java.

Según su implementación, podemos diferenciar:

- Enlazada simple: Cada nodo contiene, además de la información del elemento, un puntero al elemento siguiente.
- Doblemente enlazada: Añaden a cada nodo un nuevo puntero a su predecesor. Java las implementa en la clase LinkedList.
- Circulares: El último nodo de la lista apunta al primero.

Listas

Independientemente de la implementación, las listas disponen métodos para:

- Añadir elementos al final de la lista.
- Añadir elementos en cualquier posición intermedia de la lista.
- Borrar un elemento dada su posición.
- Comprobar el tamaño de la lista.
- Acceder u obtener el elemento de una posición de la lista.
- ...

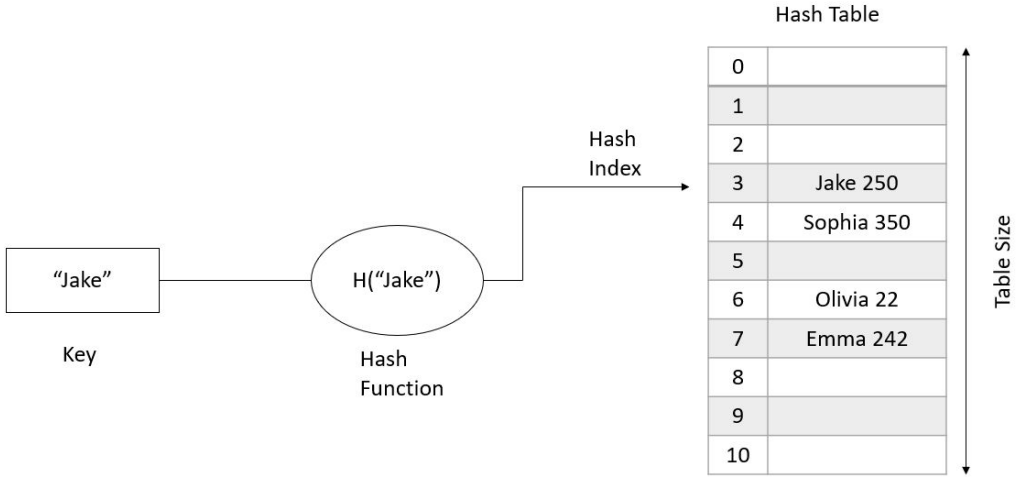
Listas. Ejemplo

```
ArrayList<String> textos = new ArrayList<>();

textos.add("Uno");
textos.add("Dos");
textos.add("Tres");
textos.add(2, "Dos y medio");        // Entre "Dos" y "Tres"

textos.remove(1);                    // Elimina "Dos"

for (int i = 0; i < textos.size(); i++) {
        System.out.println(textos.get(i));
}
```

Mapas, diccionarios o tablas hash

Los mapas almacenan asociaciones clave–valor eficientemente.

Para su implementación, se utiliza una tabla de tamaño variable en la que se almacenan los elementos en una posición dada por una función hash aplicada sobre la clave.



Mapas, diccionarios o tablas hash

Ante una colisión (la posición de la tabla donde insertar un par está ocupada por otro), la solución más empleada es el hashing abierto, creando una lista donde se enlazan dichos pares clave–valor.

Los mapas disponen de métodos para:

- Añadir elementos
- Borrar un elemento dada su clave
- Comprobar el tamaño del mapa
- Acceder u obtener un valor por su clave
- Comprobar si existe una clave dada
- …

Mapas. Ejemplo

```
HashMap<String, Producto> productos = new HashMap<>();

productos.put("111A", new Producto("111A", "Monitor LG 22 pulg", 99.95f));
productos.put("222B", new Producto("222B", "Disco duro 512GB SSD", 109.95f));
productos.put("333C", new Producto("333C", "Ratón bluetooth", 19.35f));

productos .remove("222B");

// keySet() obtiene todas las claves
for (String codigo : productos.keySet()) {
        System.out.println(productos.get(codigo));
}
```

Otros tipos de colecciones

- Pilas. Los elementos se añaden (push) y se extraen (pop) por el mismo extremo de la pila. En Java tenemos la clase Stack.
- Colas. Los elementos se añaden (enqueue) por un extremo y se extraen (dequeue) por el opuesto. Java ofrece la clase Queue.
- Conjunto. No admite elementos repetidos.
- Árboles. Representa una jerarquía de elementos en la que cada elemento tiene un único antecesor y puede tener N sucesores.
- Grafos. Colecciones donde desde cada nodo se puede acceder a otros N nodos, estableciendo distintos caminos y conexiones entre ellos.

Java Collection Framework

What can your collection do for you?

Collection class	Thread-safe alternative	Your data				Operations on your collections						
		Individual elements	Key-value pairs	Duplicate element support	Primitive support	Order of iteration			Performant 'contains' check	Random access		
						FIFO	Sorted	LIFO		By key	By value	By index
HashMap	ConcurrentHashMap	✗	✓	✗	✗	✗	✗	✗	✓	✓	✗	✗
HashBiMap (Guava)	Maps.synchronizedBiMap (new HashBiMap())	✗	✓	✗	✗	✗	✗	✗	✓	✓	✓	✗
ArrayListMultimap (Guava)	Maps.synchronizedMultiMap (new ArrayListMultimap())	✗	✓	✓	✗	✗	✗	✗	✓	✓	✗	✗
LinkedHashMap	Collections.synchronizedMap (new LinkedHashMap())	✗	✓	✗	✗	✓	✗	✗	✓	✓	✗	✗
TreeMap	ConcurrentSkipListMap	✗	✓	✗	✗	✗	✓	✗	✓*	✓*	✗	✗
Int2IntMap (Fastutil)		✗	✓	✗	✓	✗	✗	✗	✓	✓	✗	✓
ArrayList	CopyOnWriteArrayList	✓	✗	✓	✗	✓	✗	✓	✗	✗	✗	✓
HashSet	Collections.newSetFromMap (new ConcurrentHashMap<>())	✓	✗	✗	✗	✗	✗	✗	✓	✗	✓	✗
IntArrayList (Fastutil)		✓	✗	✓	✓	✓	✗	✓	✗	✗	✗	✓
PriorityQueue	PriorityBlockingQueue	✓	✗	✓	✗	✗	✓**	✗	✗	✗	✗	✗
ArrayDeque	ArrayBlockingQueue	✓	✗	✓	✗	✓**	✗	✓**	✗	✗	✗	✗

** O(log(n)) complexity, while all others are O(1) - constant time*

*** when using Queue interface methods: offer() / poll()*