Contenedores (II)
Listas y Mapas

Metodologia i Programació Orientada
a Objectes

Listas

- Una lista permite guardar elementos en un determinado orden.
- Cada elemento de una lista puede ser accesible a través de un índice, que indica su orden en la lista.
- Se diferencia de un vector
  - El tamaño no es fijo → redimensionado dinámico
  - Sus elementos no se acceden a través de [y], sino a través de los métodos `add` y `get`.
  - Si se añade un elemento en medio de la lista, el que estaba en esa posición y los siguientes se mueven una posición hacia atrás.
  - Si se elimina un elemento, los que le seguían se mueven una posición hacia adelante.
- Una lista puede verse como una “fila” de objetos, donde podemos “colar” o “sacar” elementos en medio.

Clases que implementan una Lista

- Al igual que con los conjuntos, podemos usar listas mediante varias clases cuyo uso es idéntico: ArrayList, LinkedList, Vector…
  - Pero están implementadas de diferente manera, con sus ventajas e inconvenientes
- Ejemplo de uso:

```
List<String> li = new ArrayList<>();
li.add("texto 1"); //añade al final
li.add("otro texto"); //añade al final
li.add(0, "más texto"); //añade al principio
li.remove(1); //elimina objeto en posición 1
```

Principales métodos de lista

`add(objeto)` → añade un objeto al final de la lista

`add(posición,objeto)` → añade un objeto en la posición indicada

`get(posición)` → obtiene el objeto que está en la posición indicada

`contains(objeto)` → retorna *true* si el objeto indicado está contenido en la lista

`remove(objeto)` → quita el objeto especificado por parámetro

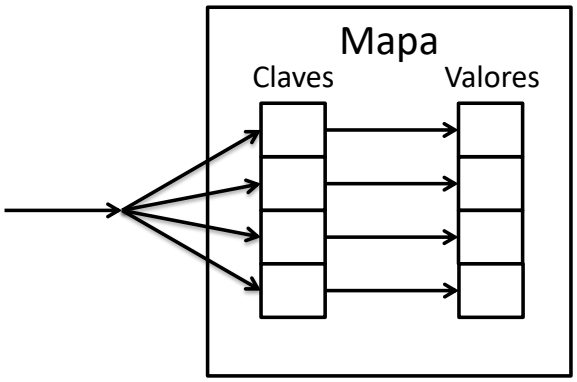
`remove(posición)` → quita el objeto especificado en la posición

`size()` → retorna el número de elementos en la lista

Más información: http://download.oracle.com/javase/6/docs/api/java/util/List.html

Mapas

- También conocidos como “Diccionarios”
- Guardan objetos que, en vez de accederse por un orden numérico (listas, arrays), se acceden mediante una clave **única** que identifica a cada uno.
- Cuando se instancia, ahora se deben poner dos tipos entre < y >:
  - El tipo de datos de la clave
  - El tipo de datos del valor asociado a la clave



Ejemplo de uso de un Mapa

- Varias clases implementan diccionarios: HashMap, TreeMap, HashTable, Properties…
- Supongamos que previamente hemos creado una clase llamada Persona, y queremos guardar las relaciones familiares con varias de esas personas.

```
Map<String,Persona> parientes = new HashMap<>();
parientes.put(“padre”, new Persona(“José Perez”));
parientes.put(“madre”, new Persona(“Maria Lopez”));
parientes.put(“hermano”, new Persona(“Manuel Perez”));
// ojo! Esto substituirá a la persona que antes se
// accedia mediante la clave “padre” por otra nueva
parientes.put(“padre”, new Persona(“Juan García”));
// este comando nos devolverá el objeto Persona
// cuyo nombre es “Manuel Perez”
Persona p = parientes.get(“hermano”);
```

Principales métodos de un Mapa

`put(clave,valor)`→ guarda un valor, que será accesible mediante la clave especificada

`get(clave)`→ accede al objeto que está guardado según la clave especificada

`remove(clave)`→ borra el objeto que está guardado según la clave especificada

`keySet()`→ devuelve un conjunto (Set) con las claves del diccionario

`values()` → devuelve una colección con los valores que hay en el diccionario

`size()`→ retorna el número de pares clave/valor que guarda el diccionario

Más información: http://download.oracle.com/javase/6/docs/api/java/util/Map.html

Comparativa de contenedores

	Conjunto (Set)	Lista (List)	Diccionario (Map)
Acceso a los elementos	Solo con un iterador o "foreach"	A través de un índice numérico, también iterando	A través de una clave, también iterando claves o valores.
Uso	Generalmente, guarda objetos a los que trataremos luego como un conjunto (iterando uno a uno)	Guarda objetos sobre los cuales nos interesa dar su orden (índice): primero, segundo, tercero… Y poder acceder a éstos según este orden/índice	Guarda objetos según unas claves que no tienen por qué ser un orden: ej. un DNI…

Ejercicios

- Crear una clase `CestaCompra`, que guarda elementos de una clase `Producto`, que tienen un nombre y un precio.
  - Crear un método para añadir nuevos productos
  - Crear un método que saque por pantalla todos los productos y el precio total de la cesta.
- Crear una clase `PeliculasFavoritas`, que debe poder guardar y mostrar las películas favoritas según un orden dado por el usuario.
  - Crear un método para añadir nuevas películas según un orden dado.
  - Crear un método que saque por pantalla las tres primeras películas favoritas del usuario.
- Crear una clase `CochesMultados`, que debe guardar un registro de coches que han sido multados por la policía.
  - Crear un método para añadir un coche multado.
  - Se debe poder buscar un coche en el registro directamente por su matrícula.
- Para los tres ejercicios anteriores, se pueden crear otras clases que creáis convenientes.

Ejercicio

- Supongamos que alguien nos ha dado hecha la clase **`Cancion`** con los siguientes métodos:
  - **`void tocar()`** → Empieza a sonar la canción
  - **`void parar()`** → Deja de sonar la canción
  - **`String toString()`** → Devuelve un String con el autor y el título de la canción
- **Programar** la clase **`ReproductorMP3`**, que contenga una lista de reproducción e implemente los siguientes métodos:
  - **`void agregarCancion(Cancion c)`** → Añade una canción a la lista de reproducción
  - **`void tocar()`** → toca la canción actual en la lista de reproducción
  - **`void parar()`** → deja de tocar la canción actual
  - **`void siguienteCancion()`** → pasa a tocar la siguiente canción, o la primera si se ha llegado al final de la lista
  - **`void imprimeLista()`** → Muestra por pantalla la información de todas las canciones en la lista de reproducción