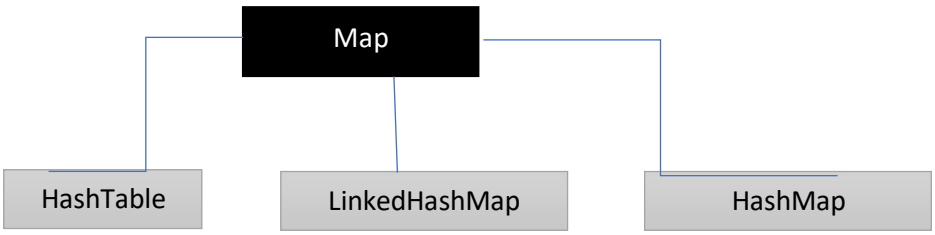
Interfaz Map



Es la raíz de todas las clases capaces de implementar mapas. Hasta la versión 1.5, los mapas eran colecciones de pares clave, valor donde tanto la clave como el valor eran de tipo Object. Desde la versión 1.5 esta interfaz tiene dos genéricos: K para el tipo de datos de la clave y V para el tipo de los valores.

Esta interfaz no deriva de Collection por lo que no usa iteradores ni ninguno de los métodos vistos anteriormente. La razón es que la obtención, búsqueda y borrado de elementos se hace de manera muy distinta. Los mapas no permiten insertar objetos nulos (provocan excepciones de tipo NullPointerException). Map define estos métodos:

Método	Uso
V get(K clave)	Devuelve el objeto que posee la clave indicada
V put(Object clave, V valor)	Coloca el par clave-valor en el mapa (asociando la clave a dicho valor). Si la clave ya existiera, sobrescribe el anterior valor y devuelve el objeto antiguo. Si esa clave no aparecía en la lista, devuelve null
V remove(Object clave)	Elimina de la lista el valor asociado a esa clave. Devuelve el valor que tuviera asociado esa clave o null si esa clave no existe en el mapa
boolean containsKey(Object clave)	Indica si el mapa posee la clave señalada
boolean containsValue(Object valor)	Indica si el mapa posee el valor señalado
void putAll(Map mapa)	Añade todo el mapa indicado, al mapa actual
Set <k> keySet()	Obtiene un objeto Set creado a partir de las claves del mapa
Collection <V> values()	Obtiene la colección de valores del mapa, permite utilizar el HashMap como si fuera una lista normal al estilo de la clase Collection (por lo tanto se permite recorrer cada elemento de la lista con un iterador)
int size()	Devuelve el número de pares clave-valor del mapa
void clear()	Elimina todos los objetos del mapa

Las operaciones fundamentales son get, put y remove. El conjunto de claves no puede repetir la clave. Hay que tener en cuenta que las claves se almacenan en una tabla hash (es decir es una estructura de tipo Set) por lo que para detectar si una clave está repetida

Clase HashTable

La clase Hashtable representa un tipo de colección basada en claves, donde los objetas almacenados en la misma (valores) no tienen asociado un índice numérico basado en su posición, sino una clave que lo identifica de forma única dentro de: la colección. Una clave puede ser cualquier tipo de objeto.

La utilización de colecciones basadas en claves resulta útil en aquellas aplicaciones en las que se requiera realizar búsquedas de objetos a partir de un dato que lo identifica. Por ejemplo, si se va a gestionar una colección de objetos de tipo "Empleado", puede resultar más práctico almacenarlos en un Hashtable, asociándoles como clave el ˝dni", que guardarlos en un ArrayList en el que a cada empleado se le asigna un índice según el orden de almacenamiento.

Creación de un hashtable

La creación de un objeto hashtable se realiza utilizando el constructor sin parámetros de la clase:

Hashtable variable_objeto = new Hashtable()

Por ejemplo,

```
Hashtable tb = new Hashtable();
```

Métodos de la clase hashtable

Los principales métodos expuestos por la clase Hashtahle para manipular la colección son los siguientes;

Object put(Object key, Object valor). Añade a la colección el objeto valor, asignándole; la clave especificada por *key*. En caso de que exista esa clave en la colección, el objeto que tenía asignada esa clave se sustituye por el nuevo objeto valor, devolviendo objeto sustituido. Por ejemplo, el siguiente código:

```
Hashtable hs=new Hashtable();

hs.put ("a21", "pepito");

System.out.println ("Antes se llamaba "+ hs.put ("a21", "luis"));
```

Mostrará en pantalla:

Antes se llamaba pepito

Bolean containsKey(Object key). Indica sí la **clave especificada existe o no en la** colección.

Object get(Object key) Devuelve el valor que tiene asociada la clave que se indica en el parámetro. En caso de que no exista ningún objeto con esa clave asociada, devolverá ***null.***

Object remove(Object key). Elimina de la colación el valor cuya clave se especifica en el parámetro. En caso de que no exista ningún objeto con esa clave, no hará nada y devolverá *null* si existe, eliminará el objeto y el mismo será devuelto por el método.

int size(). Devuelve el número de objetos almacenados en la colección.

Enumeration Keys(). Devuelve un objeto enumeration que permite iterar sobre el conjunto de claves. En el siguiente apartado se estudiará con detalle este objeto

Iteración de un hashtable: la interfaz enumeration

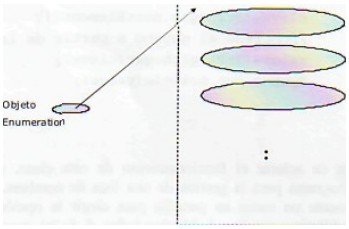
Al no estar basado en índices, un Hashtable no se puede recorrer utilizando una instrucción ***for*** con una variable que recorra las posiciones de los objetos.

Esto no significa que no se pueda iterar sobre un Hashtable, se puede hacer a través de un objeto enumeration.

Enumeration es un objeto que implemento la interfaz **java.util.Enumeration**. Las interfaces disponen de una serie de métodos que pueden ser aplicados sobre los objetos que las implementan,

Los métodos proporcionados por la interfaz Enumeration permiten recorrer una colección de objetos asociada y acceder a cada uno de sus elementos. En el caso concreto del método *keys()* de la clase Hashtable, el objeto Enumeration devuelto nos permite recorrer la colección de claves del Hashtable.

Un objeto Enumeration lo podemos imaginar como una especie de puntero o referencia, que puede ir apuntando *a* cada uno de los elementos de una colección.



Objeto enumeration asociado a una colección

La interfaz Enumeration dispone de los siguientes métodos:

Object nextElement(). La llamada al método *nextElement()* sobre un objeto enumeration, provoca que éste pase a apuntar al siguiente objeto de la colección, devolviendo el nuevo objeto apuntado. Hay que tener en cuenta que, inicialmente, un enumeration se encuentra apuntando a la posición que está antes del primer objeto de la colección. por lo que la primera llamada a *nextElement())* devolverá el primer objeto.

boolean hasMoreElements(). Indica si hay más elementos por recorrer en la colección. Cuando el objeto enumeration esté apuntando al último elemento, la llamada a este método devolverá *false.*

Con estos dos métodos, utilizando un bucle: *while,* se puede acceder a todos los elementos de la colección asociada al Enumeration.

El siguiente método recibe como parámetro un objeto Hashtable en el que se han almacenado objetos String con claves asociadas de tipo String, su misión consiste en mostrar en pantalla todos los valores almacenados, tendríamos que escribir el siguiente código.

```
public void muestraDatos (Hashtable tb)
{
  String valor, clave;
  Enumeraction e = tb.keys();
  while (e.hasMoreElements())
        {
         clave = (String) e.nextElement();
         //obtiene el objeto a partir de la clave
         valor = ( String) tb.get (clave);
         System.out.println(valor);
        }
}
```