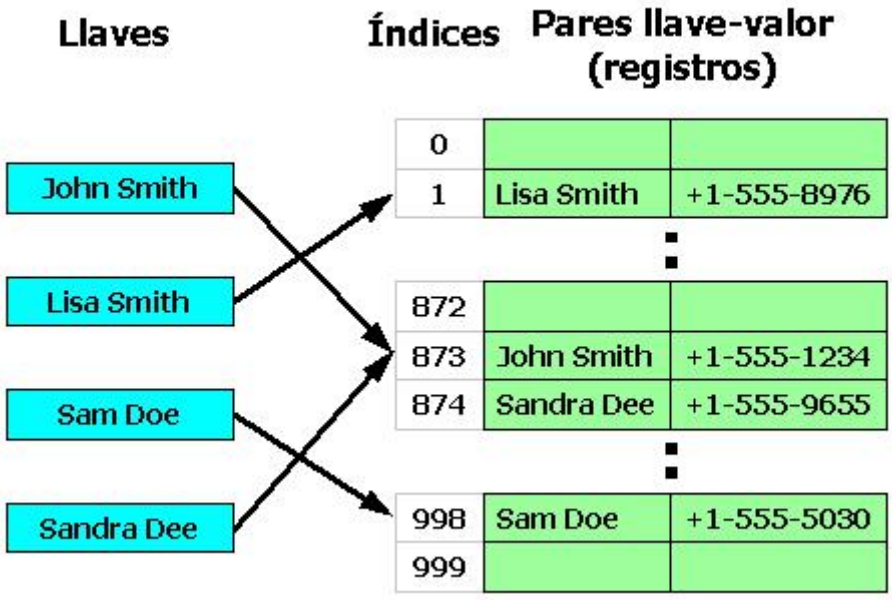
Tabla Hash.

Una *Hashtable* Java es una estructura de datos que utiliza una función hash para identificar datos mediante una llave o clave (ej. Nombre de una persona). La función hash transforma una llave a un valor índice de un arreglo de elementos. En este caso a un índice de nuestra Hashtable Java.



- ✓ Las tablas hash se suelen implementar sobre vectores de una dimensión, aunque se pueden hacer implementaciones multi-dimensionales basadas en varias claves. Como en el caso de los arrays, las tablas hash proveen tiempo constante de búsqueda promedio O(1),1 sin importar el número de elementos en la tabla. Sin embargo, en casos particularmente malos el tiempo de búsqueda puede llegar a O(n), es decir, en función del número de elementos.

- ✓ Comparada con otras estructuras de arrays asociadas, las tablas hash son más útiles cuando se almacenan grandes cantidades de información.

- ✓ Comparada con otras estructuras de arrays asociadas, las tablas hash son más útiles cuando se almacenan grandes cantidades de información.

<u>Ventajas e Inconvenientes</u>

Una tabla *hash* tiene como principal ventaja que el acceso a los datos suele ser muy rápido si se cumplen las siguientes condiciones:

- Una razón de ocupación no muy elevada (a partir del 75% de ocupación se producen demasiadas colisiones y la tabla se vuelve ineficiente).

- Una función resumen que distribuya uniformemente las claves. Si la función está mal diseñada, se producirán muchas colisiones.

Los inconvenientes de las tablas *hash* son:

- Necesidad de ampliar el espacio de la tabla si el volumen de datos almacenados crece. Se trata de una operación costosa.

- Dificultad para recorrer todos los elementos. Se suelen emplear listas para procesar la totalidad de los elementos.

- Desaprovechamiento de la memoria. Si se reserva espacio para todos los posibles elementos, se consume más memoria de la necesaria; se suele resolver reservando espacio únicamente para punteros a los elementos.

Tabla Hash en Java.

Creación de objeto del tipo Hashtable, en cuya creación hemos de indicar el tipo de dato para la clave (Key) y para el valor asociado (Value).

```
Hashtable<String, String> tablaH = new Hashtable<String, String>();
```

Para **Insertar** datos en la tabla lo haremos asociando Clave – Valor y con el método put(Key, value);

```
tablaH.put("Luis", "555555555D");
tablaH.put("Maria", "333333333D");
tablaH.put("Pepe", "444444444D");
```

Para **buscar** cualquier valor lo podemos hacer utilizando el método get(key) y con la clave asociada al valor.

```
String dni = tablaH.get("Luis");
```

Para **Eliminar** cualquier valor dado de alta en la tabla utilizaremos el método remove(key), cuyo parámetro de entrada será la clave asociada al valor. Hemos de tener en cuenta que éste método nos devolverá el valor eliminado, en el caso de que exista en la tabla la clave, en caso contrario nos devolverá Null.

```
String valorElimnado = tablaH.remove("Pepe");
```

Interface Enumeration

Las enumeraciones, definidas mediante la interfaz Enumeration, nos permiten consultar los elementos que contiene una colección de datos. Muchos métodos de clases Java que deben devolver múltiples valores, lo que hacen es devolvernos una enumeración que podremos consultar mediante los métodos que ofrece dicha interfaz.

En el caso de una Tabla Hash, podemos utilizar el método elements(), que nos devolverá un Enumeration del tipo de dato el cual esté declarado el 'valor' de la tabla hash.

Ejemplo:

```
Hashtable<String, String> tablaH = new Hashtable<String, String>();
```

En este caso tanto la clave como el valor asociado son de tipo String, pero hemos de tener en cuenta que el Enumeration debe ser del mismo tipo que el valor, por lo tanto la declaración del Enumeration tendría que ser de la siguiente manera:

```
Enumeration <String> valoresTH;
```

A continuación para asignar los valores de la Tabla Hash al Enumeration, utilizamos el método **elements()**

```
valoresTH = tablaH.elements();
```

Para poder recorrer cada uno de los elementos que tiene el objeto valoresTH, utilizaremos los metodos **hasMoreElement()**, el cual nos devuelve un boolean indicando si exiten mas elementos en el Enumeration (valoresTH), despues de ir obteniendo los valores en cada iteración con el método **nextElement():**

```
while(valoresTH.hasMoreElements()) {
	System.out.println(valoresTH.nextElement());
}
```