# Hash Table - Aplicaciones

Vilchis Salazar Jose Antonio

# hashCode

El método **hashCode** es un método definido dentro de la clase Object de Java, el cual devuelve un valor int que puede traducirse como la representación entera de dicha instancia de Object.

# hashCode

Durante la ejecución de un programa el valor hashCode particular a un objeto debe mantenerse **constante**. Sin embargo, el valor puede variar en **ejecuciones distintas** del mismo programa.

# hashCode

Objetos semánticamente **iguales** a través del método equals **deben** devolver el mismo valor hashCode.

# hashCode

Objetos semánticamente **distintos** a través del método equals **pueden** devolver el mismo valor hashCode.

# hashCode

```
1  public int hashCode() {
2      int h = 0;
3      Iterator<E> it = iterator();
4      while(it.hasNext()) {
5          // EXCLUSIVE OR
6          h = h ^ it.next().hashCode();
7          // 5-bit cyclic shift of composite code
8          h = (h << 5) | (h >>> 27);
9      }
10     return h;
11 }
```

# Funciones de compresión

La mayoría de las funciones hash tienen están dadas por una función que tiene como dominio una familia de objetos y los mapea al conjunto de los números enteros:
$f : A \to \mathbb{Z}$

# Funciones de compresión

Se debe usar una función de compresión, la cual restringe el rango de la función hash a un subconjunto finito de los números enteros, es decir, números entre cero y un natural k.
$g(f) : \mathbb{Z} \rightarrow K, K = \{0, 1, ..., k\}$

# Método de division

Una función de compresión muy simple es la de **división**:

$i \ mod \ N$

donde:

**i** es el valor devuelto por la **función hash original**. **N** es el tamaño de la tabla hash que se espera usar.

Esta función puede **dispersar uniformemente** los valores hash si N es primo.

# Método MAD

Otra función de compresión mas compleja es la de **Multiply, Add and Divide**:

$[ai + b \ mod \ p] \ mod \ N$

donde:

**i** es el valor devuelto por la **función hash original**. **N** es el tamaño de la tabla hash que se espera usar. **p** es un número primo mayor a N. **a** y **b** son numeros naturales aleatorios menores a N.

Esta función puede **reducir colisiones** los valores hash si N y p son escogidos de manera correcta.