

## Tema 9. La Tabla de Dispersión o *Hash*

### Objetivos y Bibliografía

El objetivo general de este tema es presentar la [Tabla de Dispersión](#) (*Hash*) como una Representación eficiente, No Lineal, de la EDA Diccionario. Específicamente, se introducirán los conceptos básicos que intervienen en la definición de una Tabla de Dispersión: la **función de dispersión** (*hashing*) que permite situar las Entradas de un Diccionario sobre un simple **array**, los **conflictos** que provoca la Búsqueda Dinámica de una Entrada sobre tal **array** así como los Métodos de Exploración Lineal y Cuadrático que permiten resolverlos y, en función de lo anterior, la descripción de las operaciones de Búsqueda Dinámica en la Tabla. Tras esta introducción se estará en disposición de plantear tanto la evaluación de una Tabla Hash dada, de su eficiencia medida como esfuerzo Medio de Comparación (eMC), como su implementación en Java, lo que conllevará el diseño de las clases **EntradaTablaHash** y **TablaHashCuadratica** ubicadas en el nuevo paquete **noLineales** del proyecto BlueJ **estructurasDeDatos**.

Finalmente, este tema concluirá con el estudio de las ventajas e inconvenientes que supone representar un Diccionario mediante una Tabla Hash o un Árbol Binario de Búsqueda, que a nivel experimental se puede refrendar utilizando las clases **TablaHashCuadratica** y **ABBDiccionario** como implementaciones alternativas del interfaz **Java Diccionario**.

Como bibliografía básica del tema se propone únicamente el libro de Weiss, M.A. **Estructuras de datos en Java**; específicamente, el apartado 7 de su Capítulo 6 recoge los aspectos concernientes a la definición de Tabla Hash, mientras que su Capítulo 19 contiene los dedicados a su implementación en Java.

### Índice

1. Implementación de las operaciones de un Diccionario en tiempo constante: ideas básicas
2. Función de dispersión (*hashing*): características. El método **hashCode** de la clase **Java Object**
3. Resolución de Colisiones: métodos de Exploración Lineal y Cuadrática
4. Implementación en Java de una Tabla Hash con Exploración Cuadrática: las clases **EntradaTablaHash** y **TablaHashCuadratica**

---

# Implementación en Java de una Tabla Hash con Exploración Cuadrática: las clases EntradaTablaHash y TablaHashCuadratica

## La clase EntradaTablaHash

```
package noLineales;
class EntradaTablaHash {
    Object dato; boolean activa;
    EntradaTablaHash(Object d) { this(d, true);}
    EntradaTablaHash(Object d, boolean a){ dato = d; activa = a;}
}
```

## La clase TablaHashCuadratica

```
package noLineales;
import modelos.*;import excepciones.*;
public class TablaHashCuadratica implements Diccionario{

    protected EntradaTablaHash elArray[];
    protected int capacidadDelArray, ocupadasDelArray, activasDelArray;
    protected int intentosInsercion, colisiones, inserciones;

    public TablaHashCuadratica(int capacidad){
        capacidadDelArray = siguientePrimo(2*capacidad);
        elArray = new EntradaTablaHash[capacidadDelArray];
        for ( int i = 0; i < elArray.length; i++ ) elArray[i] = null;
        ocupadasDelArray = activasDelArray = intentosInsercion = inserciones = 0;
    }
    protected static final int siguientePrimo(int n){
        if ( n % 2 == 0 ) n++;
        for ( ; !esPrimo(n); n += 2 ) ;
        return n;
    }
    protected static final boolean esPrimo (int n){
        for (int i = 3 ; i*i <= n; i += 2) if (n % i == 0) return false;
        return true;
    }
    public final boolean esVacia(){
        return ( this.activasDelArray == 0 );
    }
    public final int tamanyo(){
        return this.activasDelArray;
    }
    public final double eMC(){
        return ((double)intentosInsercion)/inserciones;
    }
    public final String toString(){
        String res = "*";
        for ( int i = 0; i < elArray.length; i++)
            if ( elArray[i] != null && elArray[i].activa )
                res += elArray[i].dato.toString()+"\n";
        return res;
    }
}
```

---

```

/**SII !esVacia() */
public final Object buscar(Object x) throws ElementoNoEncontrado{
    int posicionFinalX = buscarPos(x);
    if ( elArray[posicionFinalX] == null || !elArray[posicionFinalX].activa )
        throw new ElementoNoEncontrado("**al buscar: "+x+" no está");
    return elArray[posicionFinalX].dato;
}

/**SII !esVacia() */
public final void eliminar(Object x) throws ElementoNoEncontrado{
    int posicionFinalX = buscarPos(x);
    if ( elArray[posicionFinalX] == null || !elArray[posicionFinalX].activa )
        throw new ElementoNoEncontrado("**al eliminar: "+x+" no está");
    elArray[posicionFinalX].activa = false;
    activasDelArray--;
}

public final void insertar(Object x) throws ElementoDuplicado{
    int posicionFinalX = buscarPos(x);
    intentosInsercion += 1 + colisiones;
    inserciones++;
    if ( elArray[posicionFinalX] != null && elArray[posicionFinalX].activa )
        throw new ElementoDuplicado("**al insertar: "+x+" ya está");
    else {
        activasDelArray++;
        if ( elArray[posicionFinalX] == null ) ocupadasDelArray++;
        elArray[posicionFinalX] = new EntradaTablaHash(x);
    }
}

/** SII el grado de ocupación es MAYOR O IGUAL al 50%: REHASHING */
if ( ocupadasDelArray-1 >= elArray.length/2 ){
    EntradaTablaHash elArrayAntiguo[] = elArray;
    elArray = new EntradaTablaHash[siguientePrimo(2*elArrayAntiguo.length)];
    ocupadasDelArray = activasDelArray = 0;
    inserciones = 0;
    for ( int i = 0; i < elArrayAntiguo.length; i++)
        if ( elArrayAntiguo[i] != null && elArrayAntiguo[i].activa )
            insertar(elArrayAntiguo[i].dato);
}

protected final int buscarPos(Object x){
    int indiceHashX = x.hashCode() % elArray.length;
    if ( indiceHashX < 0 ) indiceHashX += elArray.length;
    colisiones = 0;
    while ( elArray[indiceHashX] != null && !(elArray[indiceHashX].dato).equals(x) ){
        indiceHashX += 2 * ++ colisiones - 1 ;
        if ( indiceHashX >= elArray.length ) indiceHashX -= elArray.length;
    }
    return indiceHashX;
}

public final Object[] toArray(){
    Object res[] = new Object[activas];
    int indiceRes = 0;
    for ( int i = 0; i < elArray.length; i++ )

```

---

```
        if ( elArray[i] != null && elArray[i].activa ){
            res[indiceRes]= elArray[i].dato;
            indiceRes++;
        }
    return res;
}
}
```