EDA (E.T.S. de Ingeniería Informática)
Curso 2015-2016
*Práctica 3. Uso de una Tabla de Dispersión como implementación eficiente de un Map de Imágenes*
Parte I: Implementación de la clase `TablaHash`

## Objetivos formativos y trabajo previo

Una vez realizada esta práctica el alumno debe ser capaz de implementar en Java una Tabla de Dispersión con encadenamiento separado (Hashing enlazado); en concreto:

- las operaciones básicas para calcular su factor de carga, obtener una lista de sus claves y hacer *rehashing* cuando la eficiencia de la tabla se vea comprometida.
- las operaciones para comprobar la efectividad de las funciones de `hashCode` definidas sobre las claves: cálculo de la desviación típica de las longitudes de las cubetas y obtención del histograma de ocupación.

También deberá ser capaz de mostrar gráficamente un histograma de ocupación de la tabla y razonar sobre la efectividad de la dispersión en base a la información obtenida sobre la desviación típica y del histograma, en concreto observando el número de cubetas que tienen uno o dos elementos.

Además se reforzarán los objetivos transversales a todas las prácticas de la asignatura y que están relacionados con la calidad de los programas desarrollados: utilización de paquetes para facilitar la organización, reutilización y mantenimiento del software, utilización de los mecanismos de herencia y genericidad que proporciona el lenguaje, elaboración de juegos de prueba para validar código y generación de documentación asociada al código desarrollado.

Para aprovechar al máximo la sesión en el laboratorio, antes se debe realizar una lectura comprensiva de este boletín y del código de las clases que se proporcionan a través de PoliformaT.

# Implementación de una Tabla de Dispersión

En las clases de teoría se ha estudiado la *Tabla de Dispersión* como una implementación eficiente de un *Map*. En concreto, se ha diseñado la clase `EntradaHash`, que representa un par clave-valor en la tabla y parcialmente la clase `TablaHash` que implementa el hashing enlazadado mediante un array en el que cada cubeta se representa mediante una `ListaConPI`. En esta práctica se propone que el alumno complete esta clase con métodos para calcular el factor de carga de la tabla, obtener una `ListaConPI` con sus claves y hacer *rehashing*. También debe añadir métodos que permitan comprobar la efectividad de las funciones de *hashCode* definidas sobre sus claves.

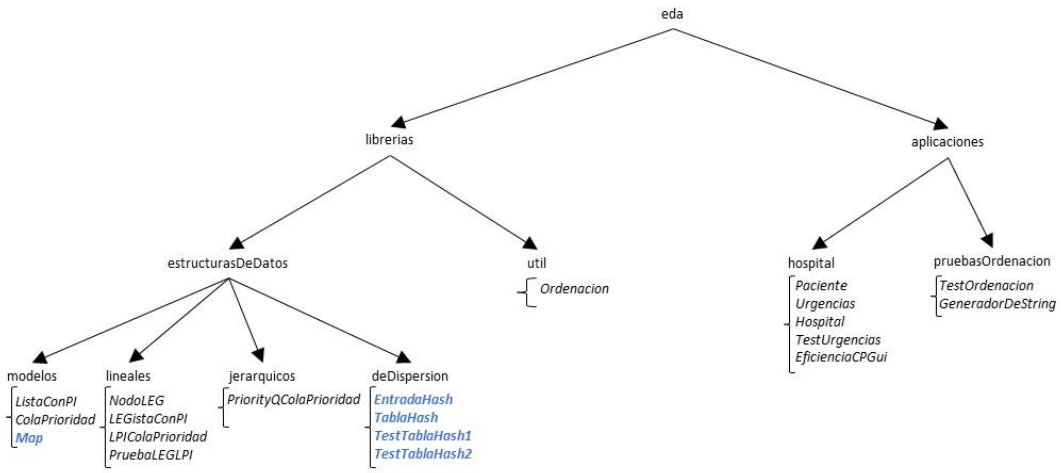
## Actividad 1: ubicación de las clases `Map`, `EntradaHash` y `TablaHash`

El alumno deberá realizar las siguientes acciones:

- Agregar al paquete *librerias.estructurasDeDatos.modelos* el interfaz `Map` disponible en *PoliformaT*. Recuérdese que *Map* es un modelo orientado a la búsqueda por clave en una colección de datos.

- Crear un nuevo paquete denominado *librerias.estructurasDeDatos.deDispersion* y ubicar en él las clases `TablaHash`, `EntradaHash` y las de prueba `TestTablaHash1` y `TestTablaHash2`, también disponibles en *PoliformaT*.

Al acabar, la estructura de proyectos, paquetes y ficheros debe ser la siguiente:



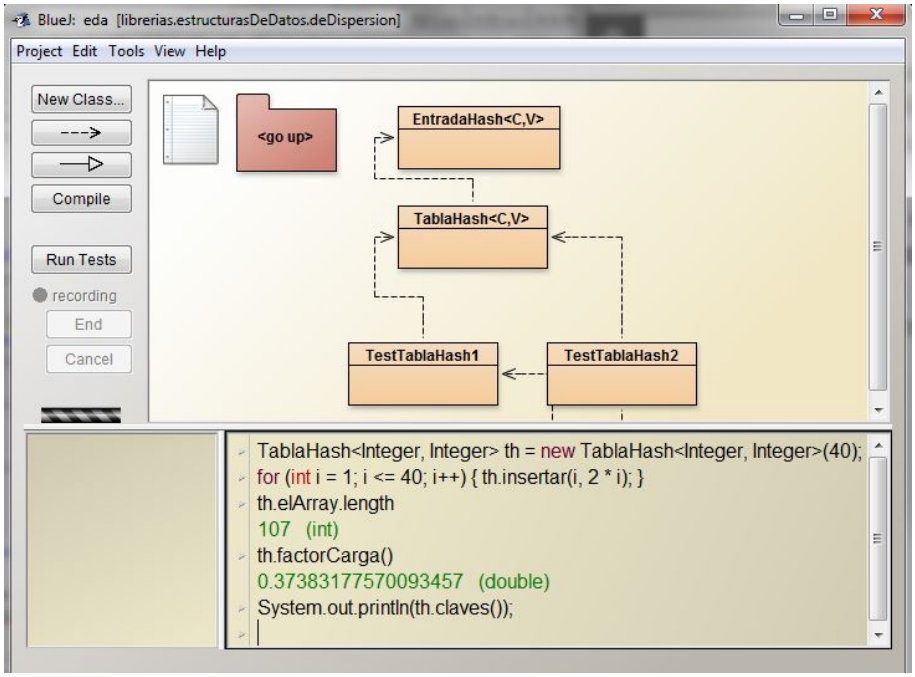
## Actividad 2: los métodos `factorCarga()`, `claves()` y `rehashing()`

El alumno deberá implementar en la clase `TablaHash` los siguientes métodos:

- `double factorCarga()`: devuelve el factor de carga de la tabla.
- `ListaConPI<C> claves()`: devuelve una *ListaConPI* que contiene las claves del *Map* actual.
- `void rehashing()`: incrementa la capacidad del array para mantener el rendimiento de la *Tabla Hash*. La nueva capacidad del array deberá ser igual al siguiente número primo del doble de la capacidad actual.

Una vez resueltos los errores de compilación y revisados los errores de estilo (opción *checkstyle* de BlueJ), se aconseja comprobar que el código desarrollado es correcto utilizando la clase `TestTablaHash1`.

En la figura siguiente se puede ver el resultado de la ejecución en el *CodePad* de BlueJ de las instrucciones para crear una `TablaHash` en la que claves y valores son de tipo `Integer`; añadir los valores del 1 al 40 como claves (asociando valores que sean el doble) y, finalmente, mostrar por pantalla el número de cubetas de la tabla (o tamaño del array), el valor del factor de carga y la lista de claves.




A continuación se debe ejecutar en el *CodePad* de BlueJ las instrucciones para:

- Crear una `TablaHash` para almacenar los primeros 100 números naturales con sus raíces cuadradas; piensa cuál debe ser el tipo de datos de los valores de la tabla.
- Mostrar por pantalla el número de cubetas de la tabla, el valor del factor de carga y la lista de claves.

## Actividad 3: los métodos `desviacionTipica()` e `histograma()`

El alumno deberá implementar en la clase `TablaHash` los siguientes métodos:

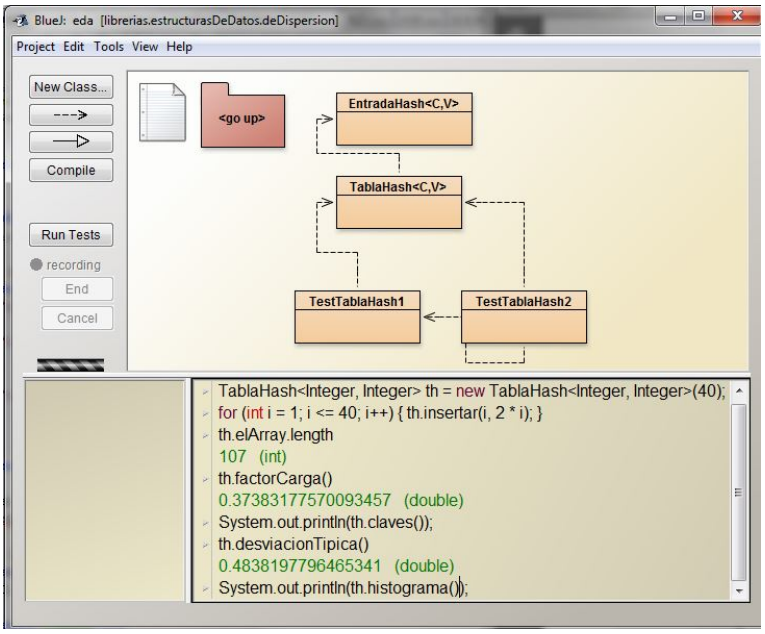
- `double desviacionTipica()`: devuelve la desviación típica de las longitudes de las cubetas. Si denotamos por $l_i$ la longitud de la cubeta i, la desviación típica $\sigma$ se define como sigue, asumiendo que hay $N$ cubetas y que la longitud media de las cubetas es $\bar{l}$:

$$\sigma = \sqrt{\frac{\sum (l_i - \bar{l})^2}{N}}$$

- `String histograma()`: devuelve un String que representa el histograma de ocupación de las cubetas de la tabla, donde cada línea consta de dos valores separados por un tabulador: longitud de cubeta y número de cubetas de esa longitud. En dicho histograma, las longitudes de cuberta a considerar son 0, 1, ... 8 y mayores o iguales que 9.

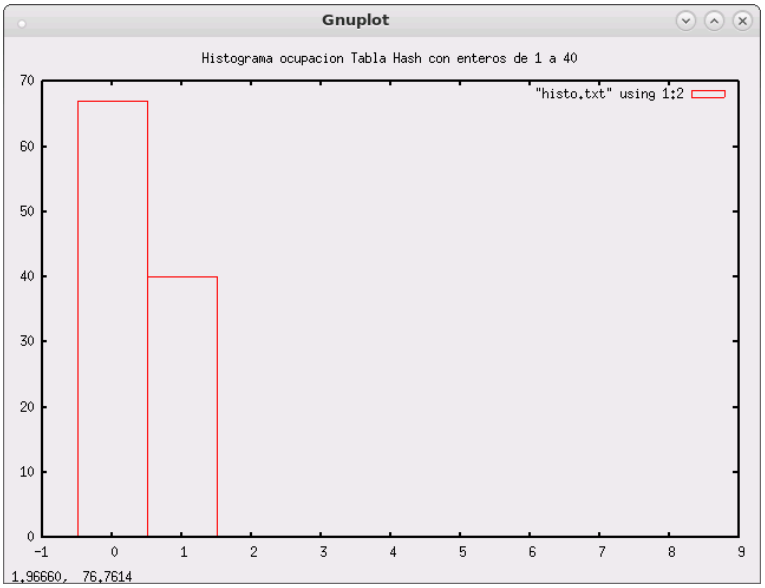
Una vez resueltos los errores de compilación y revisados los errores de estilo (opción *checkstyle* de BlueJ), se aconseja comprobar que el código desarrollado es correcto utilizando la clase `TestTablaHash2`.

En la figura siguiente se puede ver la ejecución en el *CodePad* de BlueJ de las instrucciones anteriores incluyendo las llamadas a los métodos `desviacionTipica()` e `histograma()`.




Para poder mostrar gráficamente el histograma se guardará el resultado de la llamada al método en un fichero de nombre `histo.txt`, utilizando la opción correspondiente de la ventana de ejecución de BlueJ. Seguidamente, se usará la herramienta *gnuplot*:

```
gnuplot> plot "histo.txt" using 1:2 with boxes
```




En el histograma puede comprobarse la efectividad de la dispersión (ver figura): todavía el 62 % de las cubetas están vacías (factor de carga del 0.37) y el resto (40) contienen un único elemento.

A continuación se debe ejecutar en el *CodePad* de BlueJ las instrucciones para:

- Crear una `TablaHash` para almacenar el primer millón de números naturales con sus raíces cuadradas.
- Mostrar por pantalla el número de cubetas de la tabla, el valor del factor de carga, y la desviación típica.
- Mostrar por pantalla y guardar en un fichero el histograma de ocupación de la tabla.
- Obtener gráficamente el histograma utilizando gnuplot y comprobar la efectividad de la dispersión.