Práctica 1

Parcial 1. Temas 1, 2 y 3

Implementación de una tabla hash

Objetivos

Trabajar en **parejas** para generar código original en Java que implemente una tabla hash a partir de las especificaciones.

Descripción

Se pretende implementar una tabla Hash de codificación propia. Para ello se codificarán dos clases genéricas: **Hash** y **Celda**.

La clase **Hash** representará la propia tabla. Utilizará ***direccionamiento abierto*** con ***doble hashing*** como estrategia para la resolución de colisiones. En cada posición de la tabla se almacenará un objeto de la clase **Celda**. Cada celda (ver figura 1) almacena la **clave** y el **valor** del elemento, además del **estado** de esa posición (borrado = -1, vacío = 0, ocupado = 1).

Figura 1. Esquema de la estructura utilizada para representar la tabla hash

Tanto la capacidad de la tabla como el umbral máximo del factor de carga serán datos configurables. De no especificarse, se tomarán por defecto N=7 y el 80% como α máximo.

La función hash viene definida de la siguiente manera:

- $H(clave, colisiones) = H_1(clave) + H_2(clave, colisiones)$
- $H_1(clave) = clave \bmod N$
- $H_2(clave, colisiones) = colisiones * (7 - (clave \bmod 7))$

Antes de insertar un dato, debe comprobarse que no se vaya a exceder el umbral de α. De ser así, la tabla se debe redimensionar antes de la inserción. La estrategia en este caso será la estudiada en el aula:

1. Duplicar el tamaño de la tabla original: NT = 2N
2. Buscar el siguiente número primo por encima de NT.
3. Recorrer desde el principio hasta el final la tabla original, insertando los datos que se vayan encontrando en la nueva tabla hash.
4. Insertar el dato que ha forzado el redimensionamiento.
5. Sustituir la tabla original por la nueva tabla ampliada.

Trabajo a desarrollar

Los estudiantes deberán completar tres clases:

- Hash.java
- Celda.java
- Pruebas.java

Es requisito necesario respetar la especificación detallada en el diagrama de clases adjunto (figura 2). Por tanto **no pueden modificarse** ni los nombres de los métodos ni sus cabeceras. De considerar estrictamente necesario añadir algún método extra, deberá justificarse en la documentación de la actividad.

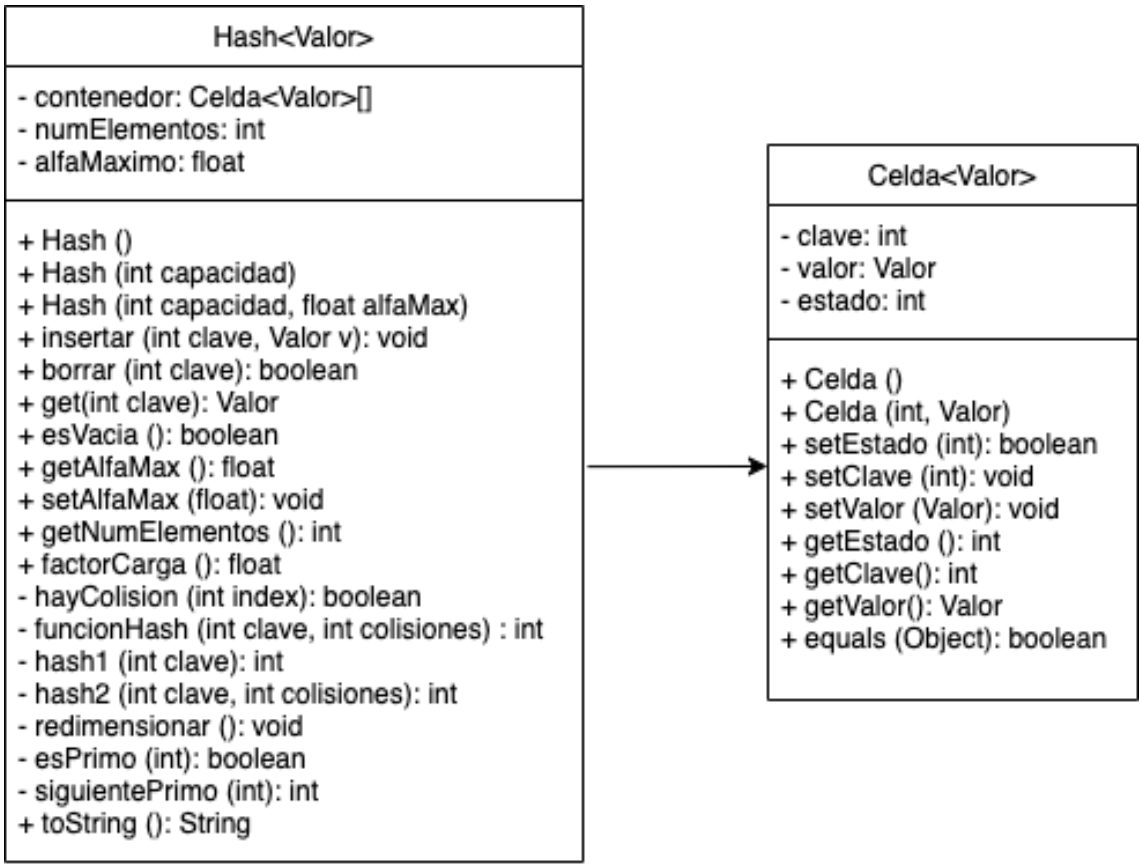

Figura 2. Diagrama de clases de la actividad

Algunos aspectos a tener en cuenta:

- Clase Hash:
  - El método **borrar(int)** devuelve true si logra eliminar el elemento. False en caso contrario.
  - **get(int)** devuelve *null* si el dato asociado a la clave indicada no está en la tabla.
  - **insertar (int, Valor)** no permite insertar duplicados. Si el elemento ya estuviese en la tabla, no lo inserta de nuevo.
  - **setAlfaMax (float)** no debe permitir asignar valores menores que 0.0 ni mayores que 1.0. Tampoco puede permitir asignar un valor inferior al factor de carga actual.
  - **toString()** "dibuja" la tabla por pantalla. Debería mostrar la celda de cada índice de la tabla: estado-clave-valor.

- Clase Celda:
  - **setEstado(int): boolean** sólo puede asignar los valores -1, 0 ó 1. En caso de que el parámetro no fuese ninguno de esos tres, no haría la asignación, y devolvería falso.

Es **condición necesaria** para superar la actividad:

- Trabajo en equipo.
- Código y documentación originales.
- Código que compile.
- Respetar el diseño de clases especificado.
- Respetar los formatos de entrega (número, nombres y formatos de los ficheros y carpetas, organización de los mismos en el comprimido, etc.).
- No abortar el flujo lógico de un bucle con sentencias break, continue o return.

Adicionalmente:

- Incluir comentarios en el código.
- En el caso de adjuntar fragmentos de código en la documentación, no debe hacerse como imagen, sino como texto.
- Se valorará el uso nombres de variables explicativos, código bien tabulado, etc.
- Utilizar correctamente bucles de recorrido fijo/variable según el caso lo requiera.

Normas de entrega

El trabajo a entregar debe ser original y desarrollado por los propios estudiantes. **No se permite entregar documentación ni código no realizado, aunque sea en parte, por los estudiantes.**

La entrega constará de tres (3) ficheros .java y un (1) fichero .pdf, que deberán organizarse y adjuntar de acuerdo a las siguientes normas:

1. Crear una carpeta raíz para almacenar todos los archivos. Tendrá un nombre de acuerdo al siguiente formato:

   "TPA-M2X. P1. GYY",

   donde X e YY serán dígitos que permitirán identificar al equipo de trabajo dentro del grupo de la asignatura. Por ejemplo, el equipo 12 del grupo M23 de TPA crearía la carpeta "TPA-M23. P1. G12".

2. Dentro de la anterior carpeta, crear dos subcarpetas:
   a. "src": contendrá los ficheros .java
   b. "doc": contendrá el fichero .pdf

3. En el documento PDF se describirán los detalles del desarrollo de la práctica:
   a. Justificará las decisiones tomada.
   b. Describirá las principales funciones/métodos (insertar, borrar, get, redimensionar, etc.).
   c. Detallará el plan de pruebas diseñado (idealmente de manera esquemática con una tabla).
   d. Comentará las dificultades encontradas y cómo se resolvieron.

   Además, el documento deberá incluir portada, índice, bibliografía, etc.

   El nombre del fichero seguirá el siguiente formato:

   "TPA-M2X. P1. GYY.**PDF**"

   En el ejemplo anterior: "TPA-M23. P1. G12.**PDF**"

4. Comprimir la carpeta original (apartado 1) en un fichero **ZIP** (no RAR, no 7Z, etc.). El nombre deberá seguir el formato:

   "TPA-M2X. P1. GYY.**ZIP**"

   En el ejemplo anterior: "TPA-M23. P1. G12.**ZIP**"

Este fichero ZIP será el que se adjunte a la tarea correspondiente en el Campus Virtual, **antes** de las **23:55h** del día fijado como fecha límite de entrega.