UNIDAD TEMÁTICA 6 – DICCIONARIOS, MAPAS & HASHING

PRACTICOS DOMICILIARIOS INDIVIDUALES - 7

EJERCICIO 1

Investiga cómo está implementado el método hashCode en Java para objetos de la clase Object. Luego investiga cómo se implementa el mismo para las clases Integer y String. Explica por qué la implementación es diferente.

EJERCICIO 2

Investiga y diagrama cómo son las estructuras internas de un HashMap. Con lo investigado en el ejercicio anterior, diagrama el estado de las estructuras luego de insertar las siguientes strings:

- Hola
- HolaMundo
- HashMap
- Colecciones

EJERCICIO 3

Implementa los métodos equals y hashCode para la siguiente clase:

```
public class Alumno : Object {
    private int ID;
    private String fullName;
    private String email;

    ...
}
```

Indica qué características deben tener las implementaciones de los métodos solicitados para mantener el contrato general para el método hashCode.

Ejercicio 1:
La principal diferencia del método HashCode de Object con el de String e Integer, es el valor que toma en cuenta para aplicarle la función Hash.
En el caso de ser un objeto de la clase String, el método tomará la cadena almacenada, mientras que en Integer tomará el número.
La diferencia es que en la clase Object no toma un elemento en particular, ya que no hay ninguno que esté determinado como más relevante. En este caso, para hacer la función Hash, se utiliza el ID de referencia en memoria.

Ejercicio 2:
Las colisiones en los HashMap de Java se resuelven con un encadenamiento simple, es decir, agregando los elementos en una LinkedList en el espacio de la tabla que corresponde
Dado que la función Hash de Java es:
s[0]*31^(n-1) + s[1]*31^(n-2) + ... + s[n-1]
Todas las strings dadas darían un hashCode distinto, por ende, no pasarían colisiones y quedarían distribuidas

Ejercicio 3:
El método hashcode debe devolver siempre el mismo hashCode al darle el mismo objeto si no sufrió cambios. También debe devolver el mismo hashCode para dos elementos que devuelvan true al ser pasados como parámetro en equals().
El método hash debe distribuir de la forma más eficiente posible los elementos, así la tabla no consigue una carga alta.