TSB - Trabajo Práctico Único [Etapa 1]

Tablas Hash (Implementación por Direccionamiento Abierto)

Para la primera entrega del Trabajo Práctico Único (Ciclo 2022), se solicita hacer una implementación rigurosa de la estructura de ***tabla hash con resolución de colisiones por direccionamiento abierto***, de tal forma que esa implementación pase los tests previstos en la JUnit que se anexa con este enunciado.

1. **Consignas y sugerencias**.
    a. Siguiendo el modelo de implementación del concepto de tabla hash con listas de desborde que se presentó en la Ficha 09, se solicita ahora implementar una clase *TSBHashTableDA* (DA por *Direccionamiento Abierto*), que modele una Tabla Hash pero con estrategia de ***direccionamiento abierto*** para la resolución de colisiones.
    b. La clase debe ser implementada en forma rigurosa, siguiendo el modelo ya presentado para la clase *TSBHashtable* de la Ficha 09. Esto implica:
        - ✓ Implementar la interface *Map<K, V>* y desde ella, los mismos métodos que sean necesarios. Para aliviar la tarea, los estudiantes pueden derivar la clase *AbstractMap*, que ya implementa en forma concreta varios de los métodos de la interface *Map*, pero dejando claro lo siguiente: los estudiantes obligatoriamente deberán dar sus propias implementaciones de los métodos que se testean en la Junit provista por la Cátedra en forma anexa a este enunciado.
        - ✓ La clase debe controlar que ninguno de los dos valores *key* y *value* de cada par almacenado sea *null* (emulando así lo que hace la clase *Hashtable* nativa de Java). Y por otra parte, la clase implementada no debe tener en cuenta el control *thread-safe* (en forma similar a lo que hace *HashMap*).
        - ✓ Definir dentro de la clase *TSBHashTableDA* una clase interna *Entry* que implemente la interface *Map.Entry<K, V>* para representar a cada par que se almacene en la tabla.
        - ✓ Definir dentro de la clase *TSBHashTableDA* las tres clases internas para gestionar las *vistas stateless* de *claves*, de *valores* y de *pares de la tabla*, incluyendo a su vez en ellas las clases internas para representar a los iteradores asociados a cada vista.
        - ✓ Redefinir en la clase *TSBHashTableDA* los métodos *equals()*, *hashCode()*, *clone()* y *toString()* que se heredan desde *Object*. Para *equals()* y *hashCode()* tener especial cuidado de respetar las convenciones de Java en cuanto a los principios mutuos que esos métodos deben cumplir.
        - ✓ Definir en la clase *TSBHashTableDA* los métodos *rehash()* y *contains(value)* que no vienen especificados por *Map*, pero son especialmente propios de la clase (emulando a *java.util.Hashtable*).
        - ✓ La clase implementada, debe testearse contra la JUnit que se anexa. El primer parámetro de evaluación será comprobar con cuanta precisión la clase pasa esos tests. Y luego los docentes harán una comprobación rápida del código fuente para chequear detalles de buenas prácticas y elementos lógicos.

- ✓ Cada grupo deberá entregar un proyecto con la clase que se haya diseñado, y que integre a la JUnit como parte del mismo. No es necesario incluir un *main()* en el proyecto.
- ✓ La calificación que el grupo obtenga en esta primera etapa, será promediada en forma ponderada con la calificación que obtenga en la segunda etapa (entrega definitiva), en la cual se solicitará aplicar la tabla diseñada para la resolución de un caso de aplicación real.
- ✓ Cualquier duda o comentario debe ser posteado en el foro general del aula virtual de la asignatura, o directamente planteado para su discusión en las horas de clase de cada curso.
- ✓ La entrega de esta primera etapa podrá hacerse en forma normal desde el mismo momento de publicación de este enunciado, y hasta el sábado 08 de octubre, pero también podrán entregarlo sin penalización por entrega tardía hasta el sábado 15 de octubre. Luego de esa fecha, se admitirá la entrega con penalización por entrega tardía, hasta el día martes 18 de octubre.