UT6 - Diccionarios, Mapas y API Colecciones

Mapas:

- Permiten almacenar elementos utilizando claves únicas y recuperarlos rápidamente.
- Almacenan pares clave-valor (k,v), donde cada clave es única.
- Operaciones principales: `tamaño()`, `estaVacio()`, `recuperar(k)`, `poner(k,v)`, `eliminar(k)`, `claves()`, `valores()`, `elementos()`.

Diccionarios:

- Almacenan pares clave-valor similares a los mapas, pero permiten múltiples entradas con la misma clave.
- Tipos de diccionarios: ordenados y desordenados.
- Operaciones principales: `tamaño()`, `estaVacio()`, `buscar(k)`, `buscarTodos(k)`, `insertar(k,v)`, `eliminar(e)`, `elementos()`.

API de Colecciones en Java:

- Las colecciones agrupan múltiples elementos en una sola unidad para almacenar, recuperar, manipular y comunicar datos agregados.
- Componentes del framework de colecciones: interfaces, implementaciones y algoritmos.
- Beneficios del uso de colecciones: reducción del esfuerzo de programación, incremento de la velocidad y calidad del programa, promoción de la reutilización del software.

Implementaciones de la API de Colecciones:

- `Set`: `HashSet`, `Tre9yeSet`, `LinkedHashSet`.
- `List`: `ArrayList`, `LinkedList`.
- `Map`: `HashMap`, `TreeMap`, `LinkedHashMap`.

Interfaz `Collection`:

- Métodos: `size()`, `isEmpty()`, `contains(Object element)`, `add(E element)`, `remove(Object element)`, `iterator()`.
- Recorridos: `for-each`, `Iterator`, operaciones agregadas y de arrays.

Interfaz `Set`:

- Colección que no puede tener elementos duplicados, modela la abstracción matemática de un conjunto.
- Implementaciones: `HashSet`, `TreeSet`, `LinkedHashSet`.
- Operaciones "bulk": unión, intersección, diferencia de conjuntos.

Interfaz `List`:

- Colección ordenada que puede contener elementos duplicados.
- Operaciones adicionales: acceso por posición (`get`, `set`, `add`, `addAll`, `remove`), búsqueda (`indexOf`, `lastIndexOf`), iteración (`listIterator`), vista de subrango (`subList`).
- Implementaciones: `ArrayList`, `LinkedList`.

Interfaz `Map`:

- Mapea claves a valores, con operaciones básicas como `put`, `get`, `remove`, `containsKey`, `containsValue`, `size`, `empty`.
- Operaciones "bulk": `putAll`, `clear`.
- Vistas de colección: `keySet`, `entrySet`, `values`.

Vistas de Colección:

- Permiten iterar sobre un `Map` como una colección de claves, valores o entradas.

Contrato entre hashCode() y equals():

- `hashCode()` debe devolver el mismo entero si no se modifica la información utilizada en las comparaciones de igualdad.
- Si dos objetos son iguales según `equals()`, deben tener el mismo `hashCode()`.
- No es necesario que objetos desiguales tengan diferentes `hashCode()`.

Implementaciones estándar de hashCode():

- Para tipos como `int`, `String`, `Float`, `Object`.
- Ejemplo de definición de `hashCode()` para números de cédula.

Uso de HashMap:

- Estructura interna de `HashMap` y parámetros de configuración.
- Ejemplo de programa para medir tiempos de inserción con diferentes valores de parámetros.

Relación entre hashCode y equals:

- Función de estos métodos en la inserción o búsqueda en un `HashMap`.
- Parámetros posibles para el constructor de `HashMap`.