Reporte Examen

Alumno: Kevin Uriel Dulche Jaime

Matricula: 2213026201

Sección 1: Tablas hash

Catálogo de productos

Crear un programa que gestione un catálogo de productos utilizando una tabla HashMap. El programa debe permitir al usuario:

- Agregar nuevos productos al catálogo (con nombre, código, precio y descripción).
- Buscar un producto por nombre, código o descripción.
- Editar la información de un producto.
- Eliminar un producto del catálogo.

- Mostrar una lista de todos los productos del catálogo ordenados por diferentes criterios (nombre, código, precio).

El programa implementa un catálogo de productos utilizando una tabla hash en Java. A continuación, se detalla cómo funciona:

- Clase Producto: Define la estructura de un producto con atributos como código, nombre, precio y descripción.
- Inicialización del catálogo: Se crea un HashMap llamado "catalogo" para almacenar objetos de tipo Producto. Los productos se agregan al catálogo utilizando su código como clave y un objeto Producto como valor.
- Mostrar el catálogo de productos: Se recorre el HashMap y se imprimen en la consola todos los productos almacenados en el catálogo.
- Búsqueda de productos: El programa busca productos en el catálogo según el criterio especificado (código, nombre o descripción) y muestra los productos encontrados.
- Edición de productos: Se edita la información de un producto en el catálogo reemplazando el objeto Producto asociado con la clave correspondiente en el HashMap.
- Eliminación de productos: Se elimina un producto del catálogo utilizando su código como clave en el HashMap.

El código demuestra el uso de la estructura de datos HashMap para gestionar un catálogo de productos de manera eficiente, permitiendo operaciones como agregar, buscar, editar y eliminar productos de manera fácil y rápida.

Sección 2: Montículo (Heap)

Pregunta b (25 puntos):

Crea un programa que une dos montículos en uno solo manteniendo la propiedad de orden. Imprime el nuevo árbol por niveles.

- **Método mergeHeaps:** Este método recibe dos montículos como parámetros y devuelve un nuevo montículo que es la unión de ambos manteniendo la propiedad de orden. Se crea un nuevo montículo vacío (**unionHeap**) y se agregan todos los elementos de los montículos originales (**heap1** y **heap2**) al nuevo montículo. Dado que **PriorityQueue** implementa un montículo mínimo por defecto, la propiedad de orden se mantiene automáticamente.
- **Método imprimirPorNivel:** Este método imprime los elementos del montículo por niveles. Se van extrayendo los elementos del montículo uno por uno hasta que esté vacío, imprimiendo cada elemento a medida que se extrae.
- **Función principal (main):** En la función principal, se crean dos montículos (**heap1** y **heap2**) y se les agregan algunos elementos como ejemplo. Luego, se imprime cada montículo por niveles utilizando el método **imprimirPorNivel**. Después, se unen los dos montículos utilizando el método **mergeHeaps**, y el resultado se imprime por niveles también.

El código demuestra cómo se puede unir eficientemente dos montículos en uno solo manteniendo la propiedad de orden, lo que es útil en aplicaciones que requieren la fusión de conjuntos ordenados, como por ejemplo en algoritmos de ordenamiento externo. La impresión por niveles facilita la visualización del árbol resultante de manera comprensible.