UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ciencias

Estructuras de Datos

Práctica 14: Mapas

Profesora:
Amparo López Gaona
Ayudante: Adrián Aguilera Moreno
Ayudante de Laboratorio: Kevin Jair Torres Valencia

Objetivos

- Implementar un mapa utilizando un árbol binario de búsqueda (BST) como estructura subyacente.
- Comprender el funcionamiento de las operaciones básicas de un mapa: inserción, búsqueda, eliminación, recorrido, consulta de tamaño y estado
- Implementar un iterador para recorrer los elementos del mapa de manera eficiente.

Introducción

¿Qué es un Mapa?

Un **mapa** (o diccionario) es una estructura de datos que almacena pares clave-valor, donde cada clave es única y permite acceder a su valor asociado. Es similar a un diccionario real, donde una palabra (clave) tiene una definición (valor).

Implementación con Árbol Binario de Búsqueda (BST)

Un BST es una estructura de datos en la que cada nodo tiene un valor (clave) y dos subárboles (izquierdo y derecho), donde:

- Todos los valores en el subárbol izquierdo son **menores** que la clave del nodo actual.
- Todos los valores en el subárbol derecho son **mayores** que la clave del nodo actual.

Esta propiedad permite realizar operaciones como inserción, búsqueda y eliminación en tiempo promedio $O(\log n)$.

Desarrollo

En esta práctica, deberán completar la implementación de un Mapa genérico, siguiendo el principio de separación de interfaz e implementación. Se te proporcionarán algunos archivos base que deberás extender.

Para ello tendrán que implementar los métodos en las siguientes clases:

- `MapaInterface<K, V>`: Define las operaciones básicas que debe implementar el mapa (agregar, obtener, eliminar, etc.).
- `Mapa<K, V>`: Es la implementación de la estructura.
  - Contiene la lógica del árbol binario de búsqueda.
  - Usa una clase interna NodoArbol para almacenar los pares clave-valor.
- `Iterador<T>`: Define los métodos hasNext() y next() para recorrer elementos.
- `IteradorClaveValor<K, V>`: Recorre el árbol en orden (izquierdo-raíz-derecho) usando una pila.
- `IteradorValores<K, V>`: Adapta `IteradorClaveValor` para devolver solo los valores.
- `MainMapa`: Demuestra el uso del mapa con operaciones básicas.

Consideraciones para su implementación en la clase `Mapa`:

- Clase `NodoArbol`: Se encuentra como clase interna de la implementación. En ella se almacena:
  - `clave`: es unica y comparable.
  - `valor`: asociado a la clave.
  - Referencias a los nodos izquierda y derecho.
- Operaciones principales:
  - `agregar(clave, valor)`: Inserta un nuevo par clave-valor manteniendo el orden del BST.
  - `obtener(clave)`: Busca un valor por su clave (retorna `null` si no existe).
  - `eliminar(clave)`: Elimina un nodo y reajusta el árbol.
  - `existeClave(clave) / existeValor(valor)`: Verifica si una clave o valor existe en el mapa.

Consideraciones para su implementación para los `Iteradores`:

- `IteradorClaveValor`: usa una pila para recorrer el árbol en orden (inorden, es decir, izquierda-Raíz-Derecha).
- `IteradorValores`: es una adaptación que solo devuelve los valores (no las claves).

Finalmente, para poder probar su implementación solo basta con ejecutar la clase `MainMapa`, ya **NO** es necesario que realicen las pruebas. Ver Figura 1 donde se muestra la salida esperada.

```
Pokémon registrados: 5

¿Quién es el Pokémon #25? Pikachu

¿Existe el Pokémon #4? true
¿Existe el Pokémon #5? false

¿Existe algún Pokémon llamado 'Pikachu'? true

=== Pokédex Completa antes de las operaciones ===
#1: Bulbasaur
#4: Charmander
#7: Squirtle
#25: Pikachu
#150: Mewtwo

=== Nombres de Pokémons ===
Bulbasaur
Charmander
Squirtle
Pikachu
Mewtwo

Eliminando al Pokémon #4...
¿Se eliminó correctamente? true
Nuevo total: 4 Pokémons

Intentando obtener al Pokémon eliminado (#4): null

=== Pokédex despues de eliminar ===
#1: Bulbasaur
#7: Squirtle
#25: Pikachu
#150: Mewtwo

Vaciar la Pokédex...
¿Está vacía la Pokédex? true
Pokémon registrados: 0

=== Pokédex despues de vaciar ===
```

Figure 1: Ejecución de la implementación de la estructura Mapa

Formato de Entrega

1. Las prácticas se entregarán en parejas.
2. NO incluir los archivos .class dentro de la carpeta.
3. Los archivos de código fuente deben estar documentados.
4. Se pueden discutir y resolver dudas entre los integrantes del grupo. Pero cualquier práctica plagiada total o parcialmente será penalizada con cero para los involucrados.
5. La práctica se debe subir al Github Classroom correspondiente.
6. La entrega en classroom debe contener el link HTTPS y SSH de su repositorio y es lo único que se debe entregar.
7. El horario y día de entrega se acordará en la clase de laboratorio y no deberá sobrepasar 2 clases de laboratorio.