Guatemala 04 de marzo de 2024
Adrián López - 231361
Luis Fernando Mendoza

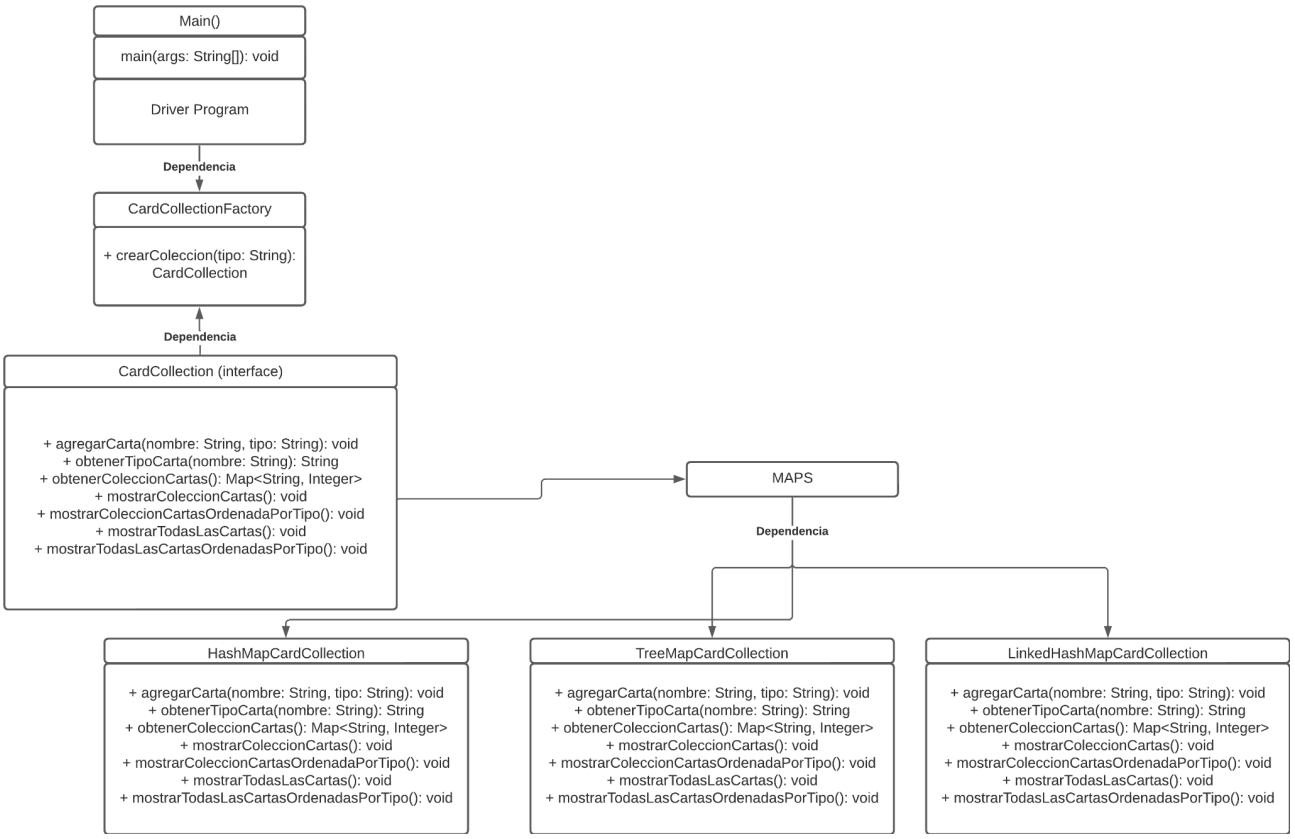
Algoritmos y Estructuras de Datos
HDT 6: Operaciones con Mapas

Objetivos:
a. Utilización de Java Collection Framework: uso de la interface MAP y sus (tres) implementaciones.
b. Uso de algoritmos polimórficos proporcionados por el Java Collection Framework.
c. Control de versiones del programa.

"Debe de realizar un programa que utilice el diseño Factory para la implementación de MAP que debe de leer el archivo de texto "cards_desc.txt" que contiene una lista de nombres de cartas y si dicha carta es monstruo, trampa o hechizo. Dichas cartas se agregan a un Mapa que contiene todas las cartas disponibles. Luego, debe de permitir al usuario el:

- *Agregar una carta a la colección.*
- *Mostrar el tipo de una carta específica.*
- *Mostrar el nombre, tipo y cantidad de cada carta que el usuario tiene en su colección.*
- *Mostrar el nombre, tipo y cantidad de cada carta que el usuario tiene en su colección, ordenadas por tipo.*
- *Mostrar el nombre y tipo de todas las cartas existentes.*
- *Mostrar el nombre y tipo de todas las cartas existentes, ordenadas por tipo."*

a. El programa utiliza el patrón de diseño Factory para seleccionar la implementación del MAP tal como solicitado. Para comprobarlo, se ha realizado el diagrama UML basado en el diseño de Factory antes de realizar la implementación del programa:



b. *Evidencia de desarrollo:*
https://github.com/adirnnn/HDT6-Map-Operations.git

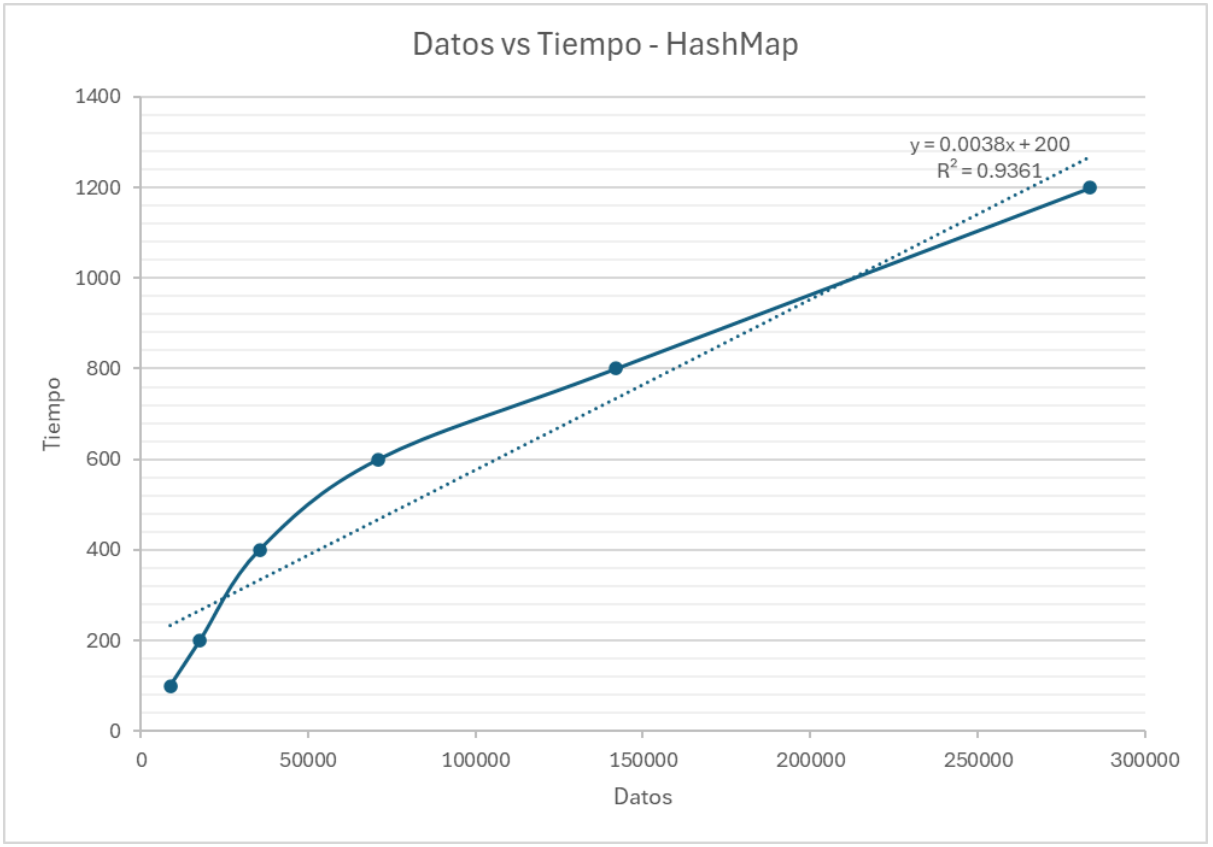
c. *Tiempos de Ejecución por implementación de MAP.*

No	***Action***	***Map***	***Time***
1	*Mostrar el nombre, tipo y cantidad de cada carta en la colección*	*HASH_MAP*	*200*
2	*Mostrar el nombre, tipo y cantidad de cada carta en la colección, ordenada por tipo*	*HASH_MAP*	*300*
3	*Mostrar el nombre y tipo de todas las cartas*	*HASH_MAP*	*300*
4	*Mostrar el nombre y tipo de todas las cartas, ordenadas por tipo*	*HASH_MAP*	*200*
5	*Mostrar el nombre, tipo y cantidad de cada carta en la colección*	*TREE_MAP*	*300*

6	*Mostrar el nombre, tipo y cantidad de cada carta en la colección, ordenada por tipo*	*TREE_MAP*	*400*
7	*Mostrar el nombre y tipo de todas las cartas*	*TREE_MAP*	*100*
8	*Mostrar el nombre y tipo de todas las cartas, ordenadas por tipo*	*TREE_MAP*	*500*
9	*Mostrar el nombre, tipo y cantidad de cada carta en la colección*	*LINKED_HASH_MAP*	*300*
10	*Mostrar el nombre, tipo y cantidad de cada carta en la colección, ordenada por tipo*	*LINKED_HASH_MAP*	*500*
11	*Mostrar el nombre y tipo de todas las cartas*	*LINKED_HASH_MAP*	*500*
12	*Mostrar el nombre y tipo de todas las cartas, ordenadas por tipo*	*LINKED_HASH_MAP*	*200*

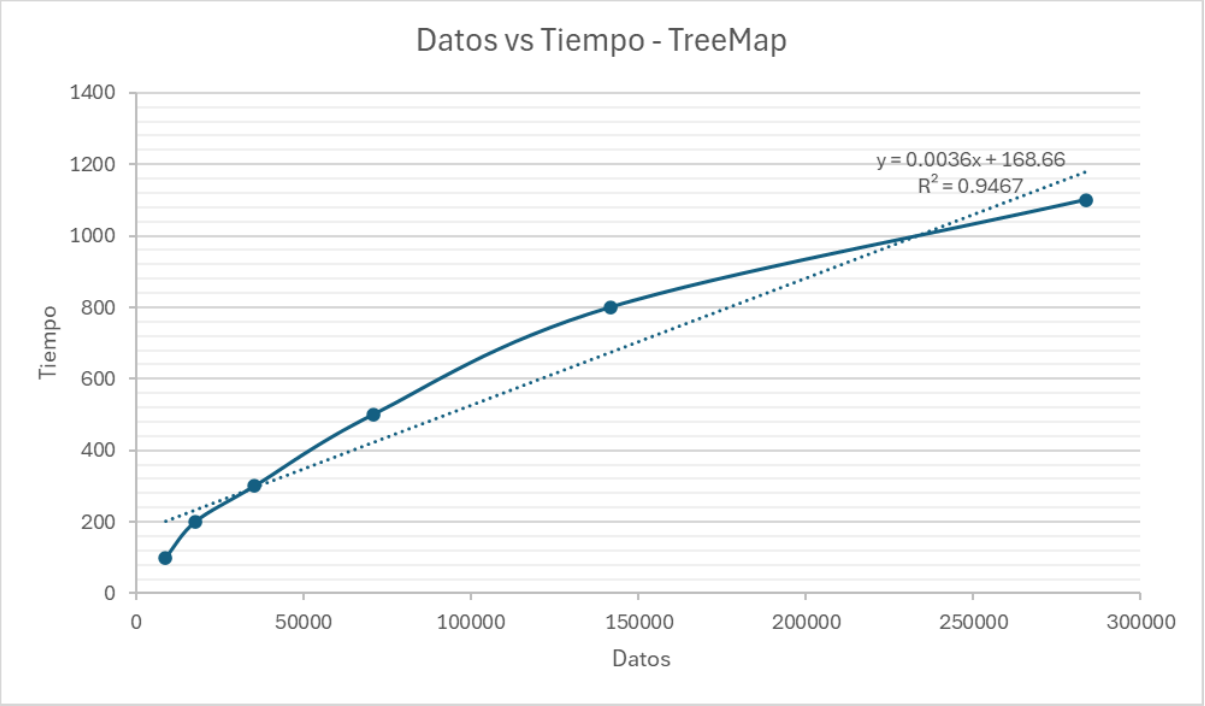
HashMap

Datos	*Tiempo*
8861	*100*
17722	*200*
35444	*400*
70888	*600*
141776	*800*
283552	*1200*



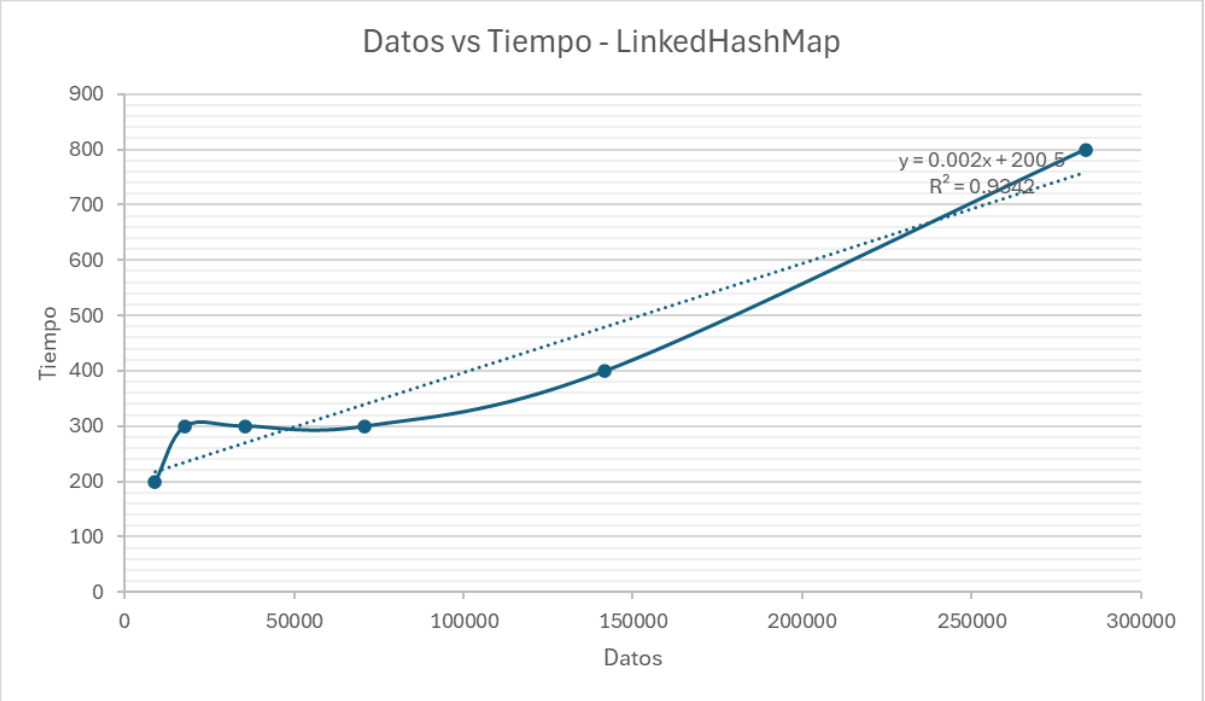
TreeMap

Datos	*Tiempo*
8861	*100*
17722	*200*
35444	*300*
70888	*500*
141776	*800*
283552	*1100*



LinkedHashMap

Datos	*Tiempo*
8861	*200*
17722	*300*
35444	*300*
70888	*300*
141776	*400*
283552	*800*



d. *Complejidad de tiempo para la implementación HashMap para mostrar todas las cartas:*

```java
// Método para mostrar todas las cartas con su tipo
1 usage  unknown
@Override
public void mostrarTodasLasCartas() {
    System.out.println("Todas las Cartas:");
    for (Map.Entry<String, String> entrada : tiposCartas.entrySet()) {
        String nombreCarta = entrada.getKey();
        String tipoCarta = entrada.getValue();
        System.out.println(nombreCarta + " - Tipo: " + tipoCarta);
    }
}
```

La complejidad temporal de la función dada **mostrarTodasLasCartas** se puede analizar de la siguiente manera:

- **Iteración de ciclo**: la función itera sobre todas las entradas en el mapa de tiposCartas usando un ciclo for. **O(n)**
- **Acceso a las entradas del mapa**: dentro del bucle, la función accede a la clave y el valor de cada entrada utilizando los métodos **getKey()** y **getValue()**. **O(1)**

Al simplificar la expresión: O(n) + O(2) la respuesta final es **O(n)**