

Hoja de trabajo No.6

Realizar: Operaciones con mapas.

Realizarse: Parejas

Objetivos:

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

Programa a realizar:

Utilice el de diseño Factory para seleccionar la implementación de MAP que usará su programa, en tiempo de corrida. El usuario debe seleccionar entre: 1)HashMap, 2)TreeMap, 3)LinkedHashMap.

El programa lee un archivo 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.

El archivo sigue la siguiente estructura: Cada carta se encuentra en una línea distinta y tiene el nombre luego el carácter | y el tipo de la carta. Un ejemplo del archivo es el siguiente:

```
Carta 1|Tipo
Nombre de Carta 2|Tipo
...
```

El archivo de cartas que se usarán para calificar la hoja se encuentra adjunto.¹

Luego de leer el archivo su programa debe permitir al usuario realizar las siguientes operaciones:

- Agregar una carta a la colección del usuario. Para esto el usuario ingresa el nombre de la carta que desea agregar a la misma. NOTA: El usuario puede tener más de una carta de cada tipo. NOTA: Si el usuario ingresa una carta que no se encuentra entre las cartas disponibles el programa debe mostrar un error.
- Mostrar el tipo de una carta específica. El usuario ingresará el nombre de la carta y se muestra el tipo de esa carta.
- 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.

Tareas:

- Su programa principal debe usar **Patron de diseño Factory** para seleccionar la implementación de MAP a utilizar.
- Debe dejar evidencia de todo el desarrollo en el repositorio de github o sistema similar para control de versiones. Indicar como acceder a su repositorio y si es necesario, agregar a su catedrático y auxiliar para que tengan acceso al mismo.
- Use un profiler para evaluar el tiempo de ejecución de su programa para mostrar las cartas. Corra su programa con las tres implementaciones y muestre los tiempos de ejecución de cada una de ellas. Diga cuál es la más rápida con el profiler.
- Calcule la complejidad de tiempo para la implementación HashMap, para mostrar todas las cartas. Indique como llegó a ese resultado.

Debe subir a Canvas todos los productos elaborados y los enlaces a su repositorio de github.

¹ La lista de cartas se obtuvo del API descrita en el siguiente [link](#) (desarrollada por [ygohub](#)).



Calificación:

| Aspecto | Puntos |
|---|------------|
| Uso del repositorio: existen más de tres versiones guardadas, la última versión es igual a la colocada en Canvas. | 10 |
| Patron Factory para seleccionar la implementación a usar en tiempo de corrida | 15 |
| Funcionamiento del programa para las operaciones 1, 2, 3 y 5 | 40 |
| Funcionamiento del programa para las operaciones 4 y 6 | 15 |
| Análisis de tiempo en ejecución con profiler y la complejidad calculada para HashMap. | 20 |
| TOTAL: | 100 |