Mapas· clase abstracta

## Mapas

### clase abstracta

Un mapa en java es una **clase abstracta de pares de valores.**

Estos pares de valores se identifican en **clave(velocidad de acceso)** y **valor(tipo de dato)**.

Si quieres acceder a cada par clave-valor de un **Map**, necesitas una forma de representar cada uno de esos pares.

Si queremos **crear un objeto mapa** tendremos que **implementar métodos abstractos** de **Map**. Aquí es donde entra la interface **Map.Entry.**

### implementación

Para implementar los métodos de Map, tenemos clases que heredan de Map con esa implementación resuelta, en función a un **orden** declarado en cada una de ellas.

El uso más común de **Map.Entry** es al **iterar sobre los elementos de un Map**. Los **Maps** en Java tienen un método llamado **entrySet()** que devuelve un **Set** de objetos **Map.Entry.** Este **Set** te permite recorrer cada par clave-valor.

La interfaz **Map.Entry** define los siguientes métodos principales:

1. **HashMap**: Clase con la implementación de Map más utilizada. Ofrece un rendimiento excelente para operaciones básicas como añadir, obtener y eliminar elementos, ya que no garantiza **ningún orden específico** de los elementos.  
   Ej: Map<Integer, Cliente> clientes = new HashMap<>();
2. **LinkedHashMap**: Esta clase, mantiene el **orden de inserción de los elementos**. Si el orden en que se añaden los clientes es importante, esta sería la opción.  
   Ej: Map<Integer, Cliente> clientes = new LinkedHashMap<>();
3. **TreeMap**: Esta clase, almacena los elementos **ordenados según las claves**. Si necesitas que tus clientes estén ordenados por su Integer (por ejemplo, su ID de cliente), TreeMap es la elección correcta.

Ej: Map<Integer, Cliente> clientes = new TreeMap<>();

### métodos

Las clases Map, comparten la mayoría de los métodos básicos (como **put, get, remove, containsKey, size, isEmpty, clear**). Las diferencias radican en cómo manejan el orden y algunas funcionalidades específicas de su implementación.