Comparison of different Map types

Here is a table that compares the different types of maps available in Java:

Map Type	Description
HashMap	A map that stores key-value pairs in a hash table. It provides fast lookups, but does not maintain the insertion order of elements.
TreeMap	A map that stores key-value pairs in a sorted tree structure. It maintains the natural ordering of keys, or allows you to specify a custom comparator to define the sort order.
LinkedHashMap	A map that stores key-value pairs in a hash table with a linked list running through it. It maintains the insertion order of elements.
ConcurrentHashMap	A thread-safe map that is designed for use in multi-threaded environments. It uses a hash table and lock striping to provide high concurrency and fast performance.

Pros and Cons

Map Type	Pros	Cons
HashMap	- Fast lookups - Efficient for most purposes	- Unordered - Not thread-safe
TreeMap	- Sorted keys - Good for data that needs to be in sorted order	- Slower lookups than HashMap - Not thread-safe
LinkedHashMap	- Maintains insertion order - Good for data that needs to preserve insertion order	- Slower lookups than HashMap - Not thread-safe
ConcurrentHashMap	- Fast lookups - Thread-safe	- Unordered

How a Map works internally?

From an algorithm perspective, a map works by using a hash function to map keys to indices in an array. When you add a new element to the map, it calculates the hash code of the key and uses it to determine the index at which the key-value pair should be stored. When you look up a value in the map, it uses the same hash code to locate the key-value pair and return the associated value.

In a hash map, the hash function is designed to distribute the keys evenly across the array, so that each index is equally likely to contain a key-value pair. This allows the map to achieve an average lookup time of O(1), meaning that the time it takes to look up a value is independent of the size of the map.

However, it is possible for the hash function to produce `collisions`, where two or more keys map to the same index. In this case, the map may need to store multiple key-value pairs at the same index, and use a collision resolution strategy, such as chaining, to handle collisions. This can cause the lookup time to degrade to O(n), where n is the number of elements stored at the index.

> ✏️ **Collision Resolution Strategy**
>
> A collision resolution strategy is a technique used to handle collisions in a hash map, which occur when two or more keys map to the same index in the array.
>
> There are several collision resolution strategies that can be used, including:
>
> 1. **Chaining**: In this strategy, each index in the array is a linked list, and each node in the list stores a key-value pair. When a collision occurs, the key-value pair is added to the end of the linked list at the index. This allows the map to store multiple key-value pairs at the same index, without requiring additional space.
> 2. **Open addressing**: In this strategy, when a collision occurs, the map searches for the next available empty index in the array to store the key-value pair. There are several probing techniques that can be used to determine the next index, such as linear probing, quadratic probing, and double hashing.