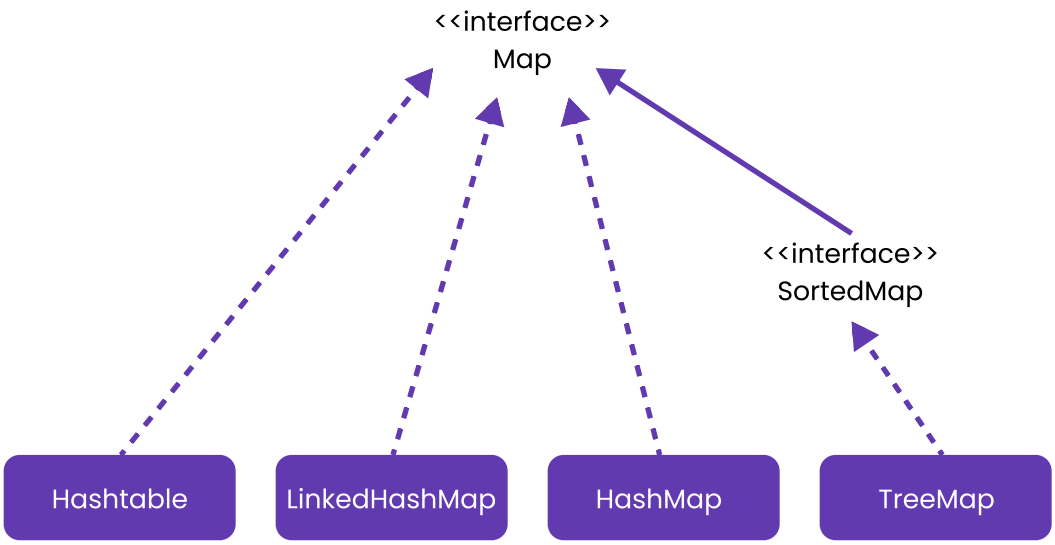
Types of HashMap in Java



HashMap is implemented as a hash table, and there is no ordering on keys or values.

TreeMap is implemented based on red-black tree structure, and it is ordered by the key.

LinkedHashMap preserves the insertion order

Hashtable is synchronized, in contrast to HashMap. It has an overhead for synchronization.
This is the reason that HashMap should be used if the program is thread-safe.

Also concurrent hash map can be used if we want to achieve synchronization in a multi threaded environment

Now we will see an example of each type of Map in java and will see the output of map. The idea is to see the order of keys for different types of maps.

HashMap can store keys in any order.TreeMap sorts the keys and stores them in order.

LinkedHashMap preserves the order of insertion of keys.

Code: LP_Code2.java

```
Value of HashMap is: {1=Piyush, 2=Athar, 3=Ajay, 4=Anil}
Value of TreeMap is: {1=Piyush, 2=Athar, 3=Ajay, 4=Anil}
Value of LinkedHashMap is: {4=Anil, 2=Athar, 3=Ajay, 1=Piyush}
```

Interview problem: Two Sum

Ques: Given an array of integers and an integer target, return indices of the two numbers such that they add up to target. Assume only 1 valid answer exists.

Eg. Input = [2,7,11,15]. target =9
Output = [0,1]

Input: [3,2,4]. target = 6
Output: [1,2]

Code: LP_Code3.java

Output

```
Enter the number of elements you want to store: 5
Enter the elements of the array:
2 7 4 1 6
Enter the target element of the array:
9
The output array is: [0,1]
```

```
Enter the number of elements you want to store: 3
Enter the elements of the array:
3 2 4
Enter the target element of the array:
6
The output array is: [1,2]
```

Approach:

We would store every element in the hashmap in the form of (array[index], index). The idea is we will traverse the original array and will check in map if the remaining element i.e. target - currentElement is present in hashmap then we found our answer.
The answer would be currentIndex and index of the remaining element.

There can be 2 edge cases in this.

- What if we found exactly half of the target. Then to confirm we have one more element of the same value, we would check the currentIndex < map.get(target-currentElement). This condition would ensure that we have another element which is ahead of this as map would have already updated the index when we are entering the elements in the array.

Interview problem: First unique Character in String

Given a string s, find the first non-repeating character in it and return its index. If it does not exist, return -1.

eg. **Input:** s = "leetcode"
Output: 0

Input: s = "loveleetcode"
Output: 2

Input: s = "aabb"
Output: No Character is found: -1