**Java ConcurrentMap Interface**

The ConcurrentMap interface of the Java collections framework provides a thread-safe map. That is, multiple threads can access the map at once without affecting the consistency of entries in a map.

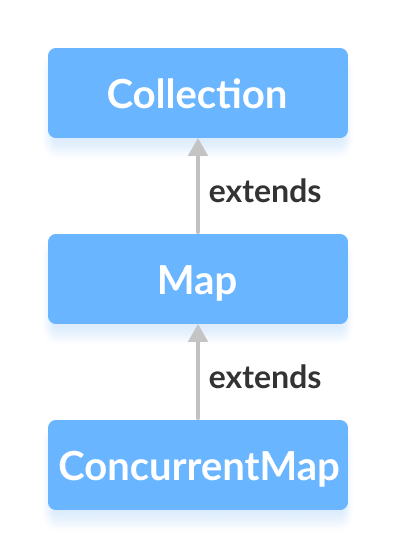
ConcurrentMap is known as a synchronized map.

It extends the Map interface.

## Class that implements ConcurrentMap

Since ConcurrentMap is an interface, we cannot create objects from it.

In order to use the functionalities of the ConcurrentMap interface, we need to use the class ConcurrentHashMap that implements it.



## How to use ConcurrentMap?

To use the ConcurrentMap, we must import the java.util.concurrent.ConcurrentMap package first. Once we import the package, here's how we can create a concurrent map.

// ConcurrentMap implementation by ConcurrentHashMap

CocurrentMap<Key, Value> numbers = new ConcurrentHashMap<>();

In the above code, we have created a concurrent map named numbers.

Here,

* Key - a unique identifier used to associate each element (value) in a map
* Value - elements associated by keys in a map

## Methods of ConcurrentMap

The ConcurrentMap interface includes all the methods of the Map interface. It is because Map is the superinterface of the ConcurrentMap interface.

Besides all those methods, here are the methods specific to the ConcurrentMap interface.

* **putIfAbsent()** - Inserts the specified key/value to the map if the specified key is not already associated with any value.
* **compute()** - Computes an entry (key/value mapping) for the specified key and its previously mapped value.
* **computeIfAbsent()** - Computes a value using the specified function for the specified key if the key is not already mapped with any value.
* **computeIfPresent()** - Computes a new entry (key/value mapping) for the specified key if the key is already mapped with the specified value.
* **forEach()** - Access all entries of a map and perform the specified actions.
* **merge()** - Merges the new specified value with the old value of the specified key if the key is already mapped to a certain value. If the key is not already mapped, the method simply associates the specified value to our key.

To learn more, visit Java ConcurrentMap official documentation.

## Implementation of ConcurrentMap in ConcurrentHashMap

import java.util.concurrent.ConcurrentMap;

import java.util.concurrent.ConcurrentHashMap;

class Main {

public static void main(String[] args) {

// Creating ConcurrentMap using ConcurrentHashMap

ConcurrentMap<String, Integer> numbers = new ConcurrentHashMap<>();

// Insert elements to map

numbers.put("Two", 2);

numbers.put("One", 1);

numbers.put("Three", 3);

System.out.println("ConcurrentMap: " + numbers);

// Access the value of specified key

int value = numbers.get("One");

System.out.println("Accessed Value: " + value);

// Remove the value of specified key

int removedValue = numbers.remove("Two");

System.out.println("Removed Value: " + removedValue);

}

}

**Output**

ConcurrentMap: {One=1, Three=3, Two=2}

Accessed Value: 1

Removed Value: 2

To learn more about ConcurrentHashMap, visit Java ConcurrentHashMap.