Map Interface:

- A Map is a collection in java used for storing key-value pairs.
- Each key must be unique, but values can be duplicated.
- It is a part of java.util package
- Common implementations: HashTable, HashMap, LinkedHashMap, TreeMap.

Why?

- Efficient retrieval of data using a key.
- Suitable for scenarios where a relationship exists between key, value pairs.

Key Methods in the Map Interface:

- put(K key, V value) : Adds a key-value pair to the map
- get(Object key) : retrieve the value associated with the key.
- remove(Object key) : Removes the key-value pair

- containsKey(Object key) : Checks if a key is present
- containsValue(Object value) :Check if a calue is present
- values(): Returns a collection of values.

Map (Java SE 21 & JDK 21)

1. HashTable:

Features:

- Synchronized and thread-safe
- Does not allow null keys or values
- Suitable for multi-threaded environments

```
package Map;

import java.util.Hashtable;

public class HashtableMethodsExample {
    public static void main(String[] args) {
        //creating a HashTable
        Hashtable<Integer, String> table = new
Hashtable<>();

        //put() method to add key-value pairs
        table.put(1, "JAVA");
        table.put(2, "Python");
        table.put(3, "C++");

        // get() method to retrieve value by key
        System.out.println("Value of 2nd key : "+
table.get(2));

        // containsKey() method - check if a key exists
or not
        System.out.println("Contains key 4 : "+
table.containsKey(3));
```

```
            // containsValue() method -
            //To remove key value pair
            table.remove(1);
            System.out.println("After Removal : " + table);
        }
    }
```

2. HashMap

Features :

- Not synchronized (Not thread-safe)
- Allows one null key and multiple null values
- Faster than HashTable in single threaded environments
- Maintains no order of elements

```
package Map;

import java.util.HashMap;

public class HashMapMethodsExample {
    public static void main(String[] args) {
        //Creating a HashMap
        HashMap<String, Integer> map = new HashMap<>();

        //put() method to add key-value pairs
        map.put("Apple", 100);
        map.put("Banana", 50 );
        map.put("Cherry", 75);

        //get() method to retrieve value by key
        System.out.println("Price of Banana : " +
map.get("Banana"));

        //containsKey() method to check if key exists
        System.out.println("Does it contains 'Apple '? "+
map.containsKey("Apple"));
```

```
        //containsValue()
        //remove()

        // values() method to get all values
        System.out.println("Values : "+ map.values());

        //entrySet() method to get all key-value pairs
        System.out.println("Entries : "+ map.entrySet());
    }
}
```

4. LinkedHashMap

Features:

- Maintains the insertion order
- Not synchronized(Not thread-safe)
- Allows one null key and multiple null values
- Slightly slower than HashMap due to insertion order

```
package Map;

import java.util.LinkedHashMap;

public class LinkedHashMapExample {
    public static void main(String[] args) {
        //Creating a LHM
        LinkedHashMap<String, String> linkedMap = new
LinkedHashMap<>();

        // put() method to add key-value pairs
        linkedMap.put("Krishna", "Manager");
        linkedMap.put("Govind", "Developer");
        linkedMap.put("Gopal", "Tester");

        //get() method - Retrieves value by key
        System.out.println("Role of Krishna : "+
linkedMap.get("Krishna"));

        //containsKey()
        //containsValue()
```

```
            //remove()
            //keySet()

            //values() method - Get all values
            System.out.println("Values : " +
    linkedMap.values());

            //entrySet()
        }
    }
```