Queue Interface

It follows the FIFO(First In First Out) principle

Elements are inserted from the rear and removed from front.

PriorityQueue Class [PriorityQueue (Java SE 21 & JDK 21)]

A class which implements the queue interface

Maintains elements in their natural order

- Not thread-safe

```
package queueExample;

import java.util.PriorityQueue;

public class PriorityQueueExample {
    public static void main(String[] args) {
        PriorityQueue<Integer> pq = new PriorityQueue<>();

        //adding elements
        pq.add(40);
        pq.add(30);
        pq.add(10);
        pq.add(20);

        System.out.println("Priority Queue: "+ pq);

        //Accessing the highest priority element
        System.out.println("Highest Priority Element: "+
pq.peek());

        // To get the size
        System.out.println("Size: "+ pq.size());


        //Removing the highest priority element
```

```
        //System.out.println("Polling: "+ pq.poll());

        //Displaying the pq
      //  System.out.println("Priority Queue after polling: "+
pq);


    }
}
```

Can we add null values?

➔ No, it throws nullpointerException

## MAP Interface

A collection which is used to store key-value pair.

Char:

1. Each key is unique, but values can be duplicated
2. Allows retrieval of data using key.

Common Implementations:

1. Hashtable
2. HashMap
3. LinkedHashMap
4. TreeMap

# HashTable [Hashtable (Java SE 21 & JDK 21)]

It is synchronized and thread-safe, which makes it suitable for multithreaded environment.

Characteristics:

1. It does not allow null keys or values
2. Performance is slightly slower than hashmap because of synchronization.

```
package mapExample;

import java.util.Hashtable;

public class HashTableExample {
    public static void main(String[] args) {
        //Creating a hashtable
        Hashtable<Integer,String> table = new Hashtable<>();

        //adding elements
        table.put(1,"JAVA");
        table.put(2,"Python");
        table.put(3, "C#");

        System.out.println("HashTable: "+ table);

        //retrieve a value by key
        System.out.println("Value of 3rd key: "+
table.get(3));


        //Check if the key exists or not
        System.out.println("Contains key 3? "+
table.containsKey(3));

        //To remove key-value pair
        table.remove(3);
```

```
        //Displaying table after removal
        System.out.println("After Removal:"+ table);
    }
}
```

1. Can we store null keys or values in the hashtable?

## HashMap

HashMap is an unsynchronized map which allows only one null key and multiple null values.

Characteristics:

Faster than hashtable

Does not maintain the order of elements

```
package mapExample;

import java.util.HashMap;

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

        //Adding key-value pairs
        map.put("Apple",100);
        map.put("Banana",50);
        map.put("PineApple", 75);

        //Retrieve value by a key
        System.out.println("Price of apple: "+
map.get("Apple"));

        //check if key exists
```

```
        System.out.println("Does cart contain banana? "+
map.containsKey("Banana"));

        //Get all values
        System.out.println("Values: "+ map.values());

        // To retrieve all key-value pairs
        System.out.println("All KV pairs: "+ map.entrySet());
    }
}
```

Store null key and values in HashMap.

TASK: Student Management System with HashMap.

| LinkedHashMap: LinkedHashMap (Java SE 21 & JDK 21) |
|---|

It is a subclass of hashmap

It maintains the insertion order of elements

Characteristics:

1. Not synchronized
2. Slower than the hashmap because it maintains the insertion order.

```
3. package mapExample;

   import java.util.LinkedHashMap;

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

           linkedHm.put("Krishna","CEO");
           linkedHm.put("Govind", "CFO");
           linkedHm.put("Gopal", "Developer");
           linkedHm.put("Rakesh", "Tester");

           //Retrieving the value by key
```

```
        System.out.println("Role of Krishna: "+
linkedHm.get("Krishna"));

        //To get all values
        System.out.println("Values of all elements: "+
linkedHm.values());


    }
}
```

When to use which map?

1. HashTable: When thread-safety is required
2. HashMap: For faster operations in a single threaded environment
3. LinkedHashMap: Whenever maintaining an insertion order is important.

TREEMAP:* TreeMap (Java SE 21 & JDK 21)