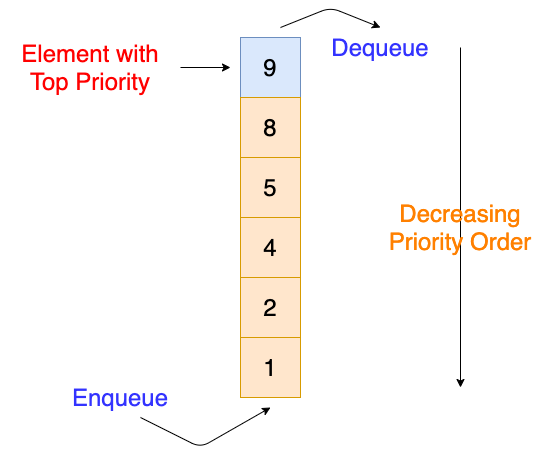
What is MinPQ?

The MinPQ class represents **a priority queue of generic keys**. It supports the usual insert and delete-the-minimum operations, along with methods for peeking at the minimum key, testing if the priority queue is empty, and iterating through the keys. This implementation uses a binary heap.

A priority queue is a special type of queue in which **each element is associated with a priority value**. And, elements are served on the basis of their priority. That is, higher priority elements are served first. However, if elements with the same priority occur, they are served according to their order in the queue.



Is Java Priority queue max or min?

By default, the priority queue in Java is **min Priority queue** with natural ordering. To make it max, we have to use a custom comparator so that head of the queue returns the greatest element in the queue.

Rafa:

**import** java.util.PriorityQueue;

public class PriorityQueueExample {

public static void main(String[] args) {

// Create a PriorityQueue that sorts elements

// by their natural ordering

PriorityQueue<Integer> pq = new PriorityQueue<>();

// Let's add some elements to the PriorityQueue

Integer [] elements = new Integer[]{8, 100, 98, 10, 2,

15, 45, 25, 76, 3, 9, 1, 63, };

for (int e: elements) {

pq.add(e);

}

// Let's iterate through the elements to see they

// are not necessarily stored in sorted order

System.out.print("Print by Iterating: ");

for(int e: pq) {

System.out.print(e + " ");

}

System.out.println();

System.out.print("Print by Retrieval: ");

// Let's remove elements one by one

while (!pq.isEmpty()) {

System.out.print(pq.remove() + " ");

}

}

}

Output:

Print by Iterating: 1 3 2 25 8 15 45 100 76 10 9 98 63

Print by Retrieval: 1 2 3 8 9 10 15 25 45 63 76 98 100