# Priority Queue

By Room 3
Hi Ethan

# What is a Priority Queue?

Priority Queue is more specialized data structure than Queue. Like ordinary queue, priority queue has same method but with a major difference. In Priority queue items are ordered by key value so that item with the lowest value of key is at front and item with the highest value of key is at rear or vice versa. So we're assigned priority to item based on its key value. Lower the value, higher the priority.

Properties of Priority Queue

- Every item has a priority associated with it.
- An element with high priority is dequeued before an element with low priority
- If two elements have the same priority, they are served according to their order in the queue

# Different Ways of Implementation

Priority queue can be implemented using the following data structures:

- Arrays
- Linked list
- Heap data structure
- Binary search tree

# How Do We Implement It?

Priority Queue can be implemented with arrays, linked lists, heap data structures, binary search trees, etc. As an extension of Queue, we can implement Priority Queue into other functions as well. For example, enque and deque would implement PQ with adding and removing elements. PQ can also be implemented into arrays by returning the front and rear elements of the queue. Also, PQ can also return if the queue is empty or even print out the queue as well.

# What is Priority Queue Useful For

When it is necessary to process objects based on priority.

Usually queues follow the  FIFO model but for some cases, certain object may have priority in terms of being processed ahead of others.

Heap Sort : Heap sort is typically implemented using Heap which is an implementation of Priority Queue.

Data compression : It is used in Huffman codes which is used to compresses data.

The hospital emergency queue is an ideal real-life example of a priority queue. In this queue of patients, the patient with the most critical situation is the first in a queue, and the patient who doesn’t need immediate medical attention will be the last. In this queue, the priority depends on the medical condition of the patients.

# The Most Basic Methods

.poll() -> This method retrieves and remove the head of the queue.

.peek() -> This method retrieves but DOESN’T remove the head of queue.

.add() -> This method adds elements to the queue.

.remove() -> This method removes elements from the queue.

.sort() -> : (

# Data Structure Implementation We Got For Priority Queue

We implemented it into String datatypes. In our case, we used our names (and Ethan's) without the last letter. From scratch and equipped it with only an array and a dream in Java, we implemented our names - 1 into a queue and then printed it out. Our peek and poll methods work by naming our queue array queueSort, then we peeked by returning the first value of the array, and polled by removing the first value of the array and returning.

Now 100% from scratch!

# Our Code

```java
import java.util.*;

public class ByGroup3 {

    private Person[] queue;
    private int size;

    public ByGroup3(){
        queue = new Person[10];
        size = 0;
    }

    public void add(Person p) {
        size++;
        if(size > queue.length) {
            int newSize = (int) (queue.length * 1.5);
            Person[] temp = new Person[newSize];
            for(int i = 0; i < queue.length; i++) {
                temp[i] = queue[i];
            }
            queue = temp;
        }
        queue[size - 1] = p;
        queue = queueSort(queue);
    }

```

```java
    public void add(Person p) {
        size++;
        if(size > queue.length) {
            int newSize = (int) (queue.length * 1.5);
            Person[] temp = new Person[newSize];
            for(int i = 0; i < queue.length; i++) {
                temp[i] = queue[i];
            }
            queue = temp;
        }
        queue[size - 1] = p;
    }

    private int findHighestPriority() {
        int highest = -1;
        int highestIndex = 0;
        for(int i = 0; i < size; i++) {
            if(queue[i].tier > highest) {
                highest = queue[i].tier;
                highestIndex = i;
            }
        }
        return highestIndex;
    }

```

Now 100% from scratch!

# Our Code

```
39      // Poll method
40      public Person poll() {
41          int index = findHighestPriority();
42          Person removed = queue[index];
43          for(int i = index; i < size - 1; i++) {
44              queue[index] = queue[index + 1];
45          }
46          size--;
47          return removed;
48      }
49
50
51      // Peek method
52      public Person peek() {
53          int index = findHighestPriority();
54          return queue[index];
55      }
56
57
58      public String toString() {
59
60          String result = "";
61          for(int i = 0; i < size; i++) {
62              result += queue[i].getName() + " ";
63          }
64          return result;
65      };
```

```
68      public static void main(String[] args) {
69
70          Person a1 = new Person(0, "etha");
71          Person a2 = new Person(5, "jasnin");
72          Person a3 = new Person(1, "larr");
73          Person a4 = new Person(3, "jame");
74          Person a5 = new Person(6, "victo");
75          Person a6 = new Person(2, "gen");
76          Person a7 = new Person(4, "edwi");
77
78          ByGroup3 priorityQueue = new ByGroup3();
79          priorityQueue.add(a3);
80          priorityQueue.add(a5);
81          System.out.println(priorityQueue.peek().name);
82  //      System.out.println(priorityQueue);
83          System.out.println(priorityQueue.poll().name);
84          System.out.println(priorityQueue.poll().name);
85      }
86
```

```
104 class Person {
105     int tier;
106     String name;
107
108     public Person(int tier, String name) {
109         this.tier = tier;
110         this.name = name;
111     }
112
113     public int getTier() {
114         return this.tier;
115     }
116
117     public String getName() {
118         return this.name;
119     }
120
121     public void setTier(int newTier) {
122         this.tier = newTier;
123     }
124
125     public void setName(String newName) {
126         this.name = newName;
```