Ans1.

This comparator defines the ordering of elements. Examples are: Bubble Sort, Merge Sort. Counting-based sorting: There's no comparison involved between elements in these types of sorting algorithms but rather work on calculated assumptions during execution. Examples are : Counting Sort, Radix Sort

Ans2.

When a function is called the local variables are stored in a stack, and it is automatically destroyed once returned. A stack is used when a variable is not used outside that function. It allows you to control how memory is allocated and deallocated. Stack automatically cleans up the object.

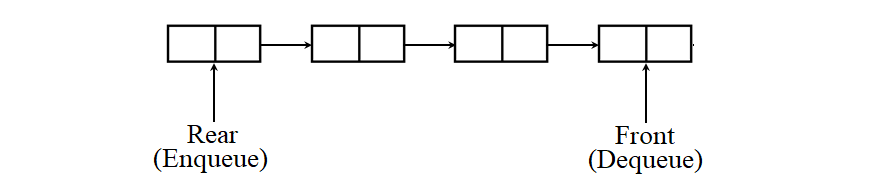
Ans3.

|  |  |
| --- | --- |
| Stack | Queue |
| The stack is based on LIFO(Last In First Out) principle | The queue is based on FIFO(First In First Out) principle. |
| Insertion Operation is called Push Operation | Insertion Operation is called Enqueue Operation |
| Deletion Operation is called Pop Operation | Deletion Operation is called Dequeue Operation |
| Push and Pop Operation takes place from one end of the stack | Enqueue and Dequeue Operation takes place from a different end of the queue |
| The most accessible element is called Top and the least accessible is called the Bottom of the stack | The insertion end is called Rear End and the deletion end is called the Front End. |
| Simple Implementation | Complex implementation in comparison to stack |
| Only one pointer is used for performing operations | Two pointers are used to perform operations |
| Empty condition is checked using  Top==-1 | Empty condition is checked using  Front==-1||Front==Rear+1 |
| Full condition is checked using  Top==Max-1 | Full condition is checked using  Rear==Max-1 |
| There are no variants available for stack | There are three types of variants i.e circular queue, double-ended queue and priority queue |
| Can be considered as a vertical collection visual | Can be considered as a horizontal collection  visual |
| Used to solve the recursive type problems | Used to solve the problem having sequential processing |

Ans4.

## 2. Simple Queue

A simple queue is the most basic queue. In this queue, **the enqueue operation takes place at the rear, while** **the dequeue operation takes place at the front:**



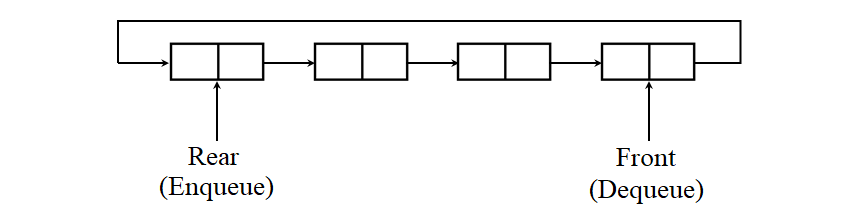
Its applications are process scheduling, disk scheduling, memory management, IO buffer, pipes, call center phone systems, and interrupt handling.

## 3. Circular Queue

A circular queue **permits better memory utilization than a simple queue**when the queue has a fixed size.

In this queue, the last node points to the first node and creates a circular connection. Thus, it allows us to insert an item at the first node of the queue when the last node is full and the first node is free.

It’s also called a ring buffer:



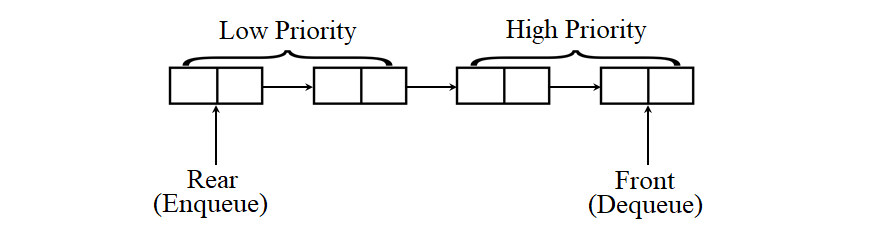
It’s used to switch on and off the lights of the traffic signal systems. Apart from that, it can be also used in place of a simple queue in all the applications mentioned above.

## 4. Priority Queue

**A priority queue is a special kind of queue in which each item has a predefined priority of service.** In this queue, the enqueue operation takes place at the rear in the order of arrival of the items, while the dequeue operation takes place at the front based on the priority of the items.

That is to say that **an item with a high priority will be dequeued before an item with a low priority**.

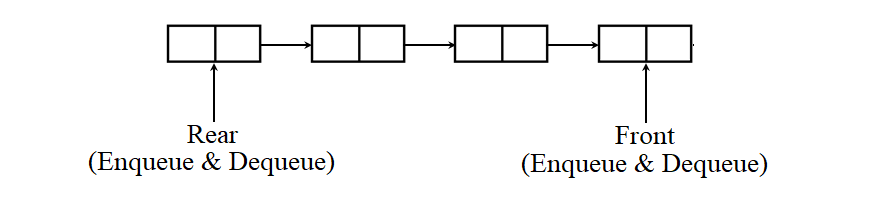
In the case, when two or more items have the same priority, then they’ll be dequeued in the order of their arrival. Hence, it may or may not strictly follow the FIFO rule:



It’s used in interrupt handling, [Prim’s algorithm](https://www.baeldung.com/java-prim-algorithm), [Dijkstra’s algorithm](https://www.baeldung.com/java-dijkstra),  [A\* search algorithm](https://www.baeldung.com/java-a-star-pathfinding), [heap sort](https://www.baeldung.com/java-heap-sort), and Huffman code generation.

## 5. Double-Ended Queue (Deque)

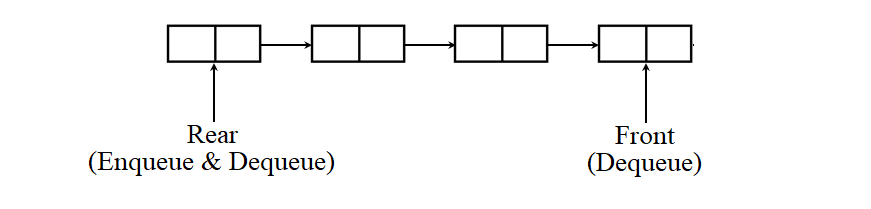
A deque is also a special type of queue. In this queue, the **enqueue and dequeue operations take place at both front and rear**. That means, we can insert an item at both the ends and can remove an item from both the ends. Thus, it may or may not adhere to the FIFO order:



It’s used to save browsing history, perform undo operations, implement A-Steal job scheduling algorithm, or implement a [stack](https://www.baeldung.com/java-stack) or implement a simple queue.

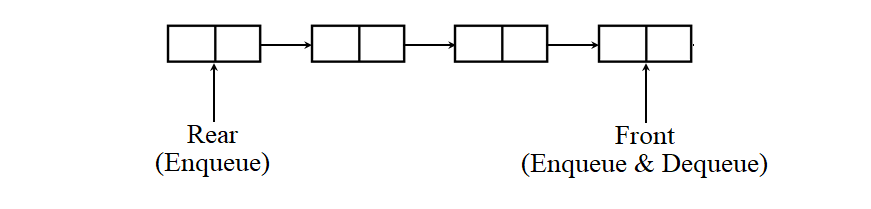
Further, it has two special cases: **input-restricted deque** and **output-restricted deque**.

In the first case, the enqueue operation takes place only at the rear, but the dequeue operation takes place at both rear and front:



An input-restricted queue is useful when we need to remove an item from the rear of the queue.

In the second case, the enqueue operation takes place at both rear and front, but the dequeue operation takes place only at the front:



An output-restricted queue is handy when we need to insert an item at the front of the queue.

Ans 5.

Use a queue when you want to get things out in the order that you put them in. Use a stack when you want to get things out in the reverse order than you put them in. Use a list when you want to get anything out, regardless of when you put them in (and when you don't want them to automatically be removed).

Ans6.

Many programming languages implement recursion by means of stacks. Generally, whenever a function (caller) calls another function (callee) or itself as callee, the caller function transfers execution control to the callee.