1. **More Abstract**

Stacks, queues, and priority queues are more abstract entities than arrays and many other data storage structures. They're defined primarily by their interface: the permissible operations that can be carried out on them. The underlying mechanism used to implement them is typically not visible to their user. For example, the underlying mechanism for a stack can be an array, as shown in this

chapter, or it can be a linked list. The underlying mechanism for a priority queue can be an array or a special kind of tree called a *heap*.

1. **Stacks**A stack allows access to only one data item: the last item inserted. If you remove this item, then you can access the next-to-last item inserted, and so on.
2. **Efficiency of Stacks**

Items can be both pushed and popped from the stack implemented in the StackX class in

constant O(1) time. That is, the time is not dependent on how many items are in the stack, and is therefore very quick. No comparisons or moves are necessary.

1. **Queues**The word *queue* is British for *line* (the kind you wait in). In Britain, to "queue up" means to

get in line. In computer science a queue is a data structure that is similar to a stack,

except that in a queue the first item inserted is the first to be removed (FIFO), while in a stack, as we've seen, the last item inserted is the first to be removed (LIFO).

1. **Efficiency of Queues**

As with a stack, items can be inserted and removed from a queue in O(1) time.

1. **Deques**

A *deque* is a double-ended queue. You can insert items at either end and delete them

from either end. The methods might be called insertLeft() and insertRight(),

and removeLeft() and removeRight().

If you restrict yourself to insertLeft() and removeLeft() (or their equivalents on

the right), then the deque acts like a stack. If you restrict yourself to insertLeft() and

removeRight() (or the opposite pair), then it acts like a queue.

A deque provides a more versatile data structure than either a stack or a queue, and is

sometimes used in container class libraries to serve both purposes. However, it's not used

as often as stacks and queues, so we won't explore it further here.

1. **Priority Queues**

A priority queue is a more specialized data structure than a stack or a queue. However,

it's a useful tool in a surprising number of situations. Like an ordinary queue, a priority

queue has a front and a rear, and items are removed from the front. However, in a priority

queue, items are ordered by key value, so that the item with the lowest key (or in some

implementations the highest key) is always at the front. Items are inserted in the proper

position to maintain the order.