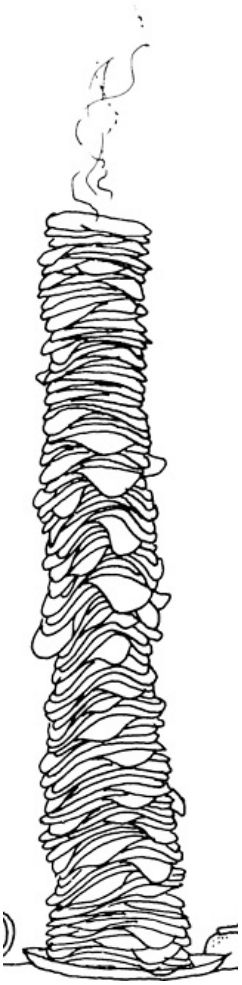
Stacks and Queues

Stacks and queues are similar and opposite data structures. They both allow access to one element at a time, but they have reversed orders.



A stack is what is called a LIFO (last-in, first-out) structure. When an item is added (or "pushed") onto a stack, the item is added to the top, while everything else is pushed down. When an item is removed ("popped") from a stack, the item that is popped is the same item that was last pushed. Think of a stack like a physical stack of pancakes; you don't want to remove a pancake other than the one on the top.

In Java, a stack is implemented as the class Stack<E>. There is not a corresponding interface; however, keep in mind that the Deque<E> interface provides stack functionality as well and is implemented by LinkedList, ArrayDeque, and others. Pushing and popping items are achieved using the push(E item) and pop() methods. The functions push() and pop() both return the item in question. In addition, there is a peek() function that will return the top item on the Stack *without* removing it. However, there is no way to access an object further down the Stack. The function empty() can be used to test if a Stack is currently empty. Stack has only one constructor, which takes no arguments and creates an empty Stack.

A queue is a FIFO (first-in, first-out) data structure. When an item is added to a queue, it is added to the back (or tail) of the queue. When an item is removed, it is removed from the front (or head). Thus the element to be removed from a queue is the element that has been on the queue for the longest span of time. Think of a queue like the people lining up (e.g. to receive the aforementioned pancakes); the first person to enter the line will be the first to leave.

Queue in Java is actually an interface and not a class. However, there are several classes implementing Queue, including (believe it or not) LinkedList, as well as PriorityQueue, LinkedBlockingQueue, ArrayDeque, etc. Some of these have other features, such as PriorityQueue, which will order its elements according to a provided comparator (or the natural ordering of the items). Queues have two sets of methods used to add and remove elements. The add(E item) method will add an item to a Queue, while the remove() method will remove one. (As stated before, the item removed will be the least recent item entered.) The element() method will take a look at the next element (similar to peek() for stacks). These all will throw an exception if there is a problem (if the queue has reached maximum capacity, or if one tries to pull from an empty queue). There are versions of these methods (offer() for add(), peek() for element(), and poll() for remove()) that will return false or null instead of throwing an exception.

Source: Oracle's website, http://docs.oracle.com/javase/7/docs/api/java/util/package-summary.html