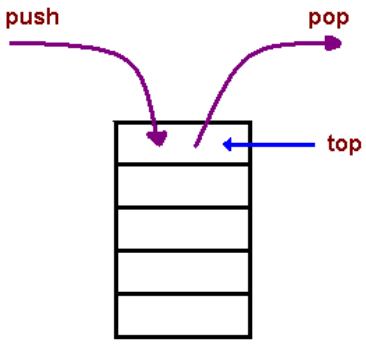
Stacks and Queues

Melissa Thai, Daniel Hartung, Shayna Oriold

Stacks

A stack is a type of data structure that lets you insert and remove elements only at one end, called the "top" of the stack. Therefore, they are removed in the order that is opposite from the order in which they have been added. This order is called "last-in, first-out" order.



The addition and removal operations associated with the stack are called "push" and "pop". Push adds an item to the top of the stack, and pop removes the item from the top.

The diagram to the right shows a visual representation of the concept of a stack:

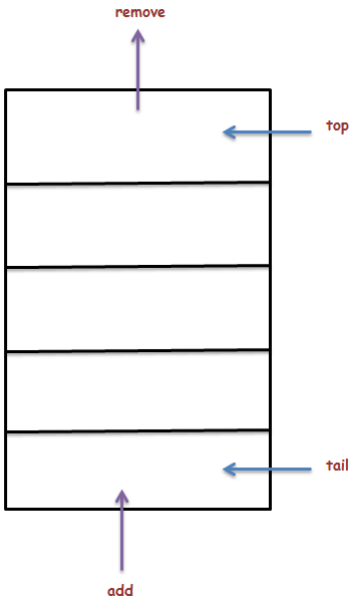
The Java library provides a simple Stack class with methods "push", "pop", and "peek" (peek gets the top element of the stack but doesn't remove it).

Example:

```
Stack<String> s = new Stack<String>();
s.push("A");
s.push("B");
s.push("C");
while(s.size() > 0) {
        System.out.print(s.pop() + " "); // Prints C B A
}
```

Queues

A queue is another type of data structure that lets you add items on one end of the queue (the "tail") and remove them from the other end of the queue (the "head"). This order is called "first-in, first-out" order.



The Queue interface in the standard Java library has methods "add" to add an element to the tail of the queue, "remove" to remove the head of the queue, and "element" to get the head element of the queue without removing it. A LinkedList is a class that implements the Queue interface.

Example:

```
Queue<String> q = new LinkedList<String>;
q.add("A");
q.add("B");
q.add("C");
while(q.size() > 0) {
        System.out.print(q.remove() + " "); // Prints A B C
}
```

Priority Queues

A priority queue collects elements, each of which has a priority. Unlike a regular queue, the priority queue doesn't use "first-in, first-out." Instead, elements are retrieved according to their priority. New items can be inserted in any order, but whenever an item is removed, it's the item with the most urgency.

Example:

```
PriorityQueue<WorkOrder> q = new PriorityQueue<WorkOrder>();
q.add(new WorkOrder(3, "Shampoo carpets"));
q.add(new WorkOrder(1, "Fix broken sink"));
q.add(new WorkOrder(2, "Order cleaning supplies"));

q.remove(); // "Fix broken sink" is removed
```

Sources

Big Java: Early Objects

http://www.cs.cmu.edu/~adamchik/15-121/lectures/Stacks%20and%20Queues/Stacks%20and%20Queues.html