Three data structures: queue, stack, and deque

We define the *queue*, *stack*, and *deque* and discuss their implementations in the Java Collections framework.

The queue (a FIFO list)

A queue is a list of items with two operations for changing it. To the right is a queue with 3 values: [*y*, *c*, *x*]. The two operations are:

```
[y,   c,   x]
|          |
front   back
```

1. add(*v*): Append element *v* to the list. This changes the list to [*y*, c, *x*, *v*].
2. remove(): Remove and return the first item of the list. This changes the list [*y*, c, *x*] to [*c*, *x*] and returns *y*.

Here's an easy way to remember what a queue is: While Cornell students stand in a line to buy hockey tickets, the British stand in a queue to buy tickets for a cricket match. A queue is also called a *FIFO list*. *FIFO* stands for *First-In-First-Out*.

Implementation of a queue

A queue (of bounded size) can be efficiently implemented in an array. Look at JavaHypertext entry "queue".

A queue can be efficiently implemented using any linked list that supports deletion in the front and insertion at the end in constant time. The first (last) element of the queue is at the front (end) of the linked list.

The stack (a LIFO list)

A *stack* is a list of items with two operations for changing it. We draw a stack with items *stacked* on top of each other. The stack to the right is the list [*x*, *c*, *y*]. Here are the two ways to change a stack.

```
y   top
c
x   bottom
```

1. push(*v*): Put *v* onto the stack. This changes the list [*x*, *c*, *y*] to [*x*, *c*, *y*, *v*].
2. pop(): Remove and return the top value of the stack. This changes the list [*x*, c, *y*] to [*x*, *c*] and returns *y*.

An example of a stack in real life is a stack of cafeteria trays. Workers add clean trays to the top, and you take the tray from the top of the stack. A stack is also called a *LIFO list*. *LIFO* stands for *Last-In-First-Out*.

Implementation of a stack

A stack (of bounded size) can be efficiently implemented using an array *b* and an **int** variable *n*: The *n* elements of the stack are in *b*[0..*n*-1], with *b*[0] being the bottom element and *b*[*n*-1] being the top element.

A stack can be efficiently implemented using a linked list. The first element is the top of the stack and the last element is the bottom. It's easy to push (prepend) an element and pop (remove) the first element in constant time.

The deque

The word *deque*, usually pronounced *deck*, is short for *double-ended queue*. A *deque* is a list that supports insertion and removal at both ends. Thus, a deque can be used as a queue or as a stack.

Stacks, queues, and deques in the Java Collection framework

Java has interface *Deque<E>*. It is implemented by classes *ArrayDeque<E>* (which implements a list in an expandable array) and *LinkedList<E>*, so these two classes can be used for a queue and for a stack.

Both *ArrayDeque* and *LinkedList* also implement interface *Queue<E>*, so you can use this interface to restrict operations to queue operations. For example, create a *LinkedList* and assign it to a *Queue* variable.

```
Queue<E> q= new LinkedList<>();
```

Thereafter, use only *q* for the *LinkedList* and operations are restricted to queue operations.

Java also has a class *Stack<E>*, which implements a stack in an expandable array. However, the Java API would rather you use an *ArrayDeque*. The problem is that there is no suitable way to restrict the operations of an *ArrayDeque* to stack operations, so we prefer to use class *Stack<E>*.