Lists, Stacks and Queues

Stacks and Queues

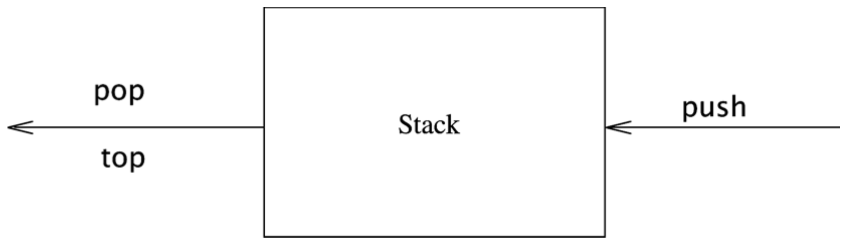
Stacks

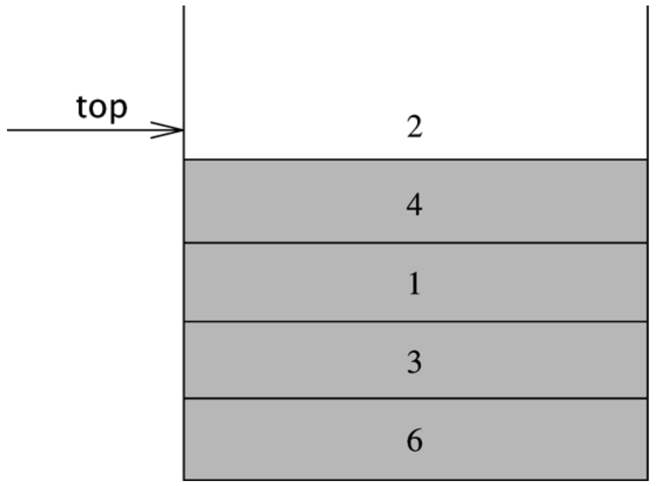
- A restricted list where insertions and deletions can only be performed at one location, the end of the list (top).
- LIFO – Last In First Out
  - Laundry Basket – last thing you put in is the first thing you remove
  - Plates – remove from the top of the stack and add to the top of the stack

Stack ADT

- Basic operations are push, pop, and top

Stack Model





Adapting Lists to Implement Stacks

- Adapter Design Pattern
- Allow a client to use a class whose interface is different from the one expected by the client
- Do not modify client or class, write adapter class that sits between them
- In this case, the List is an adapter for the Stack. The client (user) calls methods of the Stack which in turn calls appropriate List method(s).

Adapter Model for Stack

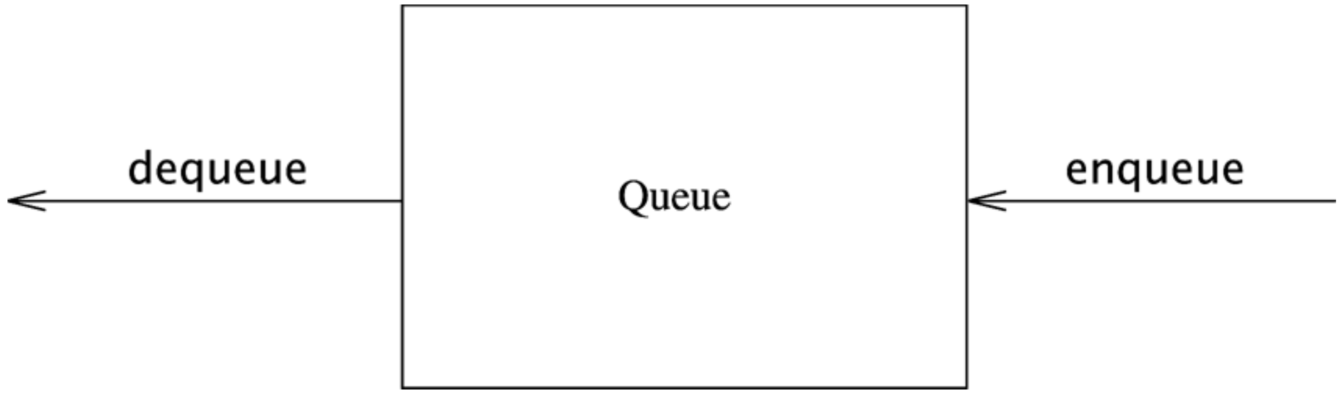
$$\begin{tikzcd}
\text{Client (Stack user)} \arrow[d, "\text{theStack.push(10)}"] \\
\text{Stack (adapter)} \arrow[d, "\text{theList.add(0, 10) ;}"] \\
\text{List (adaptee)}
\end{tikzcd}$$

Queues

- Restricted List
  - only add to head
  - only remove from tail
- Examples
  - line waiting for service
  - jobs waiting to print
- Implement as an adapter of List

Queue ADT

- Basic Operations are enqueue and dequeue



Adapter Model for Queue

$$\begin{tikzcd}
\text{Client (Queue user)} \arrow[d, "\text{theQ.enqueue(10)}"] \\
\text{Queue (adapter)} \arrow[d, "\text{theList.add(theList.size() -1, 10)}"] \\
\text{List (adaptee)}
\end{tikzcd}$$

Circular Queue

- Adapter pattern may be impractical
  - Overhead for creating, deleting nodes
  - Max size of queue is often known
- A circular queue is a fixed size array
  - Slots in array reused after elements dequeued

Circular Queue Data

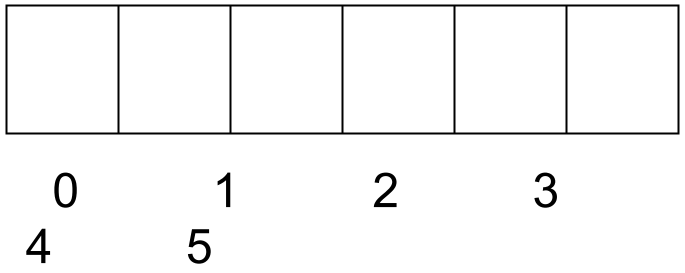
- A fixed size array
- Control Variables
  - arraySize
  - the fixed size (capacity) of the array
  - currentSize
  - the current number of items in the queue
  - Initialized to 0
  - front
  - the array index from which the next item will be dequeued.
  - Initialized to 0
  - back
  - the array index last item that was enqueued
  - Initialized to -1

Circular Queue Psuedocode

```
void enqueue( Object x ) {
        if currentSize == arraySize, throw exception             // Q
is full
        back = (back + 1) % arraySize;
        array[ back ] = x;
        ++currentSize;
}

Object dequeue( ) {
        if currentSize == 0, throw exception                     // Q
is empty
        --currentSize;
        Object x = array[ front ];
        front = (front + 1) % arraySize
        return x;
}
```

Circular Queue Example



Trace the contents of the array and the values of currentSize, front and back after each of the following operations.

1. enqueue(12)
2. enqueue(17)
3. enqueue(43)
4. enqueue(62)
5. dequeue( )
6. dequeue( )
7. enqueue(42)
8. dequeue( )
9. enqueue(33)
10. enqueue(18)
11. enqueue(99)