Queues in Ruby

- It is especially useful in threaded programming when information must be exchanged safely between multiple threads. The Queue class implements all the required locking semantics.
- The Queue class is used to synchronize communication between threads. You would use this if you were doing something with concurrency.

```ruby
require 'thread'
queue = Queue.new

producer = Thread.new do
  5.times do |i|
    sleep rand(i) # simulate expense
    queue << i
    puts "#{i} produced"
  end
end

consumer = Thread.new do
  5.times do |i|
    value = queue.shift
    sleep rand(i/2) # simulate expense
    puts "consumed #{value}"
  end
end
```

Methods

- new: creates a new queue
- <<: same as push.
- clear: remove all objects from queue.
- close: closes the queue. a closed queue cannot be re-opened.
- closed?: returns true if the queue is closed.
- empty?: returns true if the queue is empty.
- length: returns the size of the queue.
- num_waiting: returns the number of objects waiting on the queue.
- push: add an item to the end of the queue.
- shift: remove the first object from queue.
- pop: remove the last object from queue.
- size: same as length.

Stack versus Queue

Well, an array can be a stack or queue by limiting yourself to stack or queue methods (push, pop, shift, unshift).

- Stack behavior: Using **push / pop** gives **LIFO (last in first out)** behavior (stack).
- Queue behavior: Using **push / shift** gives **FIFO (first in first out)** behavior (queue).