Jeremy Asuncion
September 9th, 2015
CS146
Project 1 Write Up Questions

1. I found my previous knowledge on stacks and linked lists very helpful for this project.

2. I ran the Reverse program on the audio file for both list and array stack implementations, and for both double and generic versions.

3. I didn't use any other Java classes for my solution.

4. The underlying array starts at $n = 10$, and doubles whenever the array fills up. It follows the following pattern: $10, 20, 40, 80, 160, 320, ...$ This sequence has the following pattern:
$$a_n = 10 \cdot 2^n,$$
where a_n is the size of the array after n doublings.

   Assuming the array is size $n = 1,000,000$ for $1,000,000$ lines, then there should be about
$$\lg\left(\frac{1,000,000}{10}\right) \approx 17$$
doublings. For one billion lines, it's about 27 doublings, and for one trillion lines, it's about 37 doublings.

5. I'd implement a stack using the push and pop operations of the FIFO Queue class.

6.
Algorithm 1 Push and Pop Operations
```
procedure PUSH(Item x)
    queue.Enqueue(x)
end procedure
procedure POP
    if queue.IsEmpty() then
        Throw New EmptyStackException()
    end if
    Return queue.Dequeue()
end procedure
```

7. While using an instance of a FIFO queue would ease the burden of handling a stack, it could be slightly more inefficient to keep an instance for each stack instance. Rather, the better option would be to either implement a stack by hand, or to extend the FIFO queue class and create push and pop methods that wraps the enqueue and dequeue methods. Even then, that would add method call overhead, but it shouldn't be that bad.