Sam Lee

6/4/18

CSC 130

Professor Chidella

1. Who and what did you find helpful for this project?

The project gave me an understanding of how stacks really work. My classmates were very helpful in doing this project.

2. How did you test that your stack implementations were correct?

I ran Reverse.java to see if it gave me any errors after that I just made the new .dat file into a .wav file. I then played the file and heared if the message made any sense.

3.Other than java.util.EmptyStackException, did you use any classes from the Java framework or other class library?

(You will get a low score on this project if you use a library to implement your stacks.)

No I did not use any classes from the java framework or other class library besides what was in the Reverse.java that was given.

4. Your array stacks start with a small array and double in size if they become full. For a .dat file with 1 million lines, how many times would this resizing occur?

What about with 1 billion lines or 1 trillion lines (assuming the computer had enough memory)? Explain your answer.

1 million lines: The resizing would have occurred about 20 times.

1 billion lines: The resizing would have occurred about 30 times.

1 trillion lines: The resizing would have occurred about 40 times.

So if you start with an array with one element and doubles each time we can use 2^n.For example 0 resizing woulf be 1, 1 resizing would be 2, 2 resizing would be 4 and so on.

So if we do 2^n = 1,000,000 and we solve for n we get 19.93156857 which is about 20. We can do the same thing for the other numbers and get about 30 and 40 respectivly.

5. Instead of a DStack interface, you were given a fully-functional FIFO Queue class. How might you implement this project (i.e., simulate a Stack)

with one or more instances of a FIFO Queue?

I would use two queues one named q1 and the other q2. q1 will have n items I would dequeue on q1 and add it to the end by pushing the item until i have reached n-1.

this makes the last first then i would enqueue q2 with dequeue of q1 I would just loop this until q1 is empty

6. Write pseudocode for your push and pop operations. Assume your Queue class provides the operations enqueue, dequeue, isEmpty, and size.

void push (element d)

{

queue.enqueue(d); //Insert the element d in the queue

}

public double pop ()

{

n = queue.size(); // set the variable n to the size of the queue

for (i from 0 to n - 1) //Loop through the queue and move the element in the front to the back from index 0 to Index n - 1

{

queue\_element = queue.dequeue ();

queue.enqueue (queue\_element);

}

element = queue.dequeue(); // now store the element in the front of the queue then remove the element

return element; // return the element that was stored in the last step

}

7. Why would a stack implementation using a queue, as you described in the previous problem, be worse than your array and linked-list stack implementations?

If we do the dequeue, enqueue operation in the pop function, then it will have a execution growth upperbound of O(n) where n is the length of the queue,

and the push operation will have O(1) time bound.

If we do the array-based implementation in the pop function, then it will have an execution growth upper bound of O(1), and the push operation would have O(n) time bound.

We’ve noticed that the trade-offs between a Queue implementation of a Stack and the array-based implementation are that pop causes more processing time in Queue implementation

whereas push causes more time in Array-based implementation. I would choose Queue implementation if the program performs more push operations and choose Array-based implementation if the program performs more pop operations.