Tyler Waltze

7/19/15

MET CS 342

Homework 3

Assuming the following code exists:

```
public class Node<E> {
    public E data;
    public Node<E> link;
}

public class LinkedStack<E> implements Cloneable {
    private Node<E> top;

    // other methods here
}
```

1. Write the push method for the LinkedStack class. (25 points)

```
    public void push(Node item) {
        item.link = top;
        top = item;
    }
```

Assuming the following code exists:

```
public class Node<E> {
    public E data;
    public Node<E> link;
}

public class LinkedQueue<E> implements Cloneable {
    private int manyNodes;
    private Node<E> front;
    private Node<E> rear;

    // other methods here
}
```

1. Write the add method for the LinkedQueue class (25 points)

```
    public void add(Node item) {
        if (!isEmpty()) {
            rear.link = item;
```

```
        }

        rear = item;
    }
```

1. When writing a data structure, the design goal should be to minimize the work and error conditions for the client. Having said that, what mechanisms should/do stacks and queues provide to help clients not run into error conditions? Explain your answer. (25 points)

To some extent I disagree with the premise of the question. If the client decides to execute a pop() on a stack with no items in it, something bad is likely going to happen. In my opinion, the solution is not necessarily to provide just a function like isEmpty(). Yes, many stacks and queues currently provide this, and if a client uses it correctly then it can help keep things running smoothly. But this creates an inherent relationship between pop() and isEmpty() that requires them to always be used together, and that isn't always obvious to the client.

It's for that reason that I think a function like pop() should throw an exception, as should any function where it's possible for a similar error condition to be hit. Exceptions and checks like isEmpty() help solve the same issue, so both aren't necessary. Providing both though gives the client the option of personal preference. An if statement and a try/catch block both require the client to go out of their way to check for possible error conditions, and neither is more or less friendly.

1. If I wanted to loop through different kinds of collections in a common way what mechanism should be available for client use? (25 points)

Some sort of iterator function or object. This gives the client the ability to step through a collection of objects, individually, using a function such as next() or hasNext().