Lab 9: Linked Lists, Stacks, and Queues

You must make a `LinkedList` object that implements both the `Stack` interface and the `Queue` interface. (You should download these interfaces and compile them right away.) Your `LinkedList` class *must* use generics. In addition to the methods required by the interfaces, your it must contain the following public methods:

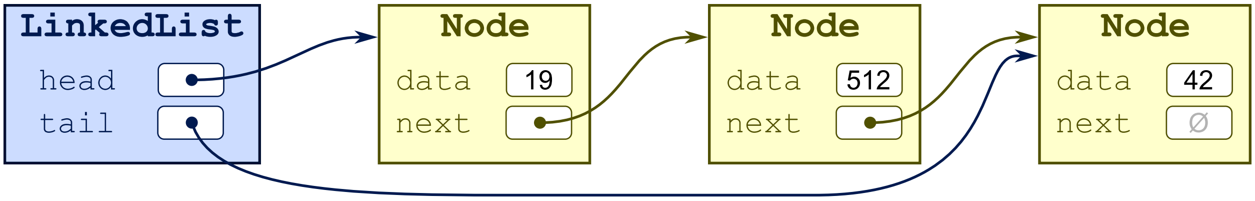
- `void addToHead(E value)`, which adds a new object to the head.
- `void addToTail(E value)`, which adds a new object to the tail.
- `boolean contains(E value)` which returns `true` if the passed value is to be found anywhere in the list. Note that this should return `true` if it encounters any object with the same value (i.e. it uses the object's `equals()` function).
- `E getHead()`, which returns the head value, without actually removing it.
- `int getSize()`, which returns the size of the `LinkedList`.
- `E removeHead()`, which removes and returns whatever object was at the head, and updates the head (and possibly the tail). Return `null` if the list was empty.
- `String toString()`, which returns a `String` representing this `LinkedList`, with each object in parentheses separated by "arrows" (`->`). (The `LinkedList` in the figure would return the `String` `"(19) -> (512) -> (42)"`.)
- `static void main(String[] args)`, which is a `main()` function for unit testing (see below).

All of these functions must work in constant time, except `contains()` and `toString()`. These will work in O(n) time. (This means that you will need to keep track of the list's tail, as well as the head.)

In addition, you will probably want to make an internal `Node` object, that holds both the internal object as well as a pointer to the next node in the list. This object might contain a contructor and its own `toString()` method.

Since the `LinkedList` must implement both the `Stack` interface and the `Queue` interface, you will need to create other methods not shown above. However, each of these methods will be one-liners, that just call one of your methods.

The `main()` function should declare a `Stack` variable, a `Queue` variable, and a plain `LinkedList` variable. All three should use different types for `E`. All will be allocated as `LinkedList`s. Use the `Stack` to test all the `Stack` methods, the `Queue` to test the `Queue` methods, and the `LinkedList` to test the remaining methods.


LinkedList
head
tail
Node
data
19
next
Node
data
512
next
Node
data
42
next
∅