CmpSci 187 Discussion #5: Implementing Stacks with Linked Lists Individual Handout

Marc Liberatore and John Ridgway

23 February 2015

Our goal is to implement the stack abstraction, given a specification in the form of an interface, and to use that implementation to reverse an input.

The following classes are in the `implementing-stacks.zip` code zip file on the course web site:

- a generic `Stack` interface
- a complete `StackUnderflowException` class
- the generic `LLNode` class used in DJW to implement linked lists
- a stub for a generic `LinkedListStack` that you will use to implement the `Stack` interface
- a minimal `Driver` class that uses your `LinkedListStack` to reverse a sequence of input strings

Note that we are using the more traditional definition of a `Stack`, so that an invocation of `pop` both removes and returns the top element.

Our idea is to reverse an input consisting of arbitrarily many strings, terminated by a single `"."`, by pushing each string onto a stack, and then popping and outputting each string.

Your task is to implement the following methods in `LinkedListStack<T>`:

- `public LinkedListStack()`, which constructs a new, empty stack;
- `public T pop() throws StackUnderflowException`, which removes and returns the top element of the stack;
- `public T peek() throws StackUnderflowException`, which returns the top element of the stack;
- `public void push(T element)`, which pushes `element` onto the stack; and
- `public boolean isEmpty()`, which returns `true` if and only if the stack is empty.

Transcribe your `Stack` implementation onto the back of this sheet, or print it out and hand it in. Be sure to write your name and your TA's name on it!