Tuesday, April 3

Tests

Next test is April 12
Cumulative with emphasis on content since first test
Through chapter 7 - queues

Symposium - April 10, 7pm, Roddy 149
Guest speaker on Big Data

Warmup

Consider a newly created character sequence with linked list using our implementation. After each method call, show the sequence’s state as represented by its instance variables. Be sure to show all the instance variables. Methods have cumulative effects.
0) ... constructed ....
1) addAfter('r');
2) addAfter('k');
3) start();
4) advance();
5) addBefore('s');
6) advance();
7) removeCurrent();
8) start();
9) removeCurrent();

seqLL assignment

due April 9 - next Monday

tips

work on it incrementally

draw pictures of situations (see warmup)

addAll

consider the problem of empty sequences
consider the problem of B.addAll(B)

make copy (copy/clone) before modifying existing list
concatenation (call clone and addAll to implement)

remove

make sure you set ALL instance variables appropriately

iterating over a collection (p. 297)

visit all the elements in a container
if class implements Iterator interface (see 3/29 notes),
can use the enhanced for loop

for (type of the collection elements item : name of the collection) {
    // do something with item
}

for example
for (String person : people) {
    System.out.println(person);
}
Stacks (chapter 6)
classic data structure accessed from one end
last-in, first-out - LIFO

stack operations:
E pop() pops/removes top item from the stack and returns it
push(E item) pushes/adds item onto top of stack
boolean isEmpty() returns true if stack is empty
int size() returns number of items in stack
E peek() returns reference to top item without removing it;
    traditionally top()

Uses:
- reverse
- activation stack
- postfix expression evaluation

Reverse:
// read ch and s.push(ch)
while (!s.isEmpty()) {
    System.out.print(s.pop());
}
System.out.println();

Postfix Expression Evaluation (p. 345):
(4 * 5) + (6 * 3) -> in-fix notation
post-fix notation -> 4 5 * 6 3 * +
sometimes called reverse Polish
to evaluate a postfix expression
    initialize a stack of doubles
    while (there is input) {
        if input is a digit
            push it onto the stack
        else // it's an operation
            use pop to get two values off the stack (2nd number is left)
            apply the operator to the two values (7 4 - results in 3)
            push the result onto the stack
        if there is only one item remaining on top stack
            answer to expression is on top of stack
    }
1 7 3 + 9 5 + * 8 5 * - +

Errors on stacks:
Stack Overflow - not enough space
Stack Underflow - not enough items
Implementation:

array

Stack implemented with an array
pop at data[used-1]
push at data[used]
first push at data[0]

linked list

Stack implemented with a linked list
pop and push at top