COMP 250

Lecture 8

stack

Sept. 25, 2017
What is a List (abstract) ?

get(i) // Returns the i-th element (but doesn't remove it)
set(i,e) // Replaces the i-th element with e
add(i,e) // Inserts element e into the i-th position
remove(i) // Removes the i-th element from list
remove(e) // Removes first occurrence of element e
            // from the list (if it is there)
clear() // Empties the list.
isEmpty() // Returns true if empty, false if not empty.
size() // Returns number of elements in the list

This operations are defined without specifying the implementation details of the data structure (arraylist, linked list).
Abstract data type (ADT)

“ADT” defines a data type by the values and operations from the user’s perspective only.

It ignores the details of the implementation.

An ADT is more abstract than a data structure.
A stack is a list. However, it typically does not have operations to access the list element \( i \) directly.
How to implement a stack?

push(e)       pop ()

array list
singly linked list
doubly linked list
How to implement a stack?

push(e)               pop ()

array list
addLast(e)          removeLast()

singly linked list

doubly linked list
How to implement a stack?

<table>
<thead>
<tr>
<th></th>
<th>push(e)</th>
<th>pop ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>array list</td>
<td>addLast(e)</td>
<td>removeLast()</td>
</tr>
<tr>
<td>singly linked list</td>
<td>addFirst(e)</td>
<td>removeFirst ()</td>
</tr>
<tr>
<td>doubly linked list</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How to implement a stack?

<table>
<thead>
<tr>
<th></th>
<th>push(e)</th>
<th>pop ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>array list</td>
<td>addLast(e)</td>
<td>removeLast()</td>
</tr>
<tr>
<td>singly linked list</td>
<td>addFirst(e)</td>
<td>removeFirst ()</td>
</tr>
<tr>
<td>doubly linked list</td>
<td></td>
<td>either row above</td>
</tr>
</tbody>
</table>
Example 1: stack of int

push(3)
push(6)
Example 1: stack of int

push(3)
push(6)
push(4)
push(1)
pop()
Example 1: stack of int

push(3)
push(6)
push(4)
push(1)
pop()
push(5)
pop()
pop()
Example 1: stack of int

push(3)
push(6)
push(4)
push(1)
pop()
push(5)
pop()
pop()

1              5
4      4      4       4      4
6     6      6      6       6      6     6
3     3     3      3      3       3      3     3

time
Example 2 - balancing parentheses

e.g. ( ( [ ] ) ) [ ] { [ ] }

To ensure proper nesting, we traverse the list and use a stack.

How?
Example 2 - balancing parentheses

e.g. \(( ( [ ] ) ) [ ] \{ [ ] \})

To ensure proper nesting, we traverse the list and use a stack.

We push left parentheses on the stack.

When we reach a right parenthesis, we compare it to top of the stack.
Example 2 - balancing parentheses

e.g. $\left( \left( \left[ \right]\right) \right) \left[ \right] \left\{ \left[ \right] \right\}$
Example 2 - balancing parentheses

e.g. \(( ( [ ) ] ) \{ [ ] \}\)

Does not match left bracket on top of stack.

BTW, each of bracket types is balanced in this example.
We refer to brackets as “tokens”. This is the more general term using in string parsing.

Algorithm: decide is parentheses are matched.
If yes, return true, else return false.

while (there are more tokens) {
    token = get next token
    if token is a left parenthesis
        push(token)
    else {                                         // token is a right parenthesis
        if stack is empty
            return false
        else {
            pop left parenthesis from stack
            if popped left parenthesis doesn’t match the right parenthesis
                return false
        }
    }
}

return stack.empty // true if stack is empty, false if not.
Example 3: HTML tags

\(<b> I am bold. </b> <i> I am italic. </i>\)

I am bold.  I am italic.
HTML Elements

An HTML *element* starts with a start tag. An HTML *element* ends with an end tag.

HTML documents consist of nested HTML *elements*.

```html
<html>
<body>
  <b> I am bold </b>
  <i> I am italic </i>
</body>
</html>
```

These tags can be thought of as brackets.
Suppose you want:

| I am bold. | I am bold and italic. | I am italic. |

What if you were to write the following?

<b> I am bold. </b>  
<i> I am bold and italic. </i>  
</b> I am italic. </i>
Suppose you want:

**I am bold.  I am bold and italic.  I am italic.**

What if you were to write the following?

<b> I am bold.  <i> I am bold and italic. </i>  I am italic. </b>

This is *officially* incorrect, because elements are not nested.

___  <i> </i>  
___  <b>  <b>  Error:  mismatch between <i>  </i>  </b>

Most web browsers will interpret it correctly, however.
I am bold.  I am bold and italic.  I am italic.

The correct way to write it is:

<b> I am bold.  <i> I am bold and italic. </i> </b>  <i> I am italic. </i>
What problems can arise if you write it incorrectly?

Suppose you are editing a html document that contains the following:

... Hello.  <b> I am bold. 

<i> I am bold and italic. </b> I am italic.  </i>

Bla bla bla ......

Q: What happens if you delete the middle line ?
What problems can arise if you do not write it correctly?

Suppose you are editing a html document that contains the following:

```html
... Hello. <b> I am bold.

<i> I am bold and italic. </b> I am italic. </i>

Bla bla bla ......
```

Q: What happens if you delete the middle line?

A: ... Hello. I am bold. Bla bla bla ......
Example 4: Stacks in Graphics

Define a ‘programming language’ for drawing simple figures like this:
Define a pen position and direction \((x, y, \theta)\) where \(\theta\) is clockwise degrees from x axis.

The initial state of the pen is \((0, 0, 0)\).
Let instructions be symbols:

D - draw unit length line in direction \((x, y)\) (changes \((x, y)\))

R - turn right 90 degrees clockwise (changes \(\theta\))

L - turn left 90 degrees counterclockwise (changes \(\theta\))

[ - push state \((x, y, \theta)\)

] - pop state, and go to that state
The initial state of the pen is \((0, 0, 0)\).

The final pen state is \((3, -1, 0)\).
The initial state of the pen is \((0, 0, 0)\).

Q: What will be the final pen state?
The initial state of the pen is \((0, 0, 0)\).

Q: What will be the final pen state?

A: \((2, 0, 0)\)
The initial state of the pen is \((0, 0, 0)\).

Q: What will be the final pen state?
The initial state of the pen is \((0, 0, 0)\).

Q: What will be the final pen state?
A: \((3, 1, 0)\)
The initial state of the pen is \((0, 0, 0)\).

Q: What if we add brackets at beginning and ending?

\[
\left[ D D \left[ R D L D \right] L D R D \right]
\]

D - draw
R - turn right 90 deg
L - turn left 90 deg
[ - push state
] - pop state
The initial state of the pen is \((0, 0, 0)\).

Q: What if we add brackets at beginning and ending?

\[
[ D \ D \ [ \ R \ D \ L \ D ] \ ] \ L \ D \ R \ D ]
\]

A: The pen state will return to \((0, 0, 0)\).
Example 5a: stack of tasks

As I work in my office, emails arrive, the phone rings, people drop by, ..... 

To make sure items all get finished, I must keep a stack. (“What was I doing when ....? “ )
Example 5b: "Call Stack"

class Demo {
    void mA() {
        mB();
        mC();
    }
    void mB() { ... }
    void mC() { ... }

    void main() {
        mA();
    }
}

class Demo {
    void mA() {
        mB();
        mC();
    }
    void mB() { ... }
    void mC() { ... }

    void main() {
        mA();
    }
}
TestSLinkedList1’s main() method calls addLast() method of SLinkedList class.
Eclipse debug mode

Call stack:
- `SLinkedList1<E>.addLast(E)` line: 85
- `TestSLinkedList1.main(String[])` line: 14

Breakpoint at line 85 in `SLinkedList1`
Quiz 1 is today

8 AM to 8 PM