COMP 250

Lecture 7

stack ADT

Sept. 21, 2016
Stack ADT (abstract data type)

push( element )
pop( )
isEmpty( )
peek( )

A stack is a list. However, one typically is not allowed to access the element directly.
How to implement a stack?

array list:

push(e) = ?

pop() = ?

singly linked list:

push(e) = ?

pop() = ?

doubly linked list:

push(e) = ?

pop() = ?
How to implement a stack?

array list:

push(e) = addLast(e)
pop() = removeLast()

singly linked list:

push(e) = addFirst(e)
pop() = removeFirst()

doubly linked list:

// either of the above
Example 1: stack of int

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

push(3)
push(6)
push(4)
push(1)
pop()
push(5)
pop()
pop()
Example 2 - balancing parentheses

To ensure proper nesting, we need a stack. We push only left parentheses on the stack.
Example 2a - balancing parentheses

To ensure proper nesting, we need a stack. We push only left parentheses on the stack.

[ ] [ ]
[ ( ( [ ] ) [ ] ] ]

Not empty
Not balanced
Example 2b - balancing parentheses

\[
\left( \left( \left[ \right] \right) \right) \left[ \right] \left\{ \left[ \right] \right\} \right)
\]

does not match

(Each type parenthesis on its own is balanced.)
we refer to brackets as “tokens”

Algorithm: decide if 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 if token is a right parenthesis {
        if stack.isEmpty()
            return false
        else {
            poppedleft = pop()
            if !(poppedleft.matches(token))
                return false
        }
    }
}
return stack.isEmpty()
e.g. 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.  \textit{I am bold and italic.}  \textit{I am italic.}

What if you were to write the following?

\textit{\textbf{I am bold. \textbf{I am bold and italic.} \textit{I am italic.}}}
Suppose you want:

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

What if you were to write the following?

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

This is incorrect, because elements are not nested.

```html
<i> </i>
<i> </i>
<i> </i>
```

Error: mismatch between `<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:

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

Goodbye .......

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:

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

Goodbye .......

Q: What happens if you delete the middle line?

A: ... Hello. I am bold. Goodbye ......
Example 3: IF - THEN - ELSE

Suppose a language allows statements of the form:

```
if bool then statement else statement
```

Such a language allows *nested* if-then-else statements, e.g.

```
if (i > 0) then if (a > 0) then b = 4 else b = 5 else c = 2
```

You can use a stack to “parse” such statements (Assignment 2).
Example 4: Stacks in Graphics

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

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 90 degrees clockwise (changes \(\theta\))

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

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

] - pop state
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?

\[
[ D \ D \ [ \ R \ D \ L \ D ] \ L \ D \ R \ D ]
\]
The initial state of the pen is \((0, 0, 0)\).

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

\[
[\text{D D [ R D L D ] L D R D }]
\]

A: The pen state will return to \((0, 0, 0)\).
This line draws figure on previous slide

\[
\begin{array}{c}
[L\ D \\
R\ [D\ D\ [\ R\ D\ L\ D]\ L\ D\ R\ D]\ L\ D\ R\ D]
\end{array}
\]
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( );
    }
}

main main main main main main main
Eclipse debug mode

TestSLinkedList classes’es main() method calls addLast() method of SLinkedList1 class.
Eclipse debug mode

Call stack:

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

Breakpoint:

```java
/**
 * add a new element to the end of the list
 * @param element the new element
 */

public void addLast(E element) {
    SNode<E> newNode = new SNode<E>(element);
    size++;
    if (head == null) {
        head = newNode;
        tail = newNode;
    }
```