

COMP 250

Lecture 7

stack ADT

Sept. 21, 2016

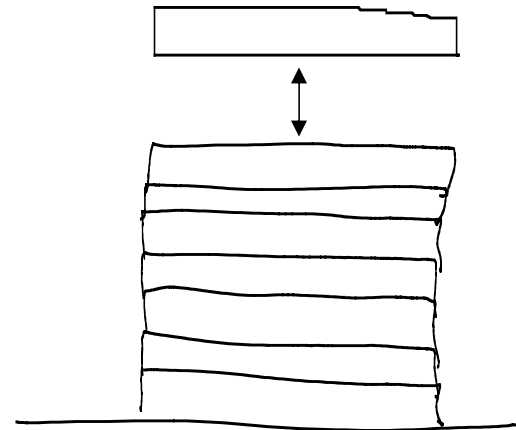
# Stack ADT (abstract data type)

push( element )

pop( )

isEmpty( )

peek( )



A stack is a list. However, one typically is not allowed to access the element  $i$  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

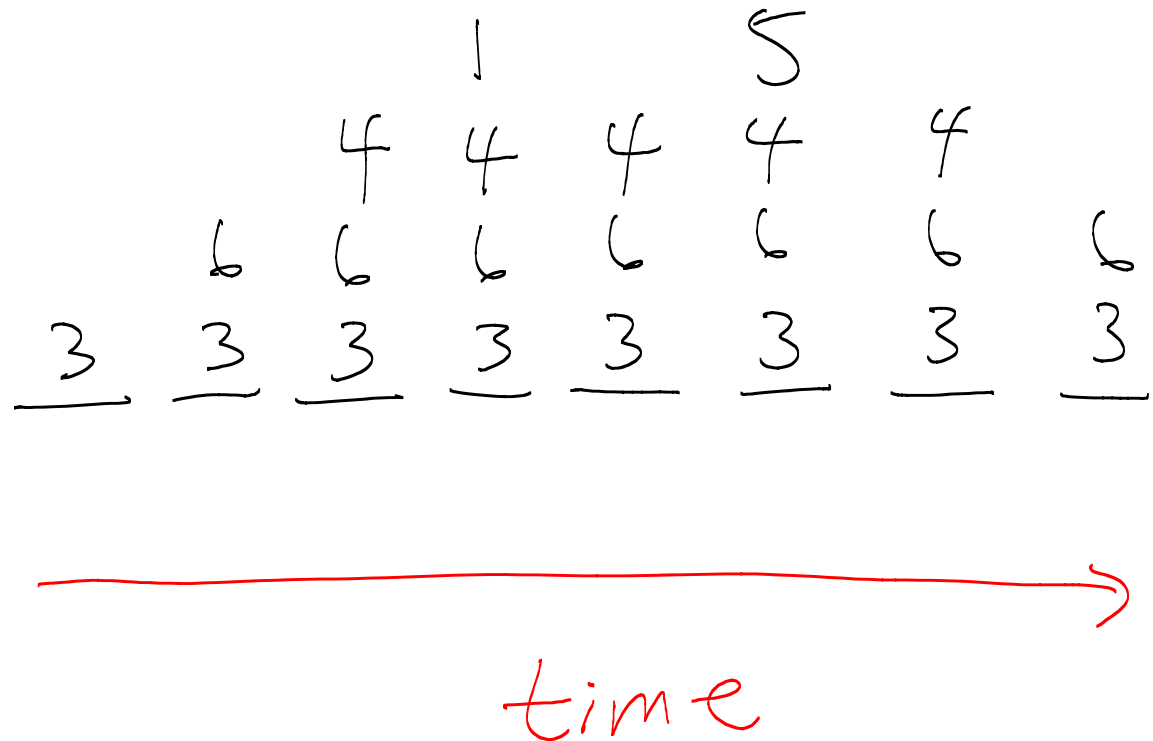
## Example 1: stack of int

push( 3 )  
push ( 6 )  
push ( 4 )  
push ( 1 )  
pop ( )  
push ( 5 )  
pop ( 2 )  
pop ( )

time →

# 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

( ( [ ] ) [ ] { [ ] }

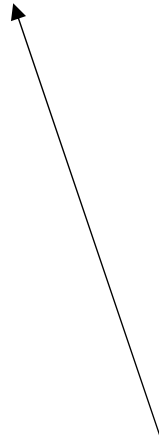
To ensure proper nesting, we need a stack.  
We push only left parentheses on the stack.

not empty  
not  
balanced



## Example 2b - balancing parentheses

( ( [ ) ] [ ] { [ ] } )



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

The diagram illustrates the nesting of HTML tags. A large red left bracket groups the entire code block. A smaller red left bracket groups the inner content, starting from the opening <body> tag and ending at the closing </body> tag. Two small red square brackets are used to highlight the opening and closing tags for the bold and italic elements: one for <b> and </b>, and another for <i> and </i>.

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

This is incorrect, because elements are not nested.

\_\_\_\_\_ `<b>` `<b>` `<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:

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

\_\_\_\_\_ < b > < b > < b > \_\_\_\_\_ < i > \_\_\_\_\_

< 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 >

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.

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

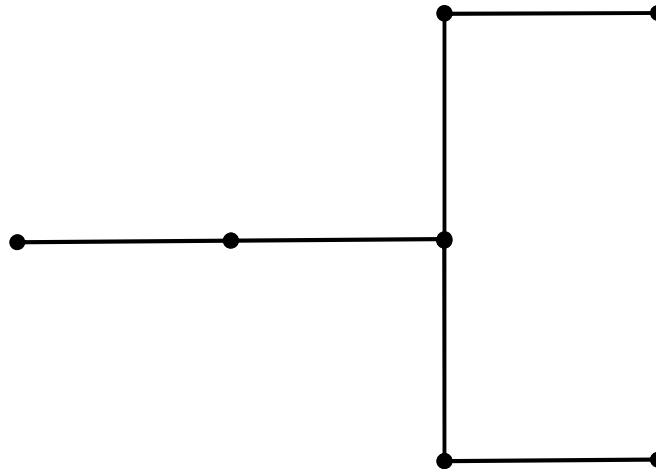
if (i > 0) then if (a > 0) then b = 4 else b = 5 else c = 2



You can use a stack to “parse” such statements (Assignment 2).

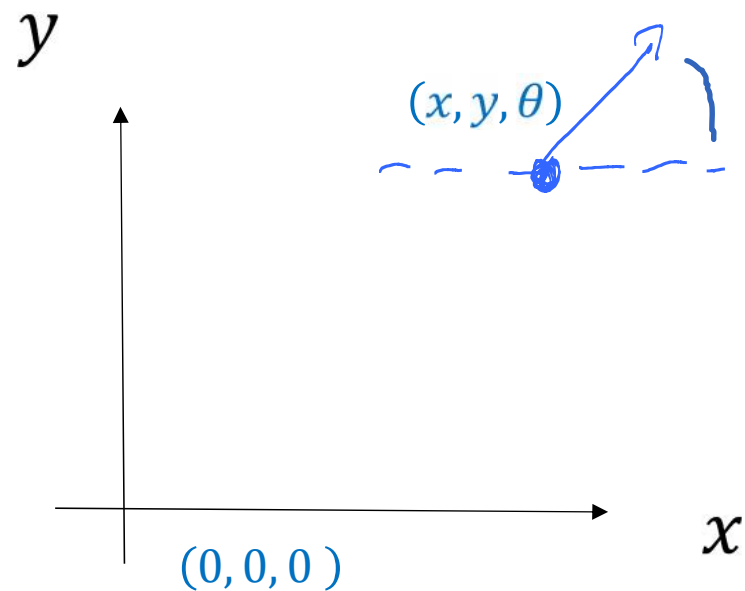
## Example 4: Stacks in Graphics

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 (changes  $(x, y)$  )

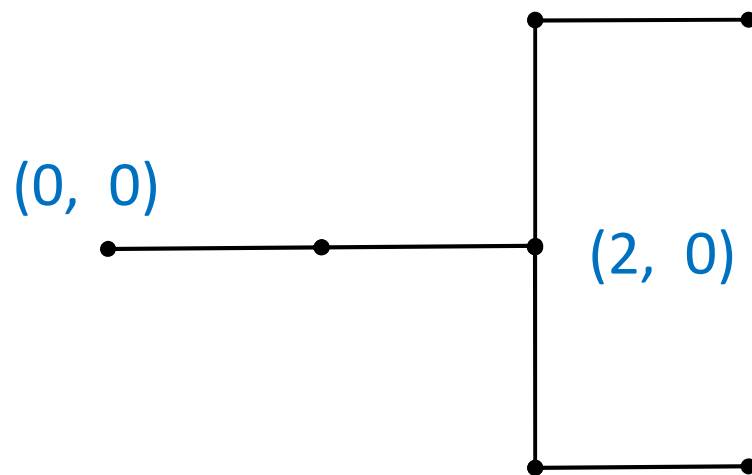
R - turn 90 degrees clockwise (changes  $\theta$  )

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

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

] - pop state

The initial state of the pen is  $(0, 0, 0)$ .



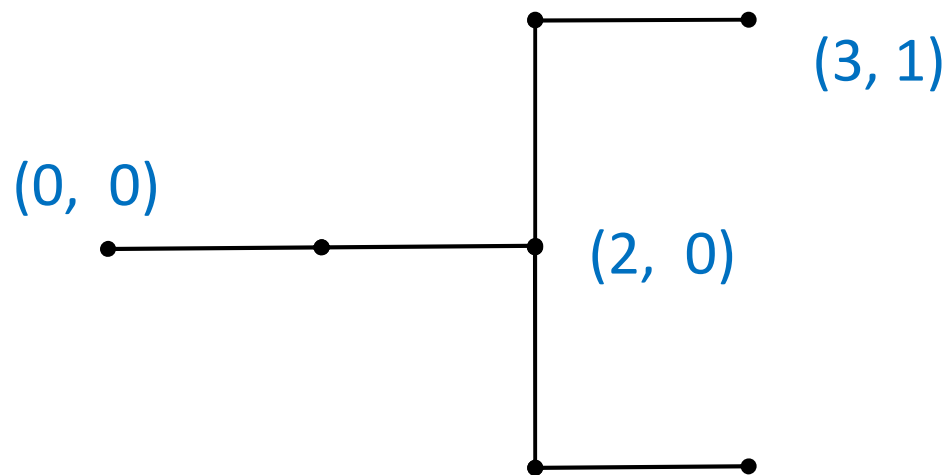
D - draw  
R - turn right 90 deg  
L - turn left 90 deg  
[ - push state  
] - pop state

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

\_\_\_\_\_  $(2, 0, 0)$  \_\_\_\_\_

Q: What will be the final pen state ?

The initial state of the pen is  $(0, 0, 0)$ .



D - draw  
R - turn right 90 deg  
L - turn left 90 deg  
[ - push state  
] - pop state

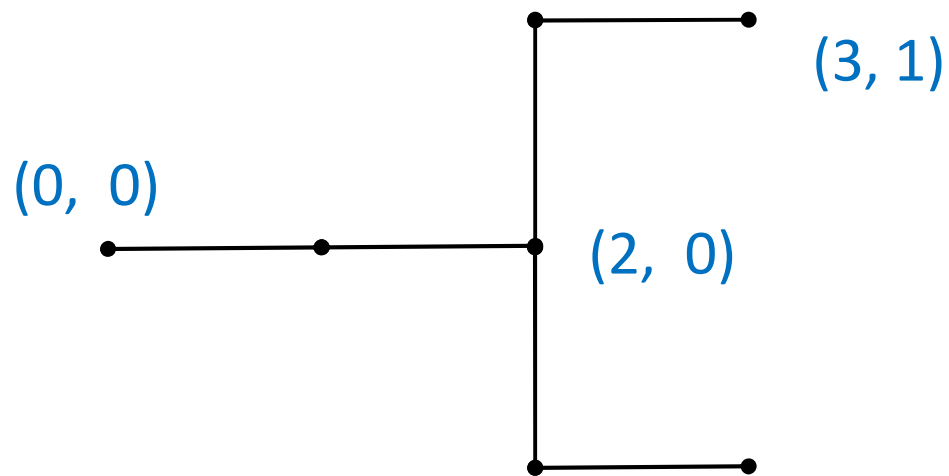
D D [ R D L D ] L D R D

\_\_\_\_\_  $(2, 0, 0)$  \_\_\_\_\_

Q: What will be the final pen state ?

A:  $(3, 1, 0)$

The initial state of the pen is  $(0, 0, 0)$ .



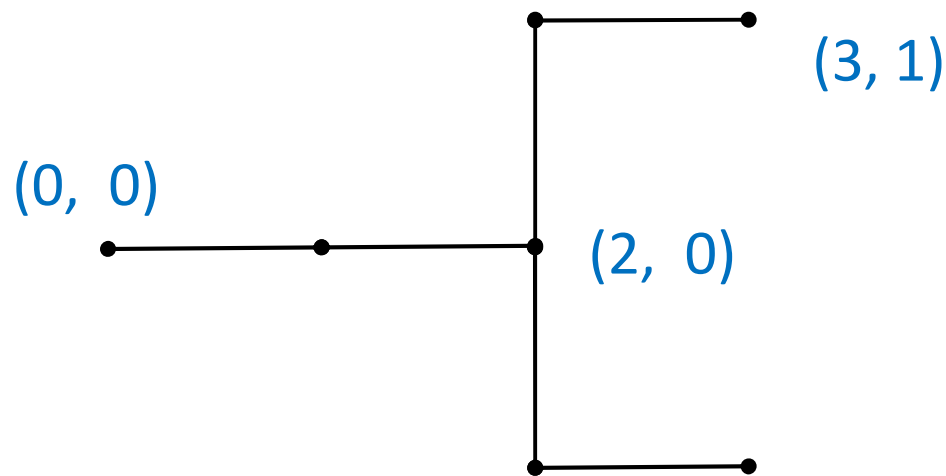
D - draw  
R - turn right 90 deg  
L - turn left 90 deg  
[ - push state  
] - pop state

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  $(0, 0, 0)$ .

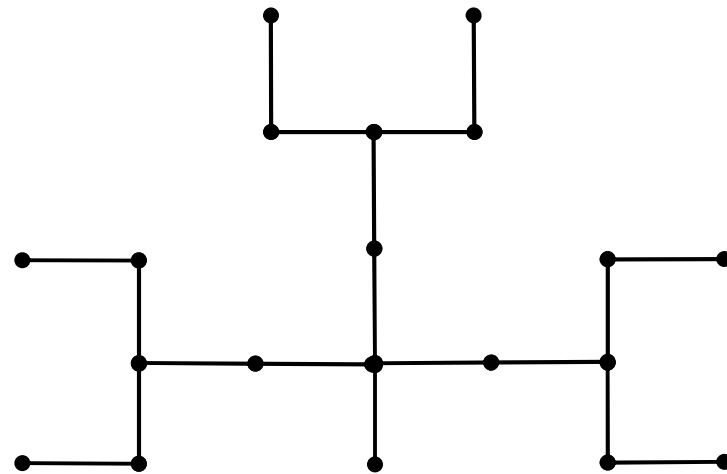


D - draw  
R - turn right 90 deg  
L - turn left 90 deg  
[ - push state  
] - pop state

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  $(0, 0, 0)$ .



(0, 0)

[ L D

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

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

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

This line draws figure on previous slide

## 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( );
    }
}

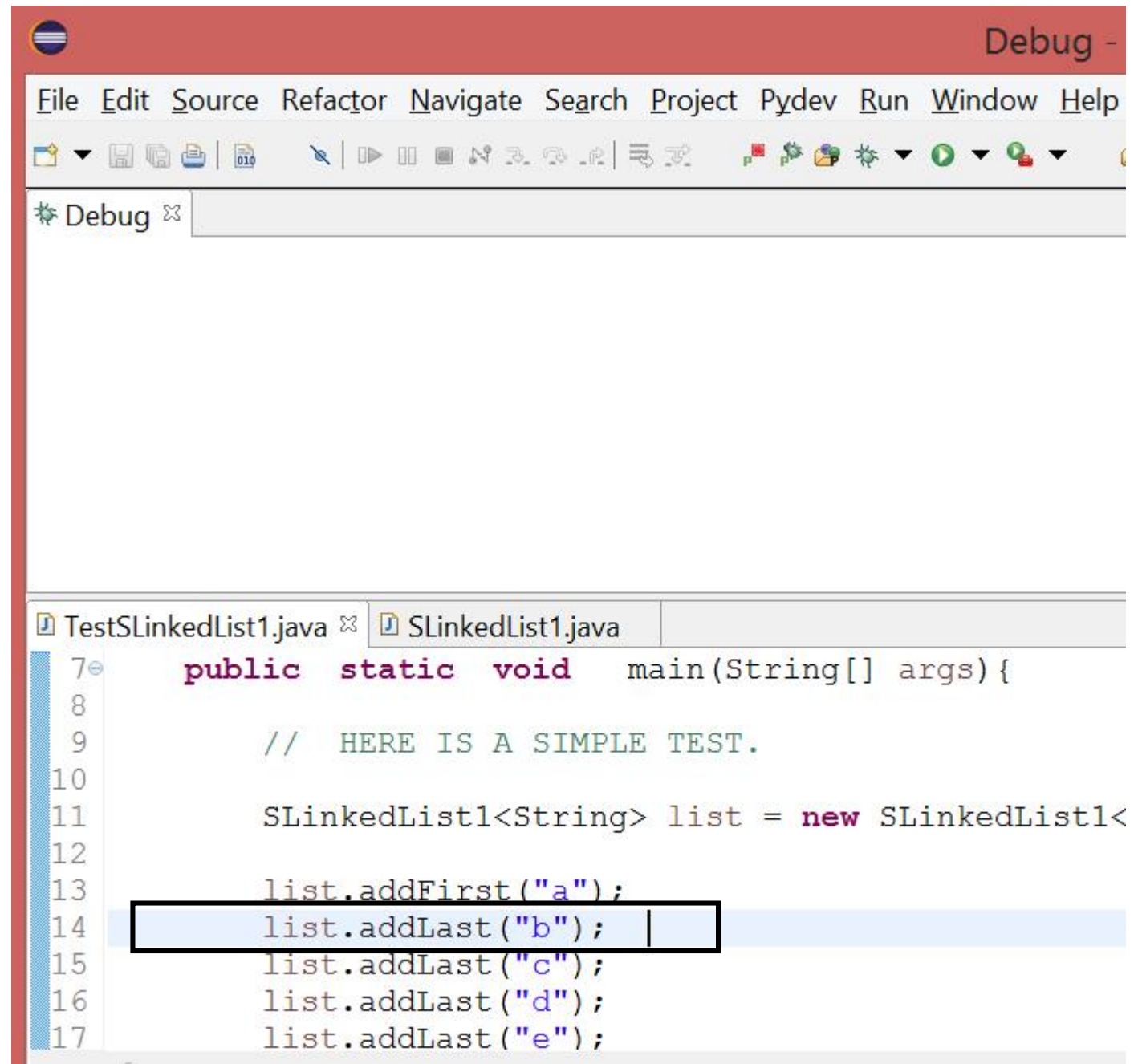
```

		mB		mC		
	mA	mA	mA	mA	mA	
<u>main</u>	<u>main</u>	<u>main</u>	<u>main</u>	<u>main</u>	<u>main</u>	<u>main</u>



## Eclipse debug mode

TestSLinkedList  
classes'es main()  
method calls  
addLast() method of  
SLinkedList1 class.



The screenshot shows the Eclipse IDE interface in Debug mode. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Pydev, Run, Window, and Help. Below the menu bar is a toolbar with various icons. The main editor area displays the code for TestSLinkedList1.java. The code is as follows:

```
7 public static void main(String[] args){
8
9     // HERE IS A SIMPLE TEST.
10
11     SLinkedList1<String> list = new SLinkedList1<
12
13     list.addFirst("a");
14     list.addLast("b");
15     list.addLast("c");
16     list.addLast("d");
17     list.addLast("e");
```

The line `list.addLast("b");` is highlighted with a blue background and a black border, indicating it is the current line of execution in the debug mode.

Eclipse debug mode

call stack

breakpoint

The screenshot shows the Eclipse IDE in Debug mode. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Pydev, Run, Window, and Help. Below the menu is a toolbar with various icons. The main window is divided into two panes. The top pane, titled 'Debug', shows the call stack for the application 'TestSLinkedList1 [Java Application]'. The stack includes 'linkedslists.TestSLinkedList1 at localhost:52013' and 'Thread [main] (Suspended (breakpoint at line 85 in SLinkedList1))'. The call stack is expanded to show two frames: 'SLinkedList1<E>.addLast(E) line: 85' and 'TestSLinkedList1.main(String[]) line: 14'. The bottom pane shows the source code for 'TestSLinkedList1.java' and 'SLinkedList1.java'. The code for 'SLinkedList1.java' is visible, showing a method 'addLast' with a breakpoint set at line 85, which is highlighted in green. The code includes a comment for adding a new element to the end of the list and a public void method 'addLast' that creates a new node and updates the head and tail pointers.

Debug -

File Edit Source Refactor Navigate Search Project Pydev Run Window Help

Debug

TestSLinkedList1 [Java Application]

linkedslists.TestSLinkedList1 at localhost:52013

Thread [main] (Suspended (breakpoint at line 85 in SLinkedList1))

SLinkedList1<E>.addLast(E) line: 85

TestSLinkedList1.main(String[]) line: 14

C:\Program Files\Java\jre7\bin\javaw.exe (Sep 19, 2016, 4:14:02 PM)

TestSLinkedList1.java SLinkedList1.java

```
78 /**
79  * add a new element to the end of the list
80  * @param element the new element
81  */
82
83 public void addLast(E element) {
84     SNode<E> newNode = new SNode<E>(element);
85     size++;
86     if (head == null) {
87         head = newNode;
88         tail = newNode;
```