COMP 250
INTRODUCTION TO COMPUTER SCIENCE

Lecture 15 – Stacks

Roman Sarrazin-Gendron, Winter 2020

Slides very much based on Michael Langer and Giulia Alberini

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).

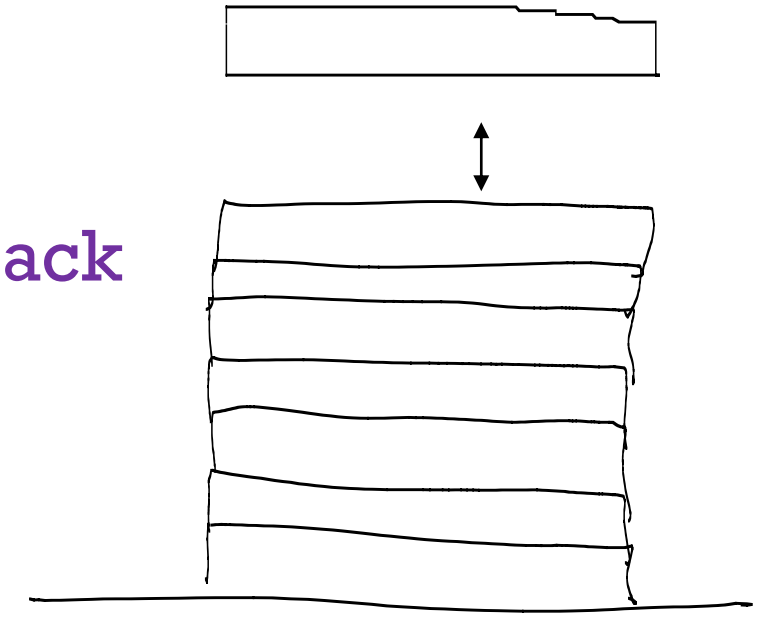
ABSTRACT DATA TYPE (ADT)

- "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.

STACK ADT

push (element) -> add element to top of stack

pop( ) -> remove and return element at top of stack



isEmpty( )

peek( )

A stack is a list. However, it typically does not have operations to access the list element *i* directly.

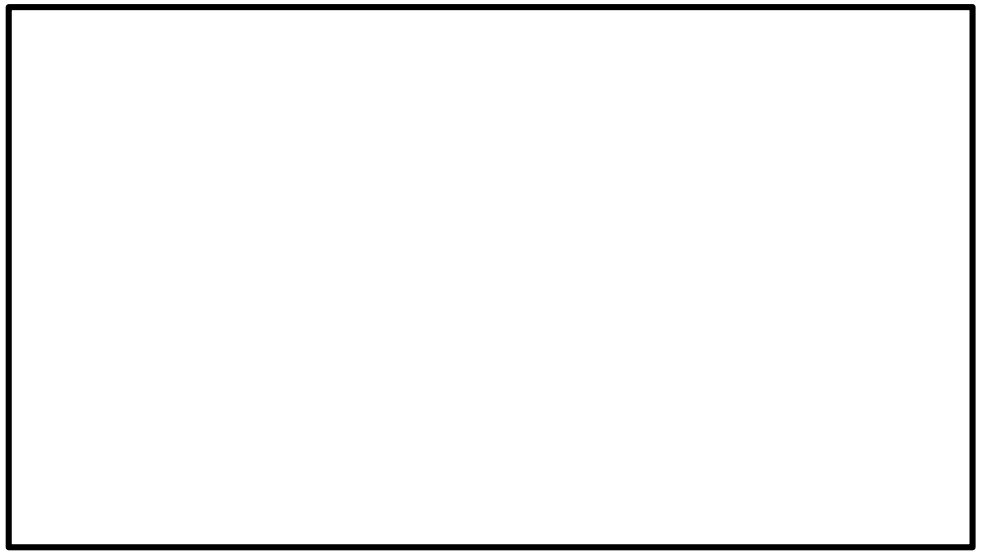
HOW TO IMPLEMENT A STACK?

push(e) pop ()

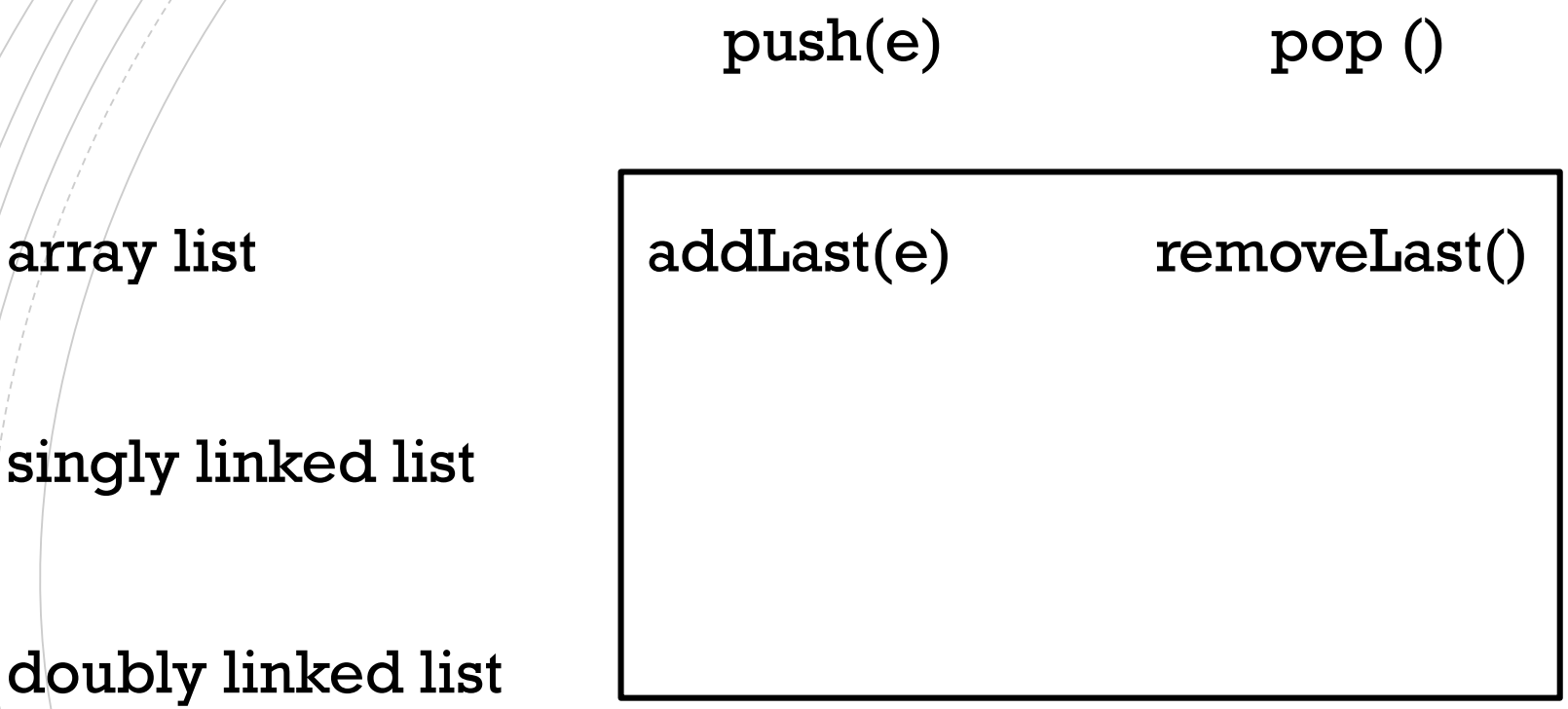
array list

singly linked list

doubly linked list



HOW TO IMPLEMENT A STACK?



HOW TO IMPLEMENT A STACK?

	push(e)	pop ()
array list	addLast(e)	removeLast()
singly linked list	addFirst(e)	removeFirst()
doubly linked list		

HOW TO IMPLEMENT A STACK?

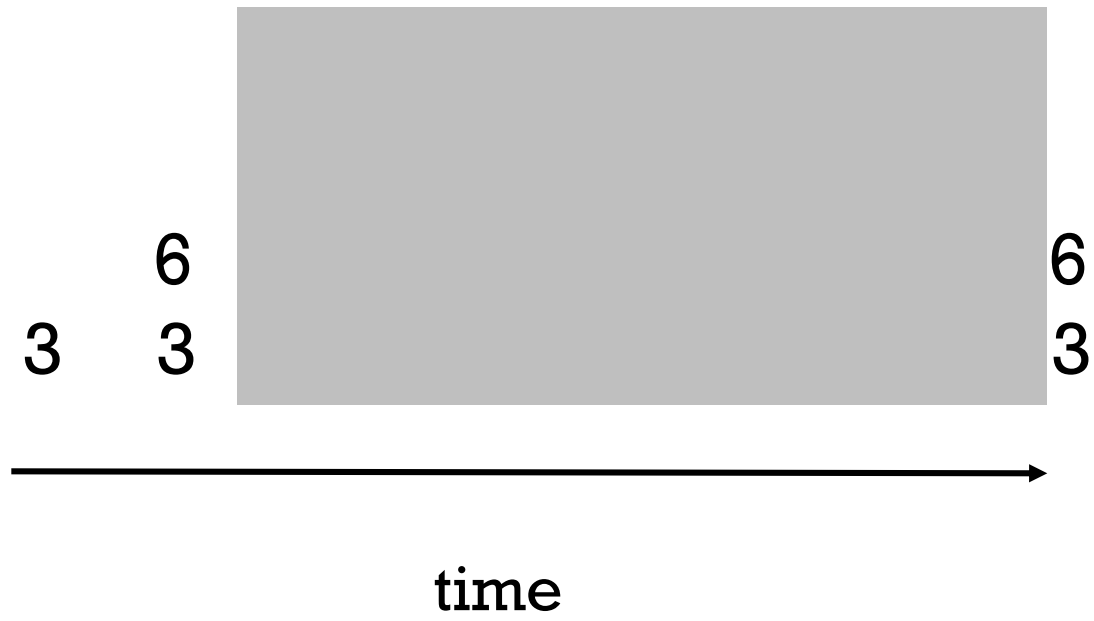
	push(e)	pop ()
array list	addLast(e)	removeLast()
singly linked list	addFirst(e)	removeFirst ()
doubly linked list	either row above	

EXAMPLE 1: STACK OF INT

```
push(3)
push(6)
```

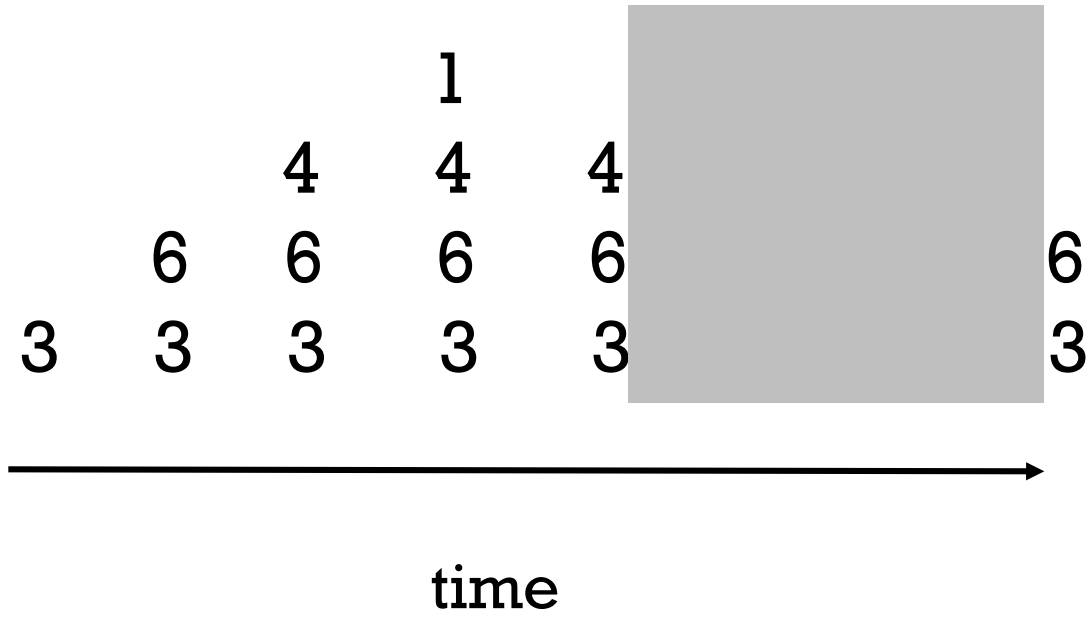
EXAMPLE 1: STACK OF INT

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



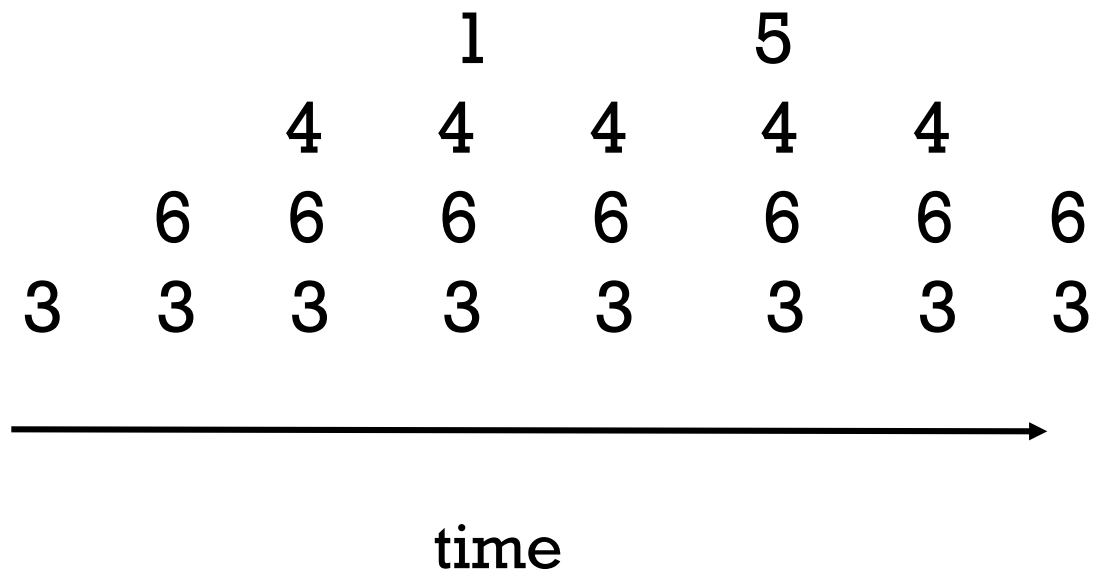
EXAMPLE 1: STACK OF INT

```
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()
```



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

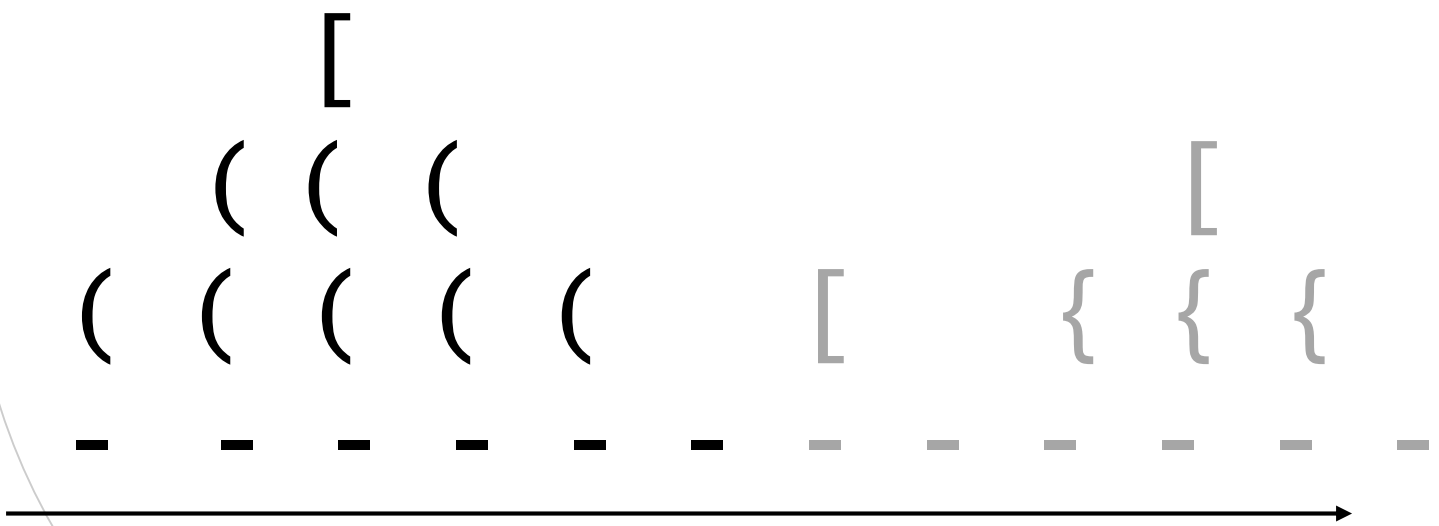
To ensure proper nesting, we traverse the list and use a stack.

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

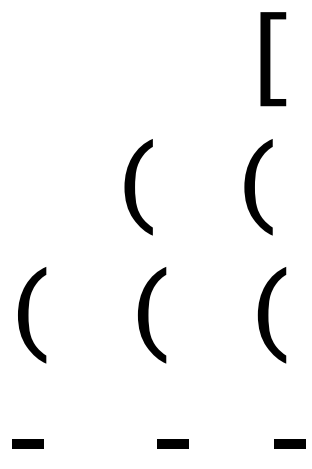
e.g. ((([]))[]{[]}



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.

ALGORITHM FOR NESTED PARENTHESES

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

 I am bold. < 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>
```

These tags can be thought of as brackets.

NESTING IN HTML

Suppose you want:

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

What if you were to write the following ?

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

NESTING IN HTML

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

What if you were to write the following ?

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

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

< i >

Error: mismatch between

Most web browsers will interpret it correctly, however.

NESTING IN HTML

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

The correct way to write it is:

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

< i >

< b > < b > < b > < i >

NESTING IN HTML

What problems can arise if you write it incorrectly?

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

… Hello. I am bold.

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

Bla bla bla ……

Q: What happens if you delete the middle line ?

NESTING IN HTML

What problems can arise if you write it incorrectly?

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

… Hello. I am bold.

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

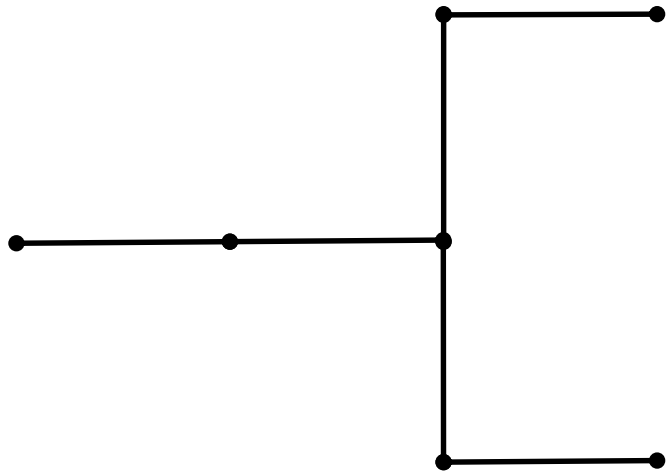
Bla bla bla ……

Q: What happens if you delete the middle line ?

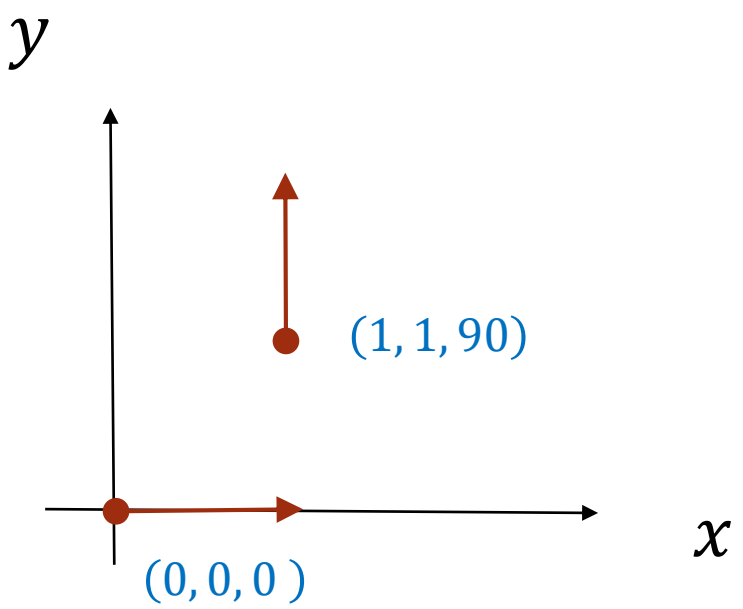
A: … Hello. **I am bold. Bla bla bla ……**

EXAMPLE 4: STACKS IN GRAPHICS

Define a 'programming language' for drawing simple figures like this:



Define a pen position and direction (x, y, θ) where θ 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 (changes (x, y))

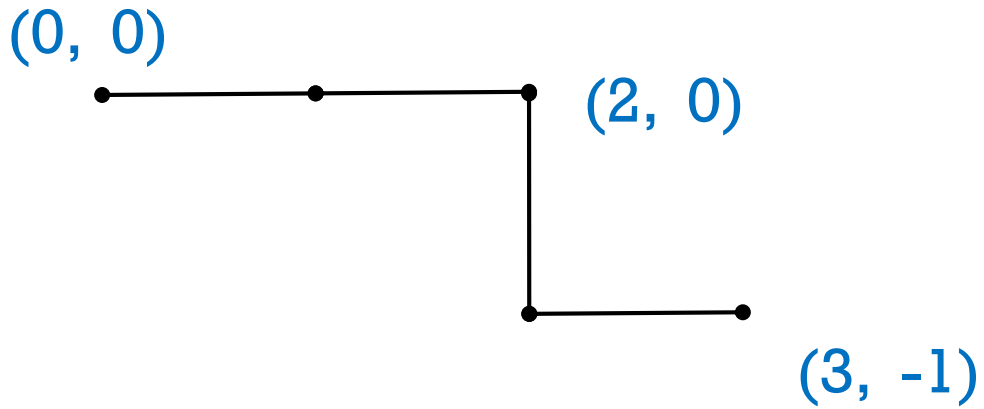
R - turn right 90 degrees clockwise (changes θ)

L - turn left 90 degrees counterclockwise (changes θ)

[- push state (x, y, θ)

] - pop state, and go to that state

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

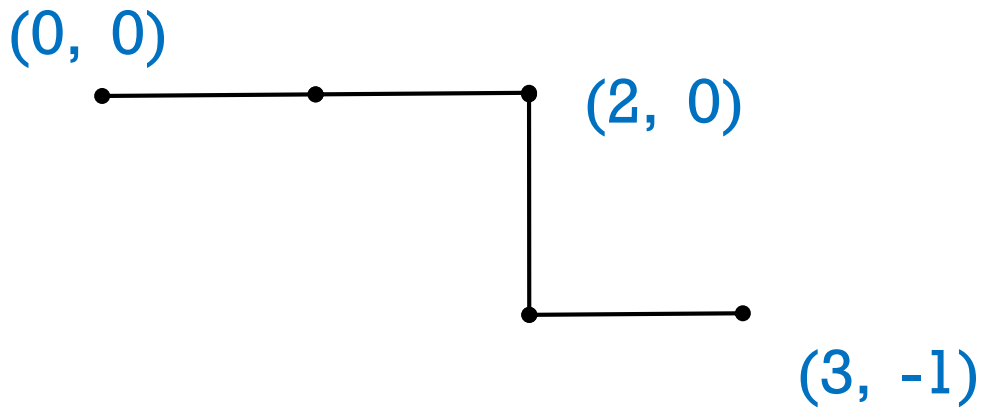


D - draw
R - turn right
L - turn left
[- push state
] - pop state

D D R D L D

The final pen state is (3, -1, 0).

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

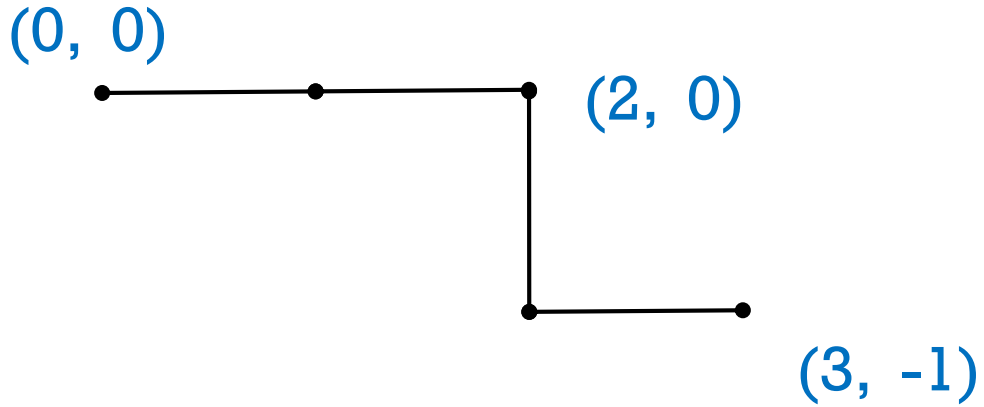


D - draw
R - turn right 90 deg
L - turn left 90 deg
[- push state
] - pop state

D D [R D L D]

Q: What will be the final pen state ?

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



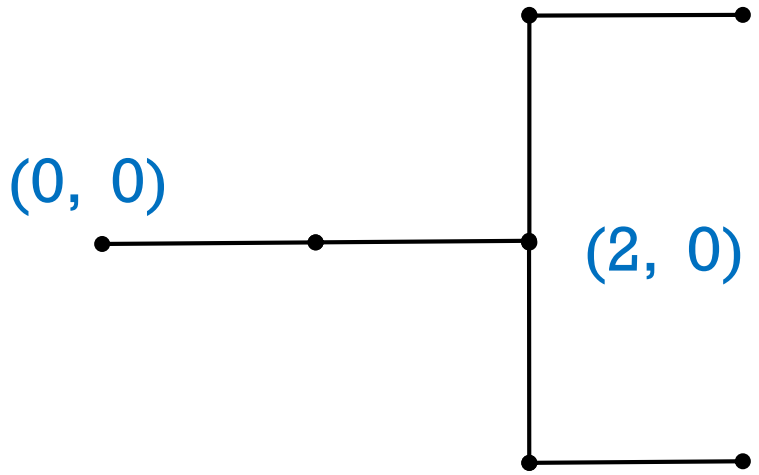
D - draw
R - turn right 90 deg
L - turn left 90 deg
[- push state
] - pop state

D D [R D L D]

Q: What will be the final pen state ?

A: (2, 0, 0)

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



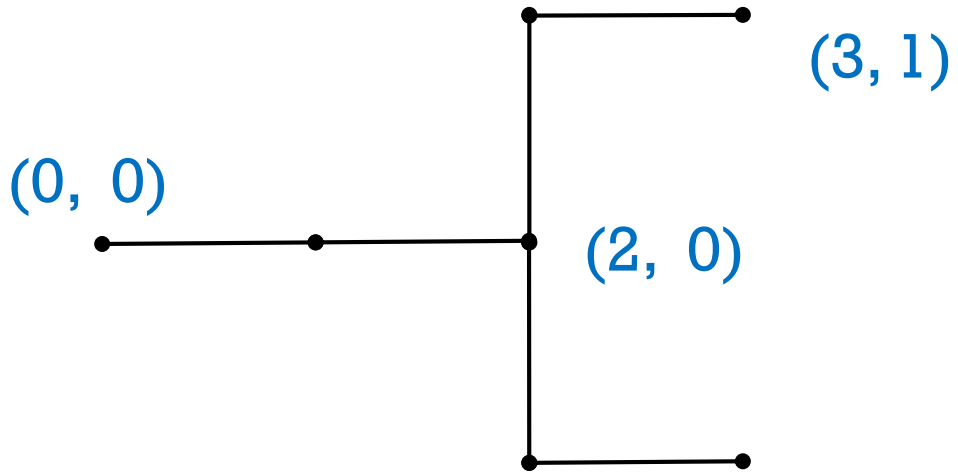
D - draw
R - turn right 90 deg
L - turn left 90 deg
[- push state
] - pop state

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

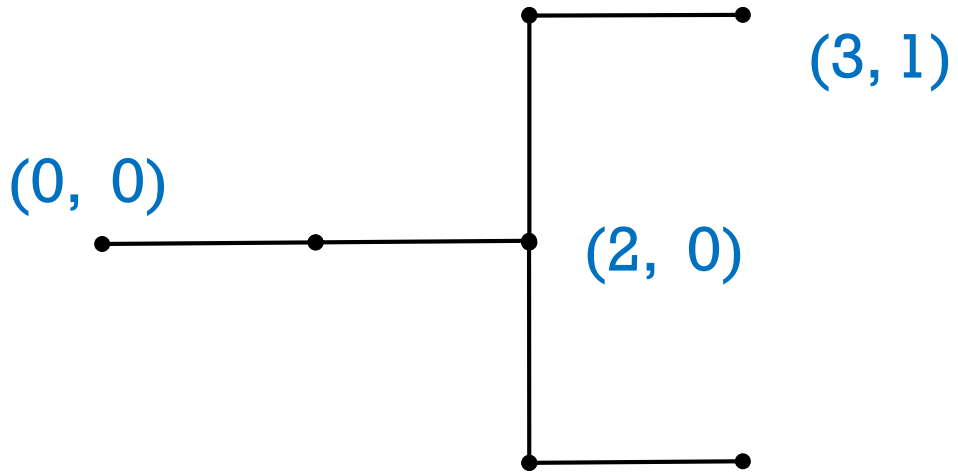
D D [R D L D] L D R D

(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).

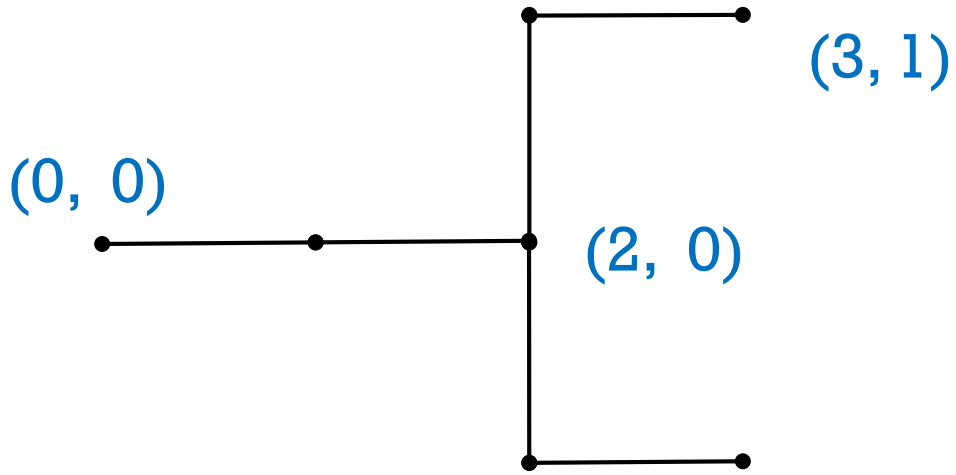


D - draw
R - turn right 90 deg
L - turn left 90 deg
[- push state
] - pop state

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).

EXAMPLE 5A : STACK OF TASKS

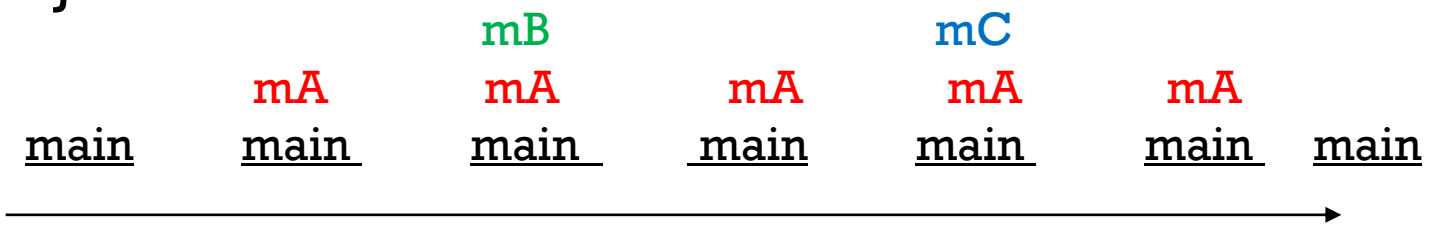
As I work in my office, emails arrive, the phone rings, people drop by, …..

To make sure items all get finished, I must keep a stack. ("What was I doing when ….? ")

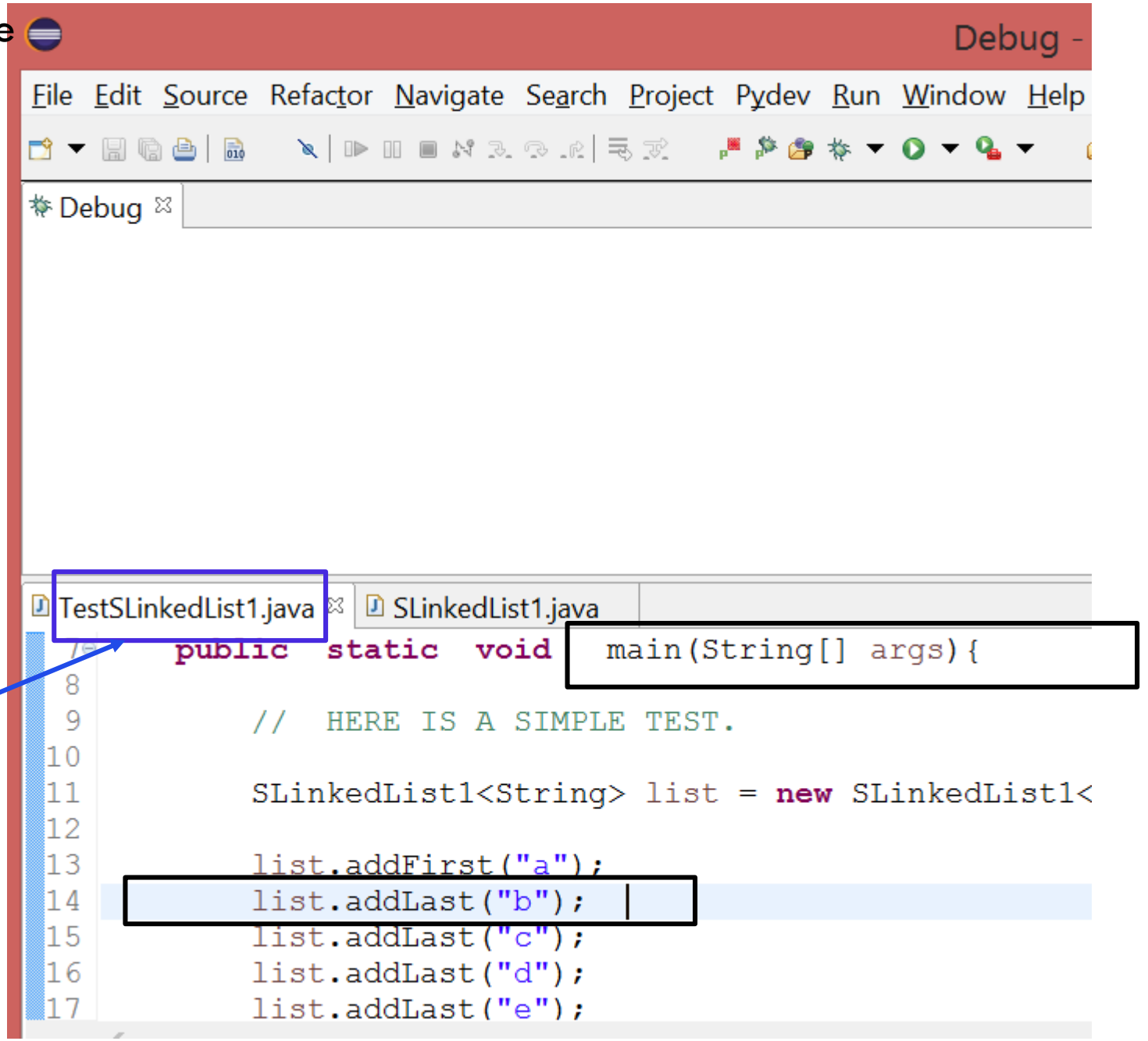
EXAMPLE 5B : "CALL STACK"

```
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( );
        }
}
```



Eclipse debug mode

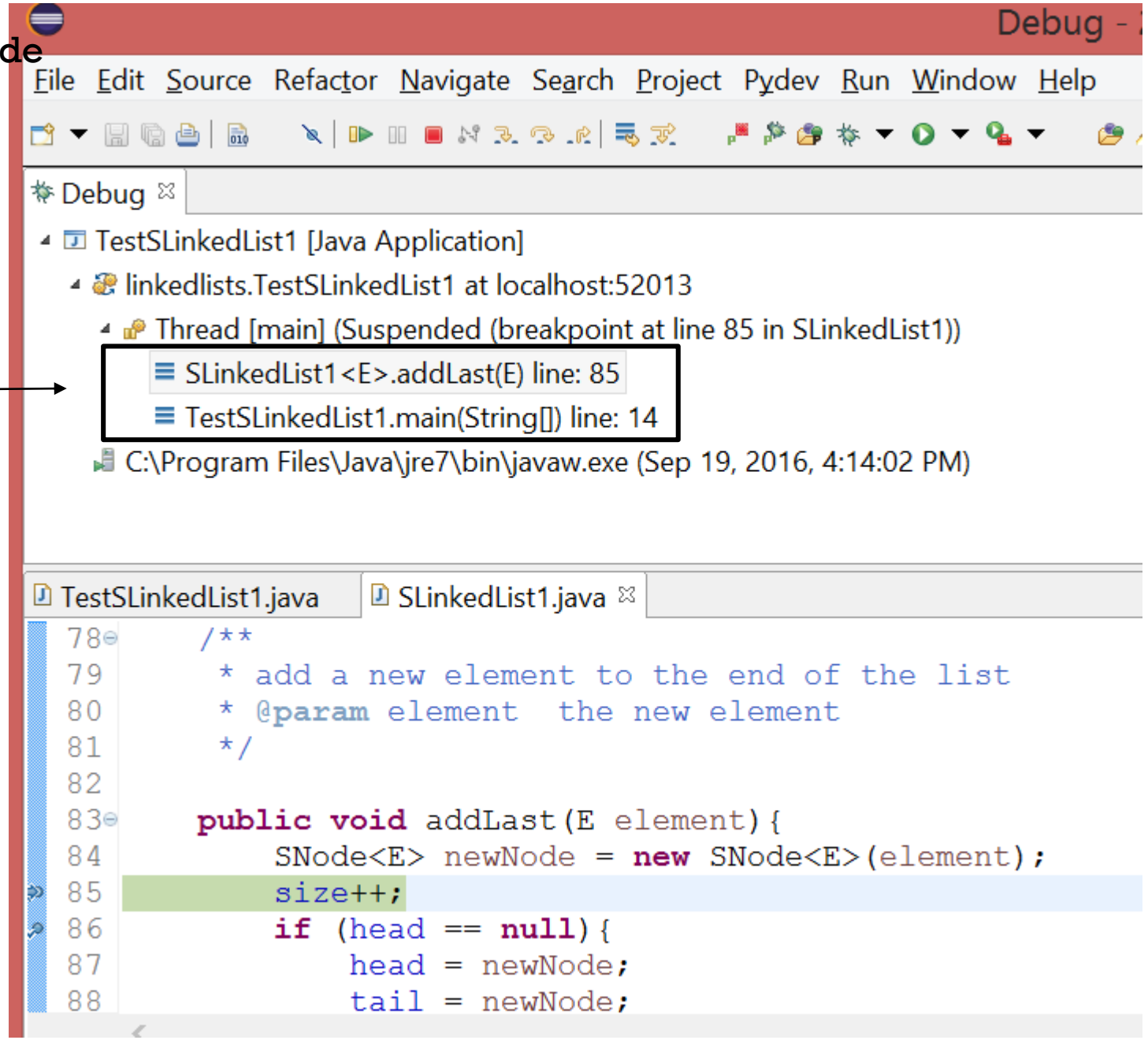


TestSLinkedList1's main() method calls addLast() method of SLinkedList class.

Eclipse debug mode

call stack →

breakpoint →



ANNOUNCEMENTS

- **Discussion board use and abuse**
  - If your posting has been deleted, it is because the information is already available (or some else has posted the same thing)
  - Please do not post requests to me or TA that are of no interest to the 600+ other students.

- **Quiz 1 is today**
  - 8 AM to 8 PM
  - Tip: save your answers as you go

    https://www.mcgill.ca/tls/learning/mycourses#quizzes