Abstract Data Type

Stack

COMP128 Data Structures

Stacks

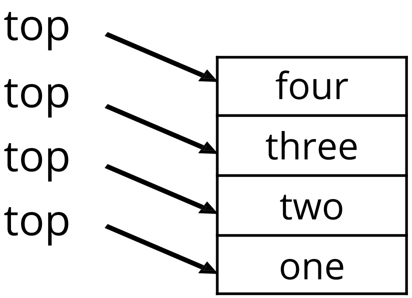
Ordered data structure with some criteria:

- New nodes can only be added to the top of the stack
- Nodes may only be removed from the top of the stack
- The depth of a stack is the number of elements it contains
- It is therefore a last-in, first-out structure (LIFO) or First-In-Last-Out (FILO) structure

Typical Operations

OPERATION	PRE-CONDITION	POST-CONDITION
push (Object item)	stack not full	stack +1, item on top of stack
pop()	stack not empty	stack –1, top item removed
peek()	stack not empty	stack same
empty()	none	stack same

Manipulation of a Stack



top
top
top
top

four
three
two
one

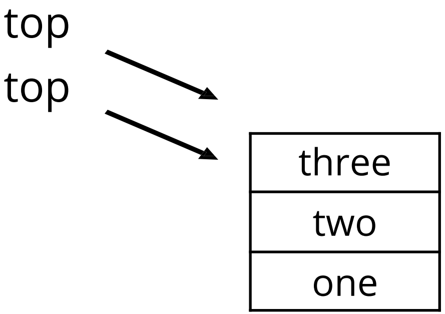
Stack with depth of 4

push(one)

push(two)

push(three)

push(four)

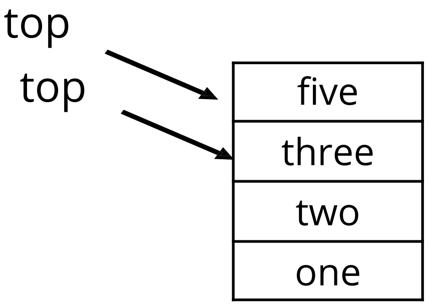


top
top

three
two
one

Stack with depth of 3

pop( )



top
top

five
three
two
one

Stack with depth of 4

push(five)

Stacks in Java

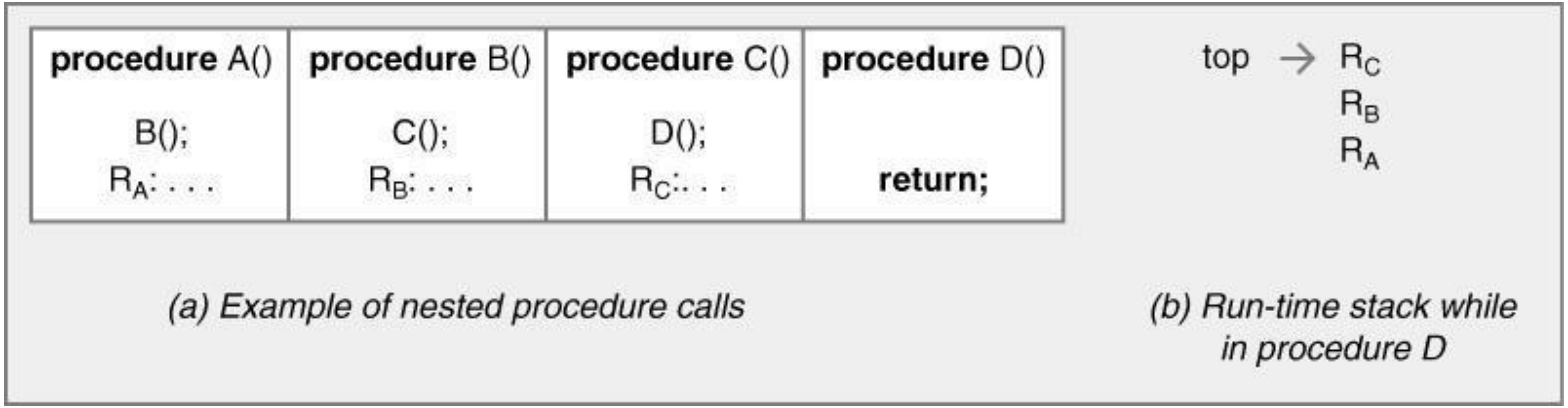
- The Stack class in java has been deprecated, i.e. you shouldn't use it.
- Instead implement as stack with the **Deque interface**, but only use the push, pop, peek, isEmpty methods.

```
Deque<String> stackOfStrings = new ArrayDeque<String>()
```

Applications of a Stack

- Reverse a text String
- Reverse the line order of a text file
- Check to see if brackets match
- Evaluation of complex expressions (intermediate values stored)
- Activation stack (method calls)
- Recursion

Using stacks to implement nested procedure calls



procedure A()	**procedure** B()	**procedure** C()	**procedure** D()
B();	C();	D();	
R_A: . . .	R_B: . . .	R_C:. . .	**return;**

(a) Example of nested procedure calls

top → R_C

R_B

R_A

(b) Run-time stack while in procedure D

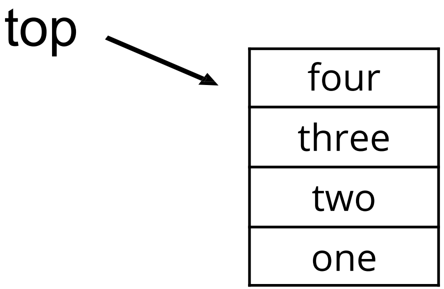
You will see more in COMP 240

In-class Activity

Palindrome Stack Activity

Implementation with an array

Implementations of a stack


top

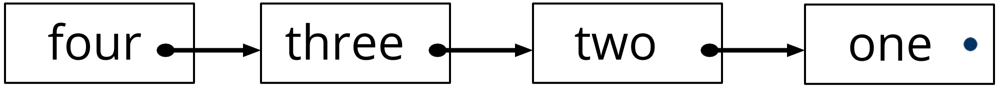
four

three

two

one

Array implementation


four

three

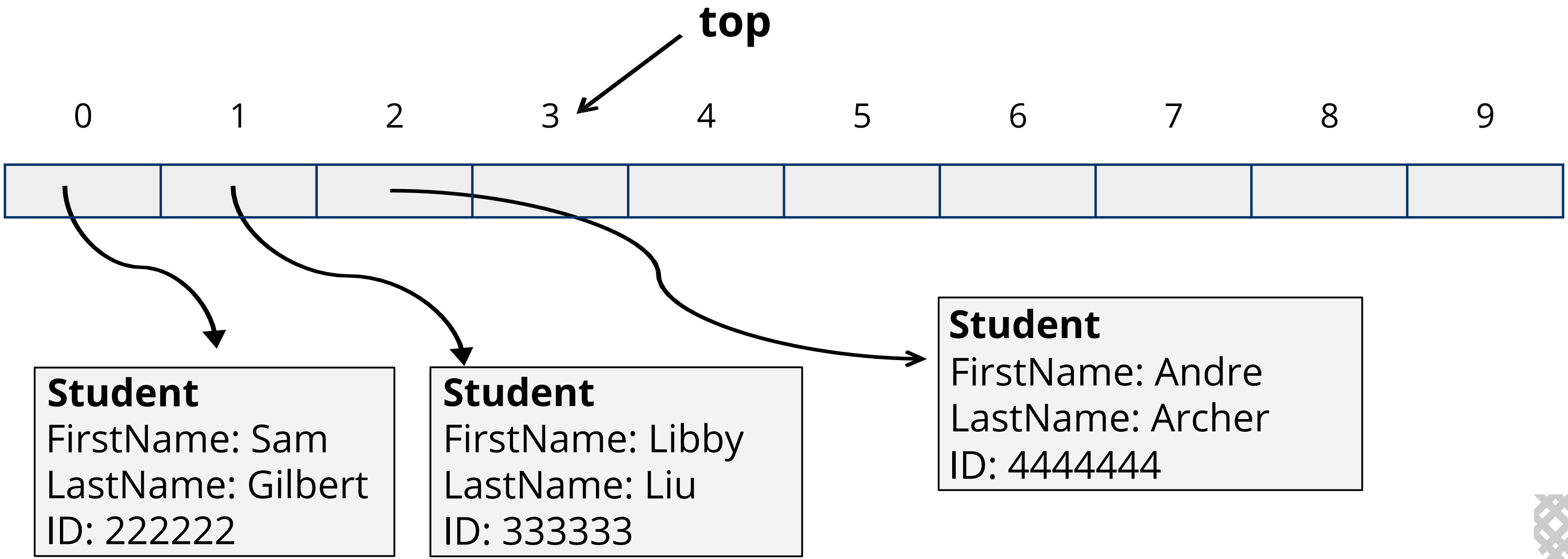
two

one

Linked list implementation

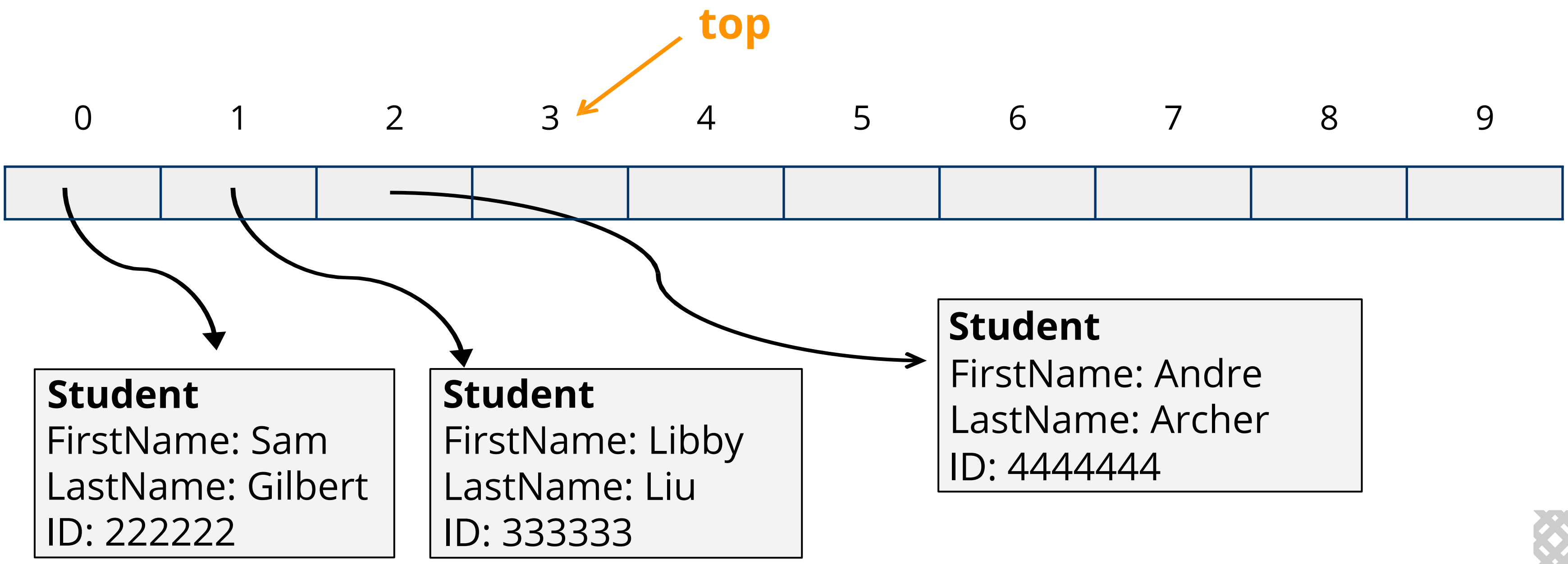
The array implementation of a stack has:

An array that contains references to Objects, such as instances of type Student An int that keeps track of the top of the stack



The array implementation of a stack has:

1. **Check if there is space and expand if necessary**



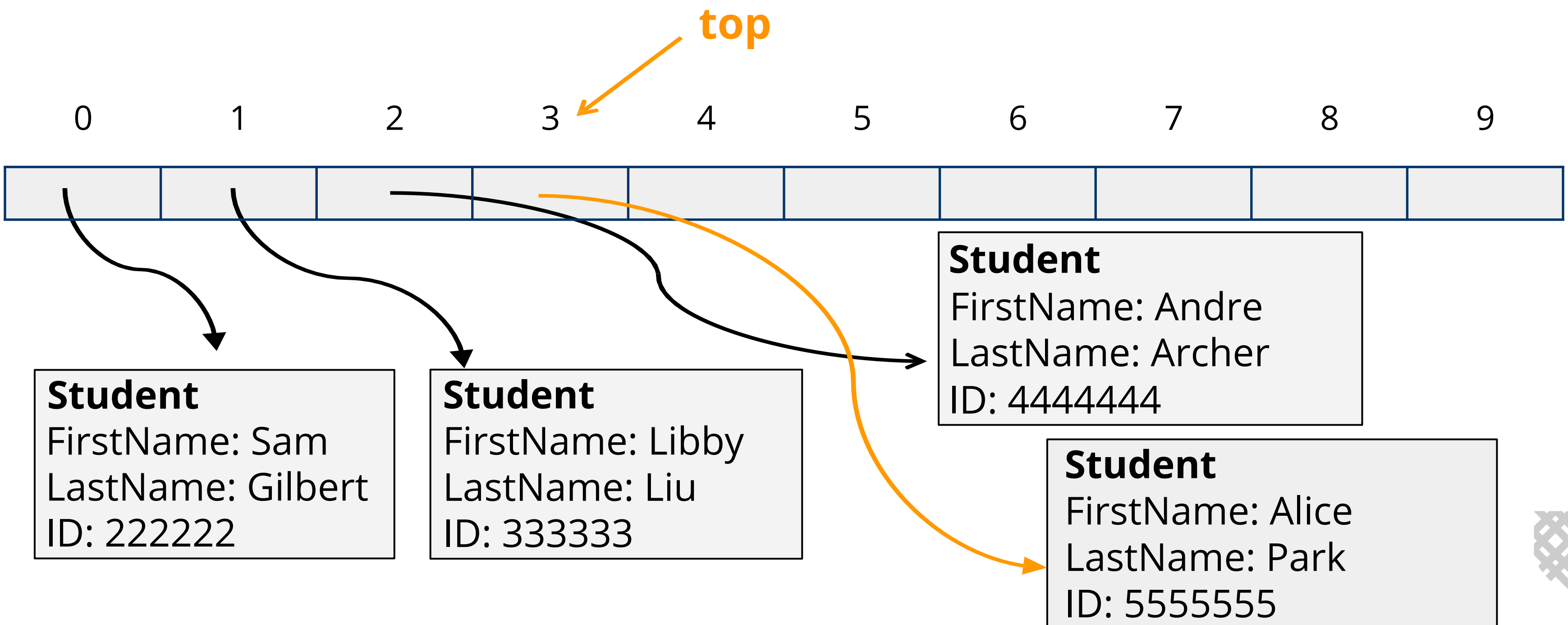
Student
FirstName: Sam
LastName: Gilbert
ID: 222222

Student
FirstName: Libby
LastName: Liu
ID: 333333

Student
FirstName: Andre
LastName: Archer
ID: 4444444

The array implementation of a stack has:

1. Check if there is space and expand if necessary
2. **Insert the object at the top position**



top

0	1	2	3	4	5	6	7	8	9

Student
FirstName: Sam
LastName: Gilbert
ID: 222222

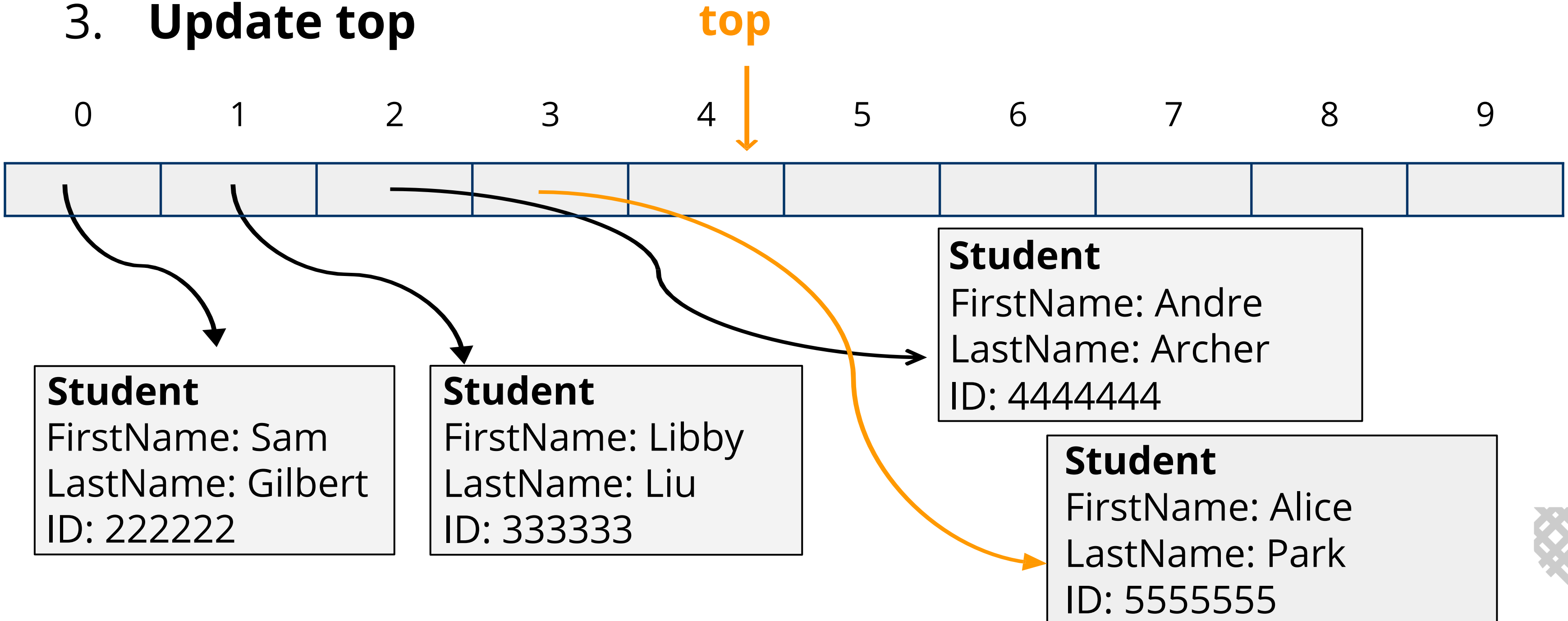
Student
FirstName: Libby
LastName: Liu
ID: 333333

Student
FirstName: Andre
LastName: Archer
ID: 4444444

Student
FirstName: Alice
LastName: Park
ID: 5555555

The array implementation of a stack has:

1. Check if there is space and expand if necessary
2. Insert the object at the top position
3. **Update top**



Student
FirstName: Sam
LastName: Gilbert
ID: 222222

Student
FirstName: Libby
LastName: Liu
ID: 333333

Student
FirstName: Andre
LastName: Archer
ID: 4444444

Student
FirstName: Alice
LastName: Park
ID: 5555555

Expanding the array

If the array is full and an object is pushed on the stack, you should:

- Create a new array that is twice the size of the current one.
- Copy all the old items into the new array
- Add the new one as usual