CS201
Data Structures and Algorithms
Revision Session 3

stack as array
stack as linked list

Stack

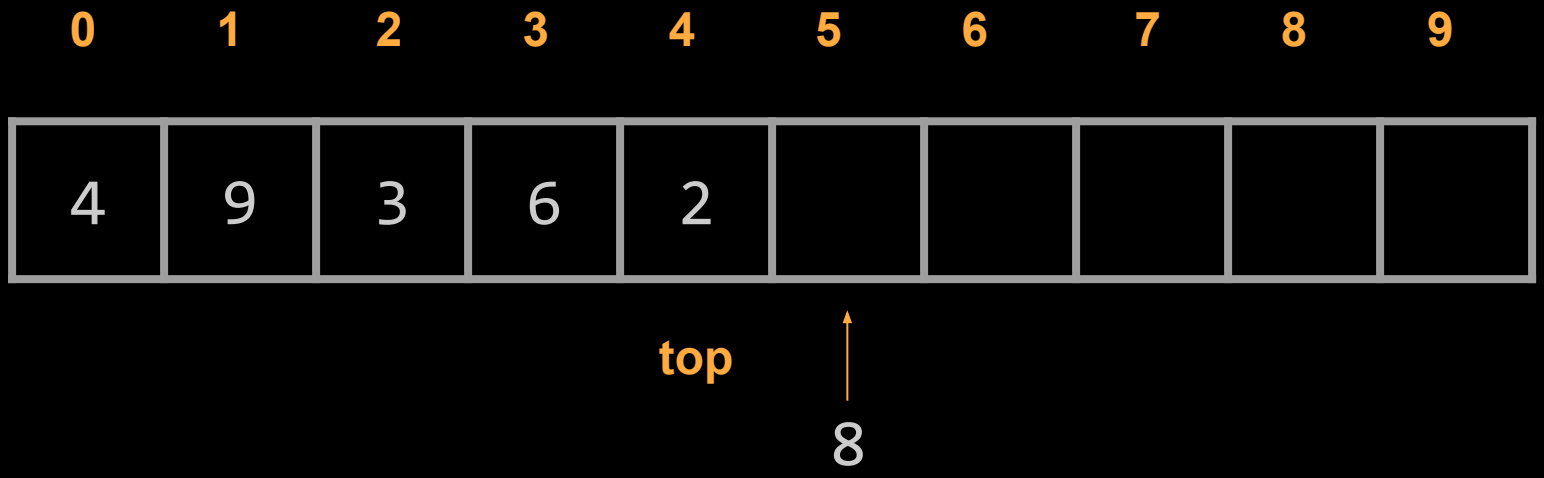
implementation:
element and stack array
element and linked list array

algorithms:
top
isFull
isEmpty
push
pop

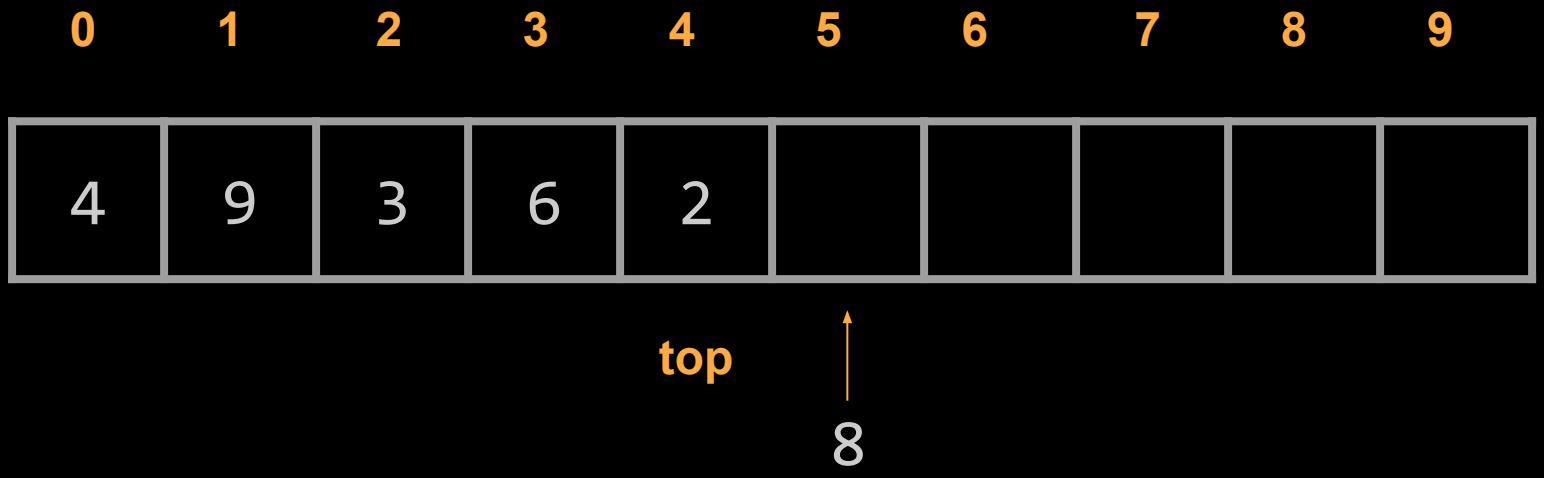
stack as array

implementation

N = 10

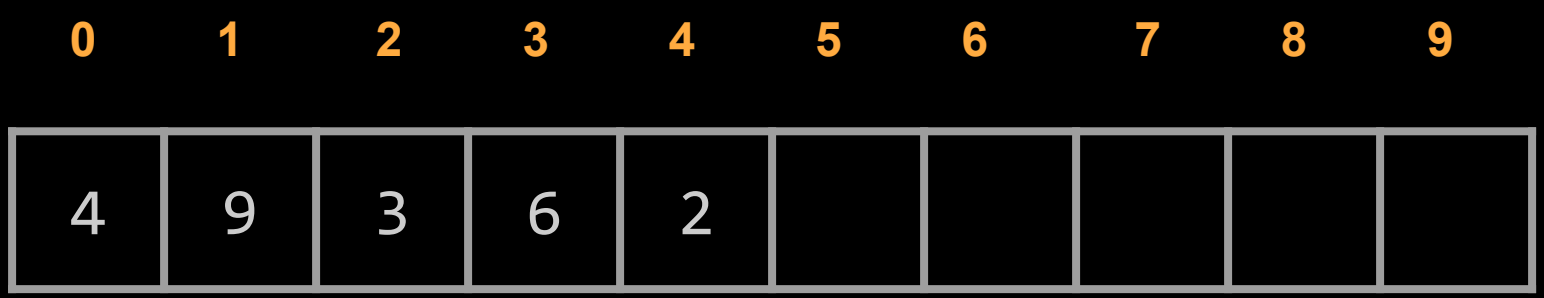


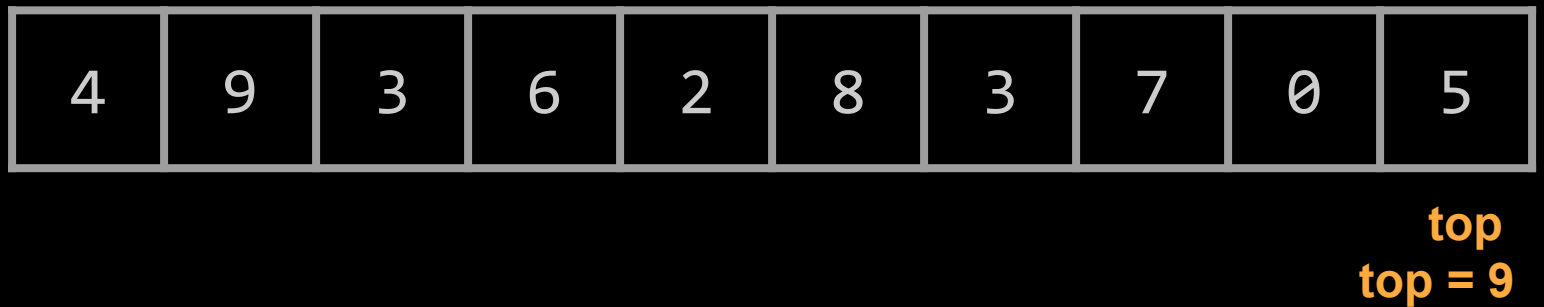
N = 10



N = 10

N = 10





```java
public class StackArray {
    4 usages
    Element[] stack;
    8 usages
    int top;  // the index showing the top element
    5 usages
    int N;    // the length of the stack array

    1 usage  neslihancesurr *
    public StackArray(int N) {
        stack = new Element[N];
        this.N = N;
        top = -1;
    }
```

```java
public class Element {
    public int data;

    3 usages  neslihancesurr *
    public Element(int data){

        this.data = data;
    }
}
```

getting the top element

```
Element top() {
    return stack[top];
}
```

the index of the last added (top) element is stored in the variable called top

this method returns the last added (top) element

full or empty?

```
public boolean isEmpty() {
    return top == -1;
}
```

```
public boolean isFull() {
    return top == (N - 1);
}
```

If no element has been added, the top field has never been incremented.

If top is at the highest index, then the stack is full.

push

```
public void push(Element newElement) {
    if (!isFull()) {
        top++;
        stack[top] = newElement;
    }
}
```

If the stack is not full, increase the top field to the next index. Then, place the new element in the empty top field.

pop

```
public Element pop(){
    if (!isEmpty()) {
        top--;
        return stack[top + 1];
    } else
        return null;
}
```

If the stack is not empty, decrease the top field to the previous index. Then, return the old top (top + 1).

stack as linked list

```java
public class Node {
    public int data;
    public Node next;

    8 usages   neslihancesurr
    public Node(int data){
        this.data = data;
        next = null;
    }
}
```

```java
public class StackLinked {
    4 usages
    Node top;

    no usages   neslihancesurr
    public StackLinked(){
        top = null;
    }
```

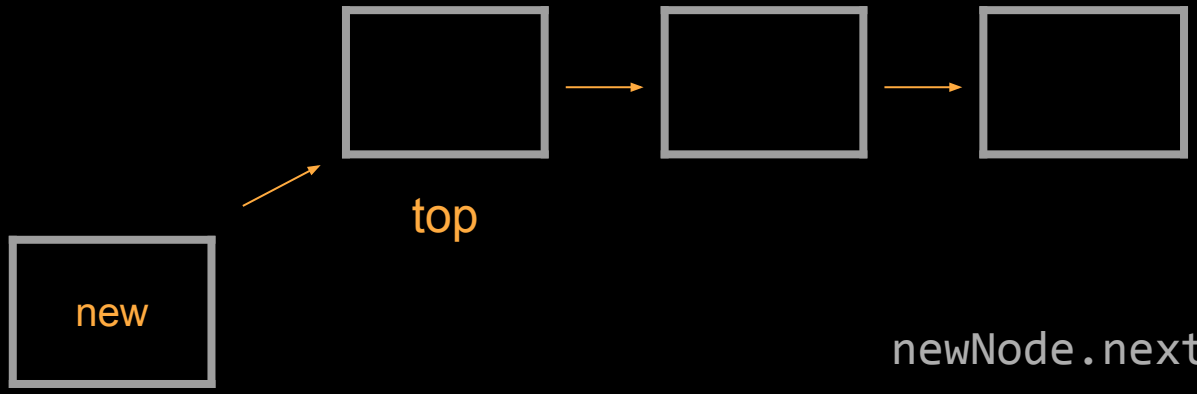
full or empty?

```
boolean isEmpty(){
    return top == null;
}
```

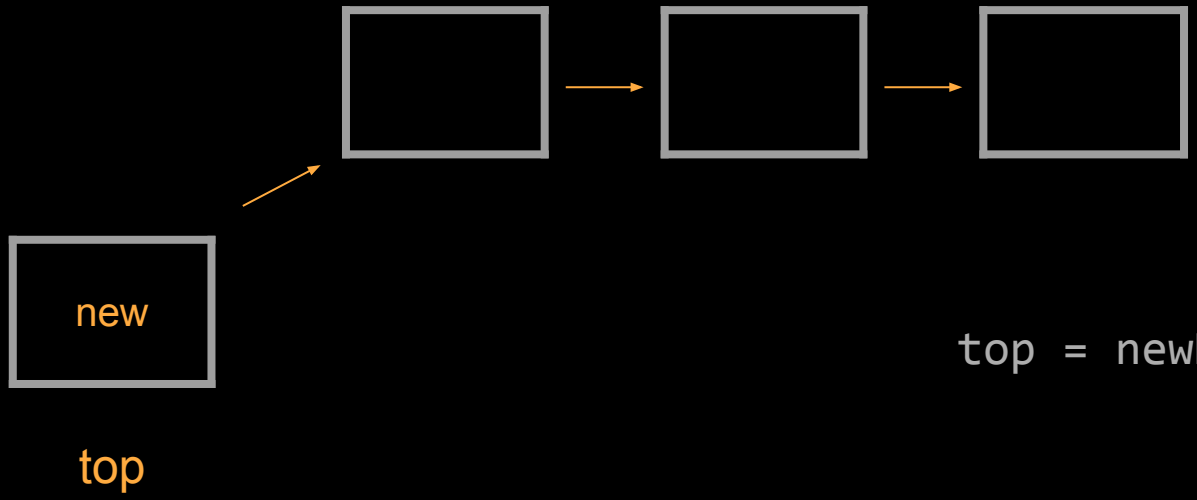
if top has never been updated,
it should be null.

push


top




```
newNode.next = top;
```




```
top = newNode;
```

```
void push(Node newNode){
    newNode.next = top;
    top = newNode;
}
```

make new node point to current top

make the new node the new top

pop

save top node in a variable

if list is not empty, move top node to the next

return the old top node

```
public Node pop(){
    Node e = top;
    if (!isEmpty()){
        top = top.next;
    }
    return e;
}
```

Stack exercise

19. Write the method that pushes the data as the *k*'th element from the top.

```
public void push(int k, int data)
```

no stack methods or external stacks

before push: 5 6 8 3 4

after push(3, 10): 5 6 8 10 3 4(top)