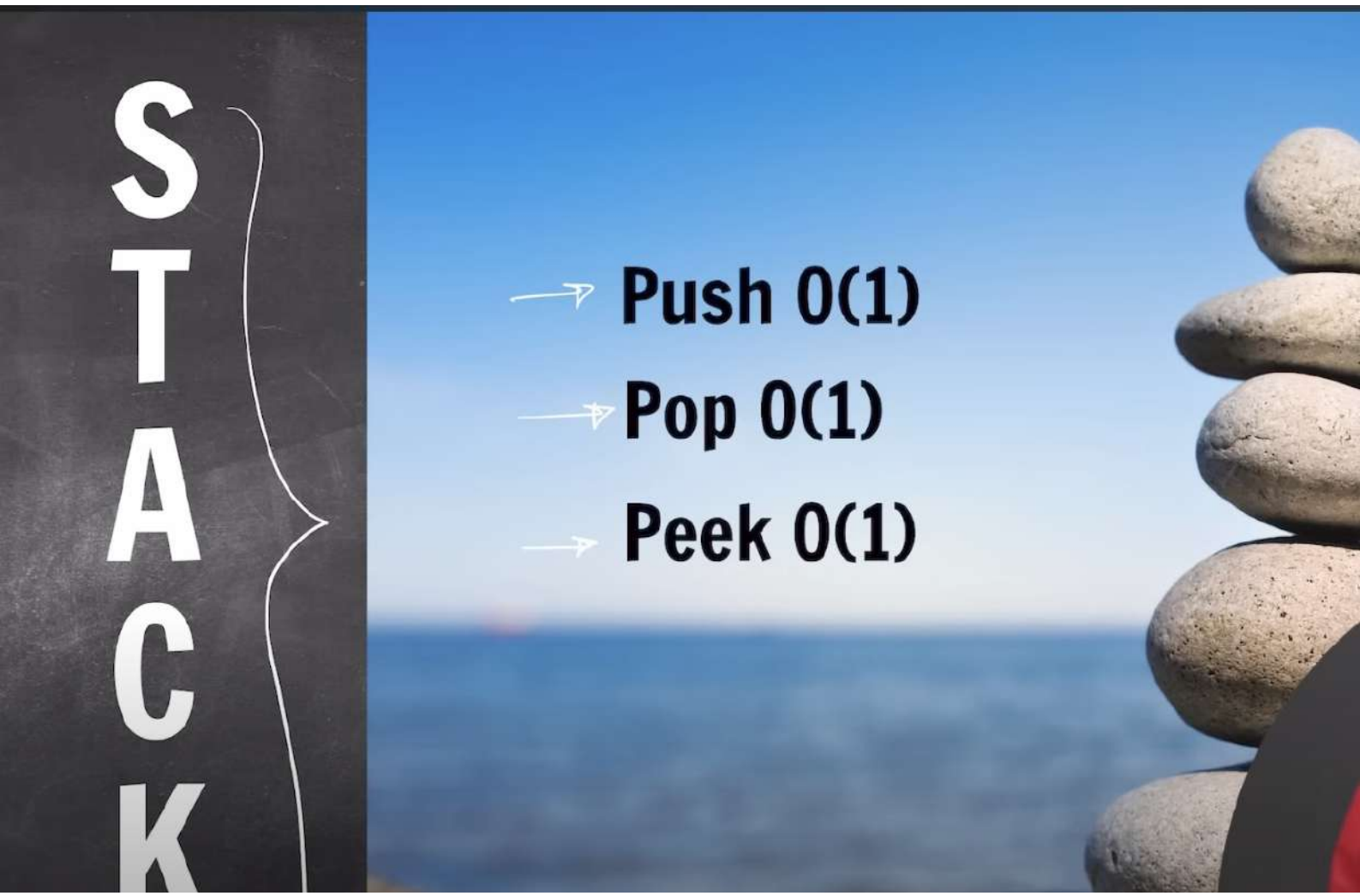
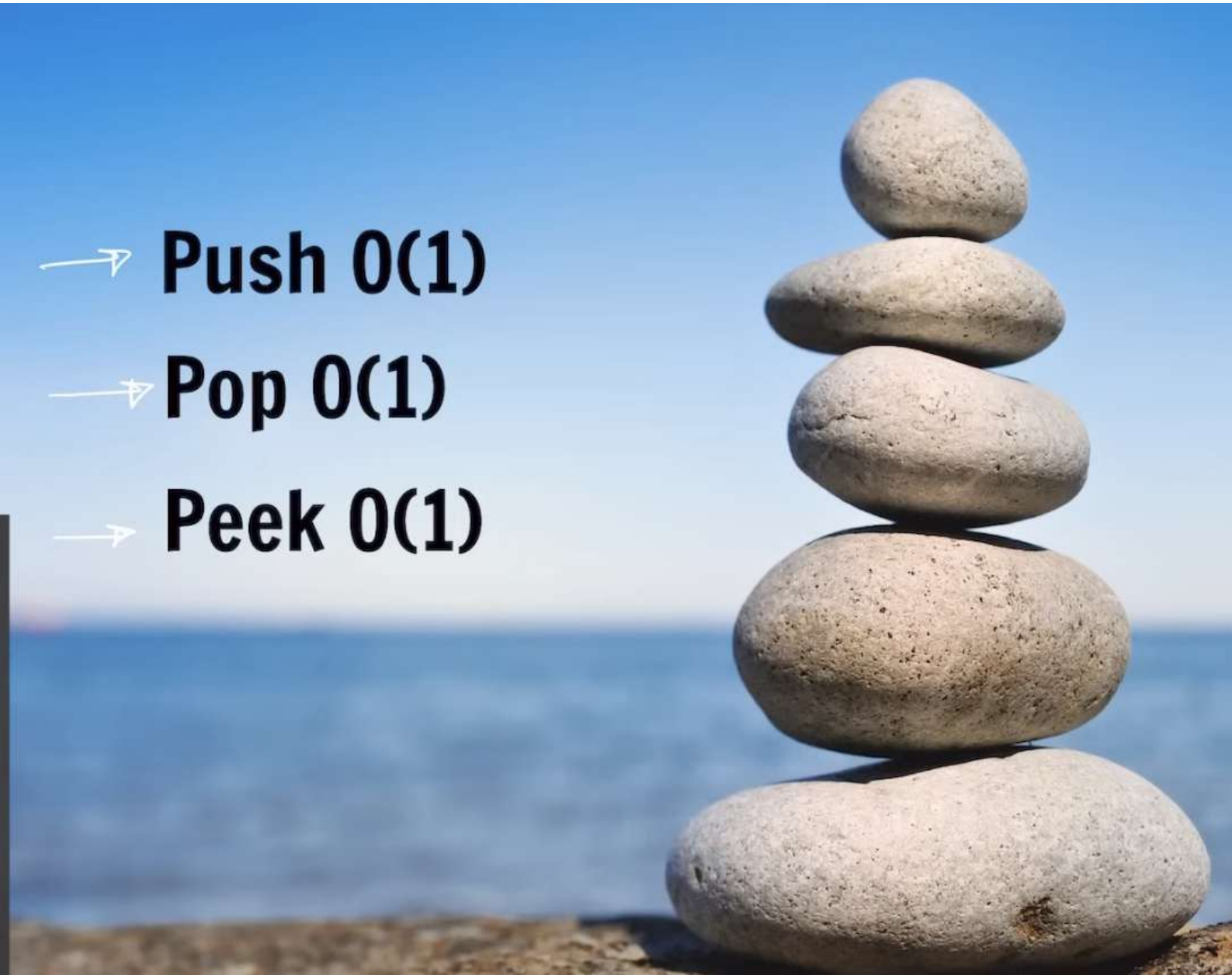
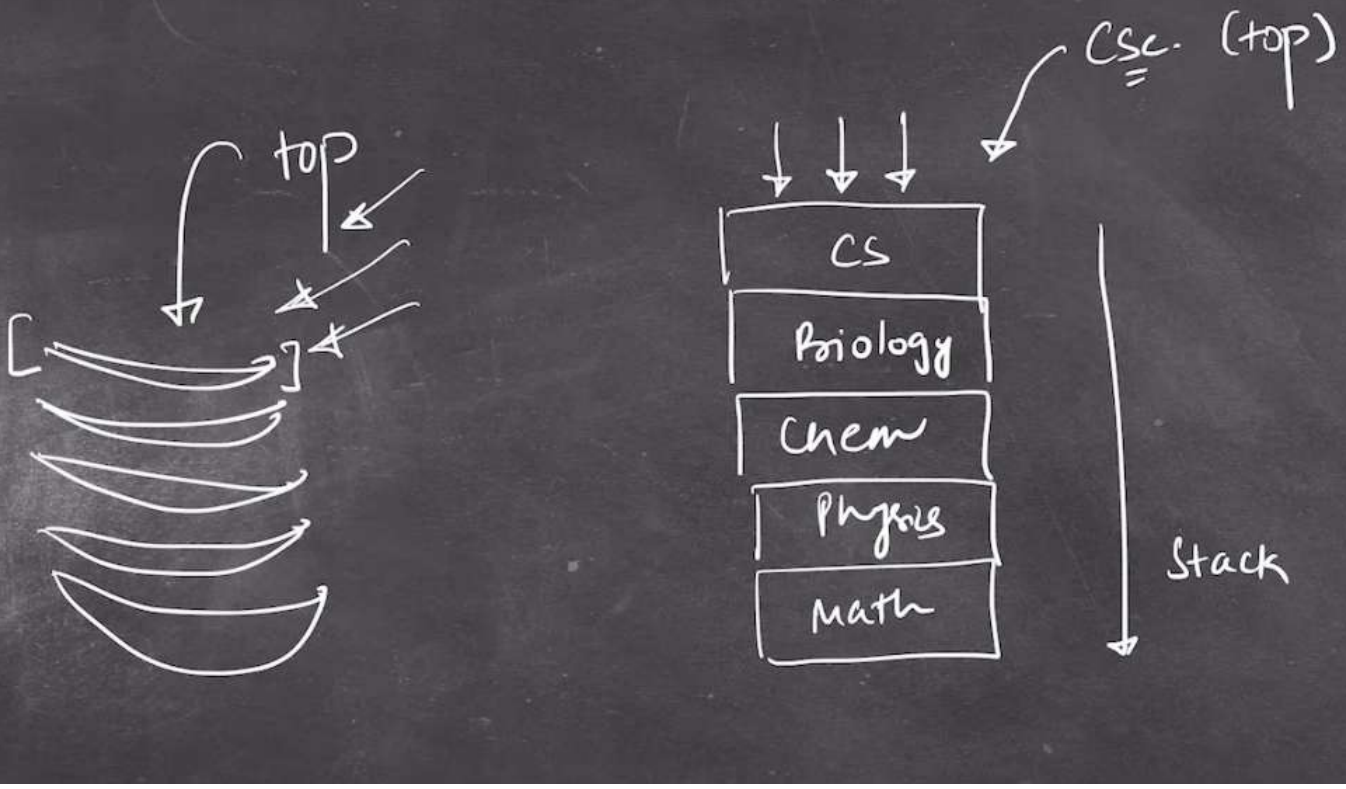

STACK
Push O(1)
Pop O(1)
Peek O(1)


Push O(1)
Pop O(1)
Peek O(1)

Real Life Examples

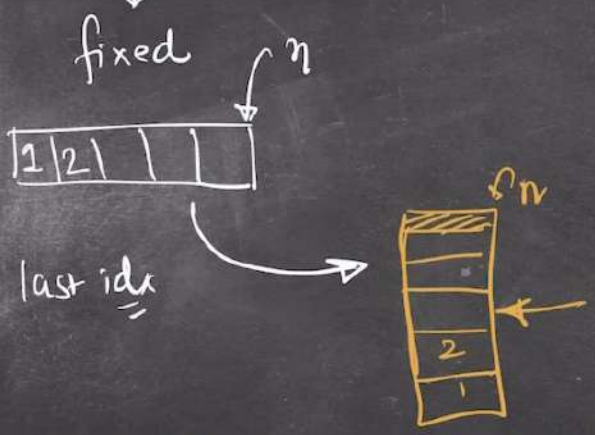


Implementation

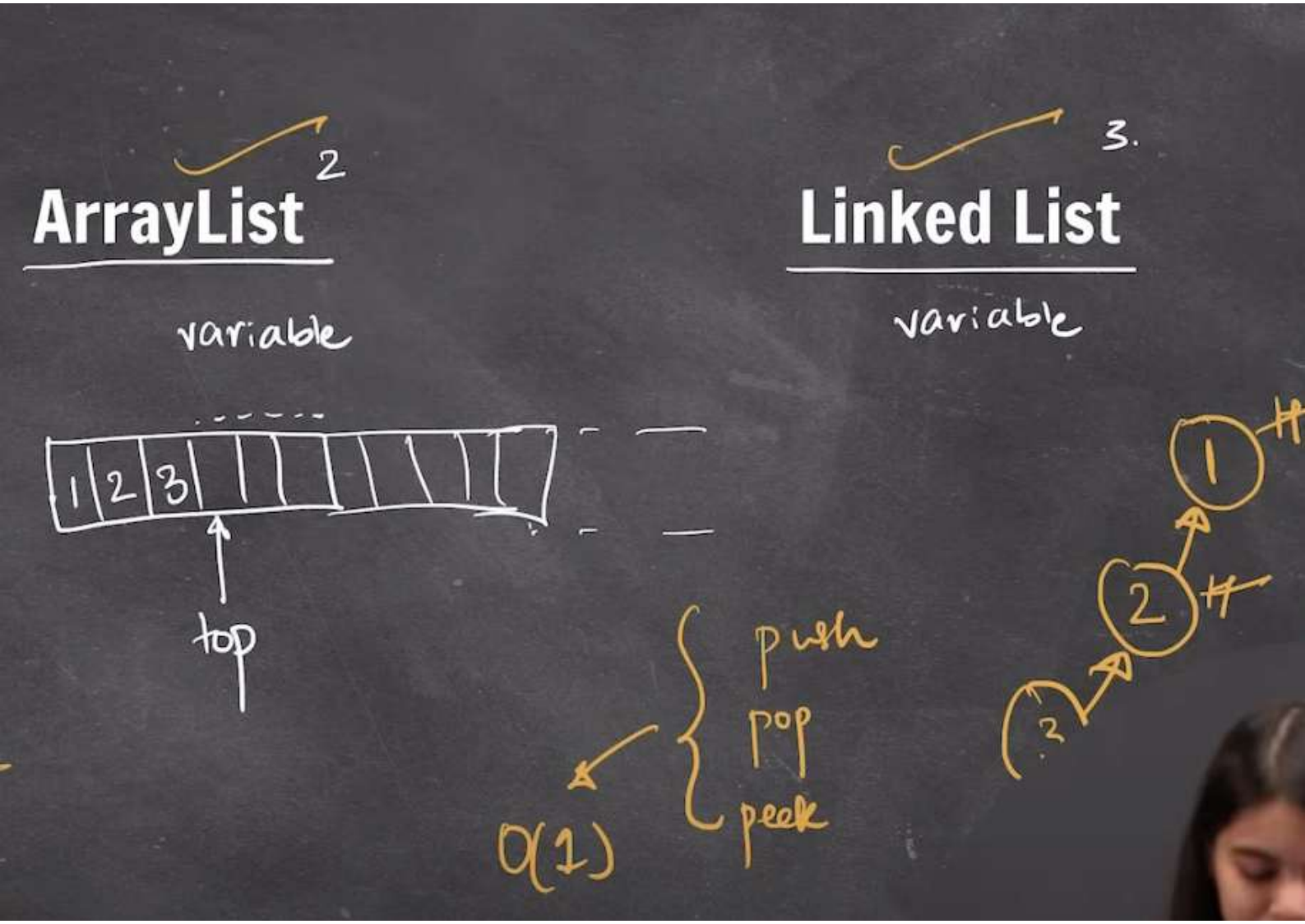
Array 1.

ArrayList 2

Linked List 3.



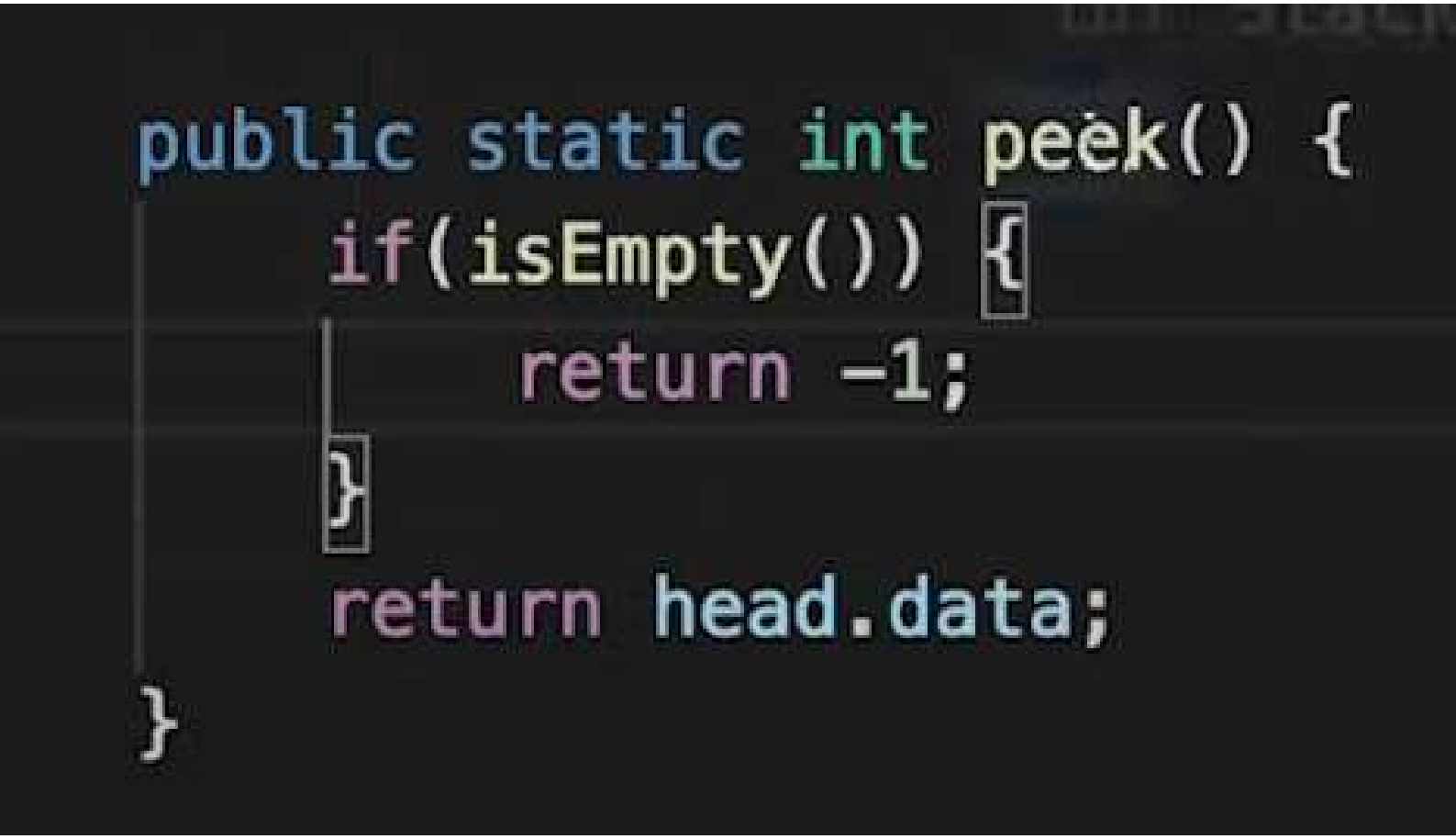
1. Stack full (when el == n)


2
ArrayList
variable
1 2 3
top
3.
Linked List
variable
1
2
3
push
pop
peek
O(1)

```java
public class StackClass {
    static class Node {
        int data;
        Node next;
        public Node(int data) {
            this.data = data;
            next = null;
        }
    }

    static class Stack {
        public Node head;
        public static boolean isEmpty() {
            return Head == null;
        }
```

```
public static void push(int data) {
    Node newNode = new Node(data);
    if(isEmpty()) {
        head = newNode;
        return;
    }
    newNode.next = head;
    head = newNode;
}
```

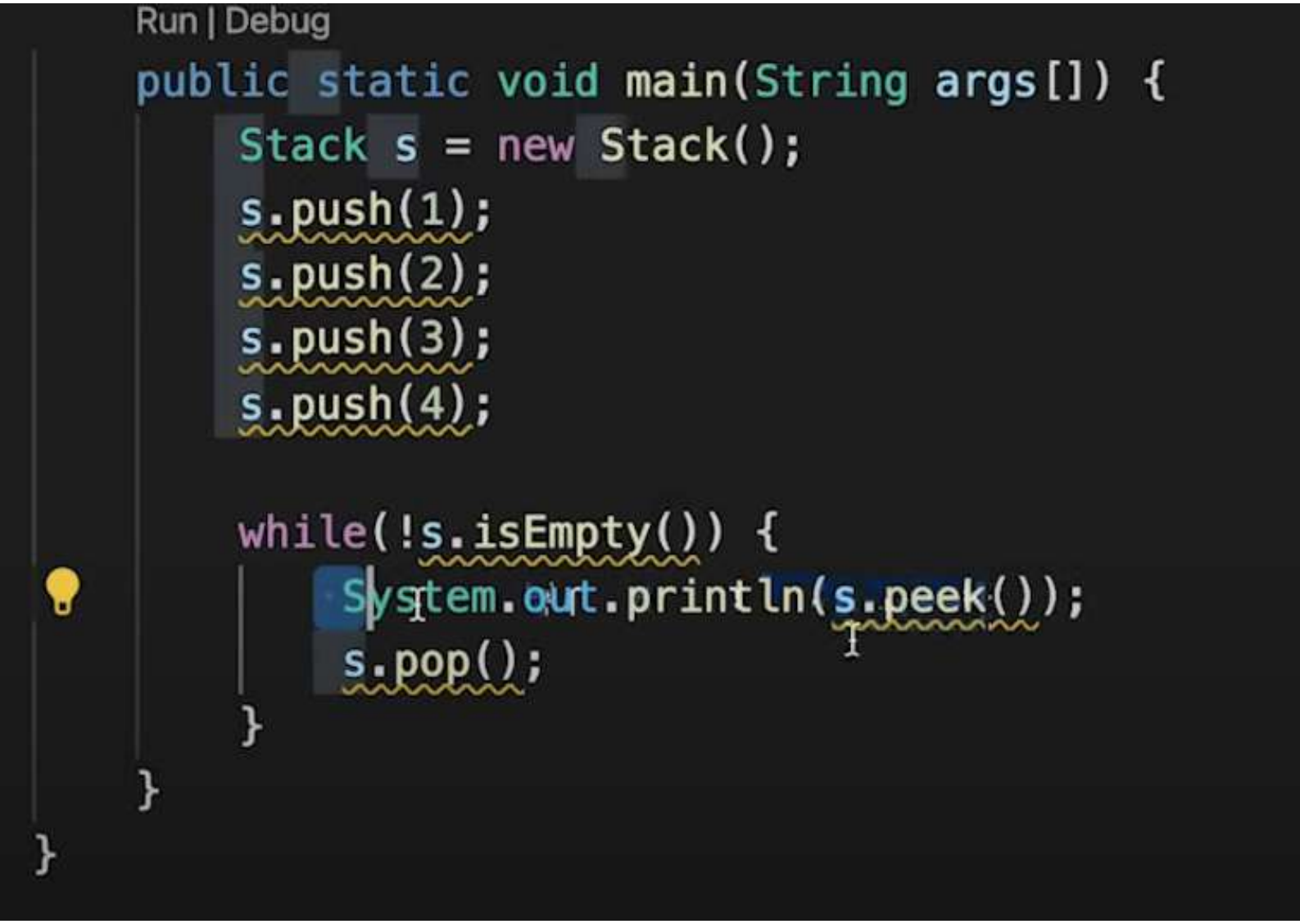
```
public static int pop() {
    if(isEmpty()) {
        return -1;
    }
    int top = head.data;
    head = head.next;
    return top;
}
```

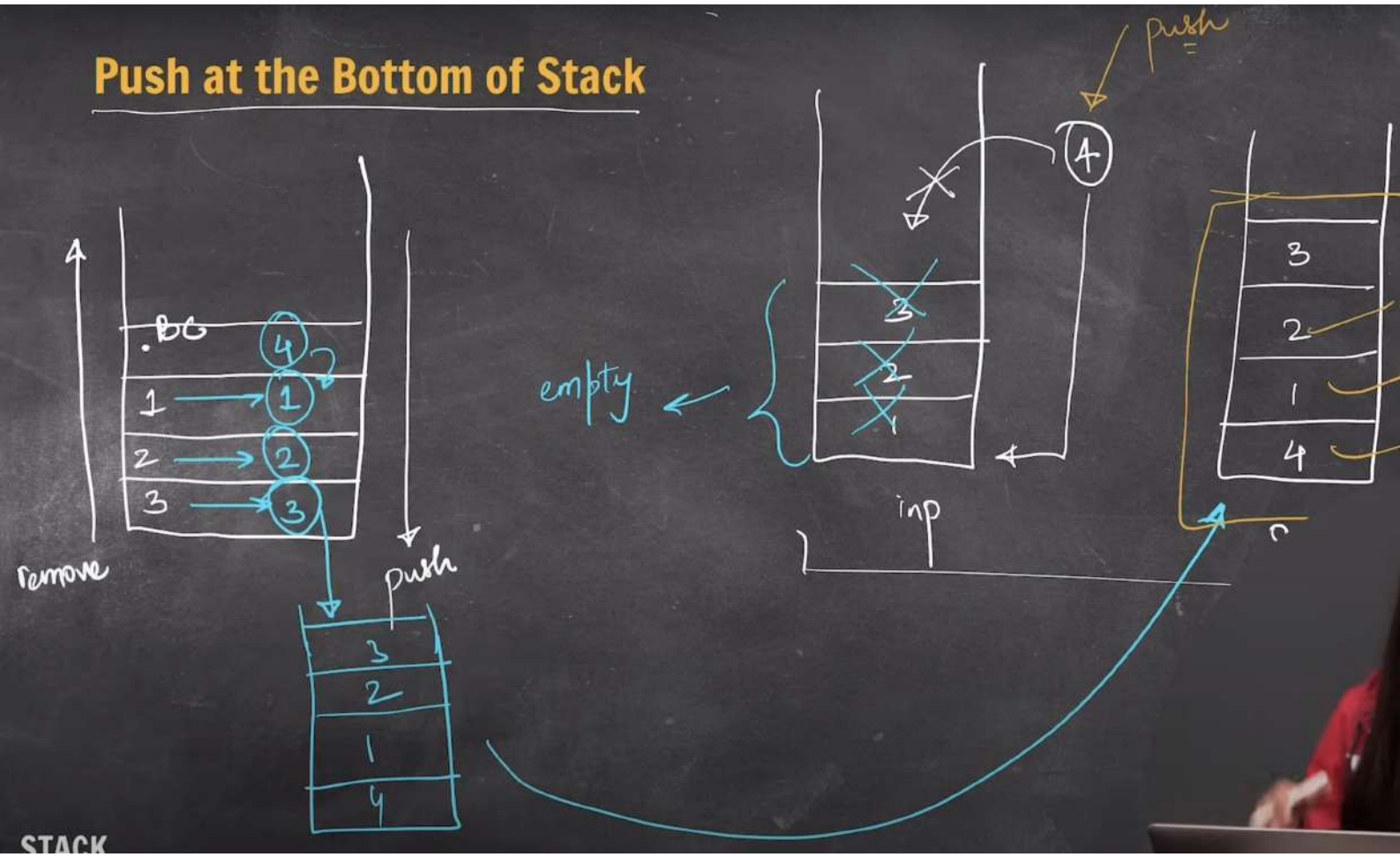

public static int peek() {
if(isEmpty()) {
return -1;
}
return head.data;
}

```java
public static void main(String args[]) {
    Stack s = new Stack();
    s.push(1);
    s.push(2);
    s.push(3);
    s.push(4);

    while(!s.isEmpty()) {
        System.out.println(s.peek());
        s.pop();
    }
}
```

```java
public class StackClass {
    class Stack {
        ArrayList<Integer> list = new ArrayList<>();
        public static boolean isEmpty() {
            return list.size() == 0;
        }

        //push
        public static void push(int data) {
            list.add(data);
        }
```

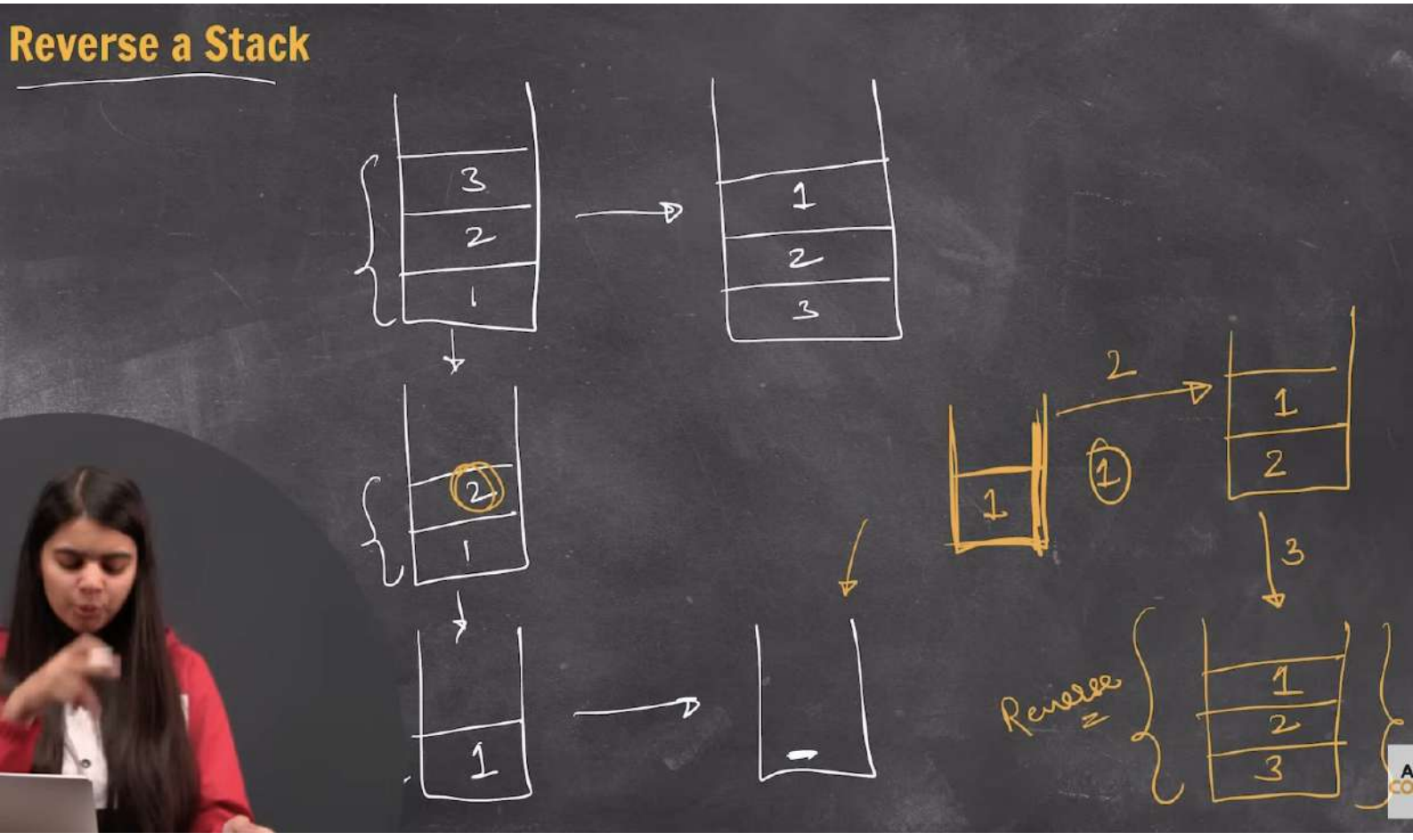
```java
    //pop
    public static int pop() {
        if(isEmpty()) {
            return -1;
        }
        int top = list.get(list.size()-1);
        list.remove(list.size()-1);
        return top;
    }

    //peek
    public static int peek() {
        if(isEmpty()) {
            return -1;
        }
        return list.get(list.size()-1);
    }
}
```


Run | Debug
public static void main(String args[]) {
Stack s = new Stack();
s.push(1);
s.push(2);
s.push(3);
s.push(4);
while(!s.isEmpty()) {
System.out.println(s.peek());
s.pop();
}
}
}


Push at the Bottom of Stack
push
4
3
2
1
4
3
2
1
empty
inp
c
remove
push
3
2
1
4
STACK

```java
import java.util.*;

public class StackClass {
    public static void pushAtBottom(int data, Stack<Integer> s) {
        if(s.isEmpty()) {
            s.push(data);
            return;
        }
        int top = s.pop();
        pushAtBottom(data, s);
        s.push(top);
    }
    Run | Debug
    public static void main(String args[]) {
        Stack<Integer> s = new Stack<>();
        s.push(1);
        s.push(2);
        s.push(3);

        pushAtBottom(data, s);

        while(!s.isEmpty()) {
```

```
Run | Debug
public static void main(String args[]) {
    Stack<Integer> s = new Stack<>();
    s.push(1);
    s.push(2);
    s.push(3);

    pushAtBottom(4, s);

    while(!s.isEmpty()) {
        System.out.println(s.peek());
        s.pop();
    }
}
}
```


Reverse a Stack
3
2
1
1
2
3
2
1
1
1
2
1
1
2
3
Reverse

```java
public static void reverse(Stack<Integer> s) {
    if(s.isEmpty()) {
        return;
    }
    int to Full screen (f) ;
    reverse(s);
    pushAtBottom(top, s);
}

Run | Debug
public static void main(String args[]) {
    Stack<Integer> s = new Stack<>();
    s.push(1);
    s.push(2);
    s.push(3);

    while(!s.isEmpty()) {
        System.out.println(s.peek());
        s.pop();
    }
}
```