Java Stack pushBottom and reverse() Explained

Stack Manipulation Explanation

GOAL:

You are doing two things:

1. pushBottom(stack, 0): This adds 0 at the bottom of the stack.

2. reverse(stack): This reverses the entire stack using recursion.

Stack Recap:

- Stack is LIFO: Last In, First Out

- push(x): Add to top

- pop(): Remove from top

- peek(): View top

Initial Stack (Before Anything):

TOP

4

3

2

1

BOTTOM

Step 1: pushBottom(stack, 0) Add 0 to the bottom

Logic:

We can't directly push to the bottom of a stack. So, we pop everything,

and once it's empty, we push 0, then push back all items one by one.

Recursive Breakdown:

1. pop 4 call pushBottom(stack, 0)

2. pop 3 call pushBottom(stack, 0)

3. pop 2 call pushBottom(stack, 0)

Java Stack pushBottom and reverse() Explained

4. pop 1 call pushBottom(stack, 0)

5. now the stack is empty push(0)

6. push(1), push(2), push(3), push(4)

After pushBottom(stack, 0):

TOP

4

3

2

1

0

BOTTOM

Step 2: reverse(stack) Reverse the stack

Logic:

1. Pop the top

2. Recursively reverse the rest

3. Use pushBottom to place popped item at the bottom

Reverse Breakdown:

- Pop 4 reverse(stack)

- Pop 3 reverse(stack)

- Pop 2 reverse(stack)

- Pop 1 reverse(stack)

- Pop 0 reverse(stack)

- Now stack is empty start rebuilding

- pushBottom(0), pushBottom(1), ..., pushBottom(4)

Final Stack (after reverse):

TOP

0

Java Stack pushBottom and reverse() Explained

1

2

3

4

BOTTOM

Output after popping:

0

1

2

3

4

Summary:

Step	What it Does
pushBottom(s, 0)	Adds 0 to bottom of the stack
reverse(s)	Reverses the entire stack

Why use recursion?

Because stack doesnt allow direct access to the bottom recursion helps go deep, then rebuild the stack layer by layer.