**Task 4: Stack Sorting In-Place**

**You must write a function to sort a stack such that the smallest items are on the top. You can use an additional temporary stack, but you may not copy the elements into any other data structure such as an array. The stack supports the following operations: push, pop, peek, and is Empty.**

**Sol:**

**Steps to Sort the Stack Using an Additional Temporary Stack**

**1. Initialize the Temporary Stack**:

• Create an empty temporary stack to hold the sorted elements.

**2. Sorting Process**:

* Pop an element from the original stack.
* While the temporary stack is not empty and the top of the temporary stack is greater than the popped element, move elements from the temporary stack back to the original stack.
* Push the popped element onto the temporary stack.
* Repeat the process until the original stack is empty.

**3. Transfer Sorted Elements Back to Original Stack**:

• After sorting is complete, transfer the elements from the temporary stack back to the original stack so that the smallest items are on top.

**Program:**

import java.util.Stack;

public class SortStack {

public static void sortStack(Stack<Integer> stack) {

Stack<Integer> tempStack = new Stack<>();

while (!stack.isEmpty()) {

int current = stack.pop();

while (!tempStack.isEmpty() && tempStack.peek() > current) {

stack.push(tempStack.pop());

}

tempStack.push(current);

}

// Transfer back to the original stack

while (!tempStack.isEmpty()) {

stack.push(tempStack.pop());

}

}

public static void main(String[] args) {

Stack<Integer> stack = new Stack<>();

stack.push(34);

stack.push(3);

stack.push(31);

stack.push(98);

stack.push(92);

stack.push(23);

System.out.println("Original Stack: " + stack);

sortStack(stack);

System.out.println("Sorted Stack: " + stack);

}

}