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 isEmpty.

Code Implementation

```
import java.util.Stack;

public class StackSorter {
    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); }
        while (!tempStack.isEmpty()){    // Transfer the sorted elements back to the original stack
            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:");
        System.out.println(stack);
        sortStack(stack);
        System.out.println("Sorted Stack:");
        System.out.println(stack);  }}
```

Explanation

1. **Initialize Temporary Stack:** Create an additional stack '**tempStack**' to assist with the sorting.
2. **Process Elements:**
   - Pop an element from the original stack and store it in a variable '**current**'.
   - While '**tempStack**' is not empty and its top element is greater than '**current**', move the top element from '**tempStack**' back to the original stack. This step ensures that elements in '**tempStack**' are in descending order.
3. **Push the Current Element:** Push 'current; onto '**tempStack**'.
4. **Repeat Until Original Stack is Empty:** Continue steps 2 and 3 until the original stack is empty.
5. **Transfer Sorted Elements Back:** Finally, move all elements from '**tempStack**' back to the original stack. Since '**tempStack**' contains elements in descending order, moving them back will place them in ascending order (smallest items on the top).