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.

```
import java.util.Stack;

public class StackSorting {
    public static void sortStack(Stack<Integer> stack) {
        Stack<Integer> tempStack = new Stack<>();

        while (!stack.isEmpty()) {
            int temp = stack.pop();

            while (!tempStack.isEmpty() && tempStack.peek() > temp) {
                stack.push(tempStack.pop());
            }

            tempStack.push(temp);
        }

        // Moving elements 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(5);
        stack.push(3);
        stack.push(8);
        stack.push(1);
        stack.push(2);

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

        sortStack(stack);

        System.out.println("Stack after sorting: " + stack);
    }
}
```