Day 12

Assignment 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.
A)
Introduction:

To sort a stack in-place such that the smallest items are on the top, you can use a temporary stack to assist with sorting. The main idea is to repeatedly pop elements from the original stack, compare them with the top element of the temporary stack, and insert them into the temporary stack in sorted order.

Here's how you can implement this approach in Java:

```
package com.example.stacksorter;
import java.util.Stack;
public class StackSorter {
 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);
    }
    while (!tempStack.isEmpty()) {
      stack.push(tempStack.pop());
    }
  }

  public static void main(String[] args) {
    Stack<Integer> stack = new Stack<>();

    stack.push(5);
    stack.push(2);
    stack.push(9);
    stack.push(1);
    stack.push(3);
    stack.push(6);
 System.out.println("Original Stack: " + stack);
    sortStack(stack);

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

Explanation:

sortStack Method:

Initializes a temporary stack (tempStack) to assist with sorting.

Iterates through the elements of the original stack (stack):

Pops elements from the original stack one by one (int temp = stack.pop()).

Moves elements from tempStack back to stack until the top of tempStack is smaller than temp.

Pushes temp onto tempStack.

Moves elements from tempStack back to stack to obtain the sorted order.

Output

```
Original Stack: [5, 2, 9, 1, 3, 6]

Sorted Stack: [9, 6, 5, 3, 2, 1]
```

Summary:

This approach effectively sorts the stack in-place using an additional temporary stack. It maintains the original stack's operations of push, pop, peek, and isEmpty without copying elements into any other data structure, as per the requirements. The elements are sorted such that the smallest items are on the top of the stack.