SortedStack

Project 05 - Planning and Executing a Maintenance Task

Author: Cole Flenniken
Java Version:
openjdk version "21.0.7" 2025-04-15 LTS
OpenJDK Runtime Environment Temurin-21.0.7+6 (build 21.0.7+6-LTS)
OpenJDK 64-Bit Server VM Temurin-21.0.7+6 (build 21.0.7+6-LTS, mixed mode, sharing)

Overview

The program implements a sorted stack that allows for the insertion of elements in the stack and a print-out of the state of the sorted stack. The stack is sorted in ascending order with the start being the bottom of the stack.

Run Program Instructions

1. Within the terminal, navigate to the directory where the SortedStack.java file resides
2. Run ‘java SortedStack.java’
3. In the case step 2 does not properly begin program execution
    a. Run ‘javac SortedStack.java’ to generate a class file
    b. Run ‘java SortedStack’ to run the class file

Generate JavaDoc Instructions

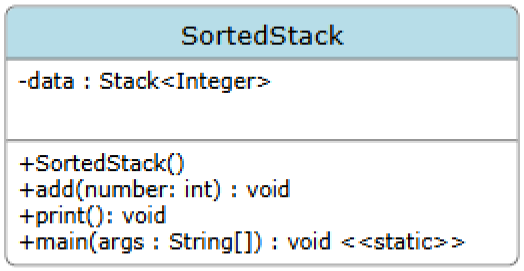
To generate documentation within the same directory as the code:

1. Within the terminal, navigate to the directory where the SortedStack.java file resides
2. Run ‘javadoc -author -version SortedStack.java’

To generate documentation within a separate directory:

1. Within the terminal, navigate to the directory you wish to generate the documentation within
2. Run ‘javadoc -author -version <SortedStack.java file path>’

Class Diagram



Code Reuse examples

1. When scanning for integers, the built-in scanner class is used. In addition, instead of writing custom logic to validate whether part of input is an integer and then read the integer, the hasNextInt() method of the scanner is used in conjunction with the nextInt() method.
2. Instead of writing custom logic to sort the stack, the Collections.sort() function is used
3. The built-in stack class is used instead of creating a custom stack.
4. Instead of writing logic to convert the state of the stack to a string, the println method call to display the stack state uses the toString override of the stack to convert the state to a string.

AI Usage:

The boilerplate for the javadoc comments was generated with Github Copilot. Most of the time it failed to provide all information I wanted so I had to add information to the comments but it did same time typing.

Maintenance Changes:

(requirement specified within lecture but otherwise not normally part of documentation)
Program:

As the name of the class no longer fit its design, the class name, constructor, and file name were changed. Otherwise, methods within the class maintained the same signature. All documentation comments were also updated. The add and print methods needed no changes but, the print method did update to provide more clarity on what sides the top and bottom of the stacks are.The constructor needed to change as the created object for underlying storage also changed.

ReadME:
References to a sortedlist were exchanged with references to the updated class, sortedstack. The class diagram was also updated.