SortedList

Project 04 - Linked List and Collections Framework

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 list that allows for the insertion of elements and the list and a print-out of the state of the sorted list.

Run Program Instructions

1. Within the terminal, navigate to the directory where the SortedList.java file resides
2. Run 'java SortedList.java'
3. In the case step 2 does not properly begin program execution
   a. Run 'javac SortedList.java' to generate a class file
   b. Run 'java SortedList' to run the class file
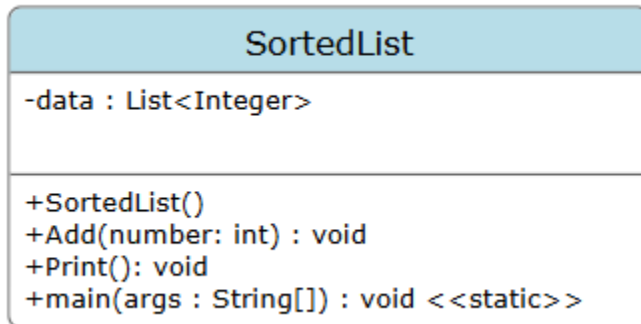
Generate JavaDoc Instructions

To generate documentation within the same directory as the code:

1. Within the terminal, navigate to the directory where the SortedList.java file resides
2. Run 'javadoc -author -version SortedList.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 <SortedList.java file path>'

Class Diagram

SortedList
-data : List<Integer>
+SortedList() +Add(number: int) : void +Print(): void +main(args : String[]) : void <<static>>

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 linked list, the Collections.sort() function is used
3. The built-in linked list class is used instead of creating a custom linked list.
4. Instead of writing logic to convert the state of the list to a string, the println method call to display the list state uses the toString override of the linkedList 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.