Sorting Visualiser – Project Documentation

Table of Contents

Introduction

Sorting Visualiser is an educational Java application designed to provide an interactive and intuitive understanding of sorting algorithms. By visualizing the step-by-step execution of various sorting techniques, the application bridges the gap between theoretical learning and practical comprehension. Users can observe real-time graphical representations of sorting processes, making complex concepts accessible and engaging.

Objectives

- Deliver a visual and interactive platform for learning and teaching sorting algorithms.
- Illustrate the internal workings and stepwise execution of multiple sorting techniques.
- Enable experimentation with different data sizes and sorting speeds.
- Support educators and students in computer science and programming courses.

System Design

Architecture Overview

The application is modularly structured into three primary Java classes:

- **SortingVisualiser.java**: Serves as the entry point and main window for the application, initializing the GUI and handling the main event loop.

- **DataStructureVisualizer.java**: Responsible for the visualization logic, rendering, and animating the sorting steps and data structures.
- **VisualizerHub.java**: Acts as the central controller, managing user interactions, algorithm selection, and coordination between visualizer components.

User Interface

- Graphical display of data elements as bars or nodes, dynamically updated during sorting.
- Intuitive controls for selecting sorting algorithms, adjusting animation speed, and resetting data.
- Real-time animation of sorting steps, allowing users to pause, resume, or reset the visualization at any point.

Implementation Details

Technologies Used

- Java (JDK 8 or higher)
- Java Swing/AWT for graphical user interface (GUI) development

Key Classes

- **SortingVisualiser.java**: Initializes the main application window, sets up the GUI, and manages the main event loop.
- **DataStructureVisualizer.java**: Contains the core logic for rendering data structures and animating the sorting process.
- **VisualizerHub.java**: Centralizes control, handling algorithm selection, user commands, and communication between components.

Sorting Process

- Data is internally represented as an array of integers.
- Each sorting algorithm is implemented as a dedicated method, which updates the array and triggers visual updates after each significant operation.
- Animation is achieved by repainting the GUI after every key step, providing a clear and continuous visualization of the sorting process.

User Guide

Installation

1. Ensure Java (JDK 8 or higher) is installed on your system. You can download it from the official Java website.
2. Download or clone the project files to your local machine.

Compilation

Open a terminal or command prompt in the project directory and execute:

```
javac SortingVisualiser.java DataStructureVisualizer.java VisualizerHub.java
```

Running the Application

After successful compilation, run the application with:

```
java SortingVisualiser
```

Using the Visualiser

- Select a sorting algorithm from the provided menu.
- Adjust the animation speed and data size as desired using the available controls.
- Click 'Start' to begin the visualization.
- Use 'Pause' to temporarily halt the animation, and 'Reset' to restore the initial data state.

Supported Algorithms

ALGORITHMS

- Bubble Sort
- Selection Sort
- Insertion Sort
- Merge Sort
- Quick Sort
- Heap Sort
- Shell Sort
- Radix Sort

DATA STRUCTURES

- Stack
- Queue
- LinkedList
- Circular LinkedList
- Doubly LinkedList

Future Enhancements

- Integration of additional data structures (e.g., trees, graphs) for visualization.
- Export functionality to save visualizations as images or videos.
- Enhanced UI/UX with modern design elements and improved accessibility.
- Multi-language support for broader accessibility.

References

- Java Official Documentation

For further details, please refer to the source code and in-line comments within each Java file. Contributions, feedback, and suggestions are welcome to help improve and expand the project.