HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATION COMMUNICATION TECHNOLOGY

OBJECTIVE-ORIENTED PROGRAMMING

IT3100E

INSTRUCTOR: PROF. TRAN THE HUNG

Class ID: 147839

Sorting Visualization

Group Members:

Bui Nguyen Minh- 20226055
Nguyen Cong Minh - 20226056
Pham Duy Hoang - 20226042
Le Van Hau - 20226038

Contents

Sorting Visualization: A Tool for Interactive Learning

FireFly

June 9, 2024

Abstract

This report presents a detailed overview of the Sorting Visualization project developed using Object-Oriented Programming principles. The project aims to visualize various sorting algorithms to enhance understanding and facilitate learning. work.

1 Introduction

1.1 Background

Sorting algorithms are fundamental concepts in computer science. Understanding their behavior and performance characteristics is crucial for both students and professionals. This article presents the development of a Java Swing application that visualizes sorting algorithms to aid learning.

1.2 Project Objectives

The primary objective of this project is to create a visual representation of sorting algorithms, specifically Merge Sort, Bubble Sort, Insertion Sort, and Selection Sort. This interactive tool allows users to witness the sorting process firsthand, improving comprehension of the algorithms' functionality and efficiency.

1.3 Assignment of members

Member	ID	Work
Bui Nguyen Minh (Leader)	20226055	Managing project and other member's assignments, Designing class diagram, Designing main Menu class and bubbleSort visualizer, Some update for visualizer screen
Nguyen Cong Minh	20226056	Designing use case diagram for project, Designing visualizer screen and some buttons with effects, Creating abstract class Sorting and InsertionSort Visualizer, Making presentation slide
Pham Duy Hoang	20226042	Designing SelectionSort Visualizer, Video demo
Le Van Hau	20226038	Designing MergeSort and MergeSort Visualizer, Writing report for project

Table 1: Team Members and Their Contributions

2 Project Functionality

The application offers functionalities such as:

- Choosing a desired sorting algorithm.
- Generating random data sets for sorting.
- Visualizing the sorting process in real-time using bar charts.
- Accessing a help menu for guidance.
- Navigating back to the main menu or exiting the application.

The application comes with limitations to ensure optimal performance:

Only non-negative integers can be used as data elements. The maximum number of elements displayed for visualization is capped at 200. Valid data set sizes for sorting algorithms range from 30 to 350 elements.

3 Project Description

3.1 Project Overview

The project aims to build an application that visualizes three sorting algorithms: Merge Sort, Counting Sort, and Radix Sort. The application helps users understand how these algorithms work through visual representation.

3.2 Project Requirements

The application has certain restrictions:

- Only non-negative (>0) numbers are allowed as array elements.
- The array size for visualization is limited to a maximum of 200 elements.
- Valid array values for sorting algorithms range from 30 to 350.

3.3 Use Case Diagram and Explanation

The use case diagram includes functionalities such as choosing a sorting algorithm, generating data, visualizing the sorting process, viewing the help menu, and navigating back to the main menu or exiting the application.

4 Technical Design

The application utilizes object-oriented programming principles for a well-structured and maintainable codebase. Class diagrams (replace with placeholders for your actual diagrams) illustrate the relationships between various functionalities within the application. These diagrams detail the functionalities of packages like Sorting, MainMenuApplication, Bars, Buttons, and Visualizer.

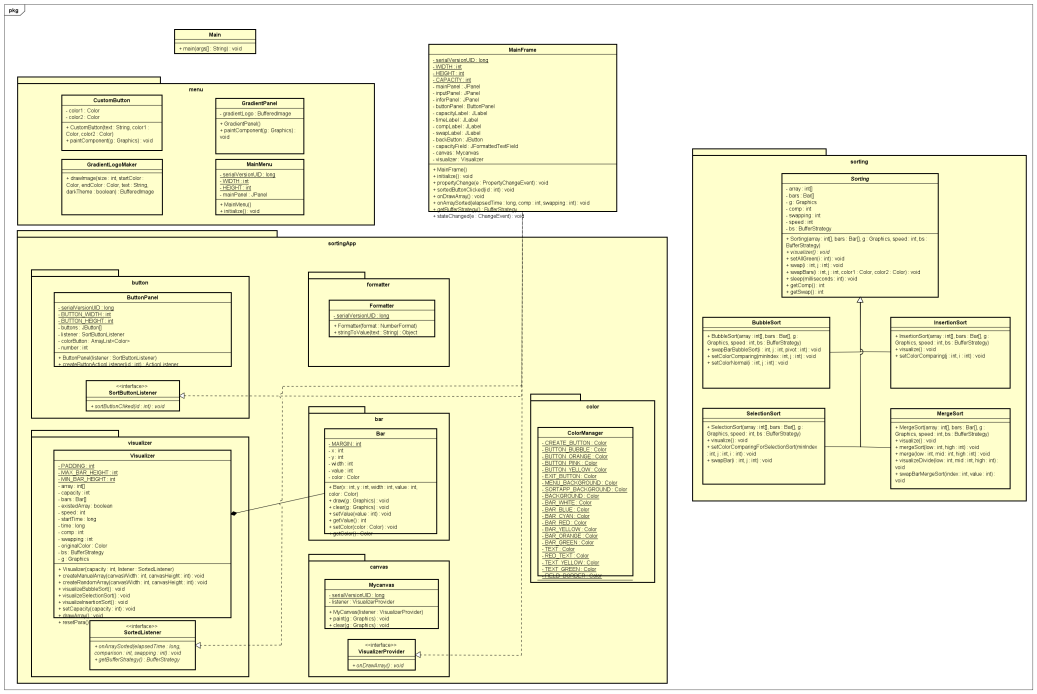

Figure 1: General Class Diagram (replace with your actual diagram)

"Our detailed class diagram and Use case diagram are available on GitHub."

5 Implementation Details

The core functionalities lie in the implementation of the chosen sorting algorithms (Merge Sort, Bubble Sort, Insertion Sort, and Selection Sort) within the Sorting package. Each algorithm is implemented within its respective class. The MainMenuApplication package handles functionalities related to the main menu and user interaction.

5.1 Class structure

```
public class MergeSort {
    public void sort(int[] array) {
        // Implementation of merge sort
    }
}
```

5.2 Algorithm Implementation

- **MergeSort:**
  - **Description:** Merge Sort is a divide and conquer algorithm that recursively divides the input array into two halves until each half contains only one element. It then merges these halves in a sorted manner.
  - **Implementation Explanation:**

```
public class MergeSort {
    public void sort(int[] array) {
        // Implementation of merge sort
    }
}
```

- **Selection Sort:**
  - **Description:** Selection Sort is a simple sorting algorithm that repeatedly finds the minimum element from the unsorted part of the array and moves it to the beginning.
  - **Implementation Explanation:**

```
public class SelectionSort {
    public void sort(int[] array) {
        // Implementation of selection sort
    }
}
```

- **Bubble Sort:**

- **Description:** Bubble Sort is a simple sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order.
- **Implementation Explanation:**

```
public class BubbleSort {
    public void sort(int[] array) {
        // Implementation of bubble sort
    }
}
```

- **Insertion Sort:**
  - **Description:** Insertion Sort is a simple sorting algorithm that builds the final sorted array one element at a time by repeatedly moving elements that are larger than the current element to one position ahead of their current position.
  - **Implementation Explanation:**

```
public class InsertionSort {
    public void sort(int[] array) {
        // Implementation of insertion sort
    }
}
```

6 Visualizations and Performance Analysis

The application generates real-time visualizations of the sorting process using bar charts. These visualizations allow users to observe how the algorithms arrange the data elements. Further analysis can be conducted to assess the performance of each sorting algorithm based on the visualizations generated.

7 Conclusion

This project successfully developed a user-friendly application that visualizes sorting algorithms. The interactive nature of the application strengthens understanding of these algorithms, making it a valuable tool for both students and program-

mers. Future work could involve incorporating additional sorting algorithms and exploring alternative visualization techniques.