Employee Management Application

21:198:335:02 Data Structures & Algorithms
Data Structures Final Project

Zeyad Rashed
Rutgers ID: 199009651
Professor Weiping (Veronica) Zhang

December 6, 2024

Abstract

This document describes the design, implementation, and analysis of the Employee Management Application. The project demonstrates key data structures and algorithms concepts, including sorting algorithms, method overloading, file I/O, and GUI design.

1 Introduction

The Employee Management Application was developed as part of the Data Structures & Algorithms course final project. This project incorporates fundamental principles discussed in *Data Structures and Algorithm Analysis in Java, Third Edition* by Clifford A. Shaffer. The application meets the following requirements:

- Management of employee records, including attributes such as ID, name, salary, and department.
- Multiple sorting options, including Bubble Sort, Heap Sort, and Comparator-based sorting.
- A JavaFX-based graphical user interface for seamless interaction.
- Performance measurement for sorting operations.

2 Design Philosophy

The project adheres to Shaffer's principle of balancing trade-offs between space and time complexity. Emphasis was placed on modularity, scalability, and maintainability. By encapsulating sorting logic in a dedicated class and delegating file I/O operations, the project follows the abstraction principles outlined in Shaffer's text.

3 Key Features and Implementation

3.1 Sorting Algorithms

Sorting is a fundamental operation in computer science. The application employs both $O(n^2)$ (Bubble Sort) and $O(n \log n)$ (Heap Sort) algorithms, allowing users to toggle between them dynamically.

```
1  /** Bubble Sort implementation for sorting employee lists */
2  public static <T> void bubbleSort(List<T> items,
       Comparator<T> comparator) {
3      for (int i = 0; i < items.size() - 1; i++) {
4          for (int j = 0; j < items.size() - i - 1; j++) {
5              if (comparator.compare(items.get(j), items.
                   get(j + 1)) > 0) {
6                  Collections.swap(items, j, j + 1);
7              }
8          }
9      }
10 }
```

3.2 Performance Measurement

The project tracks sorting times to demonstrate efficiency differences. Results are displayed in the GUI to provide feedback.

```
1  /** Measures performance of the current sorting algorithm */
2  private void sortAndMeasurePerformance(Comparator<
       Employee> comparator) {
3      long startTime = System.nanoTime();
4      if (useHeapSort) {
5          StackSorter.heapSort(employees, comparator);
6      } else {
7          StackSorter.bubbleSort(employees, comparator);
8      }
9      long duration = System.nanoTime() - startTime;
10     displayArea.setText(formatEmployeeList());
11     showAlert("Sorting Performance", "Time taken: " +
           duration + " ns", Alert.AlertType.INFORMATION);
12 }
```

3.3 Graphical User Interface

The GUI was developed using JavaFX, enabling user-friendly interaction. Features such as toggling between different algorithms and different sorting views are included.

4 Application of Theoretical Concepts

4.1 Trade-offs in Algorithm Selection

As highlighted in Shaffer's textbook (Section 7.2), the choice of algorithm depends on problem constraints. Bubble Sort is simpler but inefficient for large datasets. Heap Sort, while more complex, scales efficiently.

4.2 Abstraction and Modularity

The modular design of the application reflects the abstraction principles in Section 1.2 of Shaffer's book. Encapsulation of logic in distinct classes allows for scalability and easier debugging.

4.3 File Handling and I/O

The `EmployeeFileHandler` class demonstrates robust file operations, enabling reading, writing, and appending employee records.

5 Diagrams

5.1 UML Diagram

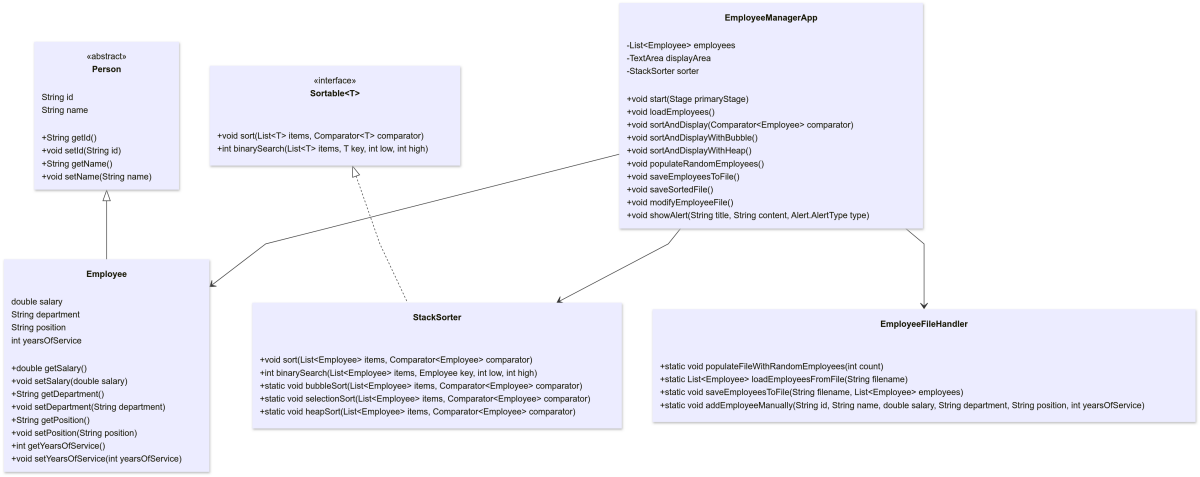

Figure 1: UML Diagram of the Employee Management Application

5.2 Flowchart

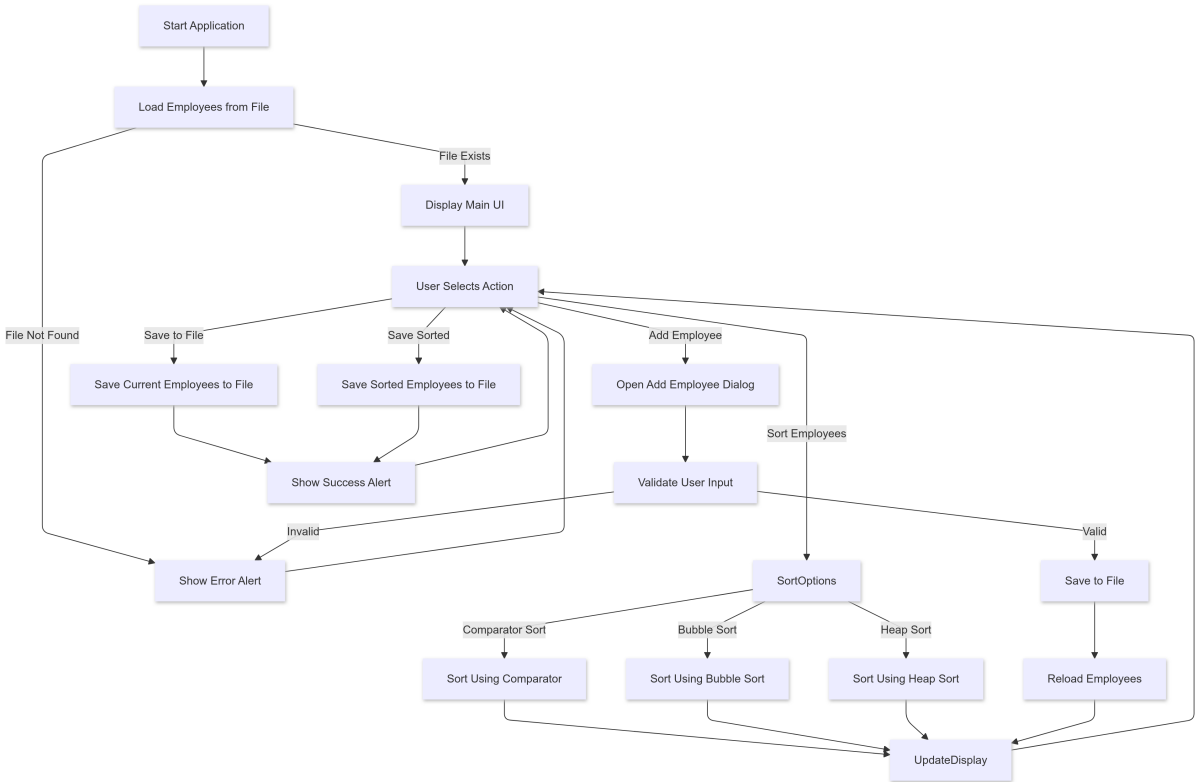

Figure 2: Flowchart of Application Workflow

6 Conclusion

The Employee Management Application showcases the practical application of data structures and algorithmic principles. By integrating theoretical knowledge with real-world implementation, the project exemplifies efficient, maintainable, and user-focused software design.

7 References

1. Shaffer, C. A. (2013). *Data Structures and Algorithm Analysis in Java, Third Edition.*

2. JavaFX Documentation: `https://openjfx.io/javadoc/23/`

3. Java Collections Framework: `https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html`