CCT College Dublin

|  |  |
| --- | --- |
| Module Title: | Algorithms & Constructs |
| Assessment Title: | CA 2 Algorithms & Constructs – 50% |
| Lecturer Name: | Dr. Muhammad Iqbal |
| Student Full Name: | Vitor Oliveira Trindade |
| Student Number: | 2024336 |
| Assessment Due Date: | 08/10/2025 |
| Date of Submission: | 10/10/2025 |

Declaration

|  |
| --- |
| By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution. |

**Solution Development and Justification of the Algorithms Used**

# Introduction and Project Context

The project proposed as part of the *Algorithms and Constructs* module aims to model and implement an employee management system for an organization, simulating the corporate structure of a tech company. The solution was designed to incorporate and apply key concepts explored in class, such as Big-O notation for algorithm efficiency, object-oriented programming principles (inheritance, composition, factory patterns), recursion, and practical usage of search and sorting algorithms.

The system was fully implemented in Java, following best practices in programming and object-oriented design. The core classes developed include Employee, EmployeeManager, EmployeeFactory, FileHandler, and Menu, all grouped under the package CA\_2. These classes are responsible for essential features such as employee registration, data reading and writing, sorting, searching, listing, and data persistence. Additionally, the system uses enum types for departments (DepartmentType) and positions (PositionType), ensuring data consistency and reducing input errors.

The interface is entirely console-based, with interactive menus implemented using enum, as required by the assignment. The system allows users to import data from a .txt file, sort employee records, search by full name, manually add new entries, and generate random employee data based on predefined structures.

One of the core requirements of the project was the implementation of custom sorting and searching algorithms. The following sections focus on the rationale behind the algorithm choices.

# Justification for the Sorting Algorithm

The algorithm chosen to sort the employee list was **Insertion Sort**, implemented manually through the insertionSortEmployees() method within the EmployeeManager class.

This choice was motivated by several factors:

* **Alignment with module requirements**: The selected algorithm fulfills the requirement to implement a custom sorting solution based on content covered in class. Insertion Sort is a classical, didactic algorithm that is easy to understand and ideal for educational purposes.
* **Efficient for small datasets**: Since this system is designed as a prototype, it is not expected to handle large volumes of data. For small lists (up to a few hundred records), Insertion Sort performs well and can even outperform more complex algorithms due to its low overhead.
* **Easy to maintain and extend**: The code is simple and straightforward, making it easy to adapt in the future for different sorting criteria (e.g., by salary, position, or company).
* **Good performance on partially sorted data**: Insertion Sort performs especially well when the list is nearly sorted — which is common after incremental additions or previous sorts.

More advanced algorithms such as Merge Sort or Quick Sort were considered but ultimately not implemented due to their added complexity. While these algorithms are more efficient for large datasets, they introduce unnecessary complications in a learning context and reduce visibility into each step of the sorting process.

# Final Considerations and Critical Reflection

The implementation of custom sorting and search algorithms not only met the technical requirements of the assignment but also provided an opportunity to apply foundational concepts from data structures and algorithms in a real-world context.

**Strengths of the approach:**

* Clear and well-organized code.
* Object-oriented structure using enums and modular design.
* Interactive console menu with enum-based options.
* Strong input validation for user-provided data.
* Integration with file I/O for persistent storage.
* Practical application of recursion and algorithm complexity awareness.

**Identified weaknesses:**

* Lack of automated unit testing affects the system’s robustness.
* Insertion Sort becomes inefficient for large datasets and would need to be replaced in a production environment.
* Sorting is executed before every search, even when the data has not changed, which could lead to unnecessary overhead.

**Potential future improvements:**

* Implement a flag to check if the list is already sorted, avoiding redundant operations.
* Use a more advanced data structure (e.g., a binary search tree) to allow for fast lookups without the need for sorting.
* Add automated unit tests to improve system reliability and support future refactoring.

# References

* Oracle (n.d.) *The Java™ Tutorials – Enum Types*. Available at: <https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html> (Accessed: 28 Apr 2025).
* GeeksforGeeks (2024) *Binary Search Algorithm*. Available at: [https://www.geeksforgeeks.org/binary-search/](https://www.geeksforgeeks.org/binary-search/%20) (Accessed: 28 Apr2025).
* GeeksforGeeks (2024) *Insertion Sort Algorithm*. Available at: https://www.geeksforgeeks.org/insertion-sort/ (Accessed: 9 Apr 2025).
* Wikipedia (2024) *Binary search algorithm*. Available at: <https://en.wikipedia.org/wiki/Binary_search_algorithm> (Accessed: 28 May 2025).
* Wikipedia (2024) *Insertion sort*. Available at: <https://en.wikipedia.org/wiki/Insertion_sort> (Accessed: 28 Apr 2025).

### Appendix

**GitHub Repository**

GitHub repository created and invitation sent to the lecturer.  
Repository link: <https://github.com/VitorOliveiraTrindadeCCT/VTR-TECH>