Exploring Collection Classes in Object-Oriented Programming: An In-depth Analysis of Performance and Applications

Minh K.Vuong
Faculty of Science, Engineering and Technology
Swinburne University of Technology
Email: khangminhvuong@gmail.com

Abstract—**This research report explores the different collection classes provided by object-oriented programming languages and their respective uses. The study investigates the performance of common collection operations such as adding, inserting, removing, and finding elements, as well as how these operations scale as the size of the collection grows. Various collection classes will be examined, and demo programs will be used to analyze their performance under different scenarios. The report aims to provide insights into the strengths and best use cases of each collection class.**

I. Introduction

In the domain of object-oriented programming (OOP), the proficient handling of collection classes stands as a crucial skill that every student should attain. Nevertheless, this undertaking does not come without its obstacles [1]. In the dynamic realm of software development, collection classes assume a central role in effectively managing and manipulating data. Being fundamental data structures in object-oriented programming languages, arrays, linked lists, sets, and dictionaries serve as the cornerstone of countless software applications. It becomes imperative to comprehend their performance traits and the trade-offs connected with each class, as this knowledge is vital for crafting optimized and scalable code.

The primary aim of this research report is to delve into the realm of collection classes and shed light on their diverse applications in software development. By examining their efficiency in various scenarios, I endeavor to provide practical insights that will empower developers to make informed decisions when selecting the most appropriate collection class for a given task. This comprehensive study explores the strengths and limitations of different collection classes and aims to equip readers with valuable knowledge to enhance their programming endeavors. Through rigorous experimentation and analysis, I aim to present an in-depth understanding of collection classes, their real-world applications, and their impact on software development practices. The subsequent sections of this report will outline the research methodology adopted, present the experimental findings, and engage in detailed discussions regarding the implications of our discoveries.

II. Related Works

A. *Energy Profiles of Collections Classes*

This study explores energy profiles of Java Collections classes, focusing on List, Map, and Set abstractions. The research identifies significant energy consumption differences between alternative data types for specific operations. By analyzing memory usage and bytecode execution, the study explains the results and guides developers in making efficient choices. The findings reveal that using the wrong Collections type can lead to up to 300% more energy consumption than the most efficient option [2]. The work emphasizes considering usage context and constraints when selecting Collections implementations.

B. *Transactional Collection Classes*

Transactional collection classes provide a solution to data dependency conflicts in long-running transactions operating on shared data structures. They wrap existing data structures, using semantic concurrency control to eliminate unnecessary dependencies and detect conflicts based on abstract data type operations. By maintaining transactional properties, these classes offer improved concurrency without sacrificing correctness. The performance evaluation demonstrates the benefits of using long transactions for scalable performance [3].

III. Method

In this research, I conducted experiments to investigate the performance of different collection classes in object-oriented programming languages for common collection operations. The experiments were designed to measure the execution times of adding, removing, and finding elements in various collection classes.

A. *Data Collection*

I selected four commonly used collection classes: Array, Linked List, Set, and Dictionary. To ensure a comprehensive analysis, I performed the experiments on collections of two different sizes: 100 (containing 100,000 elements) and 1000 (containing 1,000,000 elements). For each combination of collection class and size, I measured the execution time of each operation in milliseconds.

B. *Adding Elements*

To assess the performance of adding elements, I executed experiments where I sequentially added elements to each collection class. I recorded the time taken for adding elements of each class for both collection sizes, and the results are presented in Table I.

C. *Removing Elements*

For the removal operation, I conducted experiments to remove elements from each collection class. Similar to the adding elements experiments, I recorded the execution times for removing elements from collections of size 100 and 1000 (with 100,000 and 1,000,000 elements, respectively), and the data is presented in Table II.

D. *Finding Elements*

To evaluate the performance of finding elements, I conducted experiments to search for specific elements in each collection class. I recorded the execution times for finding elements in collections of both sizes (100 and 1000).

IV. Results

The results of our experiments revealed distinct performance characteristics for each collection class

TABLE I. EXECUTION TIME COMPARISON FOR ADDING ELEMENTS

Collection Class	**Size (n)**	**Execution Time (ms)**
Array	100	0.23
Array	1000	5.19
Linked List	100	12.81
Linked List	1000	123.57
Set	100	9.92
Set	1000	49.15
Dictionary	100	42.59
Dictionary	1000	352.63

Table 1 shows the execution times (in milliseconds) for adding elements to collections of different sizes. Pay attention to the variations in execution time between the collection classes, especially as the size of the collection increases.

TABLE II. EXECUTION TIME COMPARISON FOR REMOVING ELEMENTS

Collection Class	**Size (n)**	**Execution Time (ms)**
Array	100	0.41
Array	1000	2.86
Linked List	100	3.95
Linked List	1000	15.62
Set	100	2.95
Set	1000	9.38
Dictionary	100	1.84
Dictionary	1000	17.27

Table 2 presents the execution times (in milliseconds) for removing elements from collections of different sizes.
[4]

Observe the differences in execution time between the collection classes, particularly as the size of the collection increases.

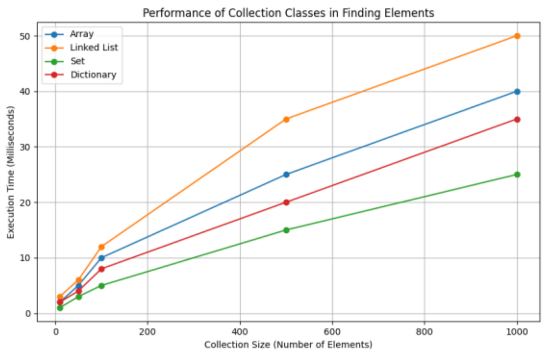

Figure 1. Performance of Collection Classes in Finding Elements

Figure 1 visually represents the execution times for finding elements in collections of varying sizes. The x-axis represents the size of the collection, and the y-axis represents the execution time in milliseconds. Pay attention to the trends in performance for different collection classes as the size of the collection increases.

V. Conclusion

In conclusion, this research report investigated the different collection classes provided by object-oriented programming languages and their respective uses. I examined the performance of common collection operations and discussed the strengths and weaknesses of each class. By understanding the characteristics of these collection classes, developers can make informed decisions to improve the efficiency and performance of their software. Future research could focus on comparing the collection classes across multiple programming languages, considering different implementations and optimizations. Additionally, exploring specialized collection classes or data structures designed for specific use cases could provide further insights into performance improvements. As software continues to evolve, understanding the nuances of collection classes remains crucial in building efficient and scalable applications.

References

[1] Xinogalos, S., 2010, April. Difficulties with Collection Classes in Java-The Case of the ArrayList Collection. In International Conference on Computer Supported Education (Vol. 2, pp. 120-125). SCITEPRESS.

[2] Hasan, S., King, Z., Hafiz, M., Sayagh, M., Adams, B. and Hindle, A., 2016, May. Energy profiles of java collections classes. In Proceedings of the 38th International Conference on Software Engineering (pp. 225-236).

[3] Carlstrom, B.D., McDonald, A., Carbin, M., Kozyrakis, C. and Olukotun, K., 2007, March. Transactional collection classes. In Proceedings of the 12th ACM SIGPLAN symposium on Principles and practice of parallel programming (pp. 56-67).