Which queue implementation LinkedList or ArrayDeque is the most efficient for different types of use cases within a Queue interface?

NAAM: GING NAM FONG
STUDENTENNUMMER: 500953220
DATUM: 14-11-2025
LOOPBAANADVISEUR / STUDENTBEGELEIDER: YAHIA ELSHERBINI, MASA MARCINKO TOPIC

Table of Contents

1. Introduction

Queues are one of the fundamental data structures, used for managing tasks, events, and data streams In Java, the Queue interface provides a standard way to organize and process elements. However, different implementations of the Queue interface vary significantly in performance and memory efficiency, depending on their underlying data structures.

This research focuses on two commonly used implementations of the Java Queue interface: LinkedList and ArrayDeque. By analyzing their structure, operational characteristics, and performance under typical use cases.

2. What is Java Queue Interface?

The Java Queue interface primarily represents a First-In, First-Out (FIFO) data structure for handling elements in the order they are processed new elements are added at the rear and removed from the front (Baeldung, 2024). However, FIFO ordering is not guaranteed for all implementations. For example, a PriorityQueue arranges elements according to their natural ordering rather than insertion order (GeeksforGeeks, 2025).

Java provides multiple implementations of the Queue interface, each with different internal structures and performance characteristics. In this research, the focus is on two commonly used implementations:

- LinkedList, which uses a doubly linked list data structure
- ArrayDeque, which uses a resizable array as its data structure

Although both implementations are a Queue operation, their underlying data structures affect memory usage, CPU cache performance, and overall performance in different ways. Understanding these differences is essential when selecting the most efficient implementation for specific use cases.

3. How does LinkedList behave as a Java Queue?

A LinkedList in Java is implemented as a doubly linked list, in which each element is stored in a node object that are created dynamically as needed containing references to the previous and next nodes.

LinkedList offers constant-time insertion and removal at both ends However, every element generates additional memory overhead due to the node and references. This makes the structure less memory-efficient and introduces overhead *(Codemia)*.

LinkedList is flexible and suitable for applications requiring frequent insertions, deletions, or sequential traversal, such as task scheduling and playlists.

4. How does ArrayDeque behave as a Java Queue?

An ArrayDeque is a resizable array to store elements. It supports constant-time operations at both ends and automatically resizes automatically as needed. It is not thread-safe, and null elements are not allowed. According to Oracle, this class is likely to be faster than LinkedList when used as a queue *(Oracle).*

ArrayDeque benefits from storing data in contiguous memory, making iteration significantly faster and improving cache efficiency. This tight memory layout reduces overhead and enables faster traversal *(Codemia)*. ArrayDeque occasionally reallocates its internal array when capacity is exceeded, but this cost is infrequent and has an amortized time complexity of O(1).

ArrayDeque is well-suited for high-performance queue, deque operations, event queues, and workloads that involve frequent iteration.

5. Which implementation performs better under typical queue operations?

Multiple Sources show that ArrayDeque consistently outperforms LinkedList for almost all queue use cases. The reasons include:

Memory Efficiency
ArrayDeque stores elements in a compact array, minimizing overhead. It is more memory-efficient than LinkedList because LinkedList must allocate a separate node object for every element, which significantly increases memory usage *(Codemia).*

Cache Locality
Contiguous memory access in ArrayDeque improves cache locality. In contrast, LinkedList allocates each element as an independent node object, which leads to poor cache behavior due to pointer navigation and non-contiguous memory placement (Java-Performance.info, 2023).

Operational Overhead
LinkedList must constantly allocate and deallocate node objects. This increases memory overhead and can contribute to higher garbage-collection activity because more small objects are repeatedly created and discarded. In contrast, ArrayDeque stores elements in a single resizable array and primarily updates index pointers, resulting in far fewer allocations *(Codemia).*

Iteration Speed

ArrayDeque iteration is much faster due to element locality. LinkedList iteration requires navigating node references, which is inherently slower.

6. Conclusion and Recommendation

Based on structural analysis, performance data, and official documentation, ArrayDeque is the most efficient queue implementation for nearly all use cases within the Java Queue interface. It delivers faster operations, reduced memory usage, better cache performance, and simpler internal management. LinkedList only provides advantages in niche cases requiring frequent middle insertions that fall outside the typical behavior of a queue.

Recommendation: Use ArrayDeque as the default queue implementation unless a specific requirement makes LinkedList necessary, which is uncommon in real queue workloads.

7. References

Geeksforgeeks. (2025). Queue Interface In Java
https://www.geeksforgeeks.org/java/queue-interface-java/

Baeldung. (2024). *Guide to the Java Queue Interface*.
https://www.baeldung.com/java-queue

Codemia. *Why is ArrayDeque better than LinkedList?*
https://codemia.io/knowledge-hub/path/why_is_arraydeque_better_than_linkedlist

JavaGuides. (2023). *ArrayDeque vs LinkedList in Java*.
https://www.javaguides.net/2023/11/arraydeque-vs-linkedlist-in-java.html

Java-Performance.info. (2023). *LinkedList Performance*.
https://java-performance.info/linkedlist-performance/

Oracle. *Class ArrayDeque*. Oracle Java Documentation.
https://docs.oracle.com/javase/8/docs/api/java/util/ArrayDeque.html

Oracle. *Class LinkedList*. Oracle Java Documentation.
https://docs.oracle.com/javase/8/docs/api/java/util/LinkedList.html