ArrayList vs LinkedList Detailed Differences

ArrayList and LinkedList are both implementations of the List interface in Java, but they differ significantly in their underlying data structures and performance characteristics.

ArrayList:

- Data Structure: Uses a dynamic array; elements stored in contiguous memory locations.
- Random Access: Fast (O(1)), elements can be accessed directly by index.
- Insertion/Deletion: Slower (O(N)) in the middle due to shifting elements; faster at the end.
- Memory Usage: More memory-efficient, minimal overhead.
- Cache Locality: Better due to contiguous allocation, which improves performance.

LinkedList:

- Data Structure: Uses a doubly linked list; each element stores references to previous and next nodes.
- Random Access: Slower (O(N)), traversal needed to reach an index.
- Insertion/Deletion: Faster (O(1)) anywhere, updates only pointers of surrounding nodes.
- Memory Usage: Less memory-efficient due to extra pointers per node.
- Cache Locality: Poorer compared to ArrayList because elements are not contiguous.

Choosing between ArrayList and LinkedList:

- Use ArrayList when:
- Frequent random access is required.
- Insertions/deletions are mainly at the end.
- Memory efficiency is important.
- Use LinkedList when:
- Frequent insertions/deletions occur, especially in the middle or ends.
- Random access by index is less frequent.