**🔗 Types of Linked Lists**

**1. Singly Linked List**

Each node links to the **next** node only.

public class ListNode

{

public int Value;

public ListNode Next;

}

✅ Use Case:

* Simple, memory-efficient list traversal (forward only).

**2. Doubly Linked List**

Each node links to the **next** and **previous** nodes.

public class DoublyListNode

{

public int Value;

public DoublyListNode Prev;

public DoublyListNode Next;

}

✅ Use Case:

* Traversing both directions.
* Deletion is easier than in singly linked list.

**3. Circular Linked List**

The **last node points to the first** node, forming a circle.

📌 Variants:

* **Singly Circular**: last node’s Next → head
* **Doubly Circular**: tail’s Next → head, and head’s Prev → tail

public class CircularListNode

{

public int Value;

public CircularListNode Next;

}

✅ Use Case:

* Round-robin scheduling
* Circular buffers

**4. Skip List (Advanced)**

* Built on multiple layers of linked lists.
* Some nodes have pointers that **skip over intermediate nodes** for fast lookup.

✅ Use Case:

* Probabilistic alternative to balanced trees
* Faster search than simple linked lists

**📌 Summary Table**

| **Type** | **Can Traverse Backward** | **Circular Option** | **Complexity** |
| --- | --- | --- | --- |
| Singly Linked List | ❌ No | ✅ Optional | Easy |
| Doubly Linked List | ✅ Yes | ✅ Optional | Moderate |
| Circular Linked List | ❌ / ✅ (depends) | ✅ Always | Moderate |
| Skip List | ❌ / Partial | ❌ | Advanced |