1. Array

- Fixed-size.
- linear collection of elements of the same type.
- Access: O(1).

Business Case: Suitable for scenarios where the number of elements is known in advance and does not change.

2. **ArrayList**

- ArrayList can grow or shrink as needed.
- Can store elements of different types.
- Does not enforce type safety (can store any type of object).

Business Case: Suitable for scenarios where the number of elements is Unknown and I don't care about performance and memory.

3. List<T>

- List<T> can grow or shrink as needed.
- enforce type safety One Type.
- Access: O(1) .

Business Case: Useful for dynamic collections where the number of elements can change (the same ArrayList but Generic).

4. Stack<T>

- LIFO (Last-In-First-Out) collection.
- Access: O(n)
- Insertion: O(1)
- Deletion: O(1)

Business Case: Ideal for scenarios where the last added element is the first to be processed.

5. Queue<T>

- FIFO (First-In-First-Out) collection.
- Access: O(n)
- Insertion: O(1)
- Deletion: O(1)

Business Case: Useful for processing tasks in the order they arrive.

6. Dictionary<TKey, TValue>

- Hash table for storing key-value pairs.
- Access: O(1)
- Insertion: O(1)
- Deletion: O(1)

Business Case: Ideal for scenarios requiring fast lookups by keys.

7. HashSet<T>

- Hash table for storing unique elements.
- Insertion: O(1).
- Access: O(1).
- Deletion: O(1).

Business Case: Suitable for collections where uniqueness is required.