C# Collection Framework - Full Detailed Guide

What is a Collection?

A collection is an object that holds a group of related elements (values or objects). Collections are more flexible than arrays because they can:

- Grow or shrink dynamically
- Store heterogeneous or homogeneous data
- Offer many built-in methods like sorting, searching, filtering, etc.

Categories of Collections

C# Collections are categorized into:

- Non-Generic Collections (System.Collections)
- Generic Collections (System.Collections.Generic)
- Concurrent Collections (System.Collections.Concurrent)
- Specialized Collections (System.Collections.Specialized)

1. Non-Generic Collections

Not type-safe. Use objects.

a) ArrayList

```
ArrayList list = new ArrayList();

list.Add(10); list.Add("Koustubh");
```

b) Hashtable

```
Hashtable ht = new Hashtable();

ht.Add("id", 101);
```

c) Stack (LIFO)

```
Stack stack = new Stack();

stack.Push(1); Console.WriteLine(stack.Pop());
```

d) Queue (FIFO)

```
Queue queue = new Queue();

queue.Enqueue("A"); Console.WriteLine(queue.Dequeue());
```

2. Generic Collections (Type-Safe)

a) List<T>

```
List<int> numbers = new List<int>();

numbers.Add(10);
```

b) Dictionary<TKey, TValue>

```
Dictionary<int, string> students = new Dictionary<int, string>();
```

c) Stack<T> and Queue<T>

Work just like non-generic but are type-safe

d) HashSet<T>

Ensures only unique items

e) SortedList<TKey, TValue>

Automatically sorted by keys

Comparison: ArrayList vs List<T>

- ArrayList: Not type-safe, legacy
- List<T>: Type-safe, recommended

LINQ with Collections

```
var evens = nums.Where(n => n % 2 == 0);
```

Summary Table

- List<T>: For dynamic arrays
- Dictionary<K,V>: For key-value fast lookup
- Stack<T>, Queue<T>: For LIFO/FIFO
- HashSet<T>: For unique elements