### 1. List<T> (Dynamic Array)

| **Method** | **Description** | **Short Example** |
| --- | --- | --- |
| **void Add(T item)** | **Adds an item to the end of the list.** | **var list = new List<int>(); list.Add(10);** |
| **void AddRange(IEnumerable<T> collection)** | **Adds the elements of the specified collection to the list.** | **list.AddRange(new[] { 20, 30 });** |
| **void Insert(int index, T item)** | **Inserts an item at the specified index.** | **list.Insert(1, 15);** |
| **void InsertRange(int index, IEnumerable<T>)** | **Inserts a collection starting at the specified index.** | **list.InsertRange(1, new[] { 5, 6 });** |
| **bool Remove(T item)** | **Removes the first occurrence of the specified item.** | **list.Remove(10);** |
| **void RemoveAt(int index)** | **Removes the element at the specified index.** | **list.RemoveAt(0);** |
| **void RemoveRange(int index, int count)** | **Removes a range of elements.** | **list.RemoveRange(0, 2);** |
| **void Clear()** | **Removes all elements from the list.** | **list.Clear();** |
| **bool Contains(T item)** | **Checks if the list contains the specified item.** | **list.Contains(20);** |
| **int IndexOf(T item)** | **Returns the index of the first occurrence of the item.** | **list.IndexOf(20);** |
| **int LastIndexOf(T item)** | **Returns the index of the last occurrence of the item.** | **list.LastIndexOf(30);** |
| **void Sort()** | **Sorts the list in ascending order.** | **list.Sort();** |
| **void Reverse()** | **Reverses the order of the elements.** | **list.Reverse();** |
| **T[] ToArray()** | **Copies the list elements to a new array.** | **int[] array = list.ToArray();** |
| **void ForEach(Action<T> action)** | **Performs the specified action on each element.** | **list.ForEach(Console.WriteLine);** |

### 2. Dictionary<TKey, TValue> (Hash Map)

| **Method (Syntax with Return Type)** | **Description** | **Short Example** |
| --- | --- | --- |
| **void Add(TKey key, TValue value)** | **Adds the specified key-value pair.** | **var dict = new Dictionary<int, string>(); dict.Add(1, "A");** |
| **bool Remove(TKey key)** | **Removes the value with the specified key.** | **dict.Remove(1);** |
| **bool ContainsKey(TKey key)** | **Checks if the dictionary contains the specified key.** | **dict.ContainsKey(1);** |
| **bool ContainsValue(TValue value)** | **Checks if the dictionary contains the specified value.** | **dict.ContainsValue("A");** |
| **bool TryGetValue(TKey key, out TValue value)** | **Tries to get the value associated with the specified key.** | **dict.TryGetValue(1, out string value);** |
| **ICollection<TKey> Keys** | **Gets a collection of keys in the dictionary.** | **foreach (var key in dict.Keys) { Console.WriteLine(key); }** |
| **ICollection<TValue> Values** | **Gets a collection of values in the dictionary.** | **foreach (var val in dict.Values) { Console.WriteLine(val); }** |
| **void Clear()** | **Removes all keys and values.** | **dict.Clear();** |

### 3. HashSet<T> (Set Collection)

| **Method (Syntax with Return Type)** | **Description** | **Short Example** |
| --- | --- | --- |
| **bool Add(T item)** | **Adds an item if it does not already exist.** | **var set = new HashSet<int>(); set.Add(10);** |
| **bool Remove(T item)** | **Removes the specified item.** | **set.Remove(10);** |
| **bool Contains(T item)** | **Checks if the set contains the specified item.** | **set.Contains(10);** |
| **void Clear()** | **Removes all elements from the set.** | **set.Clear();** |
| **void UnionWith(IEnumerable<T> other)** | **Adds elements from the specified collection.** | **set.UnionWith(new[] { 20, 30 });** |
| **void IntersectWith(IEnumerable<T> other)** | **Retains only elements present in both sets.** | **set.IntersectWith(new[] { 20 });** |

### 4. Queue<T> (FIFO)

| **Method (Syntax with Return Type)** | **Description** | **Short Example** |
| --- | --- | --- |
| **void Enqueue(T item)** | **Adds an item to the end of the queue.** | **var queue = new Queue<int>(); queue.Enqueue(10);** |
| **T Dequeue()** | **Removes and returns the item at the front of the queue.** | **int front = queue.Dequeue();** |
| **T Peek()** | **Returns the front item without removing it.** | **int front = queue.Peek();** |
| **bool Contains(T item)** | **Checks if the queue contains the specified item.** | **queue.Contains(10);** |
| **void Clear()** | **Removes all elements from the queue.** | **queue.Clear();** |

### 5. Stack<T> (LIFO)

| **Method (Syntax with Return Type)** | **Description** | **Short Example** |
| --- | --- | --- |
| **void Push(T item)** | **Adds an item to the top of the stack.** | **var stack = new Stack<int>(); stack.Push(10);** |
| **T Pop()** | **Removes and returns the top item.** | **int top = stack.Pop();** |
| **T Peek()** | **Returns the top item without removing it.** | **int top = stack.Peek();** |
| **bool Contains(T item)** | **Checks if the stack contains the specified item.** | **stack.Contains(10);** |
| **void Clear()** | **Removes all elements from the stack.** | **stack.Clear();** |

### 6. SortedDictionary<TKey, TValue>

| **Method (Syntax with Return Type)** | **Description** | **Short Example** |
| --- | --- | --- |
| **void Add(TKey key, TValue value)** | **Adds a key-value pair in sorted order.** | **var sd = new SortedDictionary<int, string>(); sd.Add(1, "A");** |
| **bool Remove(TKey key)** | **Removes the key-value pair.** | **sd.Remove(1);** |
| **bool ContainsKey(TKey key)** | **Checks if a key exists.** | **sd.ContainsKey(1);** |

### 7. SortedSet<T>

| **Method (Syntax with Return Type)** | **Description** | **Short Example** |
| --- | --- | --- |
| **bool Add(T item)** | **Adds an item in sorted order.** | **var ss = new SortedSet<int>(); ss.Add(10);** |
| **bool Remove(T item)** | **Removes the specified item.** | **ss.Remove(10);** |
| **bool Contains(T item)** | **Checks if the set contains the specified item.** | **ss.Contains(10);** |