**Non- generic Collections**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Array List | Stack | Queue | Hash Table |
| Create | **ArrayList myAL = new ArrayList();**  Implements the [IList](https://docs.microsoft.com/en-us/dotnet/api/system.collections.ilist?view=net-6.0) interface using an array whose size is dynamically increased as required. | **Stack myStack = new Stack();**  Represents a simple last-in-first-out (LIFO) non-generic collection of objects. | **Queue myQ = new Queue();**  Represents a first-in, first-out collection of objects. | **Hashtable openWith = new Hashtable();**  Represents a collection of key/value pairs that are organized based on the hash code of the key. |
| Delete | **ArrayList.Remove(Object)**  Removes the first occurrence of a specific object from the ArrayList.  **ArrayList.RemoveAt(Int32)**  Removes the element at the specified index of the ArrayList.  **ArrayList.RemoveRange(Int32, Int32)**  Removes a range of elements from the ArrayList. | **Stack.Pop**  Removes and returns the object at the top of the Stack. | **Queue.Dequeue**  Removes and returns the object at the beginning of the Queue. | **Hashtable.Remove(Object)**  Removes the element with the specified key from the Hashtable. |
| Update | **ArrayList.Add(Object)**  Adds an object to the end of the ArrayList. | **Stack.Push(Object)**  Inserts an object at the top of the Stack. | **Queue.Enqueue(Object)**  Adds an object to the end of the Queue. | **Hashtable.Add(Object, Object)**  Adds an element with the specified key and value into the Hashtable. |
| Read | foreach (var item in myAL) { Console.WriteLine(item);} | **Stack.Peek**  Returns the object at the top of the Stack without removing it.  foreach (var item in myStack)  { Console.WriteLine(item);} | **Queue.Peek**  Returns the object at the beginning of the Queue without removing it.  foreach (var item in myQ)  { Console.WriteLine(item);} | **Hashtable.GetEnumerator**  Returns an IDictionaryEnumerator that iterates through the Hashtable.  **Hashtable.GetHash(Object)**  Returns the hash code for the specified key.  **Hashtable.GetObjectData(SerializationInfo, StreamingContext)**  Implements the ISerializable interface and returns the data needed to serialize the Hashtable. |
| ReadOnly | [**ReadOnly(ArrayList)**](https://docs.microsoft.com/en-us/dotnet/api/system.collections.arraylist.readonly?view=net-6.0#system-collections-arraylist-readonly(system-collections-arraylist))  Returns a list wrapper that is read-only. |  |  | **Hashtable.IsReadOnly**  Gets a value indicating whether the Hashtable is read-only. |

**Generic Collections**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Stack | Queue | Hash Set | Sorted List |
| Create | **Stack<string> numbers = new Stack<string>();**  Represents a variable size last-in-first-out (LIFO) collection of instances of the same specified type. | **Queue<string> numbers = new Queue<string>();**  Represents a first-in, first-out collection of objects. | **HashSet<int> Numbers = new HashSet<int>();**  Represents a set of values. | **SortedList<string, string> openWith =**  **new SortedList<string, string>();**  Represents a collection of key/value pairs that are sorted by key based on the associated [IComparer<T>](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.icomparer-1?view=net-6.0) implementation. |
| Delete | **Stack<T>.Pop**  Removes and returns the object at the top of the Stack<T>. | **Queue<T>.Dequeue**  Removes and returns the object at the beginning of the Queue<T>. | **HashSet<T>.Remove(T)**  Removes the specified element from a HashSet<T> object.  **HashSet<T>.RemoveWhere(Predicate<T>)**  Removes all elements that match the conditions defined by the specified predicate from a HashSet<T> collection. | **SortedList<TKey,TValue>.Remove(TKey)**  Removes the element with the specified key from the SortedList<TKey,TValue>.  **SortedList<TKey,TValue>.RemoveAt(Int32)**  Removes the element at the specified index of the SortedList<TKey,TValue>. |
| Update | **Stack<T>.Push(T)**  Inserts an object at the top of the Stack<T>. | **Queue<T>.Enqueue(T)**  Adds an object to the end of the Queue<T>. | **HashSet<T>.Add(T)**  Adds the specified element to a set. | **SortedList<TKey,TValue>.Add(TKey, TValue)**  Adds an element with the specified key and value into the SortedList<TKey,TValue>. |
| Read | **Stack<T>.Peek**  Returns the object at the top of the Stack<T> without removing it.  foreach (var item in numbers) { Console.WriteLine(item);  } | **Queue<T>.Peek**  Returns the object at the beginning of the Queue<T> without removing it.  foreach (var item in numbers) { Console.WriteLine(item);  } | foreach (var item in Numbers) { Console.WriteLine(item);  } | **SortedList<TKey,TValue>.GetEnumerator**  Returns an enumerator that iterates through the SortedList<TKey,TValue>.  **SortedList<TKey,TValue>.**  **IndexOfKey(TKey)**  Searches for the specified key and returns the zero-based index within the entire **SortedList<TKey,TValue>.**  **SortedList<TKey,TValue>.IndexOfValue(TValue)**  Searches for the specified value and returns the zero-based index of the first occurrence within the entire SortedList<TKey,TValue>.  foreach (KeyValuePair<string, string> item in openWith){ Console.WriteLine("Key = {0}, Value = {1}", item.Key, item.Value);  } |