|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ArrayList | Stack | Queue | Hashtable |
| Create | **ArrayList myAL = new ArrayList()** | **Stack myStack = new Stack();** | **Queue myQ = new Queue()** | **Hashtable hash = new Hashtable()** |
| Delete | myAL.Clear() | myStack.Clear() | myQ.Dequeue() | hash.Clear() |
| Update | myAL.Add(object), myAL.AddRange(ICollection) | myStack.Push() | myQ.Enqueue() | hash[“key”] = “value”, hash.Add() |
| Read | myAL.Clone(), myAL.CoptTo(), myAL.FixedSize(),  myAL.Reverse(), myAL.Sort(), myAL.ToArray(), myAL.ToString() | myStack.Pop(), myStack.ToString(), myStack.CopyTo() | myQ.ToString(),myQ.TrimToSize(),  myQ.Clone() | hash[“key”], hash.Clone(), hash.ToString(), hash.CopyTo(), hash.Remove() |
| ReadOnly | myAL.BinarySearch(), myAL.Conatins(), myAL.Equals(), myAL.IndexOf(), myAL.ReadOnly() | myStack.Peek(), myStakc.Equals(), myStack.Contains() | myQ.Peek(),  myQ.Equals() | hash.Equals(), hash.KeyEquals(), |
|  |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Stack | Queue | Hash Set | Sorted List |
| **Stack<string> numbers = new Stack<string>();** | **Queue<string> numbers = new Queue<string>();** | **HashSet<int> Numbers = new HashSet<int>();** | **SortedList<string, string> openWith =**  **New SortedList<string, string>();** |
| Stack<T>.Pop | Queue<T>.Dequeue | HashSet<T>.Remove(T)  HashSet<T>.RemoveWh  ere(Predicate<T>) | SortedList<TKey,TValue>.Rem  ove(TKey)  SortedList<TKey,TValue>.Remove  At(Int32) |
| Stack<T>.Push(T) | Queue<T>.Enqueue(T) | HashSet<T>.Add(T) | SortedList<TKey,TValue>.Add(TKey, TValue) |
| Stack<T>.Peek | Queue<T>.Peek | foreach (var item in Numbers) { Console.WriteLine  (item);  } | SortedList<TKey,TValue>.GetEnu  merator  SortedList<TKey,TValue>.  IndexOfKey(TKey)  SortedList<TKey,TValue>.  SortedList<TKey,TValue>.IndexOfVa  lue(TValue) |
|  |  |  |  |