List is one of the generic collection classes present in **System.Collections.Generic** namespcae. There are several generic collection classes in System.Collections.Generic namespace as listed below.  
**1.** Dictionary   
**2.** List  
**3.** Stack  
**4.** Queue etc  
  
  
  
A List class can be used to create a collection of any type. For example, we can create a list of Integers, Strings and even complex types. The objects stored in the list can be accessed by index. Unlike arrays, lists can grow in size automatically. This class also provides methods to search, sort, and manipulate lists.  
  
public class Program  
{  
    public static void Main()  
    {  
        // Create Customer Objects  
        Customer customer1 = new Customer()  
        {  
            ID = 101,  
            Name = "Mark",  
            Salary = 5000  
        };  
  
        Customer customer2 = new Customer()  
        {  
            ID = 102,  
            Name = "Pam",  
            Salary = 7000  
        };  
  
        Customer customer3 = new Customer()  
        {  
            ID = 104,  
            Name = "Rob",  
            Salary = 5500  
        };  
  
        Customer[] arrayCustomers = new Customer[2];  
        arrayCustomers[0] = customer1;  
        arrayCustomers[1] = customer2;  
        // The following line will throw an exception, Index was outside the bounds of the array.   
        // This is because, arrays does not grow in size automatically.  
        // arrayCustomers[2] = customer3;  
          
        // Create a List of Customers. Here, we have set the size to 2. But when I add a third   
        // element the list size will automatically grow and we will not get an exception.  
        List<Customer> listCustomers = new List<Customer>(2);  
        // To add an element to the list, use Add() method.  
        listCustomers.Add(customer1);  
        listCustomers.Add(customer2);  
        // Adding an element beyond the initial capacity of the list will not throw an exception.  
        listCustomers.Add(customer3);  
  
        // Items can be retrieved from the list by index. The following code will   
        // retrieve the first item from the list. List index is ZERO based.  
        Customer cust = listCustomers[0];  
        Console.WriteLine("ID = {0}, Name = {1}, Salary = {2}",  
                 cust.ID, cust.Name, cust.Salary);  
        Console.WriteLine("------------------------------------------------");  
  
        // foreach or for loop can be used to iterate thru all the items in the list  
        // Using for loop  
        for (int i = 0; i < listCustomers.Count; i++)  
        {  
            Customer customer = listCustomers[i];  
            Console.WriteLine("ID = {0}, Name = {1}, Salary = {2}",  
                     customer.ID, customer.Name, customer.Salary);  
        }  
        Console.WriteLine("------------------------------------------------");  
  
        // Using foreach loop  
        foreach (Customer c in listCustomers)  
        {  
            Console.WriteLine("ID = {0}, Name = {1}, Salary = {2}", c.ID, c.Name, c.Salary);  
        }  
        Console.WriteLine("------------------------------------------------");  
  
        // All generic collection classes including List are strongly typed. This means   
        // if you have created a List of type Customer, only objects of type Customer   
        // can be added to the list. If you try to add an object of different type you would   
        // get a compiler error. The following line will raise a compiler error.  
        // listCustomers.Add("This will not compile");  
  
        // If you want to insert an item at a specific index location of the list, use Insert() method.   
        // The following line will insert customer3 object at index location 1.  
        listCustomers.Insert(1, customer3);  
        Console.WriteLine("ID = {0}, Name = {1}, Salary = {2}",   
               listCustomers[1].ID, listCustomers[1].Name, listCustomers[1].Salary);  
        Console.WriteLine("------------------------------------------------");  
  
        // To get the index of specific item in the list use Indexof() method  
        Console.WriteLine("Index of Customer3 object in the List = " +        
                listCustomers.IndexOf(customer3));  
        Console.WriteLine("------------------------------------------------");  
    }  
}  
  
public class Customer  
{  
    public int ID { get; set; }  
    public string Name { get; set; }  
    public int Salary { get; set; }  
}

we will discuss  
**1. Contains() function -** Use this function to check if an item exists in the list. This method returns true if the items exists, else false.  
  
**2. Exists() function -**Use this function, to check if an item exists in the list based on a condition. This method returns true if the items exists, else false.  
  
**3. Find() function -**This method searches for an element that matches the conditions defined by the specified lambda expression and returns the first matching item from the list.  
  
**4. FindLast() function -**This method searches for an element that matches the conditions defined by the specified lambda expression and returns the Last matching item from the list.  
  
**5. FindAll() function -**This method returns all the items from the list that match the conditions specified by the lambda expression.  
  
  
  
**6. FindIndex() function -**This method returns the index of the first item, that matches the condition specified by the lambda expression. There are 2 other overloads of this method which allows us to specify the range of elements to search, with in the list.  
  
**7. FindLastIndex() function -**This method returns the index of the last item, that matches the condition specified by the lambda expression. There are 2 other overloads of this method which allows us to specify the range of elements to search, with in the list.  
  
**8. Convert an array to a List -**Use ToList() method  
  
**9. Convert a list to an array -**Use ToArray() method  
  
**10. Convert a List to a Dictionary -**Use ToDictionary() method  
  
public class Program  
{  
    public static void Main()  
    {  
        // Create Customer Objects  
        Customer customer1 = new Customer()  
        {  
            ID = 101,  
            Name = "Mark",  
            Salary = 4000  
        };  
  
        Customer customer2 = new Customer()  
        {  
            ID = 102,  
            Name = "Pam",  
            Salary = 7000  
        };  
  
        Customer customer3 = new Customer()  
        {  
            ID = 104,  
            Name = "Rob",  
            Salary = 5500  
        };  
  
        Customer[] arrayCustomers = new Customer[3];  
        arrayCustomers[0] = customer1;  
        arrayCustomers[1] = customer2;  
        arrayCustomers[2] = customer3;  
  
        // To convert an array to a List, use ToList() method  
        List<Customer> listCustomers = arrayCustomers.ToList();  
        foreach (Customer c in listCustomers)  
        {  
            Console.WriteLine("ID = {0}, Name = {1}, Salary = {2}", c.ID, c.Name, c.Salary);  
        }  
        Console.WriteLine("------------------------------------------------------");  
  
        // To convert a List to an array, use ToLArray() method  
        Customer[] arrayAllCustomers = listCustomers.ToArray();  
        foreach (Customer c in arrayAllCustomers)  
        {  
            Console.WriteLine("ID = {0}, Name = {1}, Salary = {2}", c.ID, c.Name, c.Salary);  
        }  
        Console.WriteLine("------------------------------------------------------");  
  
        // To convert a List to a Dictionary use ToDictionary() method  
        Dictionary<int, Customer> dictionaryCustomers = listCustomers.ToDictionary(x => x.ID);  
        foreach (KeyValuePair<int, Customer> keyValuePairCustomers in dictionaryCustomers)  
        {  
            Console.WriteLine("Key = {0}", keyValuePairCustomers.Key);  
            Customer c = keyValuePairCustomers.Value;  
            Console.WriteLine("ID = {0}, Name = {1}, Salary = {2}", c.ID, c.Name, c.Salary);  
        }  
        Console.WriteLine("------------------------------------------------------");  
  
        // To check if an item exists in the list use Contains() function  
        // This method returns true if the items exists, else false  
        if (listCustomers.Contains(customer2))  
        {  
            Console.WriteLine("Customer2 object exists in the list");  
        }  
        else  
        {  
            Console.WriteLine("Customer2 object does not exist in the list");  
        }  
        Console.WriteLine("------------------------------------------------------");  
  
        // To check if an item exists in the list based on a condition, then use Exists() function  
        // This method returns true if the items exists, else false  
        if (listCustomers.Exists(x => x.Name.StartsWith("M")))  
        {  
            Console.WriteLine("List contains customer whose name starts with M");  
        }  
        else  
        {  
            Console.WriteLine("List does not contain a customer whose name starts with M");  
        }  
        Console.WriteLine("------------------------------------------------------");  
  
        // Find() method searches for an element that matches the conditions defined by   
        // the specified lambda expression and returns the first matching item from the list  
        Customer cust = listCustomers.Find(customer => customer.Salary > 5000);  
        Console.WriteLine("ID = {0}, Name = {1}, Salary = {2}", cust.ID, cust.Name, cust.Salary);  
        Console.WriteLine("------------------------------------------------------");  
  
        // FindLast() method searches for an element that matches the conditions defined  
        // by the specified lambda expression and returns the Last matching item from the list  
        Customer lastMatch = listCustomers.FindLast(customer => customer.Salary > 5000);  
        Console WriteLine("ID = {0}, Name = {1}, Salary = {2}", lastMatch.ID, lastMatch.Name, lastMatch.Salary);  
        Console.WriteLine("------------------------------------------------------");  
  
        // FindAll() method returns all the items from the list that  
        // match the conditions specified by the lambda expression  
        List<Customer> filteredCustomers = listCustomers.FindAll(customer => customer.Salary > 5000);  
        foreach (Customer cstmr in filteredCustomers)  
        {  
            Console.WriteLine("ID = {0}, Name = {1}, Salary = {2}", cstmr.ID, cstmr.Name, cstmr.Salary);  
        }  
        Console.WriteLine("------------------------------------------------------");  
  
        // FindIndex() method returns the index of the first item, that matches the   
        // condition specified by the lambda expression. There are 2 other overloads  
        // of this method which allows us to specify the range of elements to   
        // search, with in the list.  
        Console.WriteLine("Index of the first matching customer object whose salary is greater 5000 =" +  
            listCustomers.FindIndex(customer => customer.Salary > 5000));  
        Console.WriteLine("------------------------------------------------------");  
  
        // FindLastIndex() method returns the index of the last item,   
        // that matches the condition specified by the lambda expression.   
        // There are 2 other overloads of this method which allows us to specify   
        // the range of elements to search, with in the list.  
        Console.WriteLine("Index of the Last matching customer object whose salary is greater 5000 = " +  
            listCustomers.FindLastIndex(customer => customer.Salary > 5000));  
        Console.WriteLine("------------------------------------------------------");  
    }  
}  
  
public class Customer  
{  
    public int ID { get; set; }  
    public string Name { get; set; }  
    public int Salary { get; set; }  
}

**1. AddRange() -**Add() method allows you to add one item at a time to the end of the list, where as AddRange() allows you to add another list of items, to the end of the list.  
  
**2. GetRange() -**Using an item index, we can retrieve only one item at a time from the list, if you want to get a list of items from the list, then use GetRange() function. This function expects 2 parameters, i.e the start index in the list and the number of elements to return.  
  
**3. InsertRange() -**Insert() method allows you to insert a single item into the list at a specificed index, where as InsertRange() allows you, to insert another list of items to your list at the specified index.  
  
**4. RemoveRange() - Remove**() function removes only the first matching item from the list. **RemoveAt**() function, removes the item at the specified index in the list. **RemoveAll**() function removes all the items that matches the specified condition. **RemoveRange**() method removes a range of elements from the list. This function expects 2 parameters, i.e the start index in the list and the number of elements to remove. If you want to remove all the elements from the list without specifying any condition, then use **Clear**() function.  
  
  
  
public class Program  
{  
    public static void Main()  
    {  
        // Create Customer Objects  
        Customer customer1 = new Customer()  
        {  
            ID = 101,  
            Name = "Mark",  
            Salary = 4000,  
            Type = "RetailCustomer"  
        };  
  
        Customer customer2 = new Customer()  
        {  
            ID = 102,  
            Name = "Pam",  
            Salary = 7000,  
            Type = "RetailCustomer"  
        };  
  
        Customer customer3 = new Customer()  
        {  
            ID = 103,  
            Name = "Rob",  
            Salary = 5500,  
            Type = "RetailCustomer"  
        };  
  
        Customer customer4 = new Customer()  
        {  
            ID = 104,  
            Name = "John",  
            Salary = 6500,  
            Type = "CorporateCustomer"  
        };  
  
        Customer customer5 = new Customer()  
        {  
            ID = 105,  
            Name = "Sam",  
            Salary = 3500,  
            Type = "CorporateCustomer"  
        };  
  
          
        List<Customer> listCustomers = new List<Customer>();  
        // Add() method allows you to add one at a time to the end of the list  
        listCustomers.Add(customer1);  
        listCustomers.Add(customer2);  
        listCustomers.Add(customer3);  
  
        List<Customer> listCorporateCustomers = new List<Customer>();  
        listCorporateCustomers.Add(customer4);  
        listCorporateCustomers.Add(customer5);  
  
        // AddRange() allows you to add another list of items, to the end of the list  
        listCustomers.AddRange(listCorporateCustomers);  
  
        foreach (Customer customer in listCustomers)  
        {  
            Console.WriteLine("ID = {0}, Name = {1}, Salary = {2}, Type = {3}",  
                customer.ID, customer.Name, customer.Salary, customer.Type);  
        }  
        Console.WriteLine("------------------------------------------------------");  
  
        // GetRange() function returns a list of items from the list.  
        List<Customer> corporateCustomers = listCustomers.GetRange(3, 2);  
        foreach (Customer customer in corporateCustomers)  
        {  
            Console.WriteLine("ID = {0}, Name = {1}, Salary = {2}, Type = {3}",  
                customer.ID, customer.Name, customer.Salary, customer.Type);  
        }  
        Console.WriteLine("------------------------------------------------------");  
  
        // Remove() function removes only the first matching item from the list.  
        listCustomers.Remove(customer1);  
  
        // RemoveAt() function, removes the item at the specified index in the list.  
        listCustomers.RemoveAt(0);  
  
        // RemoveAll() function removes all the items that matches the specified condition.  
        listCustomers.RemoveAll(x => x.Type == "RetailCustomer");  
  
        foreach (Customer customer in listCustomers)  
        {  
            Console.WriteLine("ID = {0}, Name = {1}, Salary = {2}, Type = {3}",  
                customer.ID, customer.Name, customer.Salary, customer.Type);  
        }  
        Console.WriteLine("------------------------------------------------------");  
  
        // RemoveRange() method removes a range of elements from the list.   
        // This function expects 2 parameters, i.e the start index in the   
        // list and the number of elements to remove.  
        listCustomers.RemoveRange(0, 2);  
  
        // Insert() method allows you to insert a single item at a time into   
        // the list at a specificed index  
        listCustomers.Insert(0, customer1);  
        listCustomers.Insert(1, customer2);  
        listCustomers.Insert(2, customer3);  
  
        // InsertRange() allows you, to insert another list of items to your list at the specified index  
        listCustomers.InsertRange(0, listCorporateCustomers);  
  
        foreach (Customer customer in listCustomers)  
        {  
            Console.WriteLine("ID = {0}, Name = {1}, Salary = {2}, Type = {3}",  
                customer.ID, customer.Name, customer.Salary, customer.Type);  
        }  
        Console.WriteLine("------------------------------------------------------");  
  
        // If you want to remove all the elements from the list without specifying   
        // any condition, then use Clear() function.  
        listCustomers.Clear();  
  
        Console.WriteLine(" Total Items in the List = " + listCustomers.Count);  
    }  
}  
  
public class Customer  
{  
    public int ID { get; set; }  
    public string Name { get; set; }  
    public int Salary { get; set; }  
    public string Type { get; set; }  
}