**Part 77 - Sort a list of simple types in c#**

Sorting a list of simple types like int, string, char etc, is straight forward. Just invoke **Sort**() method on the list instance and the data will be automatically sorted in ascending order.  
List<int> numbers = new List<int> { 1, 8, 7, 5, 2, 3, 4, 9, 6 };  
numbers.Sort();

If you want the data to be retrieved in descending order, use **Reverse**() method on the list instance.  
numbers.Reverse();  
  
However, when you do the same thing on a complex type like **Customer**, we get a runtime invalid operation exception - Failed to compare 2 elements in the array. This because, .NET runtime does not know, how to sort complex types. We have to tell the way we want data to be sorted in the list by implementing **IComparable**interface. We will discuss this in a later video session.

So, the next obvious question - **How is the sort functionality working for simple types like int, string, char etc?**  
That is because these types (int, string, decimal, char etc) have implemented IComparable interface already.  
  
**Here is the example code used in demo:**  
public class Program  
{  
    public static void Main()  
    {  
        List<int> numbers = new List<int> { 1, 8, 7, 5, 2, 3, 4, 9, 6 };  
  
        Console.WriteLine("Numbers before sorting");  
        foreach (int i in numbers)  
        {  
            Console.WriteLine(i);  
        }  
  
        // Sort() will sort data in ascending order   
        numbers.Sort();  
  
        Console.WriteLine("Numbers after sorting");  
        foreach (int i in numbers)  
        {  
            Console.WriteLine(i);  
        }  
  
        // Use Reverse() method to retrieve data in descending order  
        numbers.Reverse();  
  
        Console.WriteLine("Numbers in descending order");  
        foreach (int i in numbers)  
        {  
            Console.WriteLine(i);  
        }  
  
        List<string> alphabets = new List<string>() { "B", "F", "D", "E", "A", "C" };  
  
        Console.WriteLine("Alphabets before sorting");  
        foreach (string alphabet in alphabets)  
        {  
            Console.WriteLine(alphabet);  
        }  
  
        alphabets.Sort();  
  
        Console.WriteLine("Alphabets after sorting");  
        foreach (string alphabet in alphabets)  
        {  
            Console.WriteLine(alphabet);  
        }  
  
        alphabets.Reverse();  
  
        Console.WriteLine("Alpabets in descending order");  
        foreach (string alphabet in alphabets)  
        {  
            Console.WriteLine(alphabet);  
        }  
  
        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 = 103,  
            Name = "Rob",  
            Salary = 5500  
        };  
  
        List<Customer> listCustomers = new List<Customer>();  
        listCustomers.Add(customer1);  
        listCustomers.Add(customer2);  
        listCustomers.Add(customer3);  
  
        Console.WriteLine("Customers before sorting");  
        foreach (Customer customer in listCustomers)  
        {  
            Console.WriteLine(customer.Name);  
        }  
  
        // Invoking Sort() on list of complex types will   
        // throw invalid operation exception, unless   
        // IComparable interface is implemented  
        listCustomers.Sort();  
              
        Console.WriteLine("Customers after sorting");  
        foreach (Customer customer in listCustomers)  
        {  
            Console.WriteLine(customer.Name);  
        }  
    }  
}  
  
public class Customer   
{  
    public int ID { get; set; }  
    public string Name { get; set; }  
    public int Salary { get; set; }  
  
}