**Lesson02 Clone Object in c#**

**Notes:-**

**1-Microsoft .Net provides support for cloning objects.**

**2-cloning objects: - an ability to create an exact copy of an object (also known as a clone).**

**3-Cloning can be of two types: shallow copy and deep copy**

**Shallow copy: - make copy of object without references to same objects**

**Deep copy: - make copy of object with reference of other objects (instance)**

**Ways of implement cloning?**

**1-Using the System.Object.MemberwiseClone method to perform a shallow copy**

**2-Using Reflection by taking advantage of the Activator.CreateInstance method**

**3-Using Serialization**

**4-By implementing the IClonable interface**

**Method 01:System.Object.MemberwiseClone**

**Example:-**

**Consider the below example**

**Employee emp = new Employee();**

**Employee clone = emp;**

**//we see that the assignment operator "=" would assign the reference varaibale on the same instance**

**//The MemberwiseClone() method defined in the System.Object class does exactly the same thing.**

**(We are here do the shallow copy of the object)**

**(We refer the separate references variable to be assign to the same instances)**

**Deep Copy allow you to create new separate instance on each separate reference variable that assign to that separate instance.**

**Shallow Copy allow you to create new separate reference variable that assign to the same instance that sharing the same properties on the separate reference variables.**

**Demo sample:-**

**We have 2 reference variable assign to the same instance**

**Customer m1 = new Customer();**

**m1.Id = 1;**

**m1.Name = "Mohammed";**

**Console.WriteLine(m1.Id + " -- " + m1.Name);**

**//we sees that the new reference variable assign to the same instance**

**Customer m2 = m1;**

**m2.Name = "Isam";**

**Console.WriteLine(m2.Id + " -- " + m2.Name);**

**Deep Copy Example:-**

**1-You can implement deep copy in many ways. One of the most preferred ways to implement a deep copy of an object is by using serialization.**

**2-You can also leverage reflection to perform a deep copy of an instance of a class.**

**public static T DeepCopy<T>(T obj){**

**if (!typeof(T).IsSerializable){**

**throw new Exception("The source object must be serializable");}**

**//the below code will make checi if the specified reference variable (object) is the same instance**

**if (Object.ReferenceEquals(obj, null)){**

**throw new Exception("The source object must not be null");}**

**T result = default(T);**

**using (var memoryStream = new MemoryStream()){**

**var formatter = new BinaryFormatter();**

**formatter.Serialize(memoryStream, obj);**

**memoryStream.Seek(0, SeekOrigin.Begin);**

**result = (T)formatter.Deserialize(memoryStream);**

**memoryStream.Close();}**

**return result;}**

**static void Main(string[] args){**

**Customer m1 = new Customer();**

**//m1.Id = 1;**

**//m1.Name = "Mohammed";**

**Customer clone = DeepCopy<Customer>(m1);**

**if (Object.ReferenceEquals(m1, clone)){**

**Console.WriteLine("References are the same.");}**

**else{Console.WriteLine("References are different.");}}**

**//we see that the result will be References are different**