https://alekseygulynin.ru/iserializable-c/  
  
что нужно сделать для того, чтобы сериализация работала корректно:

1) У класса должен быть дефолтный конструктор без параметров.

2) Этот класс должен быть помечен атрибутом "Serializable".

3) Если мы хотим сами решать, что сериализовать, а что нет — реализовываем интерфейс "ISerializable".

В этом интерфейсе определен метод "GetObjectData()". Это метод, который вызывается при сериализации. Внутри этого метода и определяется какие поля и куда добавлять. Для десериализации нужно реализовать ещё недефолтный конструктор с параметрами, которые совпадают с параметрами метода "GetObjectData()", т.е. этот метод будет вызываться, когда данные будут восстанавливаться.

На определенное поле можно повесить атрибут "NonSerialized". Это означает, что данное поле не будет участвовать в сериализации. По умолчанию все поля сериализуются.

Давайте закрепим это всё на примере кастомной сериализации класса:

// Не забываем подключать пространство имён

// using System.Runtime.Serialization;

[Serializable]

public class Room : ISerializable

{

public double Width { get; set; }

public double Length { get; set; }

[NonSerialized]

public int numberOfWindows;

public bool hasFurniture { get; set; }

// Конструктор по умолчанию

public Room()

{

}

// Метод, вызываемый при сериализации

public void GetObjectData(SerializationInfo info, StreamingContext context)

{

info.AddValue("Width", this.Width);

info.AddValue("Length", this.Length);

info.AddValue("hasFurniture", this.hasFurniture);

}

// Конструктор, вызываемый при десериализации

public Room(SerializationInfo info, StreamingContext context)

{

this.Width = (double)info.GetValue("Width", typeof(double));

this.Length = (double)info.GetValue("Length", typeof(double));

this.hasFurniture = (bool)info.GetValue("hasFurniture", typeof(bool));

}

}

Посмотрим на реализацию метода "GetObjectData()". В этом методе что определить что именно и куда мы кладём при сериализации. В простейшем случае нужен только первый параметр "SerializationInfo". Он представляет из себя нечто похожее на коллекцию "Dictionary". Сюда с помощью метода "AddValue()" добавляется ключ и значение. А при десериализации это значение будет вытаскиваться по ключу.

"StreamingContext" — это объект, который содержит дополнительную информацию о потоке, в который записываются эти данные.С помощью метода GetValue() мы по ключу получаем значение. Вторым параметром здесь указывается тип данных.

Напоминаю, что в простейшем случае этот интерфейс реализовывать не нужно.

В данной статье вы узнали, как реализовать интерфейс **ISerializable в C#**.