序列化和反序列化

命名空间：using System.Runtime.Serialization.Formatters.Binary;

序列化： 将对象转换成二进制
反序列化： 将二进制数据转换成对象
作用： 不同电脑间传输数据

步骤：

```
namespace 序列化和反序列化{
    //序列化步骤:
    //1.将类标记为可以被序列化
    //  在类的上一行用中括号+serializable标记(只有被serializable标记的类创建出的对象才能被序列化)

    //Indicates that a class can be serialized.  This class cannot be inherited.
    //被标记的类可以被序列化，但不能被继承
    [Serializable]
    public class Person{
        private string _name;
        private char _gender;
        private int _age;
        public Person(string name,char gender,int age){
            this.Name = name;
            this.Gender = gender;
            this.Age = age;
        }
        public Person(){}

        public string Name { get; set; }
        public char Gender { get; set; }
        public int Age { get; set; }
    }

    class MainFunction{
        static void Main(string[] args){
            //2.使用BinaryFormatter创建序列化对象bf
            //3.使用bf的Serialize()方法进行序列化和写入
            Person p = new Person("张三",'男',23);

            using(FileStream fsWrite = new FileStream(@"C:\Users\Admin\Desktop\duixiang.txt",FileMode.OpenOrCreate,FileAccess.Write)){
                //开始序列化对象
                BinaryFormatter bf = new BinaryFormatter();
                bf.Serialize(fsWrite,p);    //包含序列化及写入
            }
            System.Console.WriteLine("序列化成功");
```

```
            //4.接收对方发送过来的二进制数据，反序列化成对象
            //  同2.创建序列化对象bf，再使用bf的Deserialize()方法进行反序列化和读取
            //5.使用对应对象来接收反序列化后返回的object数据(注：接收时要进行强转，因此必须用相同类来接收)
            using(FileStream fsRead = new FileStream(@"C:\Users\Admin\Desktop\duixiang.txt",FileMode.OpenOrCreate,FileAccess.Read)){
                BinaryFormatter bf = new BinaryFormatter();
                p =(Person)bf.Deserialize(fsRead);//反序列化，并转换成Person对象类接收
            }
            System.Console.WriteLine(p.Name);
            System.Console.WriteLine(p.Age);
            System.Console.WriteLine(p.Gender);
            //结果为：
            //张三
            //23
            //男

        }
    }

}
```