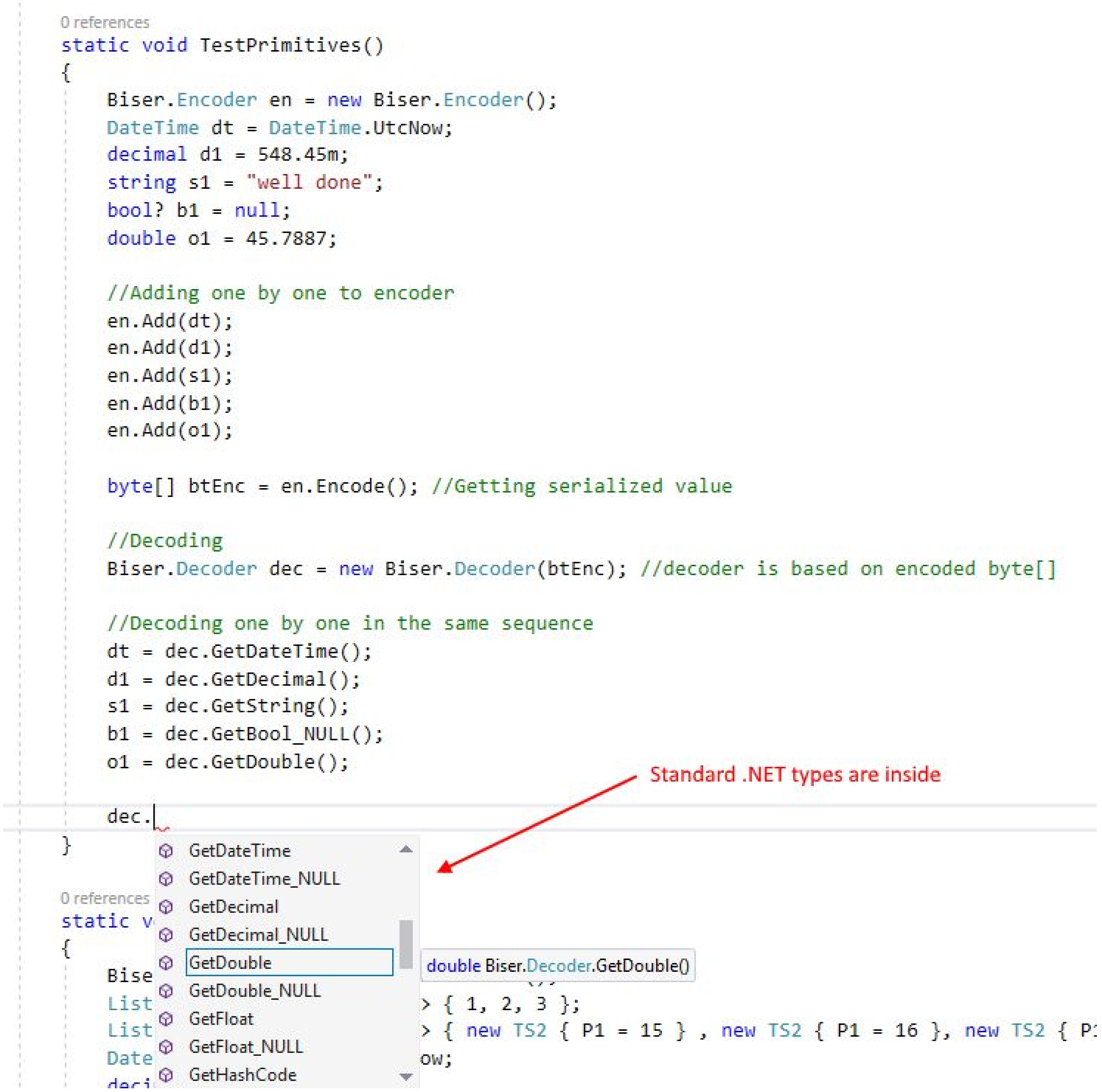
Biser .NET (binary serializer for .NET dotnet)

From simple to complex. Encoding .NET Primitives



```
0 references
static void TestPrimitives()
{
    Biser.Encoder en = new Biser.Encoder();
    DateTime dt = DateTime.UtcNow;
    decimal d1 = 548.45m;
    string s1 = "well done";
    bool? b1 = null;
    double o1 = 45.7887;

    //Adding one by one to encoder
    en.Add(dt);
    en.Add(d1);
    en.Add(s1);
    en.Add(b1);
    en.Add(o1);

    byte[] btEnc = en.Encode(); //Getting serialized value

    //Decoding
    Biser.Decoder dec = new Biser.Decoder(btEnc); //decoder is based on encoded byte[]

    //Decoding one by one in the same sequence
    dt = dec.GetDateTime();
    d1 = dec.GetDecimal();
    s1 = dec.GetString();
    b1 = dec.GetBool_NULL();
    o1 = dec.GetDouble();

    dec.
}

0 references
static v
{
    Bise
    List                           > { 1, 2, 3 };
    List                           > { new TS2 { P1 = 15 } , new TS2 { P1 = 16 }, new TS2 { P:
    Date                           ow;
    deci
```

Standard .NET types are inside

```
static void TestListDictionary()
        {
            //Encoding
```

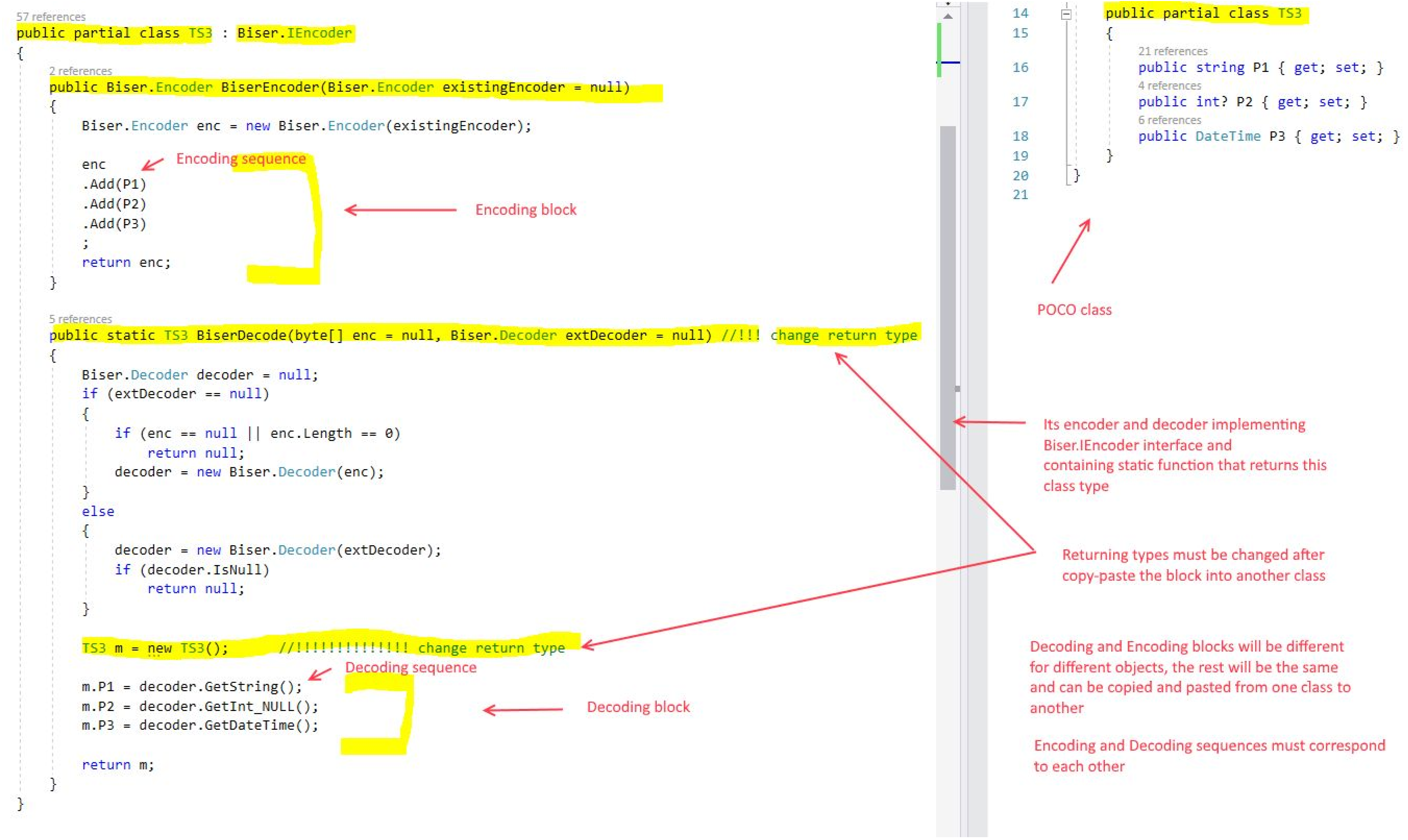
```
 Biser.Encoder enc = new Biser.Encoder();
 enc.Add((int)123);
 enc.Add(new List<string> { "Hi", "there" }, r => { enc.Add(r); });
 enc.Add((float)-458.4f);

 enc.Add(new Dictionary<uint, string> { { 1, "Well" }, { 2, "done" } }
, r => { enc.Add(r.Key); enc.Add(r.Value); });

 enc.Add((decimal)-587.7m);

 //TS4 implements IEncoder
 enc.Add(new Dictionary<uint, TS4> { { 1, new TS4 { TermId = 1 } }, { 2, new TS4 { TermId = 5 } } }
 , r => { enc.Add(r.Key); enc.Add(r.Value); });

 enc.Add(new TS4 { TermId = 188 });


 //Decoding

 var decoder = new Biser.Decoder(enc.Encode());

 Console.WriteLine(decoder.GetInt());


 //////Alternative to the following instruction. Slower than supplying List directly
 ////foreach (var item in decoder.GetCollection().Select(r => r.GetString()))
 ////{
 ////    Console.WriteLine(item);
 ////}

 List<string> lst = decoder.CheckNull() ? null : new List<string>();
 if (lst != null)
 {
     decoder.GetCollection(() => { return decoder.GetString(); }, lst, true);
     foreach (var item in lst)
         Console.WriteLine(item);
 }

 Console.WriteLine(decoder.GetFloat());


 ///////Alternative to the following instruction. Slower than supplying Dictionary directly
```

```
    //////foreach (var item in decoder.GetCollection())
    //////{
    //////    Console.WriteLine($"K: {item.GetUInt()}; V: {item.GetString()}");
    //////}

    Dictionary<uint, string> d1 = decoder.CheckNull() ? null : new Dictionary<uint, string>();
    if (d1 != null)
    {
        decoder.GetCollection(
            () => { return decoder.GetUInt(); },
            () => { return decoder.GetString(); },
            d1, true);
        foreach (var item in d1)
            Console.WriteLine(item.Key + "; " + item.Value);
    }

    Console.WriteLine(decoder.GetDecimal());

    Dictionary<uint, TS4> d2 = decoder.CheckNull() ? null : new Dictionary<uint, TS4>();
    if (d2 != null)
    {
        decoder.GetCollection(
            () => { return decoder.GetUInt(); },
            () => { return TS4.BiserDecode(extDecoder: decoder); },
            d2, true);
        foreach (var item in d2)
            Console.WriteLine(item.Key + "; " + item.Value.TermId);
    }

    Console.WriteLine(TS4.BiserDecode(extDecoder: decoder).TermId);


}
```

Encoding custom objects

In Biser serializing and deserializing functions (encoding/decoding) are supplied together with POCO class (A Plain Old CLR Objects) as a partial class extension.
These functions are very simple and are built up with the help of Biser primitives.

Use copy-paste from here (https://github.com/hhblaze/Biser/tree/master/BiserTest_Net) to create the most popular encoders/decoders



```
57 references
public partial class TS3 : Biser.IEncoder
{
    2 references
    public Biser.Encoder BiserEncoder(Biser.Encoder existingEncoder = null)
    {
        Biser.Encoder enc = new Biser.Encoder(existingEncoder);

        enc
        .Add(P1)
        .Add(P2)
        .Add(P3)
        ;
        return enc;
    }

    5 references
    public static TS3 BiserDecode(byte[] enc = null, Biser.Decoder extDecoder = null) //!!! change return type
    {
        Biser.Decoder decoder = null;
        if (extDecoder == null)
        {
            if (enc == null || enc.Length == 0)
                return null;
            decoder = new Biser.Decoder(enc);
        }
        else
        {
            decoder = new Biser.Decoder(extDecoder);
            if (decoder.IsNull)
                return null;
        }

        TS3 m = new TS3();        //!!!!!!!!!!!!!! change return type

        m.P1 = decoder.GetString();
        m.P2 = decoder.GetInt_NULL();
        m.P3 = decoder.GetDateTime();

        return m;
    }
}
```

```
public partial class TS3
    {
        21 references
        public string P1 { get; set; }
        4 references
        public int? P2 { get; set; }
        6 references
        public DateTime P3 { get; set; }
    }
}
```

Encoding sequence

Encoding block

POCO class

Its encoder and decoder implementing Biser.IEncoder interface and containing static function that returns this class type

Returning types must be changed after copy-paste the block into another class

Decoding sequence

Decoding block

Decoding and Encoding blocks will be different for different objects, the rest will be the same and can be copied and pasted from one class to another

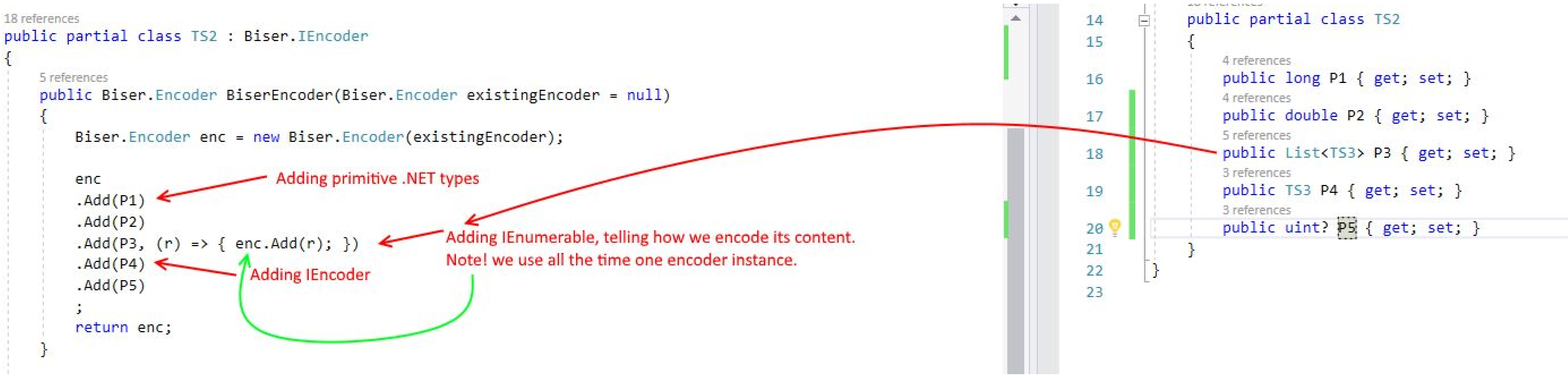
Encoding and Decoding sequences must correspond to each other

There just 3 global types to be encoded: .NET standard types, Biser.IEncoder and IEnumerable.

Encoding

It’s possible to encode primitive .NET type, IEncoder (custom object) and IEnumerable (with the content of any complexity):

.NET Primitives and IEncoder can be just added into encoder.Add function.
For IEnumerable we need to define how to encode its content:



```
18 references
public partial class TS2 : Biser.IEncoder
{
    5 references
    public Biser.Encoder BiserEncoder(Biser.Encoder existingEncoder = null)
    {
        Biser.Encoder enc = new Biser.Encoder(existingEncoder);

        enc
        .Add(P1)
        .Add(P2)
        .Add(P3, (r) => { enc.Add(r); })
        .Add(P4)
        .Add(P5)
        ;
        return enc;
    }
```

Adding primitive .NET types

Adding IEnumerable, telling how we encode its content.
Note! we use all the time one encoder instance.

Adding IEncoder

```
public partial class TS2
{
    4 references
    public long P1 { get; set; }
    4 references
    public double P2 { get; set; }
    5 references
    public List<TS3> P3 { get; set; }
    3 references
    public TS3 P4 { get; set; }
    3 references
    public uint? P5 { get; set; }
}
}
```

This trick gives us ability to encode any data sequence inside of IEnumerable, e.g:

To encode such object

public List<Tuple<string,byte[],TS3>> P8 { get; set; }

We need to make following

.Add(P8, (r) => { enc.Add(r.Item1); enc.Add(r.Item2); enc.Add(r.Item3); })

To encode Dictionary

```
public Dictionary<long,TS3> P5 { get; set; }
```

Adding

```
.Add(P5, (r) => { enc.Add(r.Key); enc.Add(r.Value); })
```

To Encode Dictionary with List as a Value

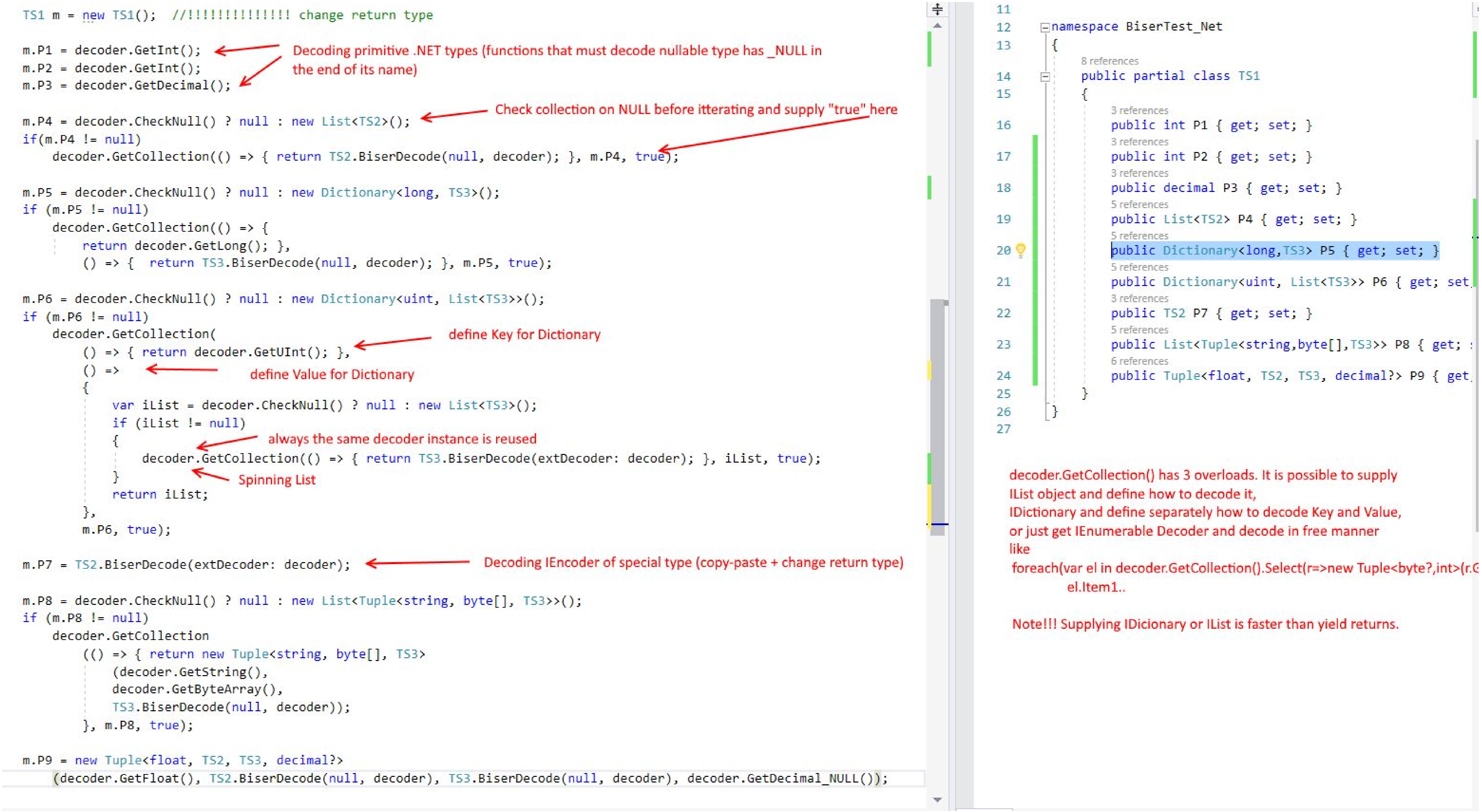
```
public Dictionary<long,List<TS3>> P6 { get; set; }
```

we need to make following

```
.Add(P6, (r) => { enc.Add(r.Key); enc.Add(r.Value, (r1) => { enc.Add(r1); }); })
```

Decoding

Primitive types are decoded in such way (example of TS1 decoder from https://github.com/hhblaze/Biser/blob/master/BiserTest_Net/TS1_Biser.cs):



```
TS1 m = new TS1();  //!!!!!!!!!!!!!! change return type

m.P1 = decoder.GetInt();
m.P2 = decoder.GetInt();
m.P3 = decoder.GetDecimal();

m.P4 = decoder.CheckNull() ? null : new List<TS2>();
if(m.P4 != null)
    decoder.GetCollection(() => { return TS2.BiserDecode(null, decoder); }, m.P4, true);

m.P5 = decoder.CheckNull() ? null : new Dictionary<long, TS3>();
if (m.P5 != null)
    decoder.GetCollection(() => {
        return decoder.GetLong(); },
        () => {  return TS3.BiserDecode(null, decoder); }, m.P5, true);

m.P6 = decoder.CheckNull() ? null : new Dictionary<uint, List<TS3>>();
if (m.P6 != null)
    decoder.GetCollection(
        () => { return decoder.GetUInt(); },
        () =>
        {
            var iList = decoder.CheckNull() ? null : new List<TS3>();
            if (iList != null)
            {
                decoder.GetCollection(() => { return TS3.BiserDecode(extDecoder: decoder); }, iList, true);
            }
            return iList;
        },
        m.P6, true);

m.P7 = TS2.BiserDecode(extDecoder: decoder);

m.P8 = decoder.CheckNull() ? null : new List<Tuple<string, byte[], TS3>>();
if (m.P8 != null)
    decoder.GetCollection
        (() => { return new Tuple<string, byte[], TS3>
            (decoder.GetString(),
            decoder.GetByteArray(),
            TS3.BiserDecode(null, decoder));
        }, m.P8, true);

m.P9 = new Tuple<float, TS2, TS3, decimal?>
    (decoder.GetFloat(), TS2.BiserDecode(null, decoder), TS3.BiserDecode(null, decoder), decoder.GetDecimal_NULL());
```

Decoding primitive .NET types (functions that must decode nullable type has _NULL in the end of its name)

Check collection on NULL before itterating and supply "true" here

define Key for Dictionary

define Value for Dictionary

always the same decoder instance is reused

Spinning List

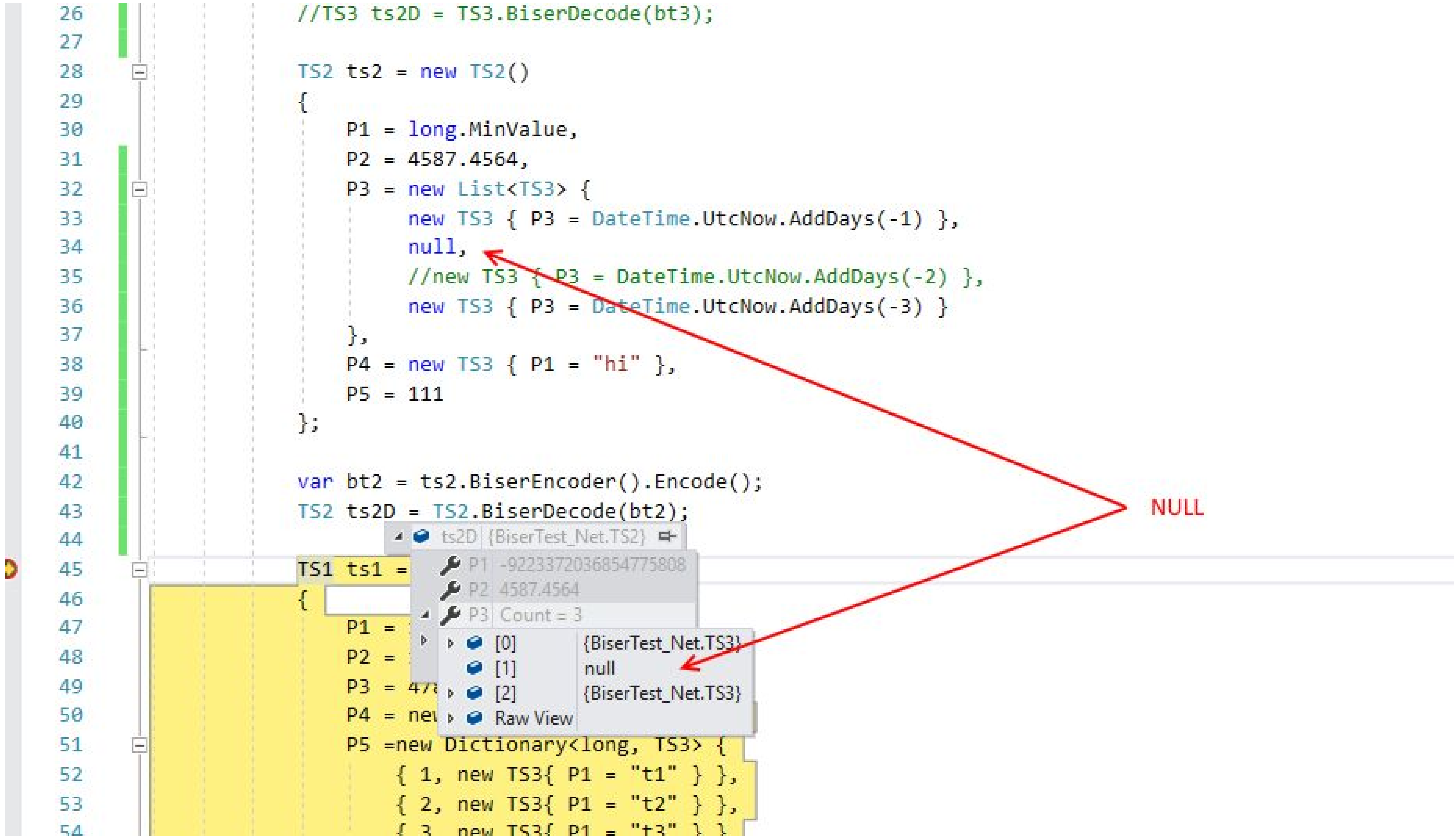
Decoding IEncoder of special type (copy-paste + change return type)

```
namespace BiserTest_Net
{
    8 references
    public partial class TS1
    {
        3 references
        public int P1 { get; set; }
        3 references
        public int P2 { get; set; }
        3 references
        public decimal P3 { get; set; }
        5 references
        public List<TS2> P4 { get; set; }
        5 references
        public Dictionary<long,TS3> P5 { get; set; }
        5 references
        public Dictionary<uint, List<TS3>> P6 { get; set
        3 references
        public TS2 P7 { get; set; }
        5 references
        public List<Tuple<string,byte[],TS3>> P8 { get; :
        6 references
        public Tuple<float, TS2, TS3, decimal?> P9 { get
    }
}
```

decoder.GetCollection() has 3 overloads. It is possible to supply
IList object and define how to decode it,
IDictionary and define separately how to decode Key and Value,
or just get IEnumerable Decoder and decode in free manner
like
foreach(var el in decoder.GetCollection().Select(r=>new Tuple<byte?,int>(r.G
        el.Item1..

Note!!! Supplying IDicionary or IList is faster than yield returns.

IEnumerables can have null values inside:

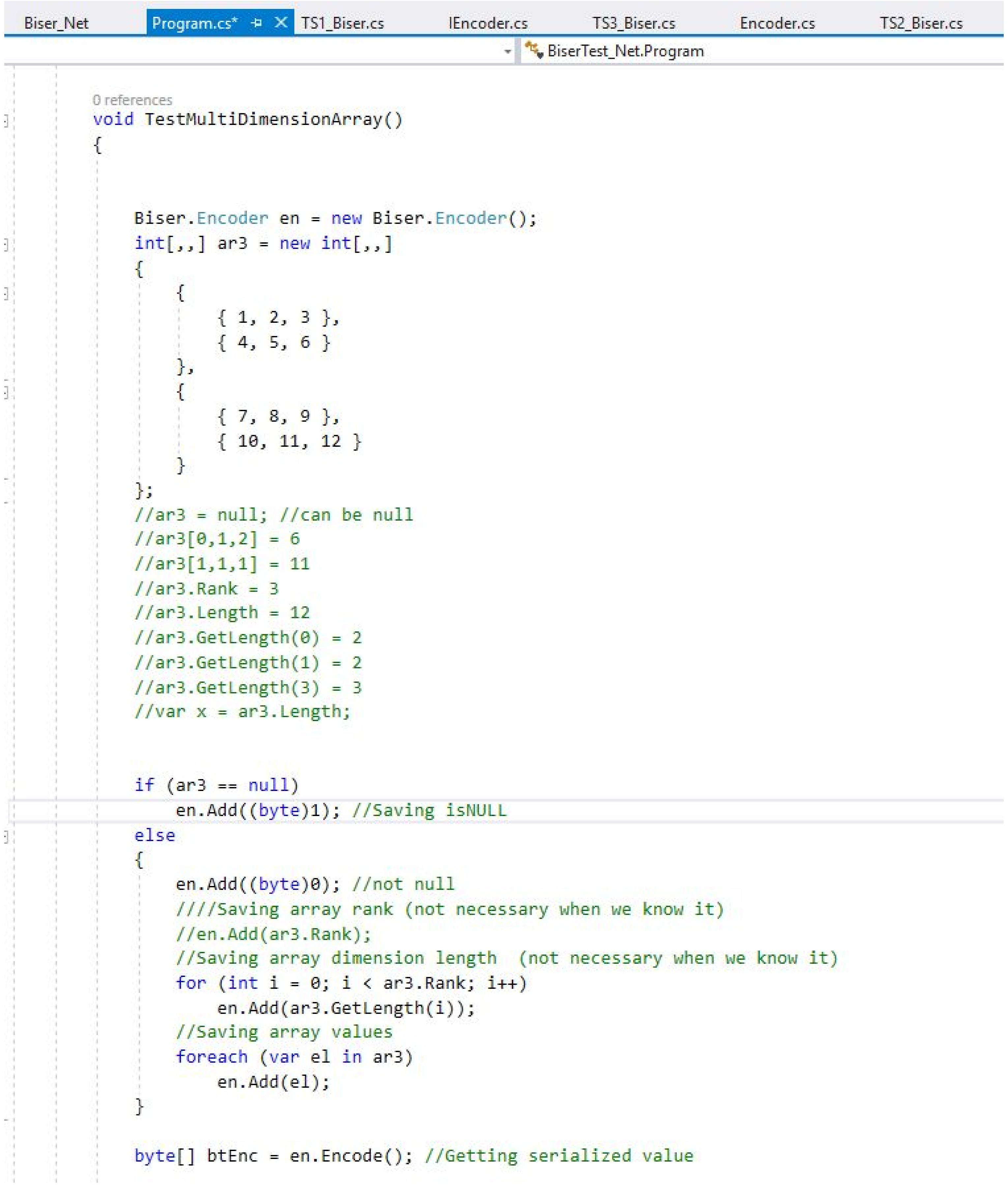


```
//TS3 ts2D = TS3.BiserDecode(bt3);

TS2 ts2 = new TS2()
{
    P1 = long.MinValue,
    P2 = 4587.4564,
    P3 = new List<TS3> {
        new TS3 { P3 = DateTime.UtcNow.AddDays(-1) },
        null,
        //new TS3 { P3 = DateTime.UtcNow.AddDays(-2) },
        new TS3 { P3 = DateTime.UtcNow.AddDays(-3) }
    },
    P4 = new TS3 { P1 = "hi" },
    P5 = 111
};

var bt2 = ts2.BiserEncoder().Encode();
TS2 ts2D = TS2.BiserDecode(bt2);

TS1 ts1 =
{
    P1 =
    P2 =
    P3 = 47
    P4 = ne
    P5 =new Dictionary<long, TS3> {
        { 1, new TS3{ P1 = "t1" } },
        { 2, new TS3{ P1 = "t2" } },
        { 3, new TS3{ P1 = "t3" } }
```

```
ts2D {BiserTest_Net.TS2}
    P1  -9223372036854775808
    P2  4587.4564
    P3  Count = 3
        [0]       {BiserTest_Net.TS3}
        [1]       null
        [2]       {BiserTest_Net.TS3}
        Raw View
```

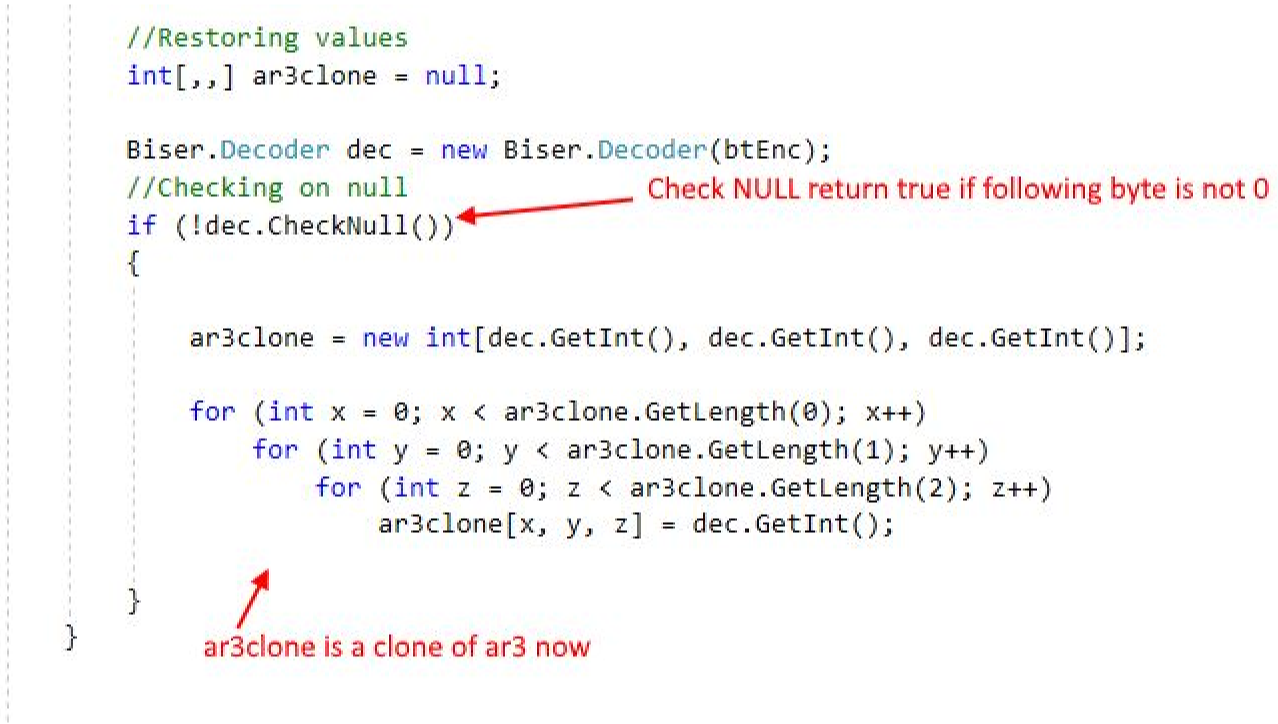
Beads on a string. N-dimensional arrays and custom serialization language

Putting all values that we are interested just one after another like beads on a string, later in the same sequence we will get them back:



```
void TestMultiDimensionArray()
{


    Biser.Encoder en = new Biser.Encoder();
    int[,,] ar3 = new int[,,]
    {
        {
            { 1, 2, 3 },
            { 4, 5, 6 }
        },
        {
            { 7, 8, 9 },
            { 10, 11, 12 }
        }
    };
    //ar3 = null; //can be null
    //ar3[0,1,2] = 6
    //ar3[1,1,1] = 11
    //ar3.Rank = 3
    //ar3.Length = 12
    //ar3.GetLength(0) = 2
    //ar3.GetLength(1) = 2
    //ar3.GetLength(3) = 3
    //var x = ar3.Length;


    if (ar3 == null)
        en.Add((byte)1); //Saving isNULL
    else
    {
        en.Add((byte)0); //not null
        ////Saving array rank (not necessary when we know it)
        //en.Add(ar3.Rank);
        //Saving array dimension length  (not necessary when we know it)
        for (int i = 0; i < ar3.Rank; i++)
            en.Add(ar3.GetLength(i));
        //Saving array values
        foreach (var el in ar3)
            en.Add(el);
    }

    byte[] btEnc = en.Encode(); //Getting serialized value
```

Deserializing values:

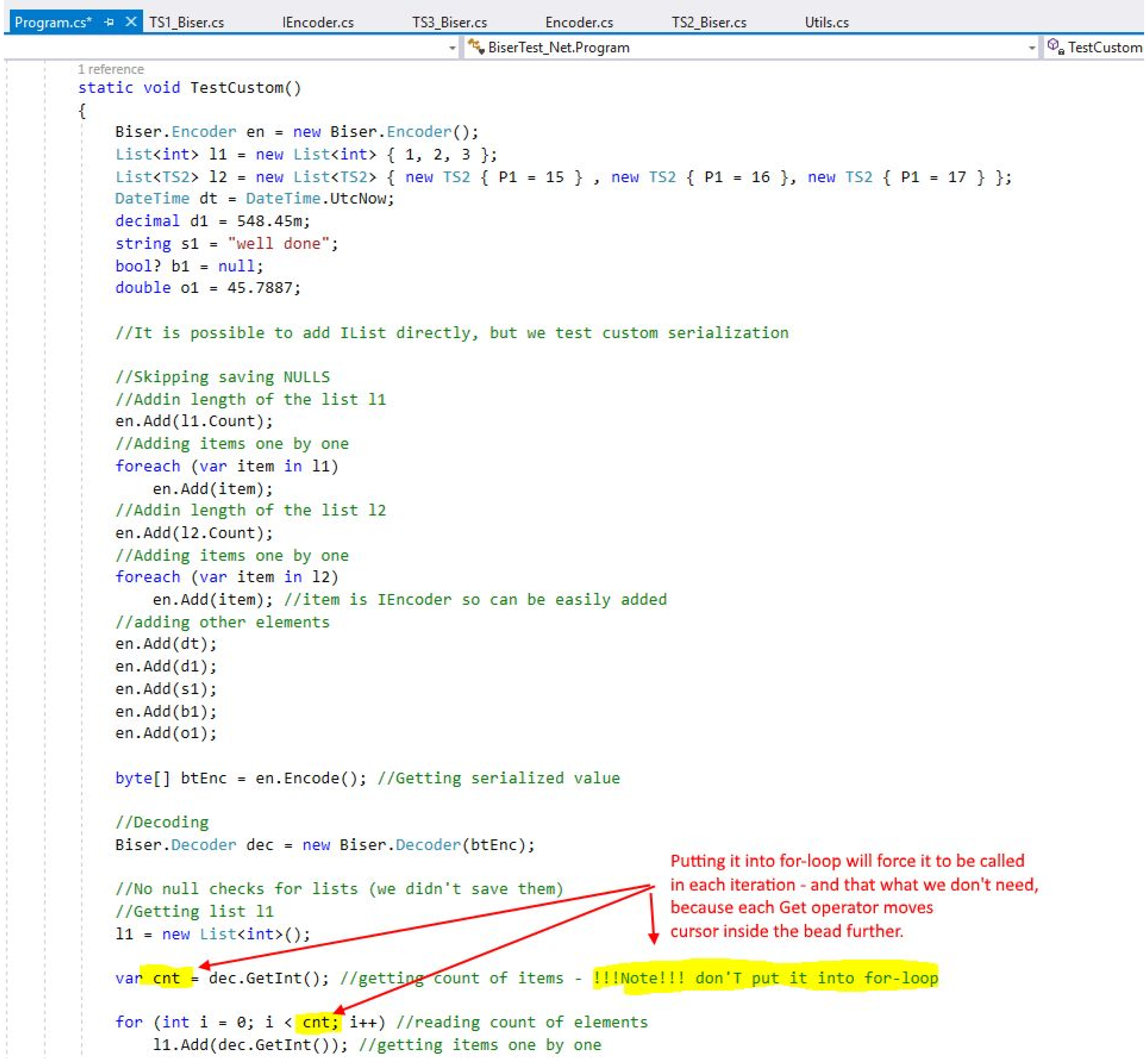


```
    //Restoring values
    int[,,] ar3clone = null;

    Biser.Decoder dec = new Biser.Decoder(btEnc);
    //Checking on null
    if (!dec.CheckNull())
    {

        ar3clone = new int[dec.GetInt(), dec.GetInt(), dec.GetInt()];

        for (int x = 0; x < ar3clone.GetLength(0); x++)
            for (int y = 0; y < ar3clone.GetLength(1); y++)
                for (int z = 0; z < ar3clone.GetLength(2); z++)
                    ar3clone[x, y, z] = dec.GetInt();

    }
}
```

Check NULL return true if following byte is not 0

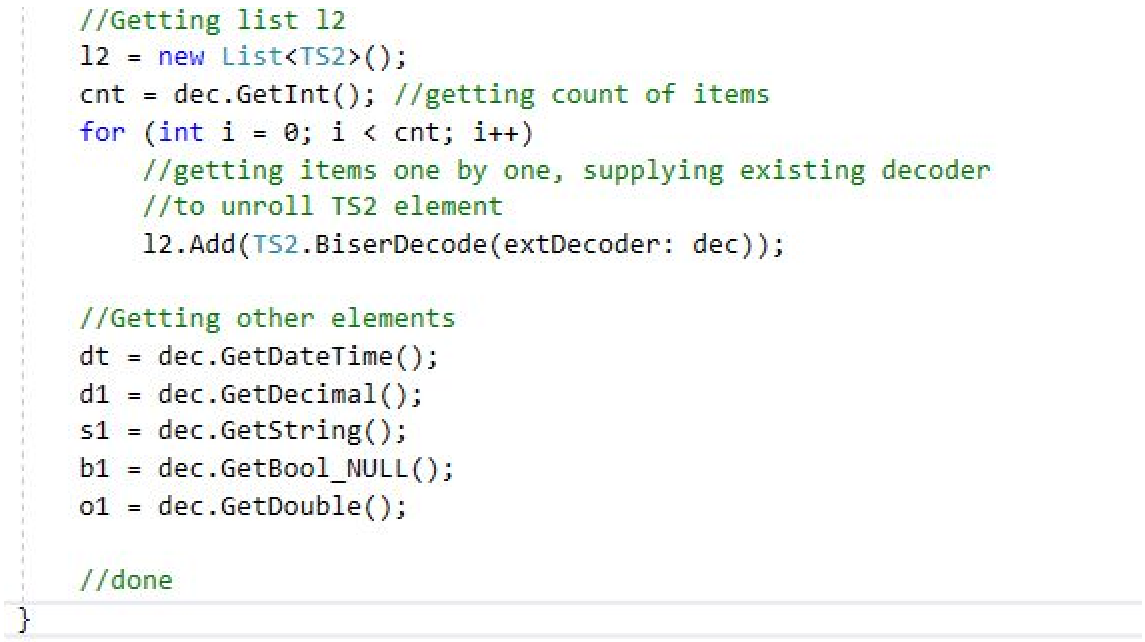
ar3clone is a clone of ar3 now

Custom serialization:



```
static void TestCustom()
{
    Biser.Encoder en = new Biser.Encoder();
    List<int> l1 = new List<int> { 1, 2, 3 };
    List<TS2> l2 = new List<TS2> { new TS2 { P1 = 15 } , new TS2 { P1 = 16 }, new TS2 { P1 = 17 } };
    DateTime dt = DateTime.UtcNow;
    decimal d1 = 548.45m;
    string s1 = "well done";
    bool? b1 = null;
    double o1 = 45.7887;

    //It is possible to add IList directly, but we test custom serialization

    //Skipping saving NULLS
    //Addin length of the list l1
    en.Add(l1.Count);
    //Adding items one by one
    foreach (var item in l1)
        en.Add(item);
    //Addin length of the list l2
    en.Add(l2.Count);
    //Adding items one by one
    foreach (var item in l2)
        en.Add(item); //item is IEncoder so can be easily added
    //adding other elements
    en.Add(dt);
    en.Add(d1);
    en.Add(s1);
    en.Add(b1);
    en.Add(o1);

    byte[] btEnc = en.Encode(); //Getting serialized value

    //Decoding
    Biser.Decoder dec = new Biser.Decoder(btEnc);

    //No null checks for lists (we didn't save them)
    //Getting list l1
    l1 = new List<int>();

    var cnt = dec.GetInt(); //getting count of items - !!!Note!!! don'T put it into for-loop

    for (int i = 0; i < cnt; i++) //reading count of elements
        l1.Add(dec.GetInt()); //getting items one by one
```

Putting it into for-loop will force it to be called in each iteration - and that what we don't need, because each Get operator moves cursor inside the bead further.



```
    //Getting list l2
    l2 = new List<TS2>();
    cnt = dec.GetInt(); //getting count of items
    for (int i = 0; i < cnt; i++)
        //getting items one by one, supplying existing decoder
        //to unroll TS2 element
        l2.Add(TS2.BiserDecode(extDecoder: dec));

    //Getting other elements
    dt = dec.GetDateTime();
    d1 = dec.GetDecimal();
    s1 = dec.GetString();
    b1 = dec.GetBool_NULL();
    o1 = dec.GetDouble();

    //done
}
```

If the length of the collection is known in advance, it is possible to economize 1 byte for NULL representation by adding collection length equal to -1. Integrated encoder.Add(IEnumerable) doesn't know the length of the collection in advance and works a bit different than in this example, effectively storing all necessary information without iterating collection twice.

Biser extensions

There is a set of useful extensions are concentrated in BiserExtensions.cs

It can be copied and pasted into your project or used directly from DLL. Note that decoding extensions based on IDecoder interface for custom objects. Also automatic creation of instances (like in Biser.BiserExtensions.BiserDecodeList example) is not always efficient as an explicit instance creation, though for someone it can be very handy.

TS5 implements IEncoder and IDecoder

```
static void TestBE1()
      {
            //Testing extensions with IDecoder and Biser.Extension interface

            var ttz= ((int)15).BiserEncode();
            var btx = (new HashSet<TS5> { new TS5 { TermId = 15 }, new TS5 { TermId = 16 }, new TS5 { TermId = 17 } }).BiserEncodeList();
            var ttzD = ttz.BiserDecode<int>();
            var btxD = btx.BiserDecodeHashSet<TS5>();
      }
```

Biser is integrated into DBreeze database and is a part of Raft.Net