56)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

public class TestAttribute : Attribute

{

public string att1 { get; set; }

public string att2 { get; set; }

public double att3 { get; set; }

public int att4 { get; set; }

public TestAttribute()

{

}

public TestAttribute(string at1, string at2, double at3, int at4)

{

att1 = at1;

att2 = at2;

att3 = at3;

att4 = at4;

}

}

namespace ConsoleApplication1

{

class Program

{

static void Main(string[] args)

{

TestAttribute att = new TestAttribute();

att.att1 = "Colton";

att.att2 = "Parry";

att.att3 = 10.00;

att.att4 = 67;

Console.WriteLine(att.att1);

Console.WriteLine(att.att2);

Console.WriteLine(att.att3);

Console.WriteLine(att.att4);

Console.ReadLine();

}

}

}

57

public \_ddataIndexer this[int index]

{

get

{

if (index >= -20.22D & index < 34000.55D )

return \_ddata[index];

else

throw new Exception(invalid);

}

set

{

index %= MIN\_ID;

if (index >= -20.22D & index < 34000.55D )

\_ddata[index] =value;

else

throw new Exception(invalid);

}

}

public \_fdataIndexer this[int index]

{

get

{

if (index >= -20.22D & index < 34000.55D )

return \_fdata[index];

else

throw new Exception(invalid);

}

set

{

index %= MIN\_ID;

if (index >= -20.22D & index < 34000.55D )

\_fdata[index] =value;

else

throw new Exception(invalid);

}

}

58)

public void \_setCdata(decimal d)

{

if(d >= 1.00M && d < 15000000.505M)

{

\_cdata = d;

}

else

{

throw Exception("Not a valid number");

}

}

public decimal \_getCdata()

{

return \_cdata;

}

59)

We don’t know the names of underlying variables for the automatic properties? Why do we care?

60)

61)

public static int extMethod(this string str)

{

string pattern = "[ou]?";

int validString = 0 ;

int invalidString = -1;

if (Regex.IsMatch(str, pattern))

{

return validstring.index;

}

else

{

return invalidString;

1

Console.WriteLine();

return str;

}

62)

3

3

5

4

6

5

3

63) System.Object

64)

65)

public static Stuff<Stuff, double, char> operator + (Stuff<T> lop, Stuff<T> rop)

{

return new Data(lop.\_operand + rop.\_operand);

}

public static Stuff<Stuff, double, char> operator + (Stuff<T> lop)

{

return new Data(lop.\_operand + lop.\_operand);

}

66)

67) var data =

from n in sData

where n == "^[Fr][aeiou]+"

group n by n into g

select g;

foreach (var g in data)

{

Console.WriteLine(g);

}

68) No it is not a value type, Random is a method performed on number value types to return a random selection of that value type.

69) No you cannot inherit from the string class, that’s why strings are immutable. The string class is sealed and that’s why you cannot inherit from it. You can however write extension methods to the string class.

70)

s0 = “ABCD”

S1 = “” an empty string;

S2 = default value of what ever string called;

71) depends on the initial value of i

The first thing that happens is that I is post incremented because of that parenthesis, then it is multiplied by itself, and then assigned to itself.

Thus is it one more than what I has been declared as multiplied by that number.

I = 0 returns 1

I = 1 returns 4

I = 2 returns 9

Etc….