15.–18. 09. 2008
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Puzzlen.Net

Unerwartetes zum Mitgrübeln

Oliver Szymanski

MATHEMA Software GmbH

Haha

```
class Program
{
    static void Main(string[] args)
    {
        Console.Write("H" + "a");
        Console.WriteLine('H' + 'a');
        Console.ReadKey();
    }
}
```

Haha

```
class Program
{
    static void Main(string[] args)
    {
        Console.Write("H" + "a");
        Console.WriteLine('H' + 'a');
        Console.ReadKey();
    }
}
```

- ✔ HaHa
- ✔ Exception
- ✔ Kompilierfehler
- ✔ Nichts davon

Haha

```
class Program
{
    static void Main(string[] args)
    {
        Console.Write("H" + "a");
        Console.WriteLine('H' + 'a');
        Console.ReadKey();
    }
}
```

- ✗ HaHa
- ✗ Exception
- ✗ Kompilierfehler
- ✓ Ha169

Modulo & Co.

```
class Program
{
    static void Main(string[] args)
    {
        int MOD = 3;
        int[] histogram = new int[MOD];

        int i = int.MinValue;
        do
        {
            histogram[Math.Abs(i) % MOD]++;
        } while (i++ != int.MaxValue);

        foreach (int c in histogram)
        {
            Console.WriteLine(c);
        }

        Console.ReadKey();
    }
}
```

Modulo & Co.

```
class Program
{
    static void Main(string[] args)
    {
        int MOD = 3;
        int[] histogram = new int[MOD];

        int i = int.MinValue;
        do
        {
            histogram[Math.Abs(i) % MOD]++;
        } while (i++ != int.MaxValue);

        foreach (int c in histogram)
        {
            Console.WriteLine(c);
        }

        Console.ReadKey();
    }
}
```

- ✔ Endlosschleife
- ✔ Exception
- ✔ x \n x \n x
- ✔ 0 \n 0 \n 0

Modulo & Co.

```
class Program
{
    static void Main(string[] args)
    {
        int MOD = 3;
        int[] histogram = new int[MOD];

        int i = int.MinValue;
        do
        {
            histogram[Math.Abs(i) % MOD]++;
        } while (i++ != int.MaxValue);

        foreach (int c in histogram)
        {
            Console.WriteLine(c);
        }

        Console.ReadKey();
    }
}
```

- ✗ Endlosschleife
- ✓ Exception
- ✗ x \n x \n x
- ✗ 0 \n 0 \n 0

ColorPoint

```
class Point
{
    private readonly int x, y;
    private readonly string name;

    public Point(int x, int y)
    {
        this.x = x;
        this.y = y;
        this.name = MakeName();
    }

    protected virtual string MakeName()
    {
        return "[" + x + ", " + y + "]";
    }

    public override String ToString()
    { return name; }
}
```

ColorPoint

```
class ColorPoint : Point
{
    private readonly string color;

    public ColorPoint(int x, int y, string color) : base(x, y)
    {
        this.color = color;
    }

    protected override string MakeName()
    {
        return base.MakeName() + " : " + color;
    }
}

static void Main(string[] args)
{
    Console.WriteLine(new ColorPoint(3, 4, "blau"));
}
```

ColorPoint

```
class ColorPoint : Point
{
    private readonly string color;

    public ColorPoint(int x, int y, string color) : base(x, y)
    {
        this.color = color;
    }

    protected override string MakeName()
    {
        return base.MakeName() + " : " + color;
    }
}

static void Main(string[] args)
{
    Console.WriteLine(new ColorPoint(3, 4, "blau"));
}
```

- [0, 0]
- [3, 4] :
- [3, 4]
- [3, 4] : blau

ColorPoint

```
class ColorPoint : Point
{
    private readonly string color;

    public ColorPoint(int x, int y, string color) : base(x, y)
    {
        this.color = color;
    }

    protected override string MakeName()
    {
        return base.MakeName() + " : " + color;
    }
}

static void Main(string[] args)
{
    Console.WriteLine(new ColorPoint(3, 4, "blau"));
}
```

- ✗ [0, 0]
- ✓ [3, 4] :
- ✗ [3, 4]
- ✗ [3, 4] : blau

Elementary

```
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine(12345 + 5432l);
    }
}
```

Elementary

```
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine(12345 + 5432l);
    }
}
```

- 66666
- Exception
- 17777
- Kompilierfehler

Elementary

```
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine(12345 + 5432l);
    }
}
```

- ✗ 66666
- ✗ Exception
- ✓ 17777
- ✗ Kompilierfehler

Exceptions

```
class Program
{
    static int test() {
        try {
            throw new Exception("Error 1");
        } catch (Exception e) {
            throw new Exception("Error 2");
        } finally {
            throw new Exception("Error 3");
        }
    }

    static void Main(string[] args)
    {
        try
        { Console.WriteLine(test()); }
        catch (Exception e)
        { Console.WriteLine(e.Message); }
    }
}
```

Exceptions

```
class Program
{
    static int test() {
        try {
            throw new Exception("Error 1");
        } catch (Exception e) {
            throw new Exception("Error 2");
        } finally {
            throw new Exception("Error 3");
        }
    }

    static void Main(string[] args)
    {
        try
        { Console.WriteLine(test()); }
        catch (Exception e)
        { Console.WriteLine(e.Message); }
    }
}
```

- ✔ Error 1
- ✔ Error 2
- ✔ Error 3
- ✔ Kompilierfehler

Kurzer Einschub

```
class Program
{
    static int test() {
        try {
            throw new Exception("Error 1");
        } catch (Exception e) {
            throw new Exception("Error 2");
        } finally {
            return 3;
        }
    }

    static void Main(string[] args)
    {
        try
        { Console.WriteLine(test()); }
        catch (Exception e)
        { Console.WriteLine(e.Message); }
    }
}
```

Kurzer Einschub

```
class Program
{
    static int test() {
        try {
            throw new Exception("Error 1");
        } catch (Exception e) {
            throw new Exception("Error 2");
        } finally {
            return 3;
        }
    }

    static void Main(string[] args)
    {
        try
        { Console.WriteLine(test()); }
        catch (Exception e)
        { Console.WriteLine(e.Message); }
    }
}
```

- Error 1
- Error 2
- 3
- Kompilierfehler

Kurzer Einschub

```
class Program
{
    static int test() {
        try {
            throw new Exception("Error 1");
        } catch (Exception e) {
            throw new Exception("Error 2");
        } finally {
            return 3;
        }
    }

    static void Main(string[] args)
    {
        try
        { Console.WriteLine(test()); }
        catch (Exception e)
        { Console.WriteLine(e.Message); }
    }
}
```

- ✗ Error 1
- ✗ Error 2
- ✗ 3
- ✓ Kompilierfehler

Was ist nun hiermit?

```
class Program
{
    static int test() {
        try {
            throw new Exception("Error 1");
        } catch (Exception e) {
            throw new Exception("Error 2");
        } finally {
            throw new Exception("Error 3");
        }
    }

    static void Main(string[] args)
    {
        try
        { Console.WriteLine(test()); }
        catch (Exception e)
        { Console.WriteLine(e.Message); }
    }
}
```

- ✔ Error 1
- ✔ Error 2
- ✔ Error 3
- ✔ Kompilierfehler

Was ist nun hiermit?

```
class Program
{
    static int test() {
        try {
            throw new Exception("Error 1");
        } catch (Exception e) {
            throw new Exception("Error 2");
        } finally {
            throw new Exception("Error 3");
        }
    }

    static void Main(string[] args)
    {
        try
        { Console.WriteLine(test()); }
        catch (Exception e)
        { Console.WriteLine(e.Message); }
    }
}
```

- ✗ Error 1
- ✗ Error 2
- ✓ Error 3
- ✗ Kompilierfehler

Finally and Exit

```
class Program
{
    static void Main(string[] args)
    {
        try
        {
            Environment.Exit(0);
        }
        finally
        {
            Console.WriteLine("ohoh");
        }
    }
}
```

Finally and Exit

```
class Program
{
    static void Main(string[] args)
    {
        try
        {
            Environment.Exit(0);
        }
        finally
        {
            Console.WriteLine("ohoh");
        }
    }
}
```

- Program endet
- "ohoh"
- Exception
- Kompilierfehler

Finally and Exit

```
class Program
{
    static void Main(string[] args)
    {
        try
        {
            Environment.Exit(0);
        }
        finally
        {
            Console.WriteLine("ohoh");
        }
    }
}
```

- ✔ Program endet
- ✗ "ohoh"
- ✗ Exception
- ✗ Kompilierfehler

Field Override

```
class Base
{
    public string name = "Snoopy";
}

class Derived : Base
{
    private string name = "Charly";
}

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine(new Derived().name);
    }
}
```

Field Override

```
class Base
{
    public string name = "Snoopy";
}

class Derived : Base
{
    private string name = "Charly";
}

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine(new Derived().name);
    }
}
```

- Exception
- Kompilierfehler
- "Snoopy"
- "Charly"

Field Override

```
class Base
{
    public string name = "Snoopy";
}

class Derived : Base
{
    private string name = "Charly";
}

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine(new Derived().name);
    }
}
```

- ✗ Exception
- ✗ Kompilierfehler
- ✓ "Snoopy"
- ✗ "Charly"

Und so?

```
class Base
{
    public string name = "Snoopy";
}

class Derived : Base
{
    public string name = "Charly";
}

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine(new Derived().name);
    }
}
```

Und so?

```
class Base
{
    public string name = "Snoopy";
}

class Derived : Base
{
    public string name = "Charly";
}

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine(new Derived().name);
    }
}
```

- ✔ Exception
- ✔ Kompilierfehler
- ✔ "Snoopy"
- ✔ "Charly"

Und so?

```
class Base
{
    public string name = "Snoopy";
}

class Derived : Base
{
    public string name = "Charly";
}

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine(new Derived().name);
    }
}
```

- ✗ Exception
- ✗ Kompilierfehler
- ✗ "Snoopy"
- ✓ "Charly"

Incrementer

```
class Program
{
    static void Main(string[] args)
    {
        int j = 0;
        for (int i = 0; i < 100; i++)
            j = j++;
        Console.WriteLine(j);
        Console.ReadKey();
    }
}
```

Incrementer

```
class Program
{
    static void Main(string[] args)
    {
        int j = 0;
        for (int i = 0; i < 100; i++)
            j = j++;
        Console.WriteLine(j);
        Console.ReadKey();
    }
}
```

- Endlosschleife
- 99
- 100
- 0

Incrementer

```
class Program
{
    static void Main(string[] args)
    {
        int j = 0;
        for (int i = 0; i < 100; i++)
            j = j++;
        Console.WriteLine(j);
        Console.ReadKey();
    }
}
```

- ✗ Endlosschleife
- ✗ 99
- ✗ 100
- ✓ 0

Infinity Loop

```
while (i == i + 1)
   {
      Console.WriteLine("tada");
   }
```

Infinity Loop

```
while (i == i + 1)
{
        Console.WriteLine("tada");
}
```

i?

Infinity Loop

```
while (i == i + 1)
{
        Console.WriteLine("tada");
}
```

✔ double i = double.PositiveInfinity

Long Division

```
class Program
{
    static void Main(string[] args)
    {
        const long MICROS_PER_DAY = 24 * 60 * 60 * 1000 * 1000;
        const long MILLIS_PER_DAY = 24 * 60 * 60 * 1000;
        Console.WriteLine(MICROS_PER_DAY / MILLIS_PER_DAY);
    }
}
```

Long Division

```
class Program
{
    static void Main(string[] args)
    {
        const long MICROS_PER_DAY = 24 * 60 * 60 * 1000 * 1000;
        const long MILLIS_PER_DAY = 24 * 60 * 60 * 1000;
        Console.WriteLine(MICROS_PER_DAY / MILLIS_PER_DAY);
    }
}
```

- Laufzeitfehler
- 1000
- 5
- Kompilierfehler

Long Division

```
class Program
{
    static void Main(string[] args)
    {
        const long MICROS_PER_DAY = 24 * 60 * 60 * 1000 * 1000;
        const long MILLIS_PER_DAY = 24 * 60 * 60 * 1000;
        Console.WriteLine(MICROS_PER_DAY / MILLIS_PER_DAY);
    }
}
```

- ✗ Laufzeitfehler
- ✗ 1000
- ✗ 5
- ✓ Kompilierfehler

MaxLoop

```
class Program
{
    static void Main(string[] args)
    {
        int count = 0;
        int MAX = int.MaxValue;
        int START = MAX - 1;

        for (int i = START; i <= MAX; i++)
            count++;

        Console.WriteLine(count);
    }
}
```

MaxLoop

```
class Program
{
    static void Main(string[] args)
    {
        int count = 0;
        int MAX = int.MaxValue;
        int START = MAX - 1;

        for (int i = START; i <= MAX; i++)
            count++;

        Console.WriteLine(count);
    }
}
```

- ✔ 100
- ✔ 101
- ✔ Laufzeitfehler
- ✔ Nichts davon

MaxLoop

```
class Program
{
    static void Main(string[] args)
    {
        int count = 0;
        int MAX = int.MaxValue;
        int START = MAX - 1;

        for (int i = START; i <= MAX; i++)
            count++;

        Console.WriteLine(count);
    }
}
```

- ✗ 100
- ✗ 101
- ✗ Laufzeitfehler
- ✓ Endlosschleife

Not Himself: Multiple Persönlichkeiten?

```
Console.WriteLine(i != i);
```

Not Himself: Multiple Persönlichkeiten?

```
Console.WriteLine(i != i);
```

↙ i ?

Not Himself: Multiple Persönlichkeiten?

```
Console.WriteLine(i != i);
```

- ✓ double i = double.NaN;

Nullable

```
class Program
{
    static void Main(string[] args)
    {
        int? i = null;
        int j = i ?? 3;

        Console.WriteLine(j);
    }
}
```

Nullable

```
class Program
{
    static void Main(string[] args)
    {
        int? i = null;
        int j = i ?? 3;

        Console.WriteLine(j);
    }
}
```

- Null
- Exception
- 3
- Kompilierfehler

Nullable

```
class Program
{
    static void Main(string[] args)
    {
        int? i = null;
        int j = i ?? 3;

        Console.WriteLine(j);
    }
}
```

- ✗ Null
- ✗ Exception
- ✓ 3
- ✗ Kompilierfehler

Oddity

```
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine(isOdd(3));
    }

    static bool isOdd(int i) {
        return i % 2 == 1;
    }
}
```

Oddity

```
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine(isOdd(3));
    }

    static bool isOdd(int i) {
        return i % 2 == 1;
    }
}
```

- Immer richtig
- Nie richtig
- 50% richtig
- 75% richtig

Oddity

```
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine(isOdd(3));
    }

    static bool isOdd(int i) {
        return i % 2 == 1;
    }
}
```

- ✗ Immer richtig
- ✗ Nie richtig
- ✗ 50% richtig
- ✓ 75% richtig

Overloading

```
class Program
{
    private Program(object o)
    {
        Console.WriteLine("object");
    }

    private Program(double[] d)
    {
        Console.WriteLine("doubleArray");
    }

    static void Main(string[] args)
    {
        new Program(null);
    }
}
```

Overloading

```
class Program
{
    private Program(object o)
    {
        Console.WriteLine("object");
    }

    private Program(double[] d)
    {
        Console.WriteLine("doubleArray");
    }

    static void Main(string[] args)
    {
        new Program(null);
    }
}
```

- Laufzeitfehler
- "object"
- "doubleArray"
- Kompilierfehler

Overloading

```
class Program
{
    private Program(object o)
    {
        Console.WriteLine("object");
    }

    private Program(double[] d)
    {
        Console.WriteLine("doubleArray");
    }

    static void Main(string[] args)
    {
        new Program(null);
    }
}
```

- ✗ Laufzeitfehler
- ✗ "object"
- ✓ "doubleArray"
- ✗ Kompilierfehler

PGMain

```
class Program
{
    private static Random rnd = new Random();
    static void Main(string[] args)
    {
        StringBuilder word = null;
        switch (rnd.Next(2)) {
            case 1:
                word = new StringBuilder('P');
                break;
            case 2:
                word = new StringBuilder('G');
                break;
            default:
                 word = new StringBuilder('M');
                break;
        }
        word.Append('a').Append('i').Append('n');
        Console.WriteLine(word);
    }
}
```

PGMain

```
class Program
{
    private static Random rnd = new Random();
    static void Main(string[] args)
    {
        StringBuilder word = null;
        switch (rnd.Next(2)) {
            case 1:
                word = new StringBuilder('P');
                break;
            case 2:
                word = new StringBuilder('G');
                break;
            default:
                word = new StringBuilder('M');
                break;
        }
        word.Append('a').Append('i').Apper
        Console.WriteLine(word);
    }
}
```

- ✔ Pain / Gain / Main
- ✔ Immer Pain
- ✔ Immer Main
- ✔ Nichts davon

PGMain

```
class Program
{
    private static Random rnd = new Random();
    static void Main(string[] args)
    {
        StringBuilder word = null;
        switch (rnd.Next(2)) {
            case 1:
                word = new StringBuilder('P');
                break;
            case 2:
                word = new StringBuilder('G');
                break;
            default:
                word = new StringBuilder('M');
                break;
        }
        word.Append('a').Append('i').Apper
        Console.WriteLine(word);
    }
}
```

- ✗ Pain / Gain / Main
- ✗ Immer Pain
- ✗ Immer Main
- ✓ Nichts davon

Boxing

```
class Program
{
    static void DoIt(int i)
    {
        Console.WriteLine("int: " + i);
    }

    static void DoIt(object o)
    {
        Console.WriteLine("object: " + o);
    }

    static void Main(string[] args)
    {
        object o = 3;
        DoIt(o);
    }
}
```

Boxing

```
class Program
{
    static void DoIt(int i)
    {
        Console.WriteLine("int: " + i);
    }

    static void DoIt(object o)
    {
        Console.WriteLine("object: " + o);
    }

    static void Main(string[] args)
    {
        object o = 3;
        DoIt(o);
    }
}
```

- Laufzeitfehler
- "int: 3"
- Kompilierfehler
- "object: 3"

Boxing

```
class Program
{
    static void DoIt(int i)
    {
        Console.WriteLine("int: " + i);
    }

    static void DoIt(object o)
    {
        Console.WriteLine("object: " + o);
    }

    static void Main(string[] args)
    {
        object o = 3;
        DoIt(o);
    }
}
```

- ✗ Laufzeitfehler
- ✗ "int: 3"
- ✗ Kompilierfehler
- ✓ "object: 3"

Static Init

```
class Program
{

    private static bool init = false;

    static Program()
    {

        Thread t = new Thread(new ThreadStart(run));
        t.Start();
        try
        {  t.Join();  }
        catch (ThreadInterruptedException e)
        {  throw new Exception(e.Message, e);  }
    }

    static void run()
    {  init = true;  }
}
```

Static Init

```
class Program
{

    private static bool init = false;

    static Program()
    {

        Thread t = new Thread(new ThreadStart(run));
        t.Start();
        try
        {  t.Join();  }
        catch (ThreadInterruptedException e)
        {  throw new Exception(e.Message, e);  }
    }

    static void run()
    {  init = true;  }
}
```

- ✔ Exception
- ✔ true
- ✔ true / false
- ✔ Nichts davon

Static Init

```
class Program
{

    private static bool init = false;

    static Program()
    {

        Thread t = new Thread(new ThreadStart(run));
        t.Start();
        try
        {  t.Join();  }
        catch (ThreadInterruptedException e)
        {  throw new Exception(e.Message, e);  }
    }

    static void run()
    {  init = true;  }
}
```

- ✗ Exception
- ✗ true
- ✗ true / false
- ✓ Nichts davon

15.–18. 09. 2008
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Vielen Dank!

Oliver Szymanski

MATHEMA Software GmbH