OOP as if you meant it

Messaging as a programming model

Ralf Westphal

info@ralfw.de

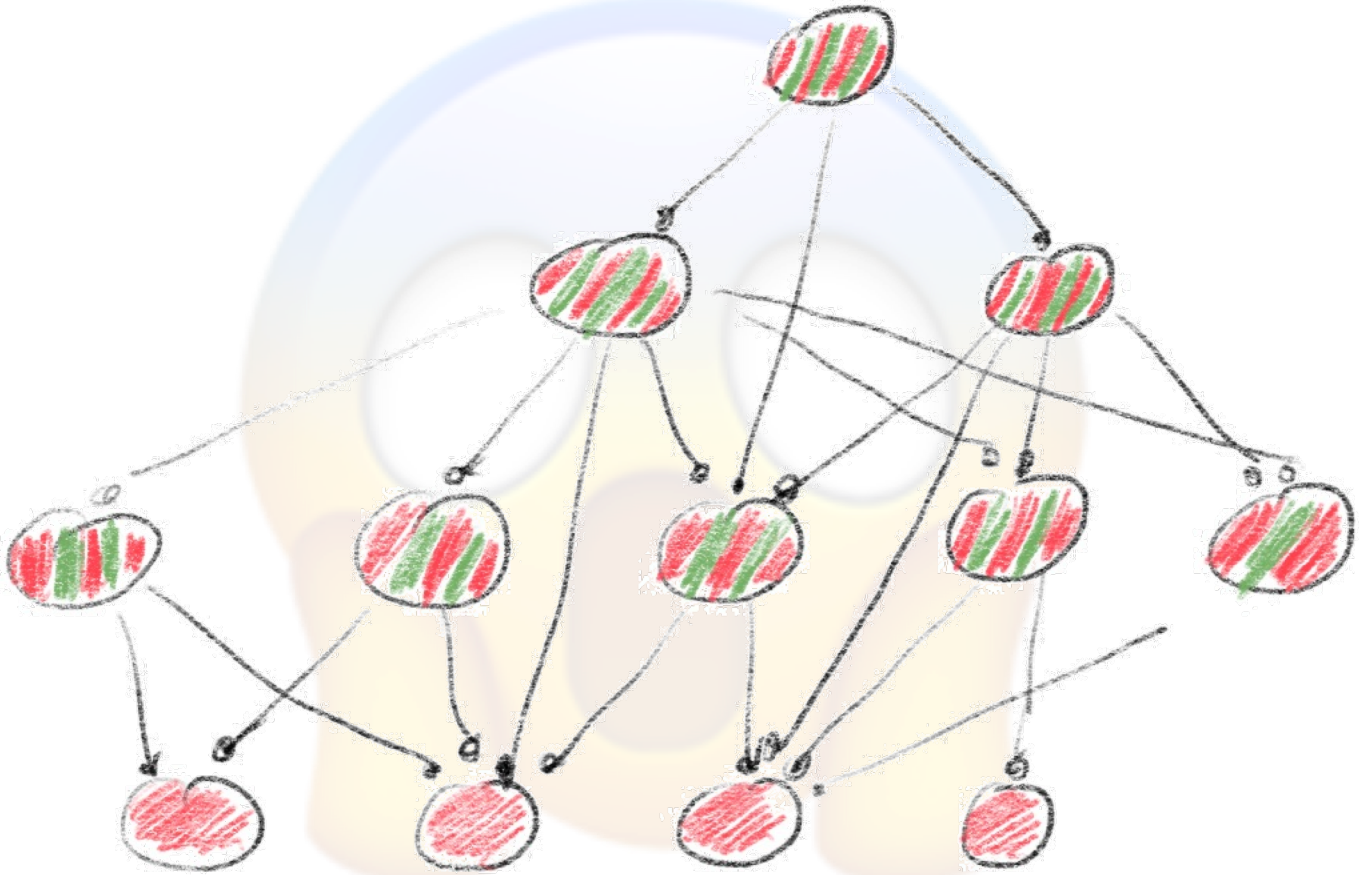
@ralfw (DE/EN)

www.ralfw.de (DE)

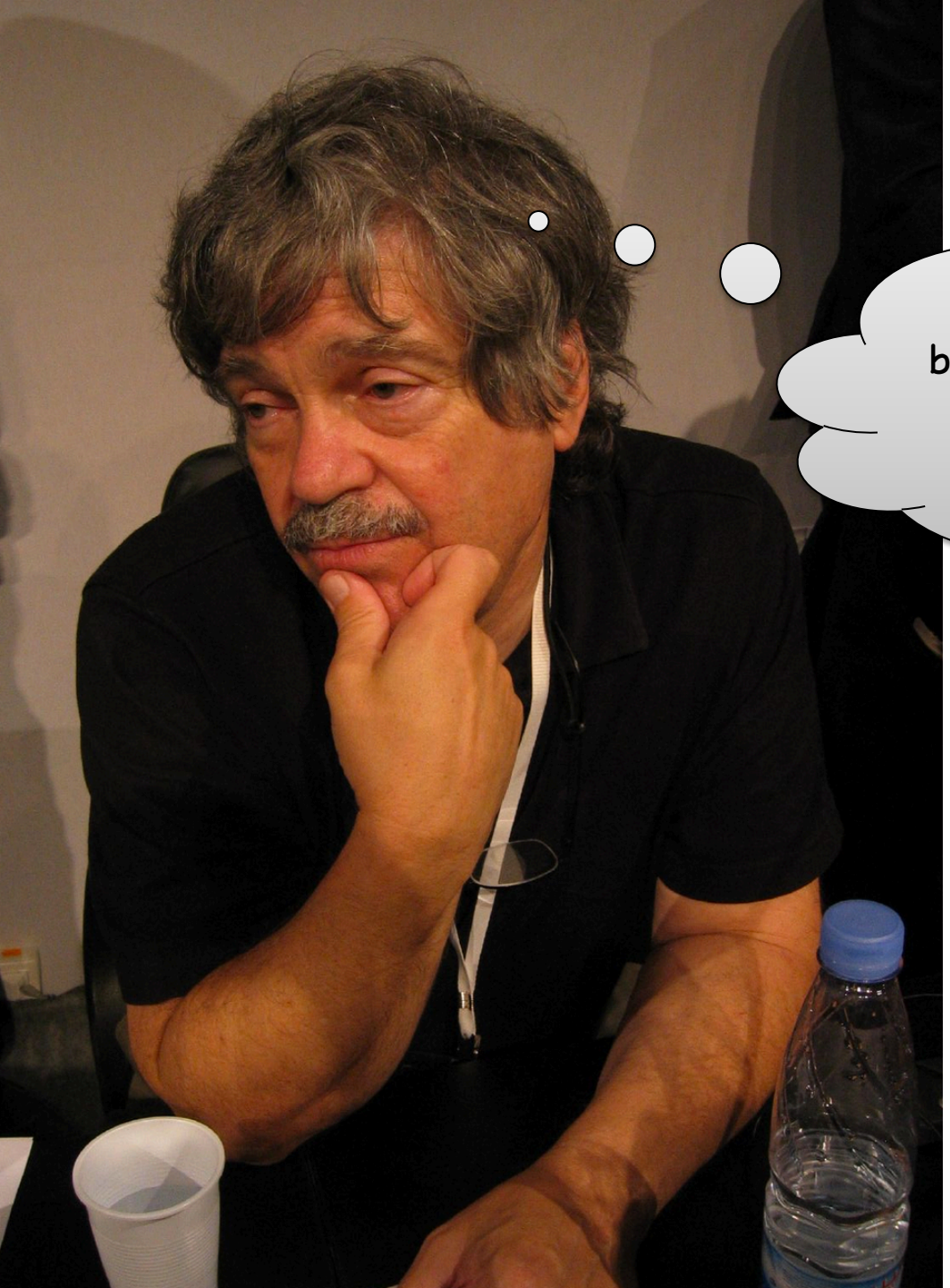
https://ralfw.de/category/english/ (EN)

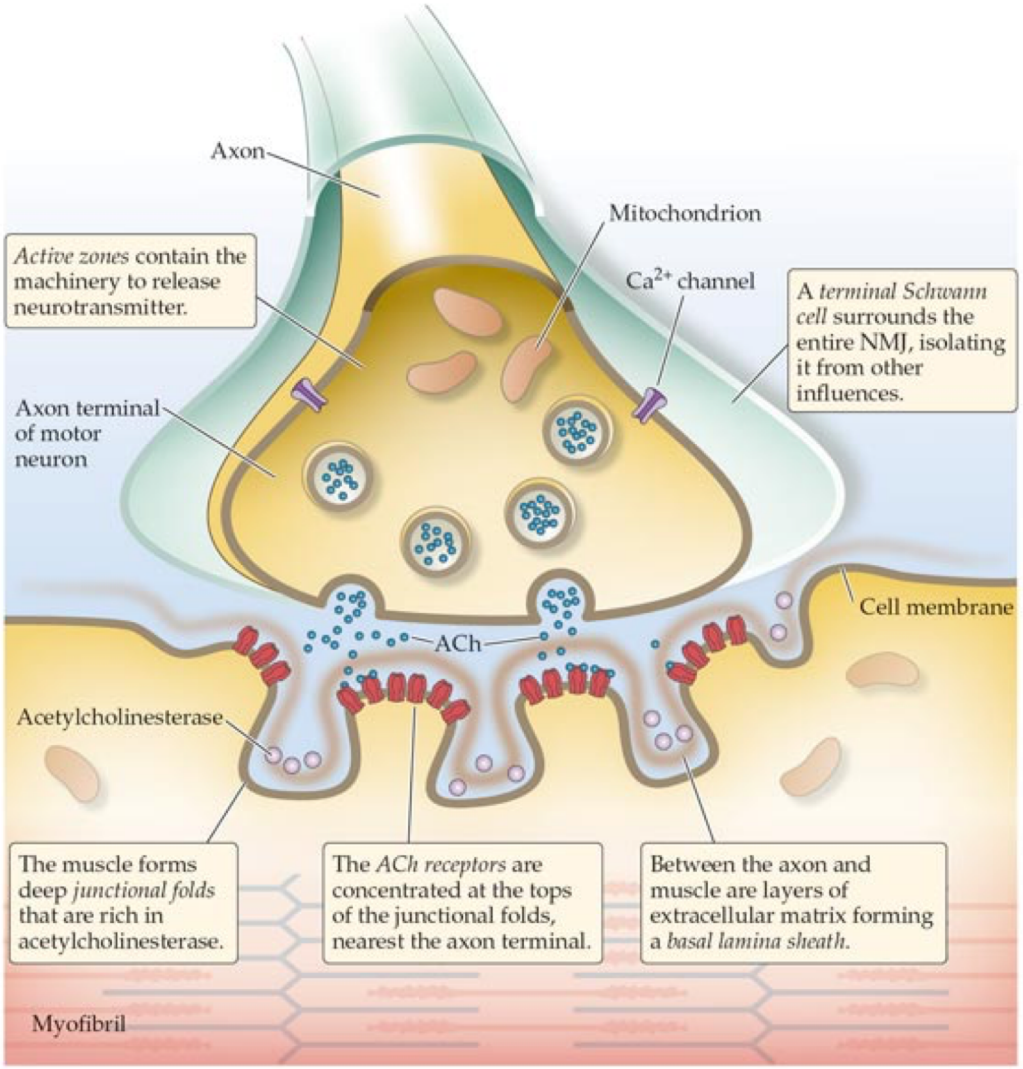
DevDays

sli.do #K100



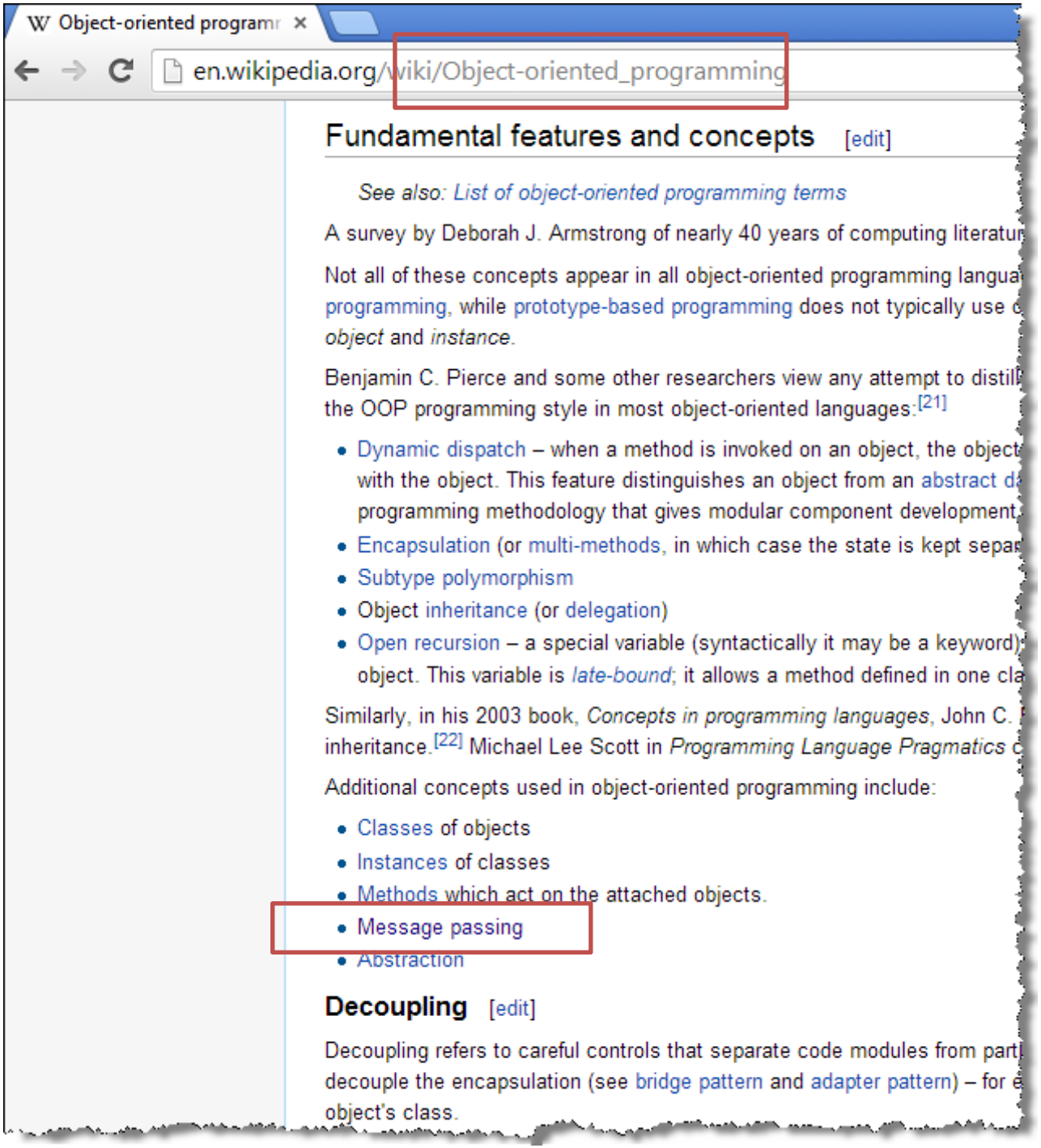
Concepts


„[I´m thinking] of objects
being like biological cells [...]
only able to communicate
with messages."



THE NEUROMUSCULAR JUNCTION


What´s lacking in today´s
object-orientation?
„The big idea is messaging."



Fundamental features and concepts [edit]

See also: List of object-oriented programming terms

A survey by Deborah J. Armstrong of nearly 40 years of computing literatur

Not all of these concepts appear in all object-oriented programming langua programming, while prototype-based programming does not typically use c *object* and *instance*.

Benjamin C. Pierce and some other researchers view any attempt to distil the OOP programming style in most object-oriented languages:[21]

- Dynamic dispatch – when a method is invoked on an object, the object with the object. This feature distinguishes an object from an abstract da programming methodology that gives modular component development.
- Encapsulation (or multi-methods, in which case the state is kept separ
- Subtype polymorphism
- Object inheritance (or delegation)
- Open recursion – a special variable (syntactically it may be a keyword) object. This variable is *late-bound*; it allows a method defined in one cla

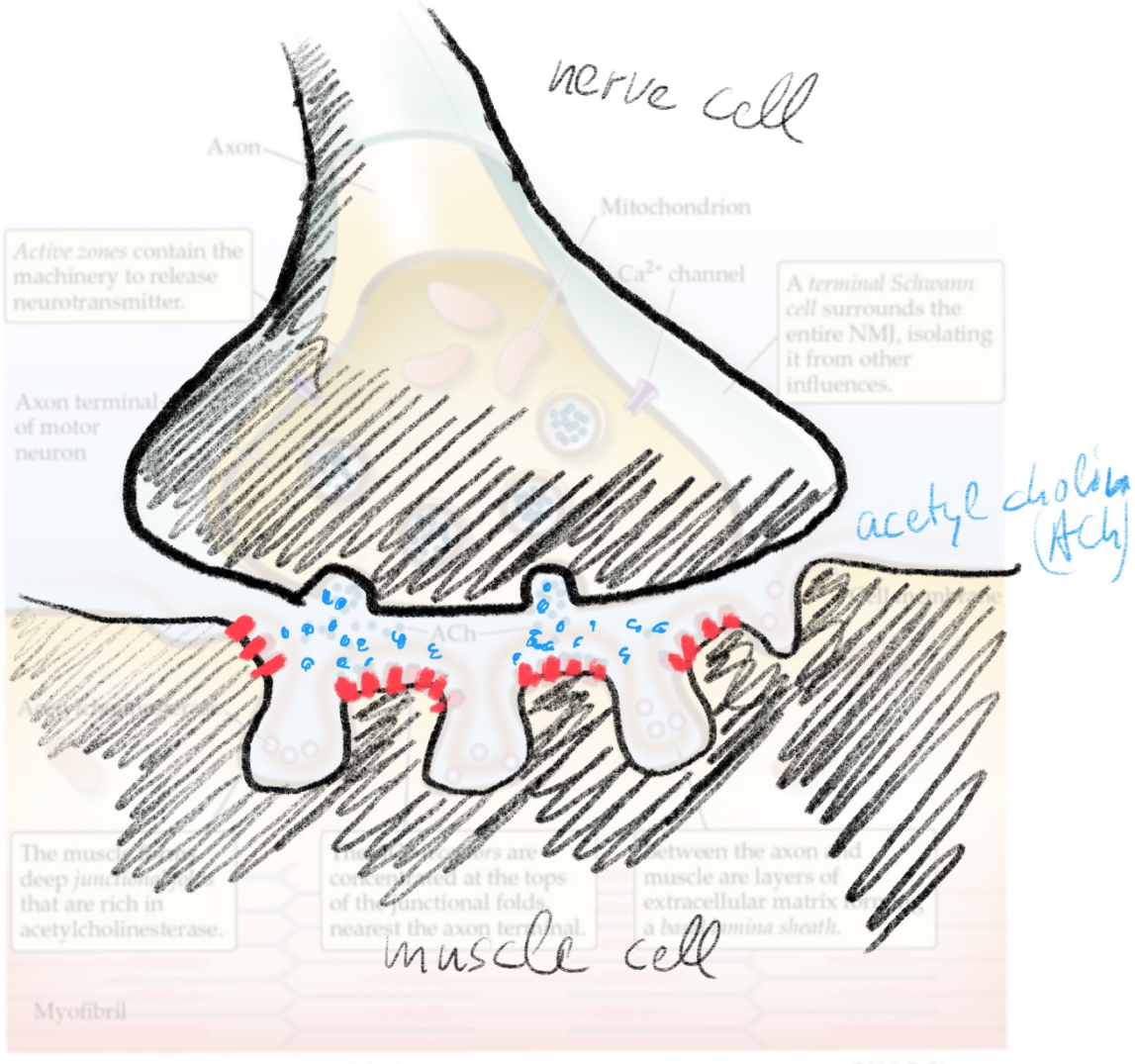
Similarly, in his 2003 book, *Concepts in programming languages*, John C. inheritance.[22] Michael Lee Scott in *Programming Language Pragmatics* c

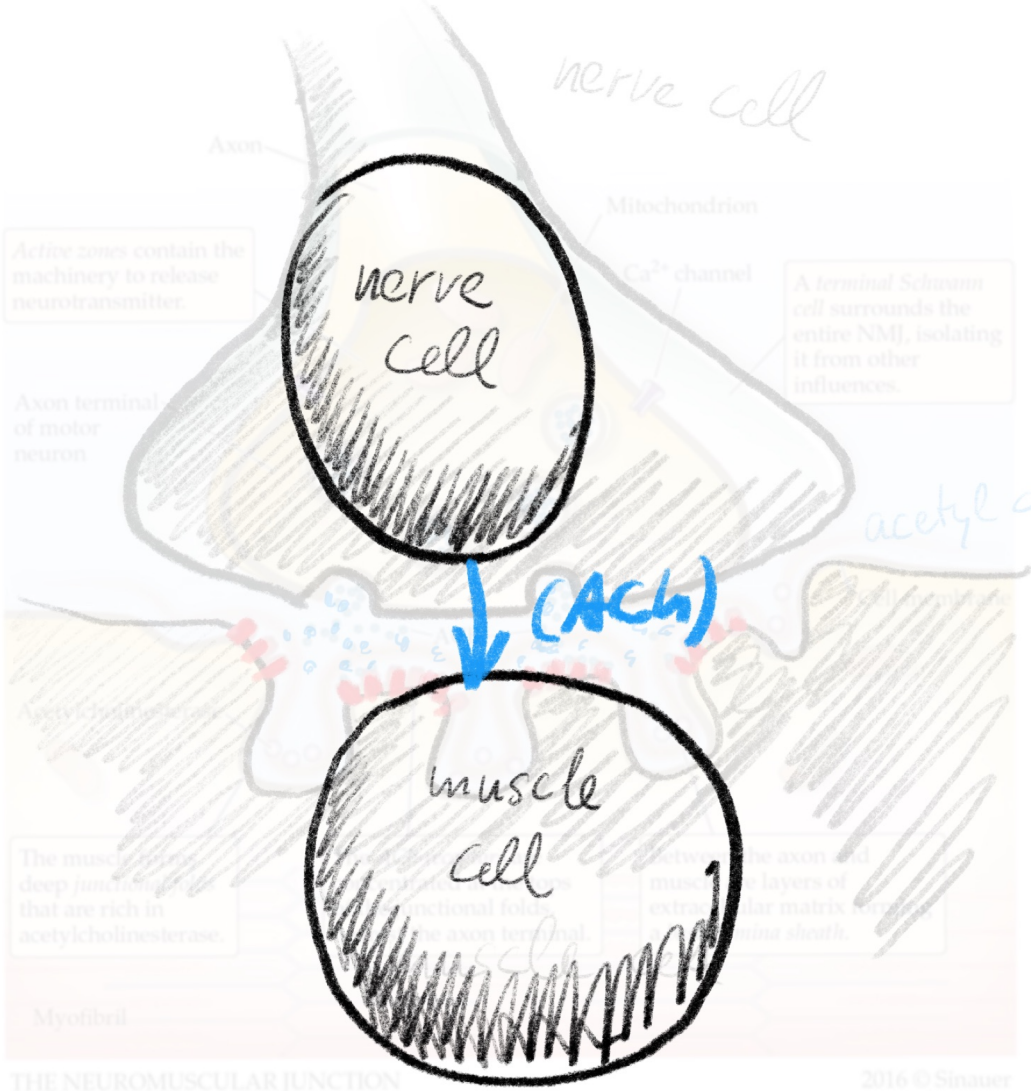
Additional concepts used in object-oriented programming include:

- Classes of objects
- Instances of classes
- Methods which act on the attached objects.
- Message passing
- Abstraction

Decoupling [edit]

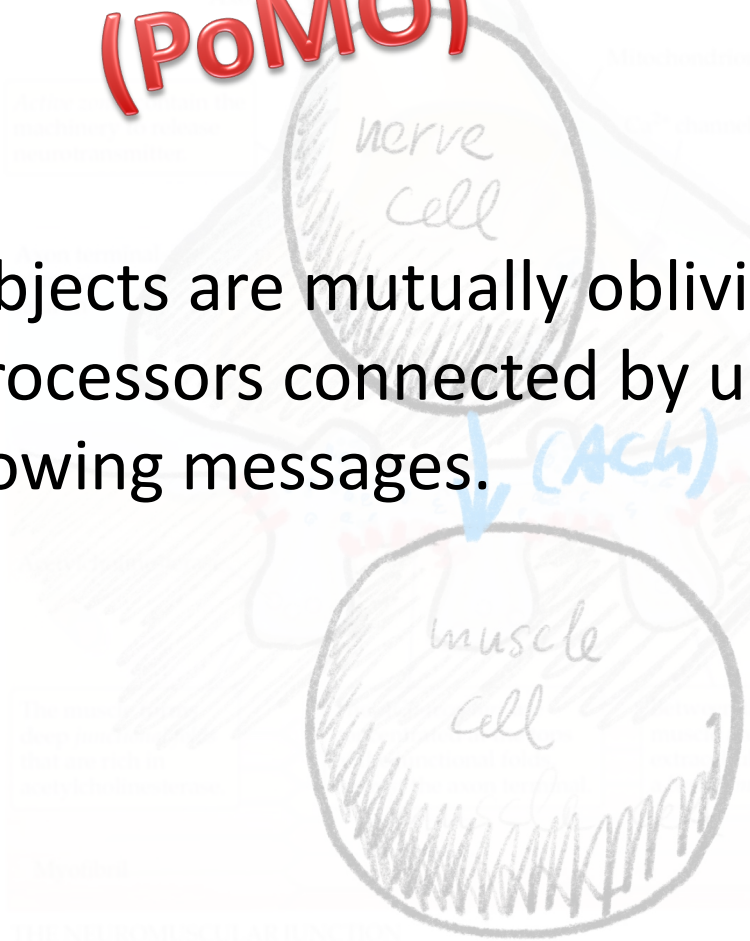
Decoupling refers to careful controls that separate code modules from parti decouple the encapsulation (see bridge pattern and adapter pattern) – for e object's class.

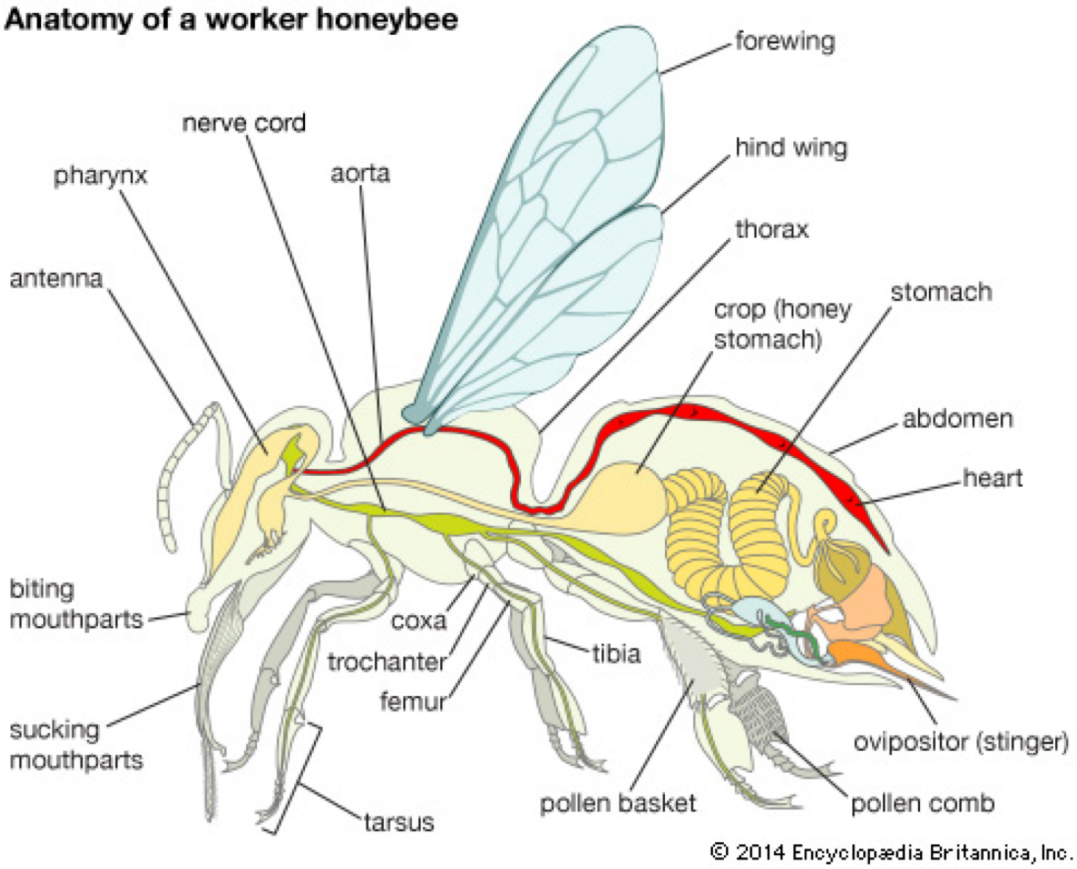

nerve cell
Axon
Mitochondrion
Active zones contain the machinery to release neurotransmitter.
Ca^{2+} channel
A terminal Schwann cell surrounds the entire NMJ, isolating it from other influences.
Axon terminal of motor neuron
acetyl choline (ACh)
ACh
muscle cell
Myofibril
THE NEUROMUSCULAR JUNCTION
2016 © Sinauer

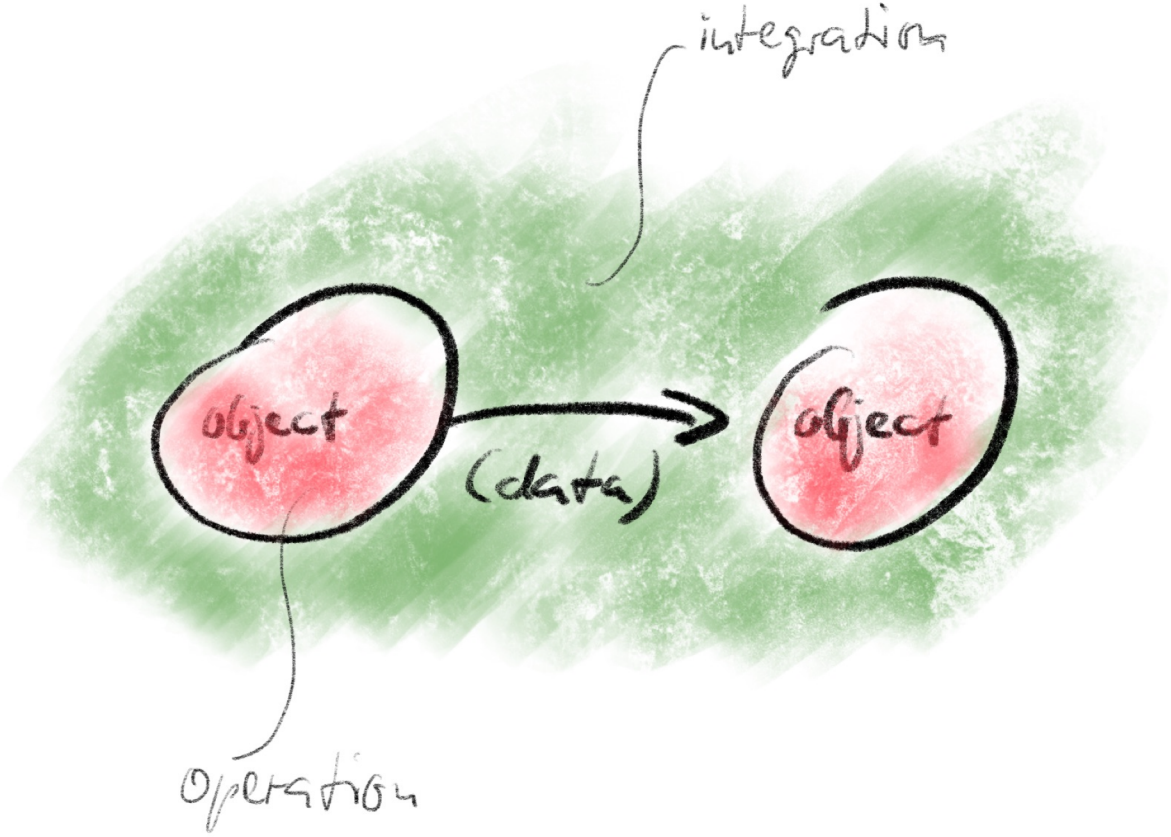

nerve cell
(ACh)
muscle cell

Principle of Mutual Oblivion (PoMO)

Objects are mutually oblivious data processors connected by unidirectionally flowing messages.




Anatomy of a worker honeybee
forewing
nerve cord
hind wing
pharynx
aorta
thorax
antenna
stomach
crop (honey stomach)
abdomen
heart
biting mouthparts
coxa
trochanter
tibia
femur
sucking mouthparts
ovipositor (stinger)
pollen basket
pollen comb
tarsus
© 2014 Encyclopædia Britannica, Inc.

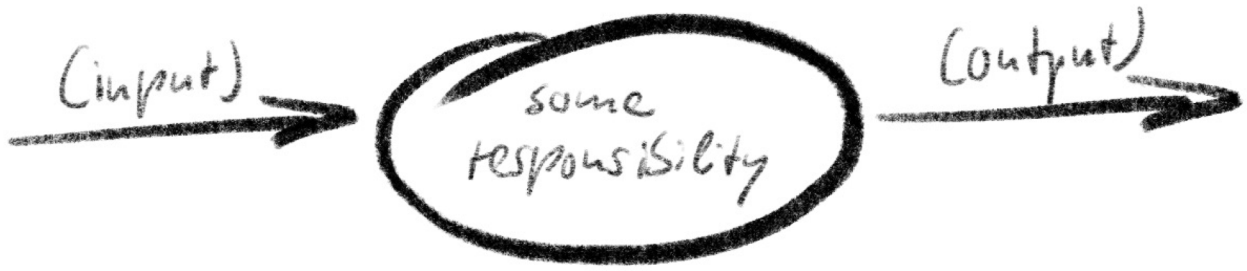

integration
object
(data)
object
operation

Integration Operation Segregation Principle (IOSP)

Objects are either operational, i.e. they contain only behavior creating logic[1)], or they are integrational, i.e. they wire together other objects into processes.

[1)] Expressions, control statements, I/O (API calls)

Translation




```
TOutput SomeResponsibility(TInput input) {...}

void SomeResponsibility(TInput input,
                        Action<TOutput> onOutput) {...}

class SomeResponsibility {
  public void Process(TInput input) {
    ...
    onOutput(...);
    ...
  }

  public event Action<TOutput> onOutput;
}
```

1D flow

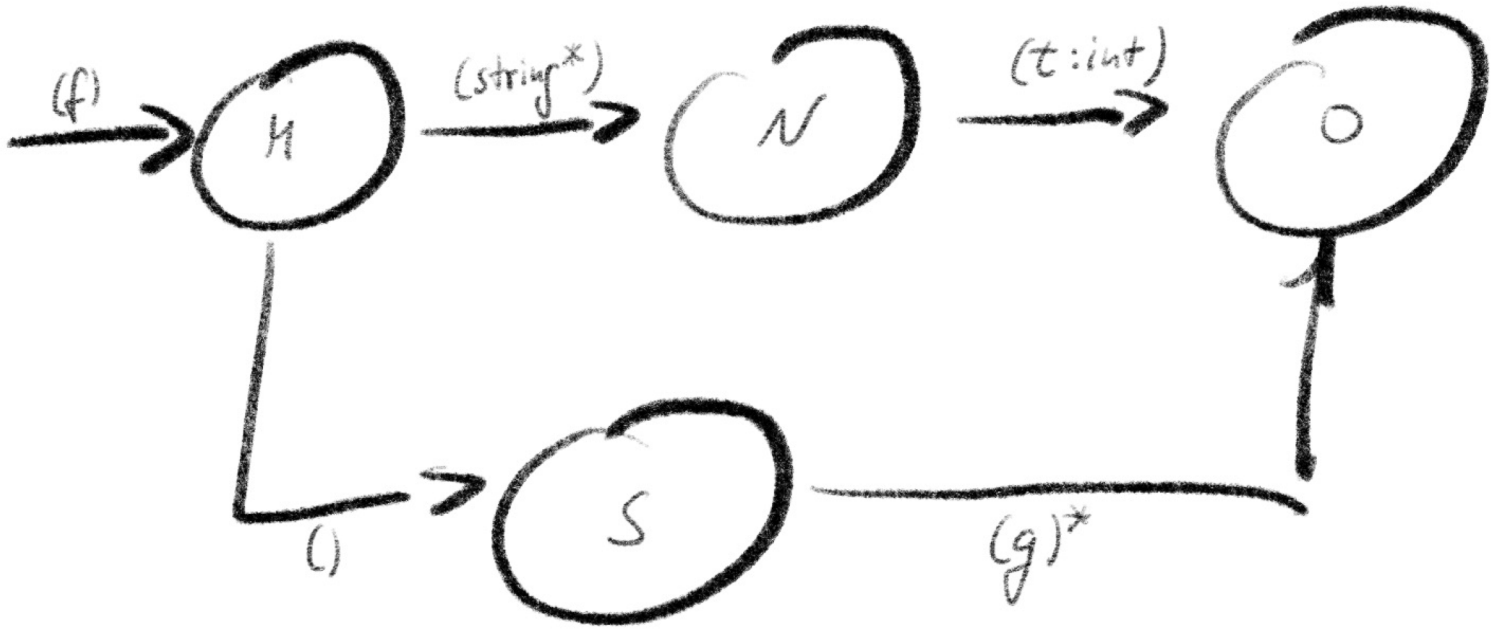
```
var s = A(x);
var stats = B(s);
C(stats);
```

```
x |> A |> B |> C
```

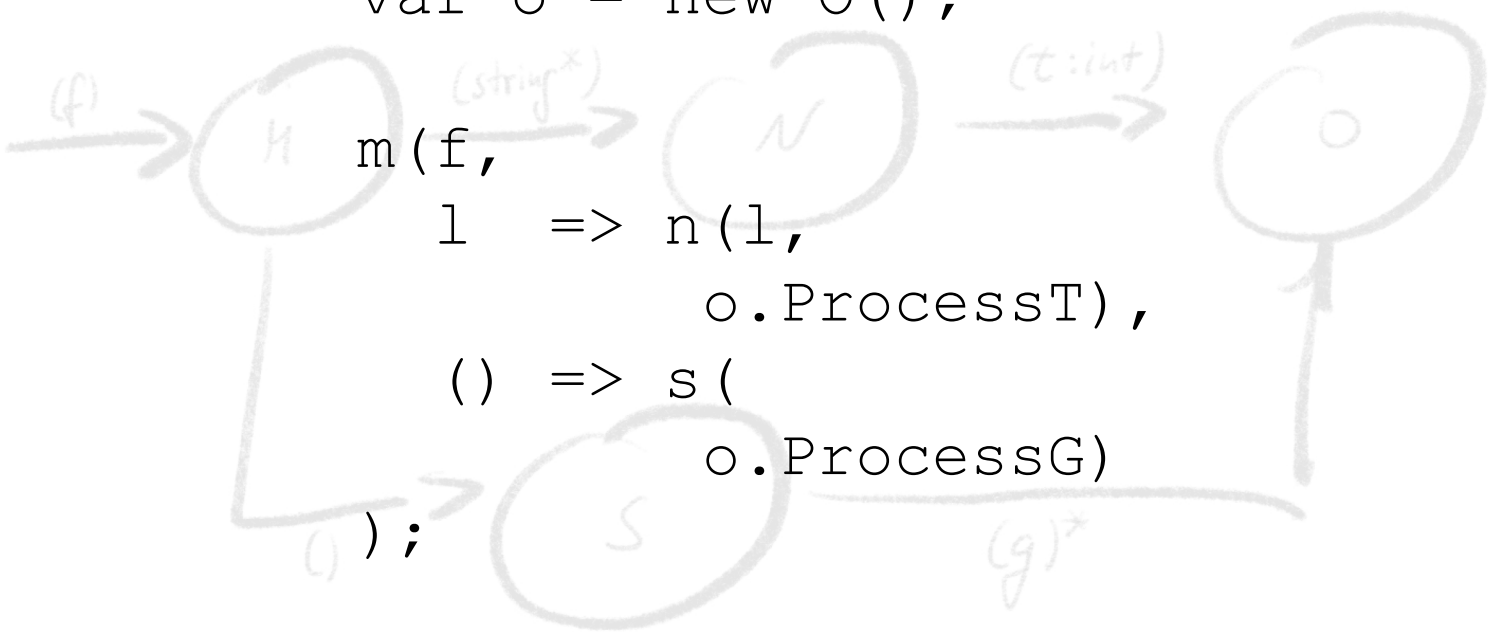
```
var a = new A();
var b = new B();
var c = new C();

a.onS += b.Process;
b.onStats += c.Process;

a.Process(x);
```

```
A(x, s =>
   B(s,
     C));
```

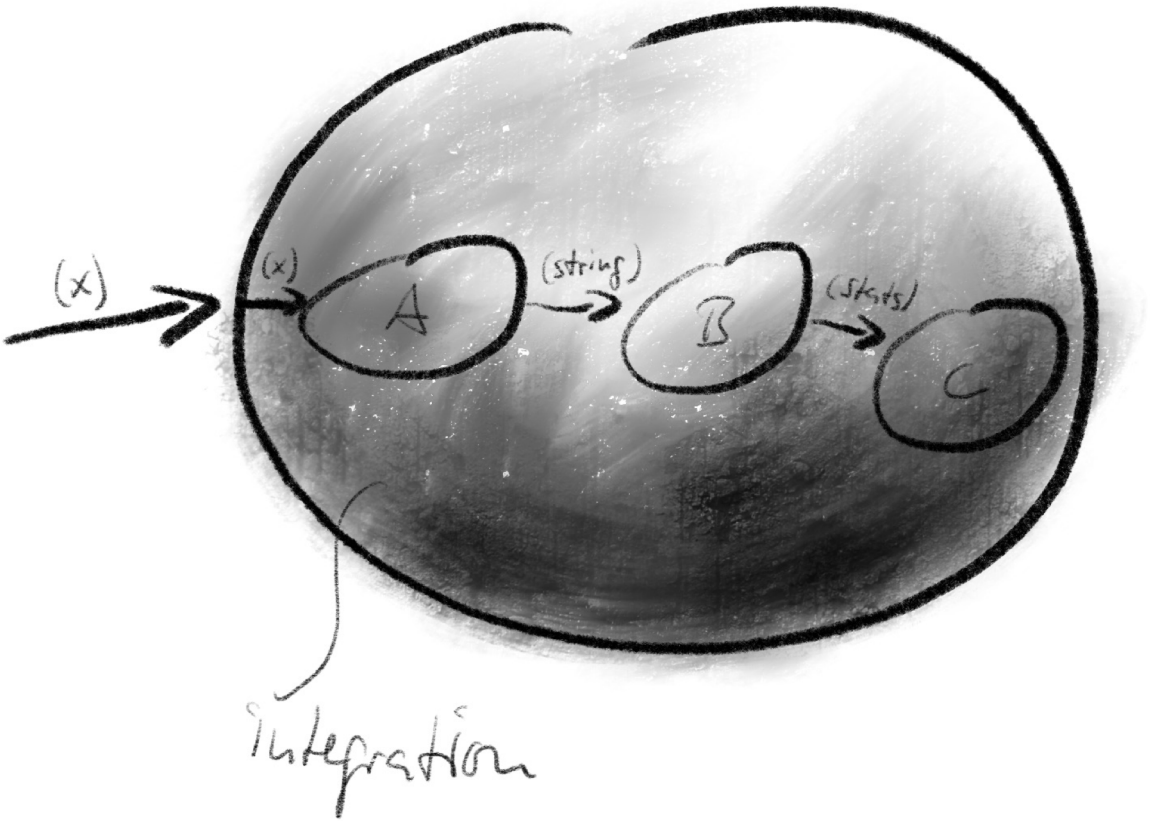
2D flow



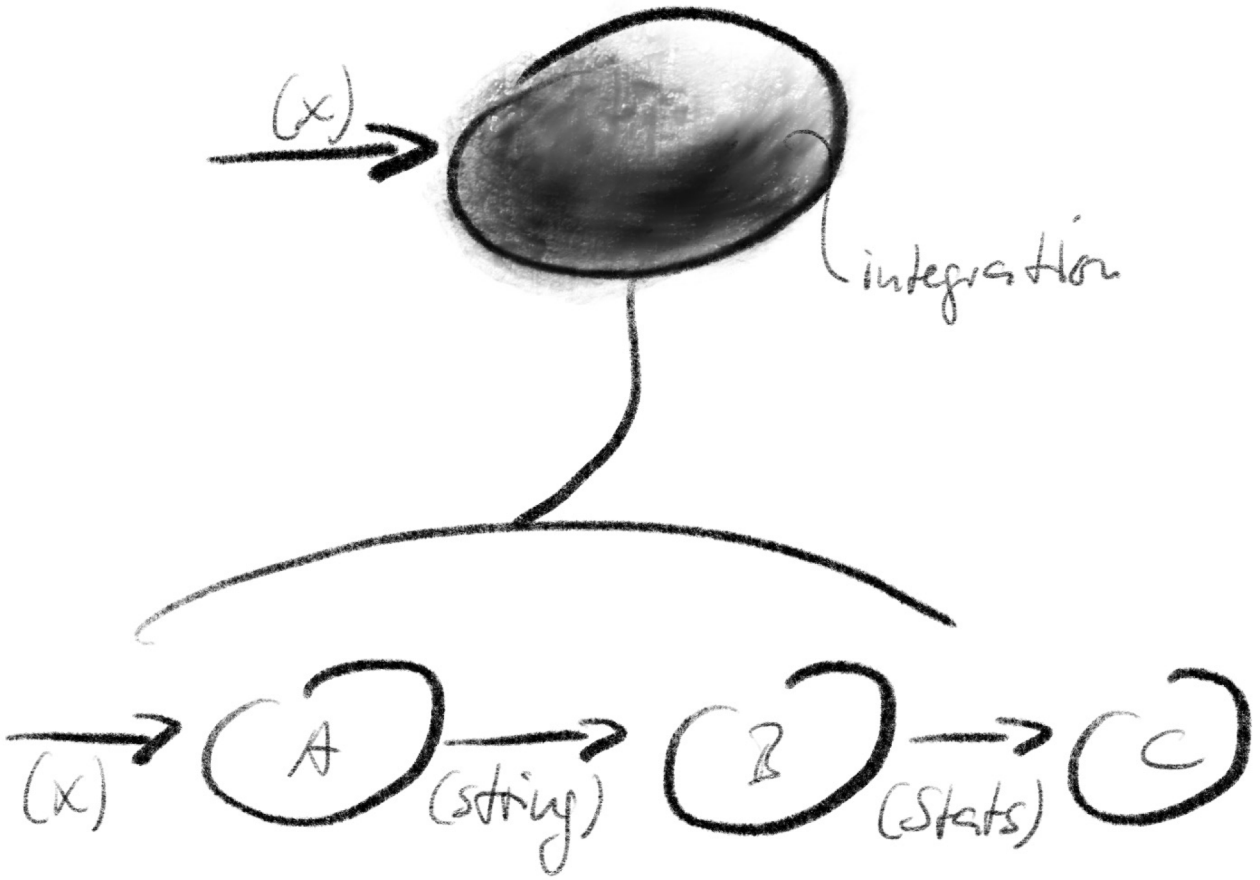
```
var o = new O();

m(f,
   l  => n(l,
           o.ProcessT),
   () => s(
           o.ProcessG)
);
```



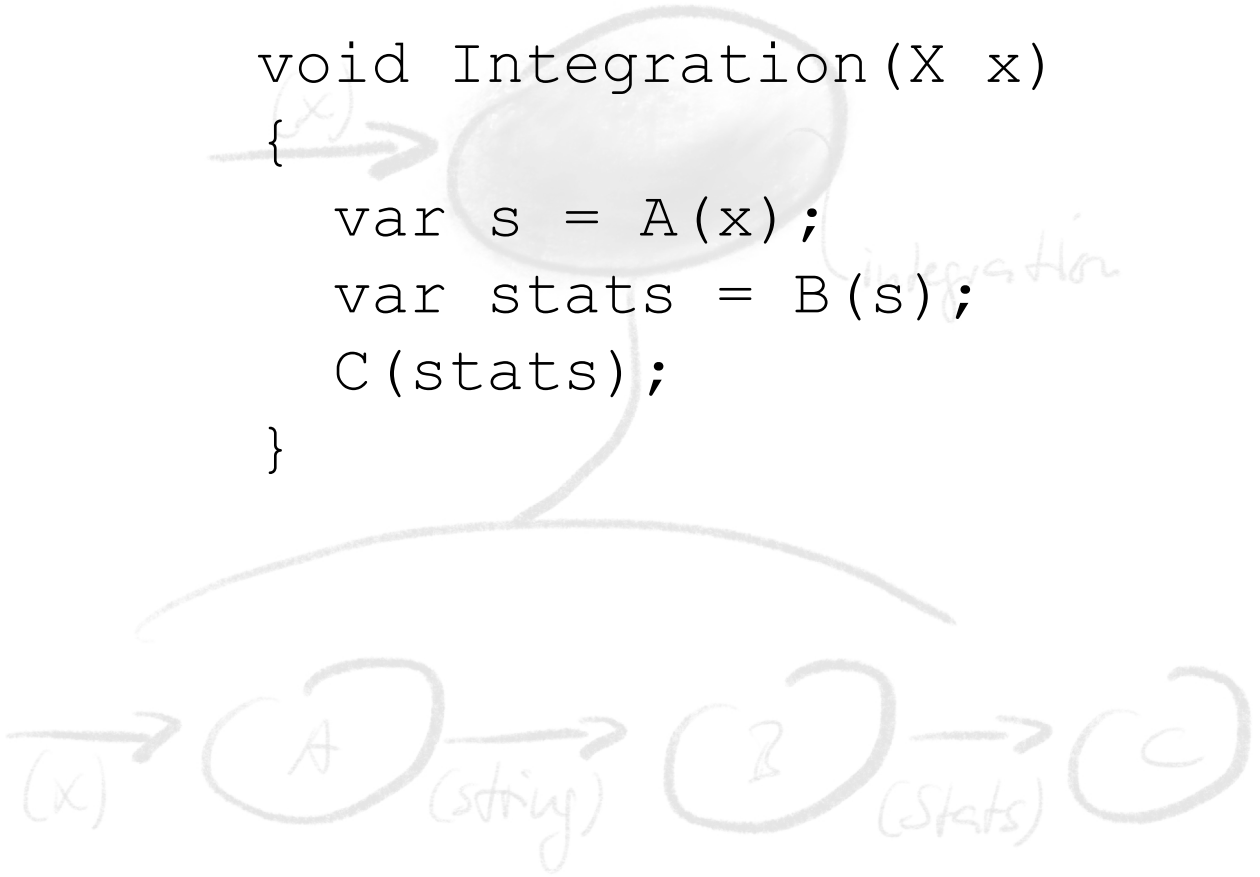
3D flow

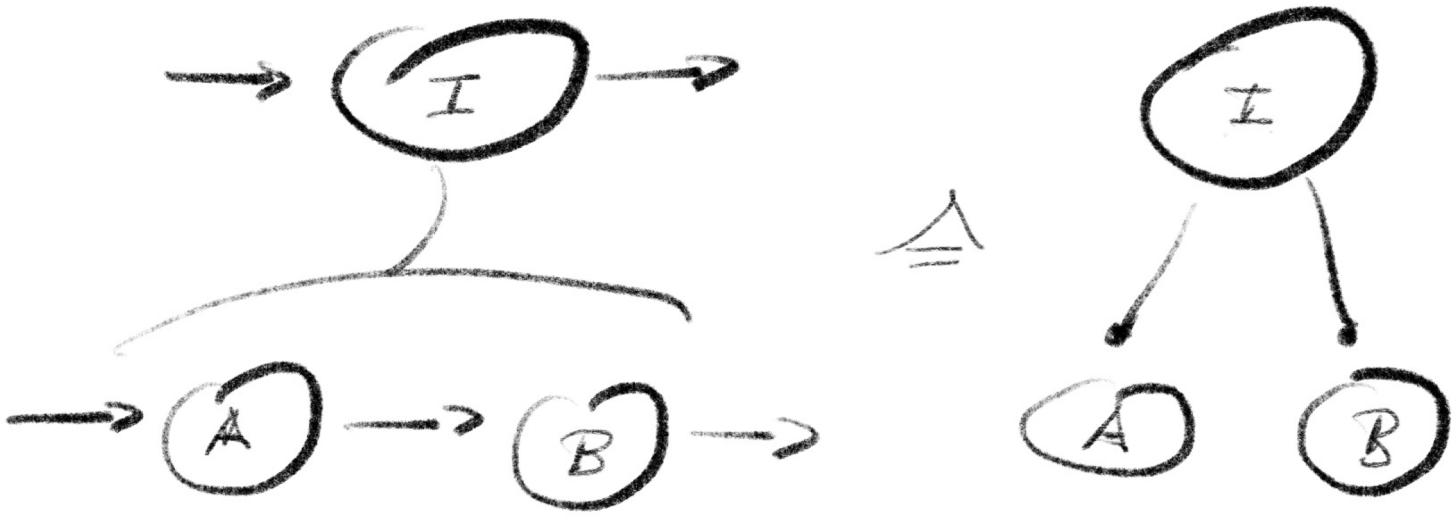


3D flow



```
void Integration(X x)
{
  var s = A(x);
  var stats = B(s);
  C(stats);
}
```

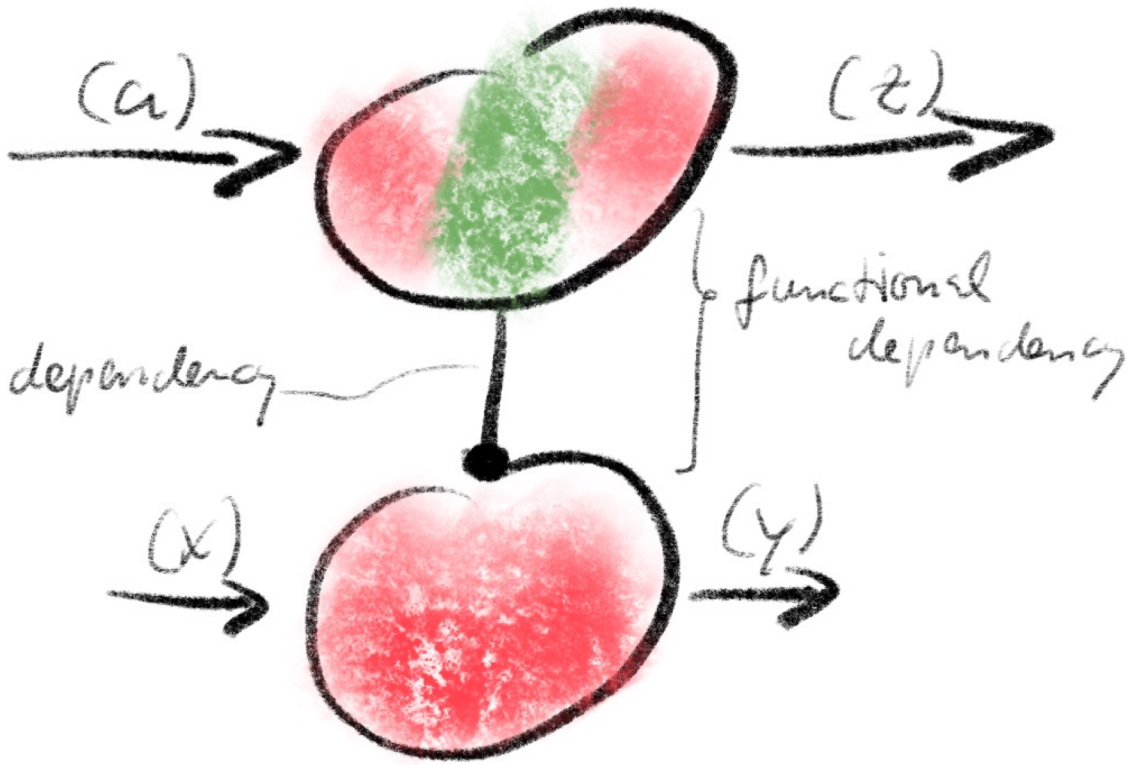



I
A
B
≙
I
A
B

Functions are the new objects

Functional dependencies

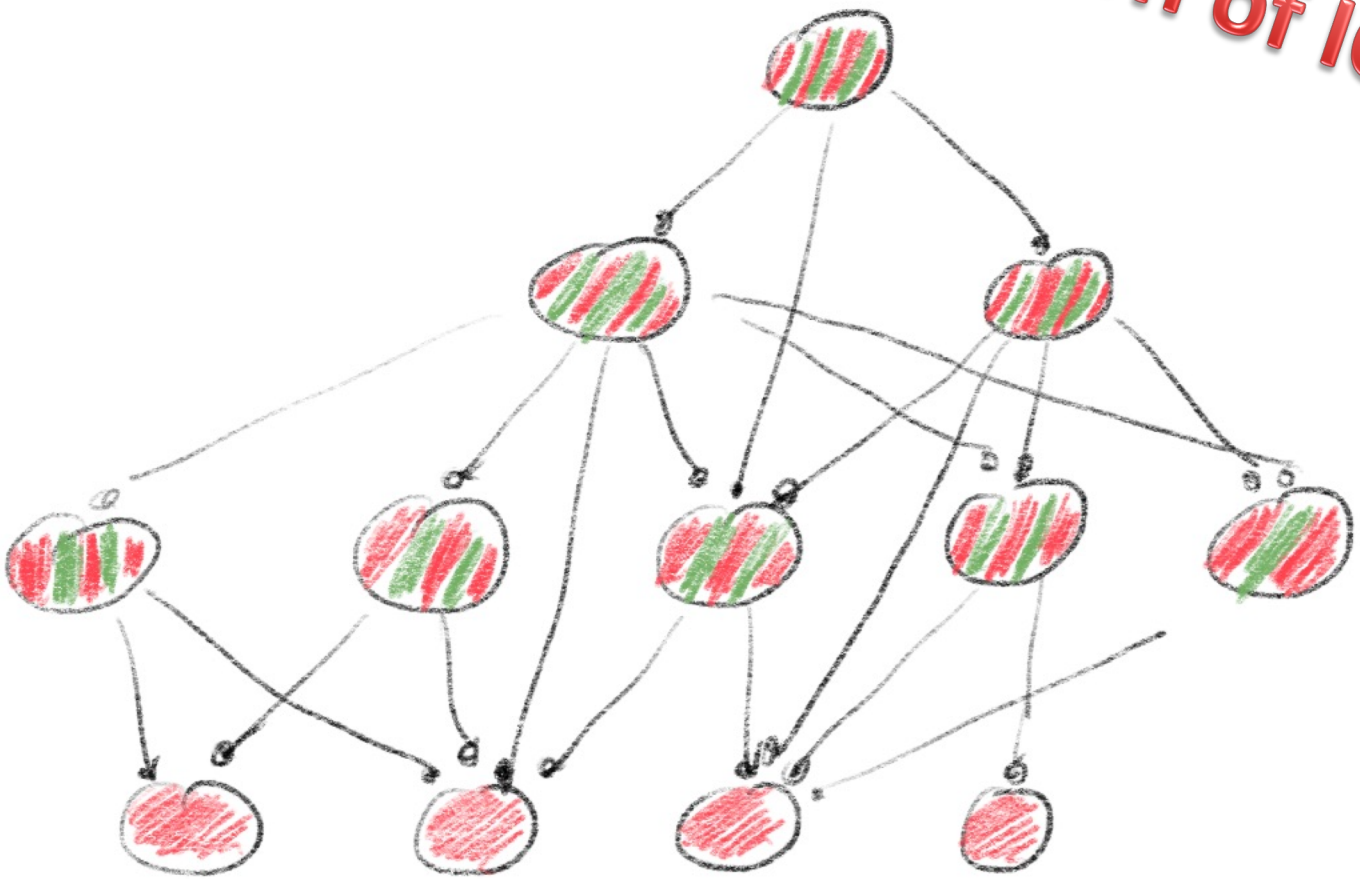
considered harmful!


(a)
(z)
functional dependency
dependency
(x)
(y)

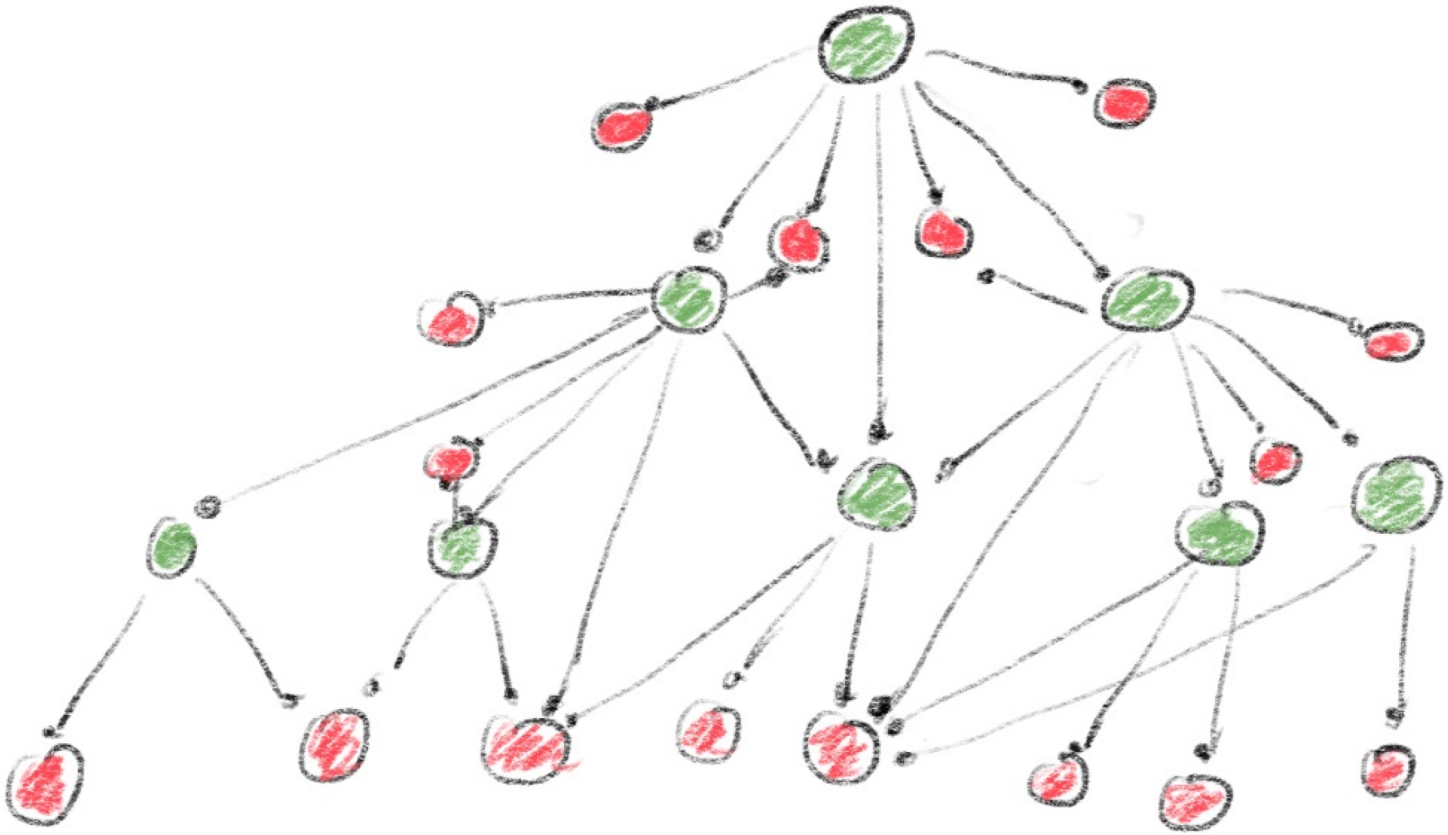
Violation of IOSP!

```
void F(X x) {
  var y = … + sin(x) * …;
  var z = G(y);
  return y * … + z * …;
}
```

Violation of IOSP!

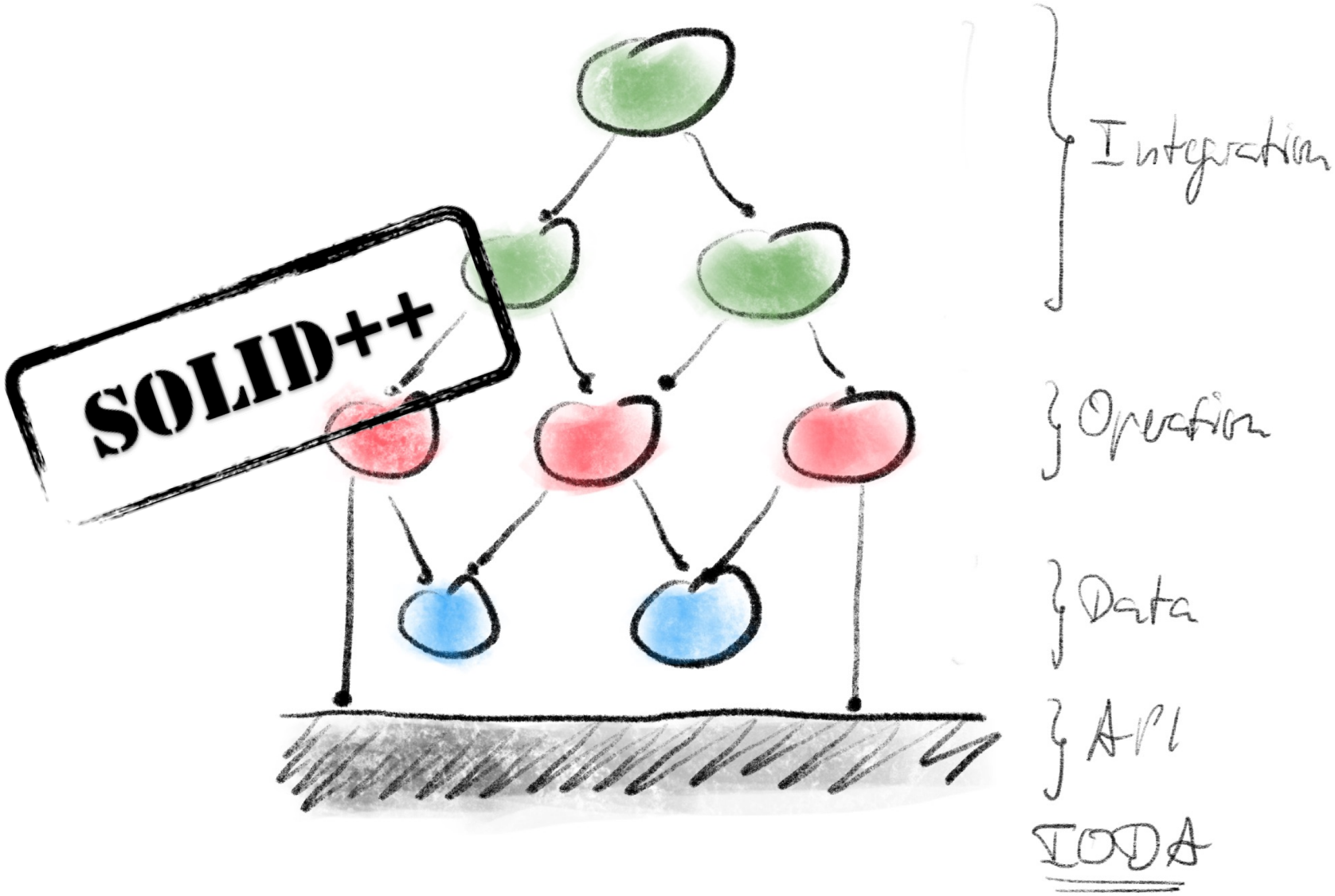


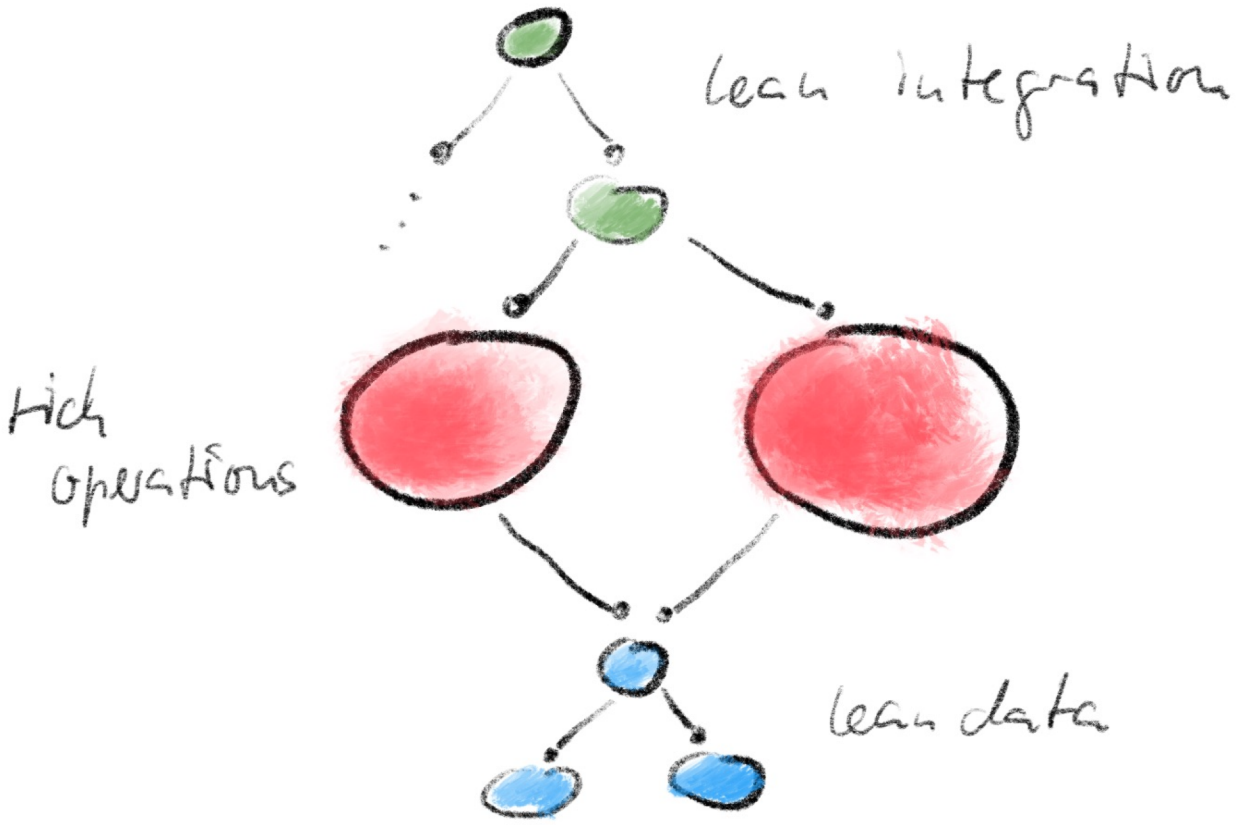
This is your code! 😱

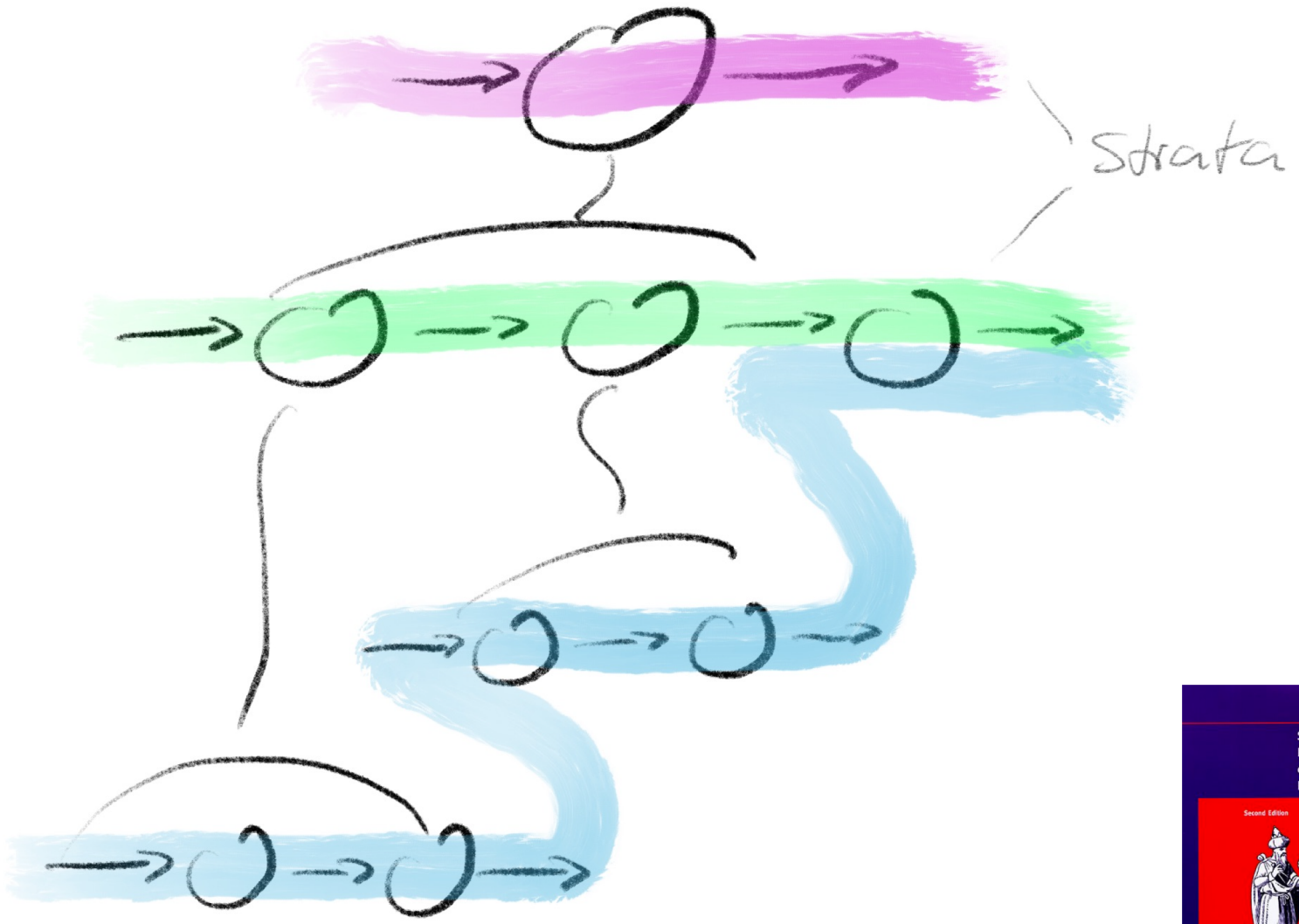


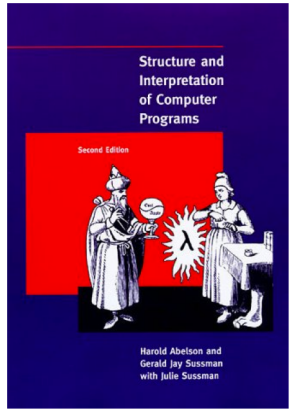
The future of your code! 😍

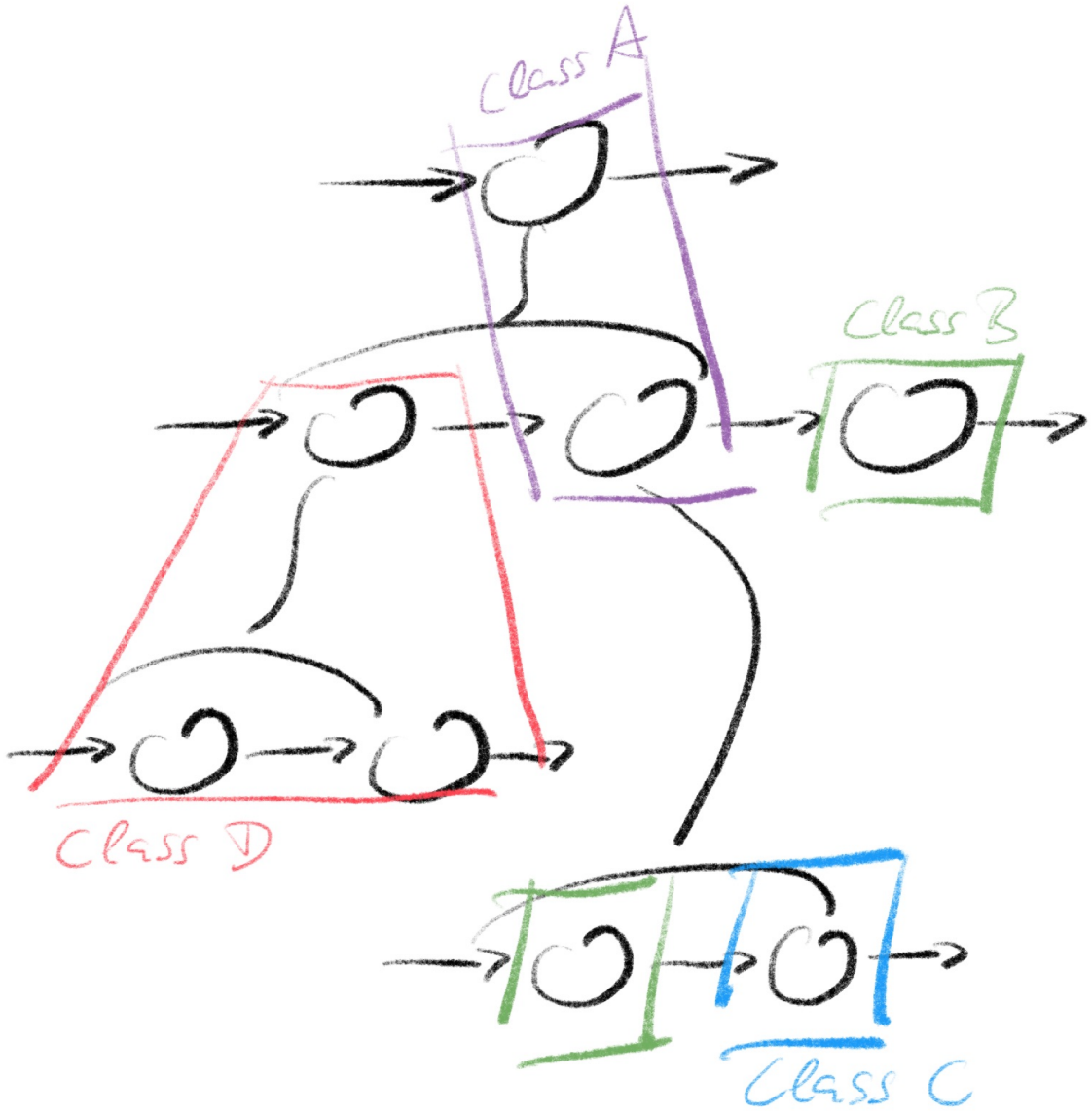
Architectural implications


SOLID++
Integration
Operation
Data
API
IODA


lean integration
rich
operations
lean data


strata


Structure and
Interpretation
of Computer
Programs
Second Edition
Harold Abelson and
Gerald Jay Sussman
with Julie Sussman


Class A
Class B
Class D
Class C

Demo

Hexen hexen

Tun Sie das

Mit Stream.Read() Bytes aus einem Stream lesen

Bei Textdateien funktioniert der Hex-Dumper ausgezeichnet. Aber es gibt ein Problem. Versuchen Sie, mit `File.WriteAllBytes()` ein Array mit Bytes, die alle größer als 127 sind, in eine Datei zu schreiben und diese dann mit dem Dumper ausgeben zu lassen. Alle Bytes werden als »fd« eingelesen! Grund dafür ist, dass **`StreamReader` zur Verarbeitung von Textdaten gedacht ist,** diese aber nur Bytes mit Werten unter 128 enthalten können. Lassen Sie uns das beheben – indem wir die Bytes mit der Methode **`Stream.Read()`** direkt aus dem Stream lesen. Und als Schmankerl werden wir die Sache gleich so ausführen wie in einem echten Hex-Dump-Programm: Wir lassen es einen Dateinamen als **Befehlszeilenargument** entgegennehmen.

Erstellen Sie eine Konsolenanwendung und **geben Sie ihr den Namen `hexdumper`**. Den Code für das Programm sehen Sie auf der gegenüberliegenden Seite. So sieht es aus, wenn Sie das Programm ausführen:

Geben Sie kein Argument an, liefert das Programm eine Fehlermeldung und bricht die Ausführung ab.

Auch wenn Sie einen Dateinamen angeben, den es nicht gibt, bricht das Programm mit einem Fehler ab.

Übergeben Sie einen gültigen Dateinamen, wird ein Hex-Dump des Inhalts auf der Konsole ausgegeben.

```
C:\Users\LS-WIN7>Hexdump.exe
Verwendung: Hexdump Datei
C:\Users\LS-WIN7>Hexdump.exe gibts_nicht.datei
Keine derartige Datei: gibts_nicht.datei
C:\Users\LS-WIN7>Hexdump.exe karte1.dat
```

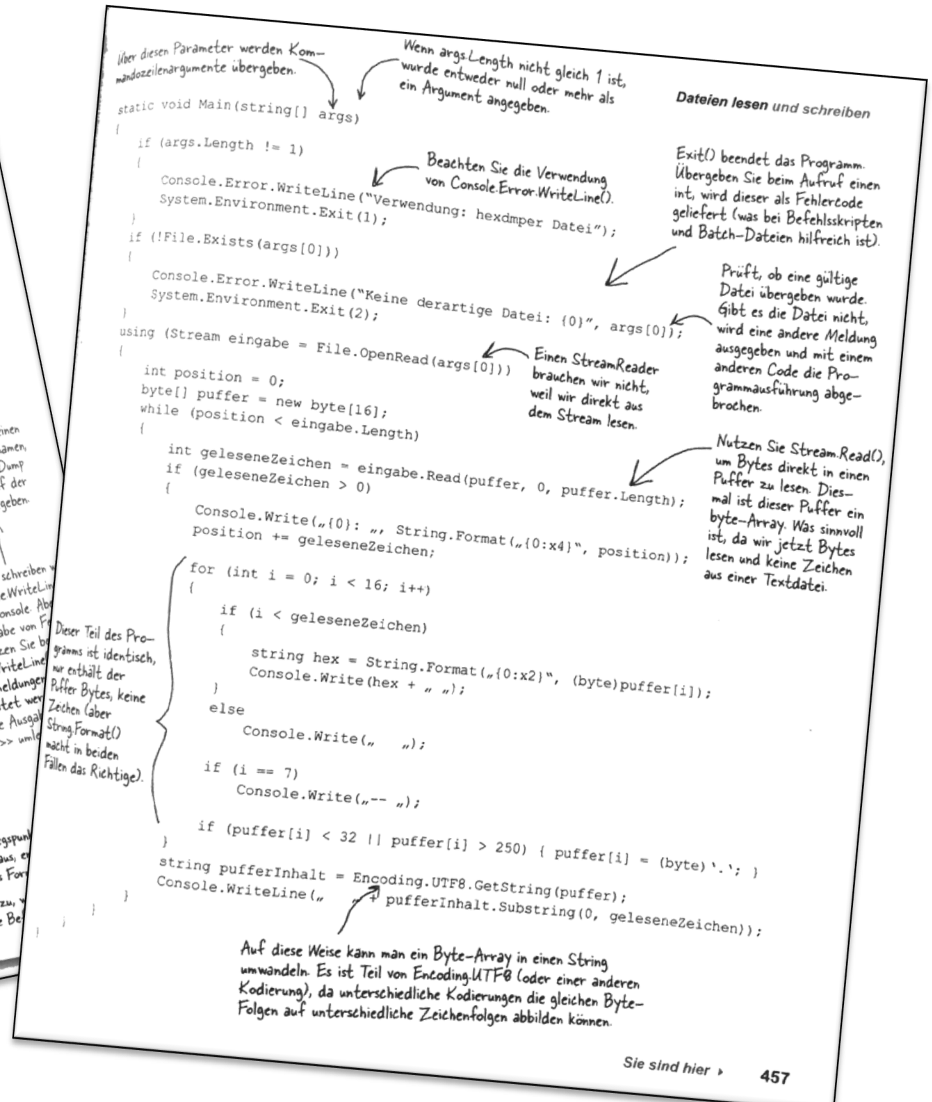
Gewöhnlich schreiben wir mit Console.WriteLine() auf die Konsole. Aber zur Ausgabe von Fehlermeldungen nutzen Sie besser Console.Error.WriteLine(). Fehlermeldungen können umgeleitet werden, wie die Ausgabe oder >> umleiten …

Befehlszeilenargumente

Wenn Sie eine Konsolenanwendung anlegen, erstellt Visual Studio eine Klasse Program mit einem Einstiegspunkt … Befehlszeilenargumenten aus, … Deklaration: **static void Main(string[] args)**. Führen Sie Ihr Programm mit Befehlszeilenargumenten aus, … meter **args** diese Argumente. Das gilt nicht nur für Konsolenanwendungen, sondern auch für Windows For…

Sie können dem Programm auch beim Debugging Kommandozeilenargumente übergeben. Wählen Sie dazu, … gramm in der IDE im Debugger ausführen, im Projekt-Menü >>Eigenschaften…<< und geben Sie die Be… dann im Tab Debugging an.

456 Kapitel 9



Dateien lesen und schreiben

Über diesen Parameter werden Kommandozeilenargumente übergeben.

Wenn args.Length nicht gleich 1 ist, wurde entweder null oder mehr als ein Argument angegeben.

```
static void Main(string[] args)
{
  if (args.Length != 1)
  {
    Console.Error.WriteLine("Verwendung: hexdmper Datei");
    System.Environment.Exit(1);
  }
  if (!File.Exists(args[0]))
  {
    Console.Error.WriteLine("Keine derartige Datei: {0}", args[0]);
    System.Environment.Exit(2);
  }
  using (Stream eingabe = File.OpenRead(args[0]))
  {
    int position = 0;
    byte[] puffer = new byte[16];
    while (position < eingabe.Length)
    {
      int geleseneZeichen = eingabe.Read(puffer, 0, puffer.Length);
      if (geleseneZeichen > 0)
      {
        Console.Write("{0}: ", String.Format("{0:x4}", position));
        position += geleseneZeichen;

        for (int i = 0; i < 16; i++)
        {
          if (i < geleseneZeichen)
          {
            string hex = String.Format("{0:x2}", (byte)puffer[i]);
            Console.Write(hex + " ");
          }
          else
            Console.Write("   ");

          if (i == 7)
            Console.Write("-- ");

          if (puffer[i] < 32 || puffer[i] > 250) { puffer[i] = (byte)'.'; }
        }
        string pufferInhalt = Encoding.UTF8.GetString(puffer);
        Console.WriteLine("   " + pufferInhalt.Substring(0, geleseneZeichen));
      }
    }
  }
}
```

Beachten Sie die Verwendung von Console.Error.WriteLine().

Exit() beendet das Programm. Übergeben Sie beim Aufruf einen int, wird dieser als Fehlercode geliefert (was bei Befehlsskripten und Batch-Dateien hilfreich ist).

Prüft, ob eine gültige Datei übergeben wurde. Gibt es die Datei nicht, wird eine andere Meldung ausgegeben und mit einem anderen Code die Programmausführung abgebrochen.

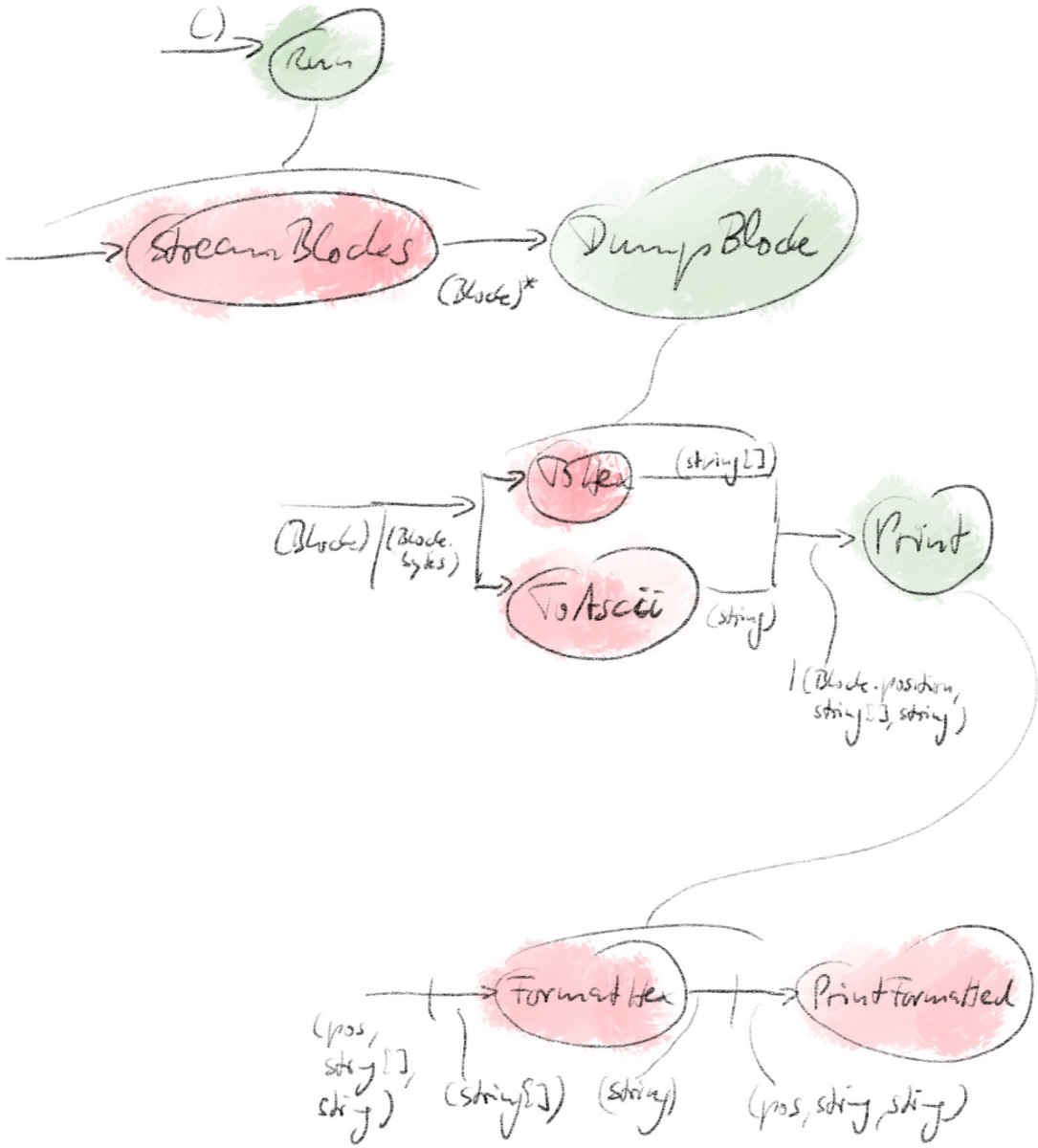
Einen StreamReader brauchen wir nicht, weil wir direkt aus dem Stream lesen.

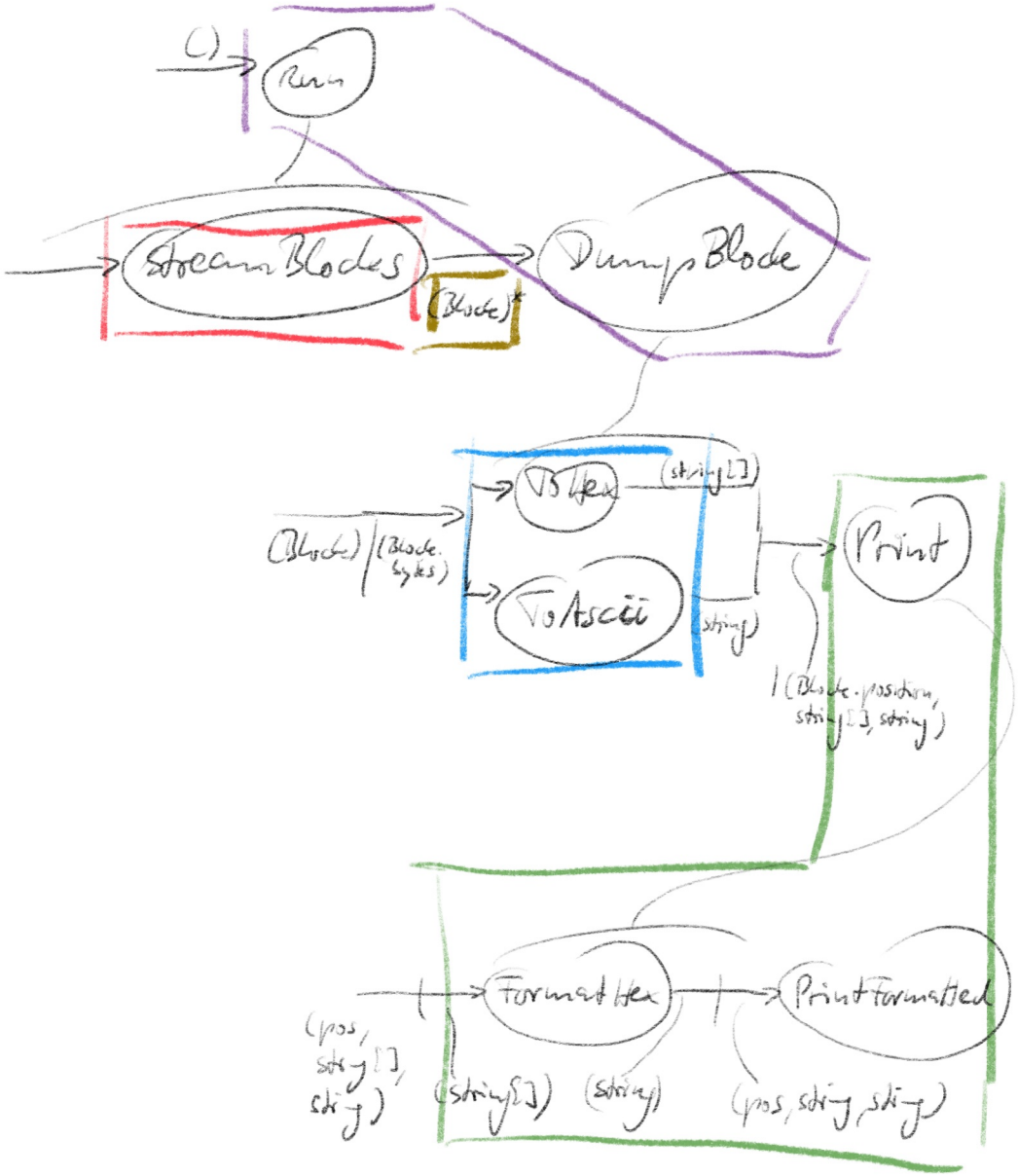
Nutzen Sie Stream.Read(), um Bytes direkt in einen Puffer zu lesen. Diesmal ist dieser Puffer ein byte-Array. Was sinnvoll ist, da wir jetzt Bytes lesen und keine Zeichen aus einer Textdatei.

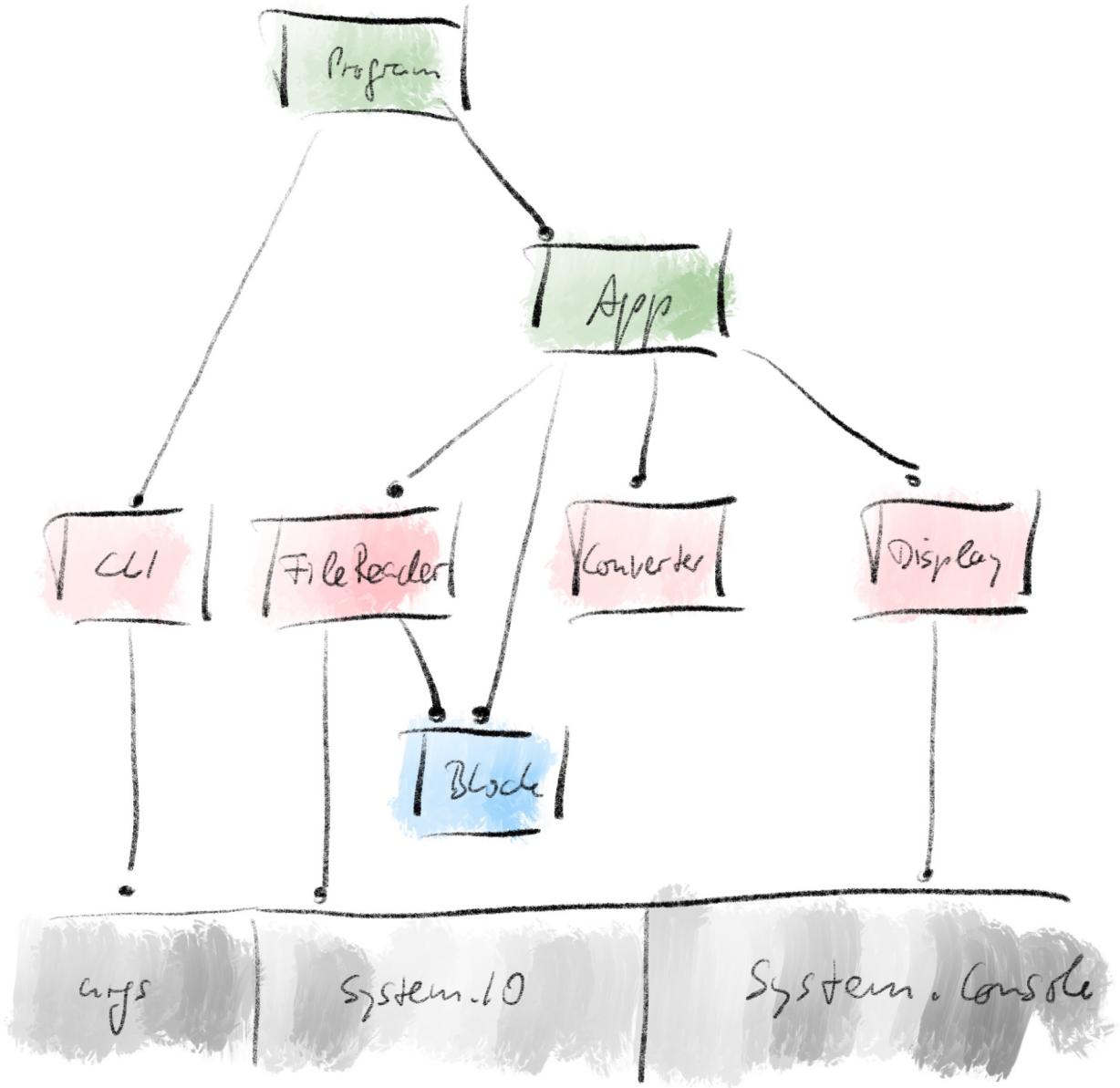
Dieser Teil des Programms ist identisch, nur enthält der Puffer Bytes, keine Zeichen (aber String.Format() macht in beiden Fällen das Richtige).

Auf diese Weise kann man ein Byte-Array in einen String umwandeln. Es ist Teil von Encoding.UTF8 (oder einer anderen Kodierung), da unterschiedliche Kodierungen die gleichen Byte-Folgen auf unterschiedliche Zeichenfolgen abbilden können.

Sie sind hier ▸ 457


()
Run
StreamBlocks
(Block)*
DumpBlock
(Block)
(Block.bytes)
ToHex
ToAscii
(string[])
(string)
Print
(Block.position, string[], string)
FormatHex
PrintFormatted
(pos, string[], string)
(string[])
(string)
(pos, string, string)


()
Run
StreamBlocks
(Block)*
DumpBlock
(Block)
(Block.bytes)
ToHex
ToAscii
(string[])
(string)
Print
(Block.position, string[], string)
FormatHex
PrintFormatted
(pos, string[], string)
(string[])
(string)
(pos, string, string)


Program
App
CLI
FileReader
Converter
Display
Block
args
System.IO
System.Console

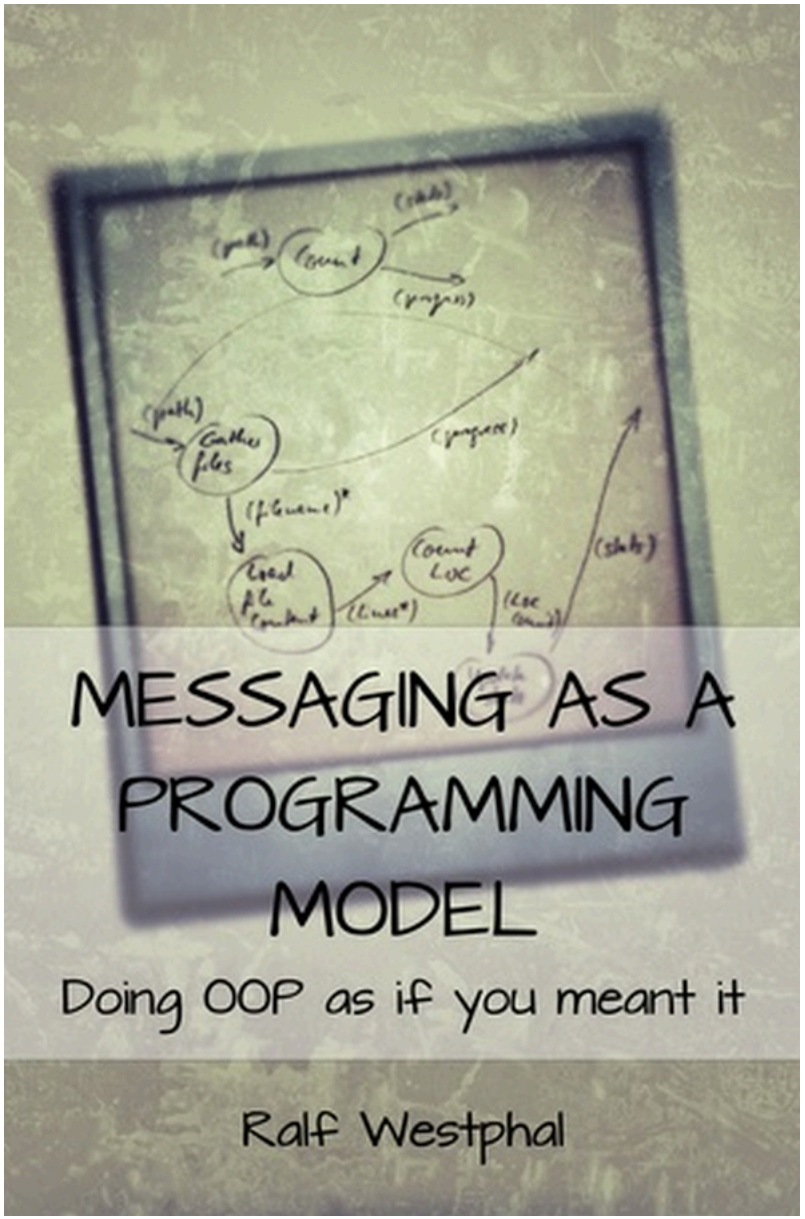

„I invented the term object-oriented, and I can tell you that C++ wasn't what I had in mind.“

Classes are secondary

- share data between „objects“ (master data)
- achieve polymorphy (master architectura decisions)
- aggregate functions (master number of „moving parts“)
- hide integration (master detail)

Summary

- Object-orientation (OO) today is suffering from massive functional dependencies.
- Going back to the roots of the term OO provides a fresh ideas for how to structure code differently for…
  - easier testing
  - less accidentally complexity
  - Increased understandability
- …and it puts behavior first!



https://leanpub.com/messaging_as_a_programming_model

Resources

- Ralf Westphal, Messaging as a Programming Model, https://leanpub.com/messaging_as_a_programming_model
  - an eBook made from: Ralf Westphal, OOP as if you meant it, http://geekswithblogs.net/theArchitectsNapkin/archive/2013/09/08/oop-as-if-you-meant-it.aspx
- Dr. Alan Kay on the Meaning of „Object-Oriented Programming“, http://userpage.fu-berlin.de/~ram/pub/pub_jf47ht81Ht/doc_kay_oop_en
- Alan Kay´s Definition of Object Oriented, http://c2.com/cgi/wiki?AlanKaysDefinitionOfObjectOriented

Speaker

- Ralf Westphal is a freelance consultant, project coach, and trainer on software architectural topics as well as software team organization.
- He is the co-founder of the German“ Clean Code Developer" initiative to increase software quality, currently counting more than 4300 members since 2009.