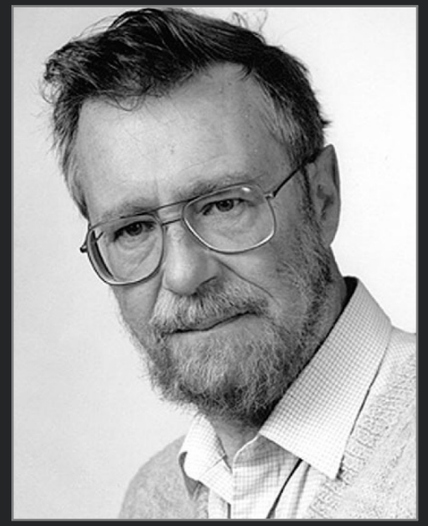
Object Thinking

and Domain Driven Design

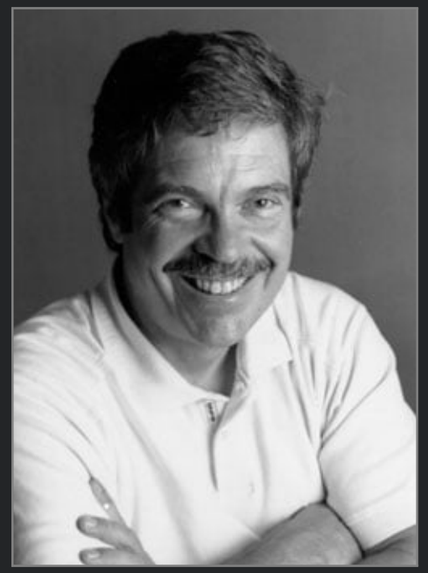
Yegor Bugayenko

Lecture #5 out of 16
90 minutes



“Object-oriented programs are offered as alternatives to correct ones”

— Edsger W. Dijkstra (1989)



“I invented the term *object-oriented*, and I can tell you I did not have C++ in mind”

— Alan Kay (1997)

“Object-oriented programming offers a sustainable way to write spaghetti code”

— Paul Graham (2003)

“C++ is a horrible language. C++ leads to really, really bad design choices. ... idiotic object model crap.”

— Linus Torvalds (2007)

- The Philosophy of OOP
- What is an Object?
- Three Most Evil Parts of OOP
- Domain Driven Design
- Elegant Objects
- Books, Venues, Call-to-Action

Chapter #1:

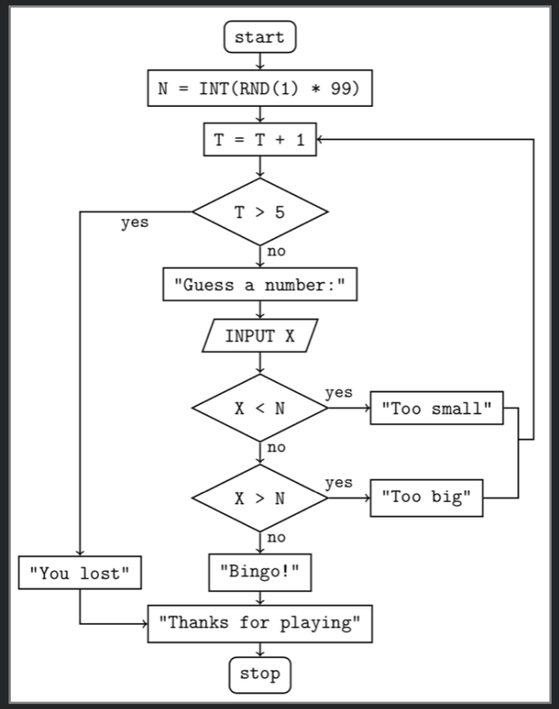
The Philosophy of OOP

The Era of GOTO

```
10 N = INT(RND(1) * 100)
20 T = T + 1
30 IF T > 5 THEN GOTO 120
40 PRINT "Guess a number in 0..99 range: "
50 INPUT X
60 IF X < N THEN PRINT "Too small."
70 IF X > N THEN PRINT "Too big."
80 IF X = N THEN GOTO 100
90 GOTO 20
100 PRINT "Bingo!"
110 GOTO 130
120 PRINT "You lost, sorry. It was: ", N
130 PRINT "Thanks for playing with me!"
```

Structured Programming

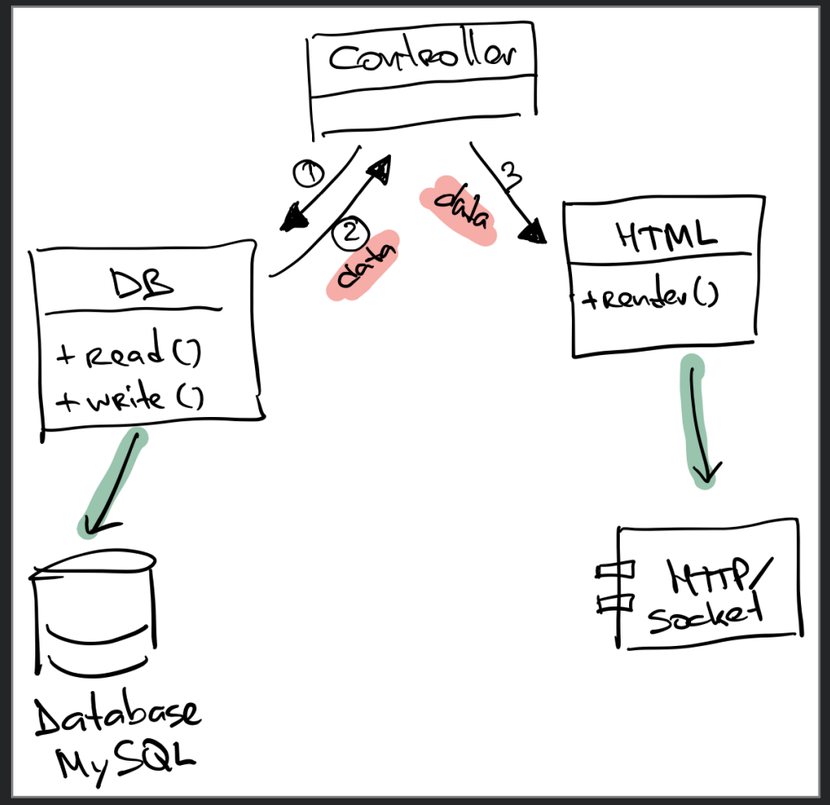
```
Uses sysutils;
Var
  N, X, T: Integer;
Begin
  Randomize;
  N := Random(100);
  T := 0;
  While True Do Begin
    T := T + 1;
    If (T > 5) Then Begin
      Writeln('You lost, sorry. It was: ' + IntToStr(N));
      Break;
    End;
    Write('Guess a number in 0..99 range: ');
    ReadLn(X);
    If (X = N) Then Begin
      Writeln('Bingo!');
      Break;
    End;
    If X < N Then
      Writeln('Too small.');
    If X > N Then
      Writeln('Too big.');
  End;
  Writeln('Thanks for playing with me!');
End.
```



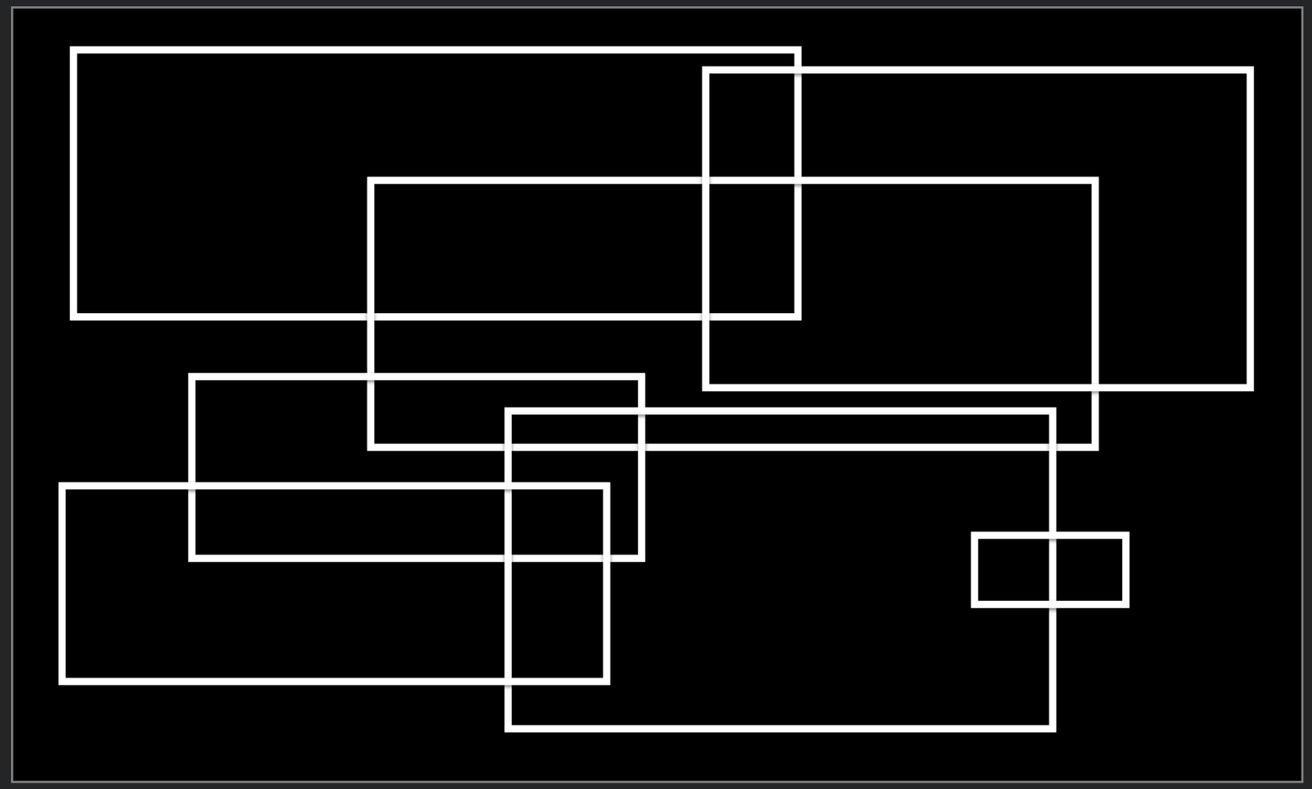
Procedural Programming

```
atomic_long_set(&acct->count, 1);
init_fs_pin(&acct->pin, acct_pin_kill);
acct->file = file;
acct->needcheck = jiffies;
acct->ns = ns;
mutex_init(&acct->lock);
INIT_WORK(&acct->work, close_work);
init_completion(&acct->done);
mutex_lock_nested(&acct->lock, 1);
pin_insert(&acct->pin, mnt);
rcu_read_lock();
old = xchg(&ns->bacct, &acct->pin);
mutex_unlock(&acct->lock);
pin_kill(old);
mnt_drop_write(mnt);
mntput(mnt);
```

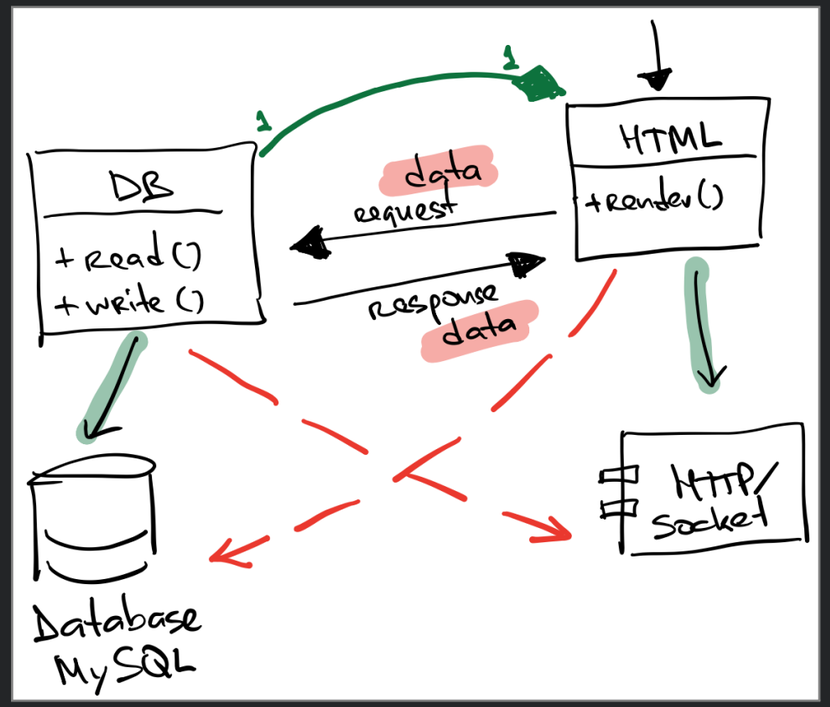
Object-Oriented Programming ... Not!



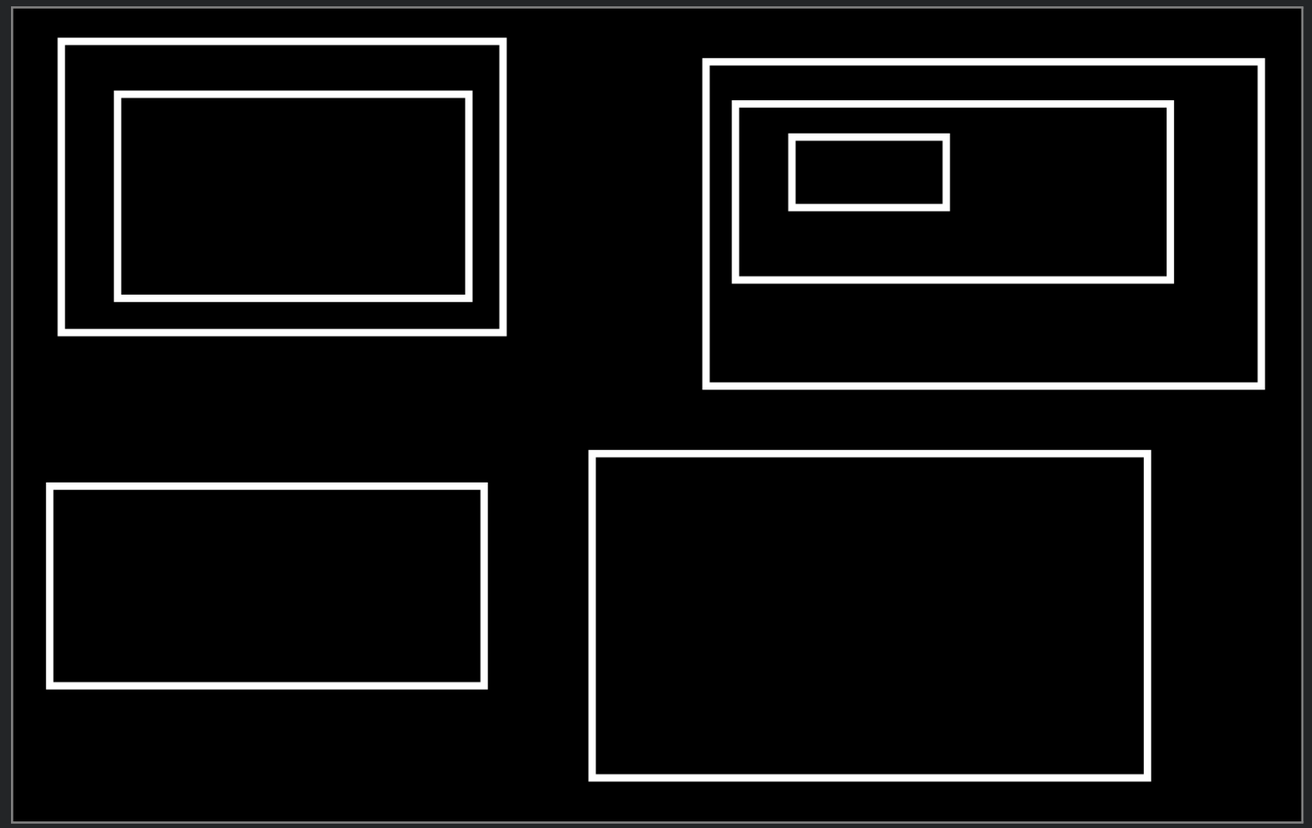
Spaghetti OOP Code



OOP Done Right



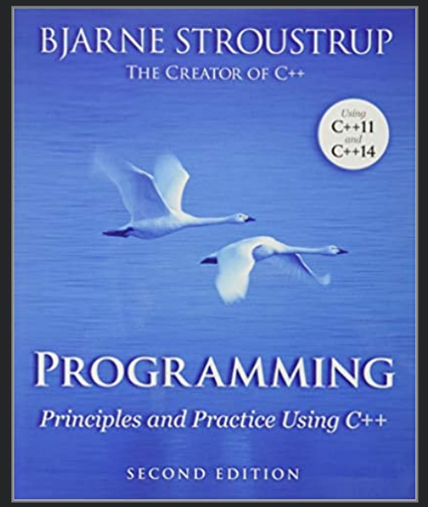
Elegant OOP Code



Chapter #2:

What is an Object?

C++



"An object is some memory that holds a value of some type"

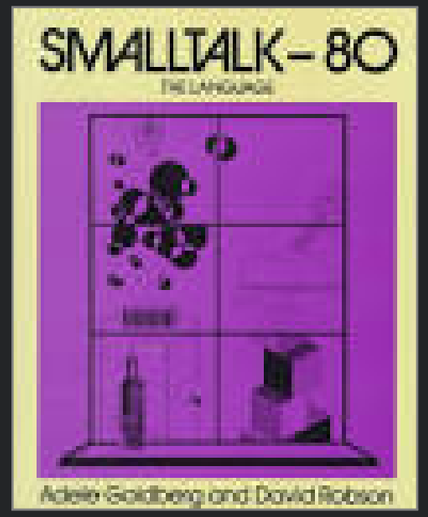
— *Programming Principles and Practice Using C++* by Bjarne Stroustrup

Wiki

“Objects may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods”

— Wikipedia

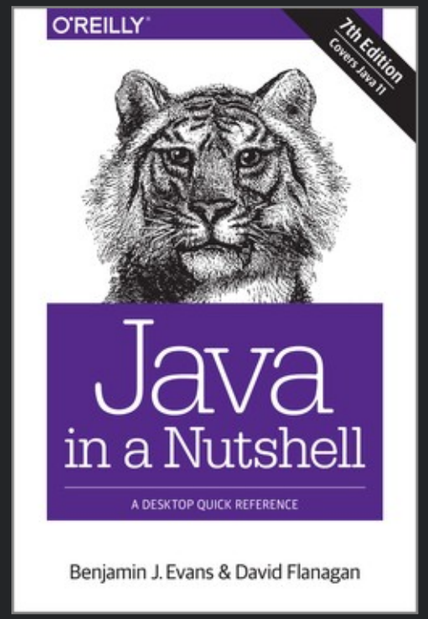
Smalltalk



"An object consists of some private memory and a set of operations"

— *Smalltalk-80: The Language and Its Implementation* by Adele Goldberg et al., p. 6

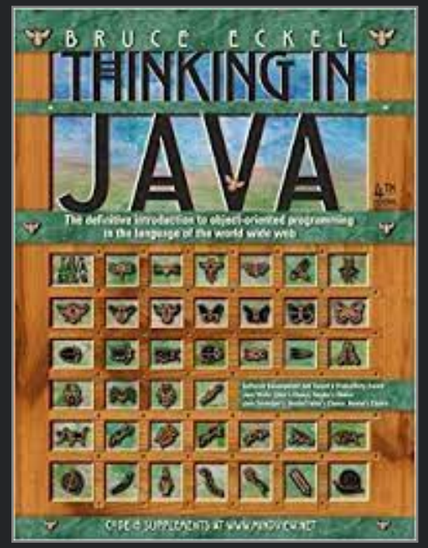
Java



"A class is a collection of data fields that hold values and methods that operate on those values"

— *Java in a Nutshell* by Ben Evans

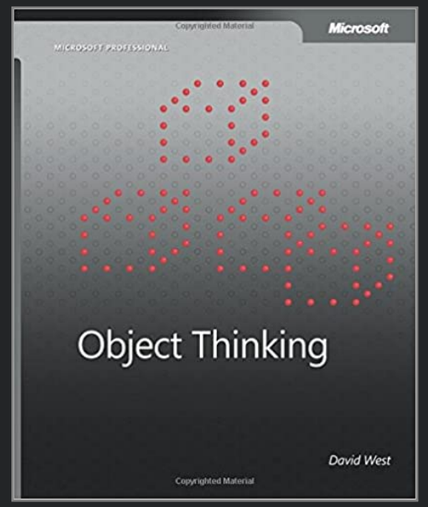
Eckel



“Each object looks quite a bit like a little computer — it has a state, and it has operations that you can ask it to perform”

— *Thinking in Java* by Bruce Eckel, p. 16

West



“An object is the equivalent of the quanta from which the universe is constructed”

— *Object Thinking* by David West, p. 66

Chapter #3:

Three Most Evil Parts of OOP

1. Static Methods

```
void transform(File in, File out) {
  Collection<String> src = FileUtils.readLines(in, "UTF-8");
  Collection<String> dest = new ArrayList<>(src.size());
  for (String line : src) {
    dest.add(line.trim());
  }
  FileUtils.writeLines(out, dest, "UTF-8");
}
```

```
void transform(File in, File out) {
  Collection<String> src = new Trimmed(
    new FileLines(new UnicodeFile(in))
  );
  Collection<String> dest = new FileLines(
    new UnicodeFile(out)
  );
  dest.addAll(src);
}
```

https://www.yegor256.com/2014/05/05/oop-alternative-to-utility-classes.html →

2. Mutability vs. Immutability

```
Email email = new SimpleEmail();
email.setHostName("smtp.googlemail.com");
email.setSmtpPort(465);
email.setAuthenticator(new DefaultAuthenticator("user", "pwd"));
email.setFrom("yegor256@gmail.com", "Yegor Bugayenko");
email.addTo("dude@jcabi.com");
email.setSubject("how are you?");
email.setMsg("Dude, how are you?");
email.send();
```

```
Postman postman = new Postman.Default(
  new SMTP("smtp.googlemail.com", 465, "user", "pwd")
);
Envelope envelope = new Envelope.MIME(
  new Array<Stamp>(
    new StSender("Yegor Bugayenko <yegor256@gmail.com>"),
    new StRecipient("dude@jcabi.com"),
    new StSubject("how are you?")
  ),
  new Array<Enclosure>(
    new EnPlain("Dude, how are you?")
  )
);
postman.send(envelope);
```

https://www.yegor256.com/2014/11/07/how-immutability-helps.html →

Benefits of Immutability

- immutable objects are simpler to construct, test, and use
- truly immutable objects are always thread-safe
- they help to avoid temporal coupling
- their usage is side-effect free (no defensive copies)
- identity mutability problem is avoided
- they always have failure atomicity
- they are much easier to cache
- they prevent NULL references, which are bad

w.yegor256.co
09/objects-sh
mmutable.html
→

3. NULL References

```
public Employee getByName(String name) {
  int id = database.find(name);
  if (id == 0) {
    return null;
  }
  return new Employee(id);
}
```

w.yegor256.co
13/why-null-i
s-bad.html →

Null References, The Billion Dollar Mistake
presentation by Tony Hoare, watch here.

NULL Object

```
public Employee getByName(String name) {
  int id = database.find(name);
  if (id == 0) {
    return Employee.NOBODY;
  }
  return Employee(id);
}
```

Fail Fast vs. Fail Safe

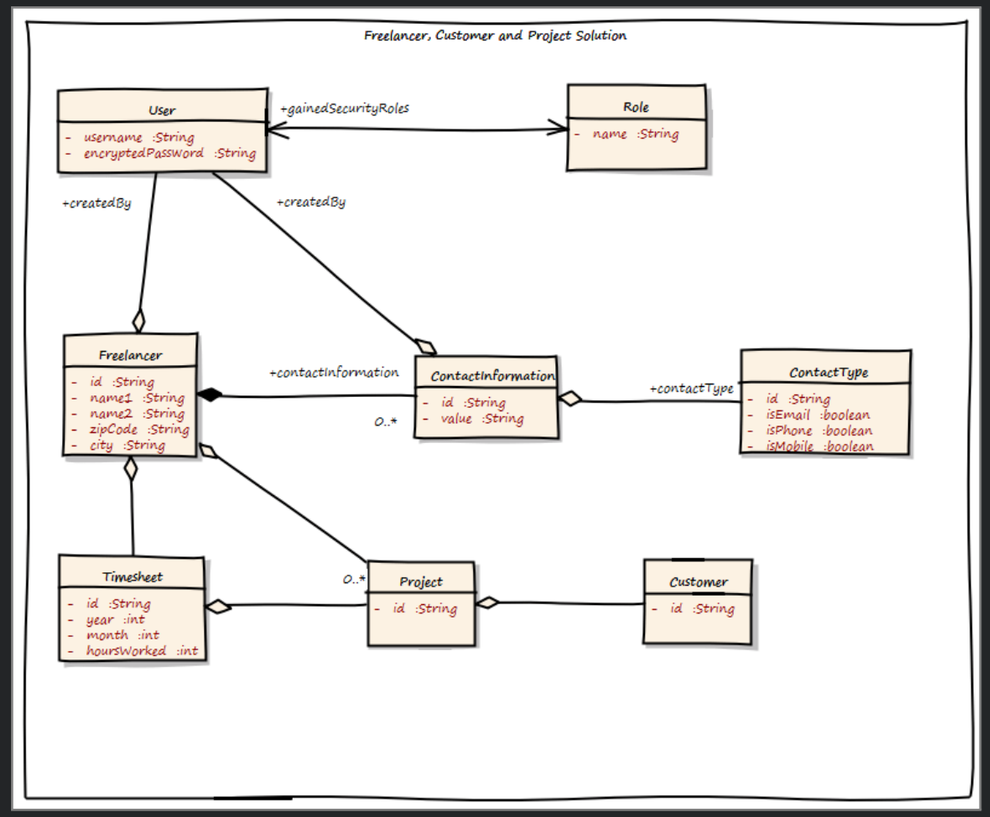
```
public Employee getByName(String name) {
  int id = database.find(name);
  if (id == 0) {
    throw new EmployeeNotFoundException(name);
  }
  return Employee(id);
}
```

https://www.yegor256.com/2015/08/25/fail-fast.html →

Chapter #4:

Domain Driven Design

Names of Objects Done Right



Chapter #5:

Elegant Objects

Elegant Objects (EO)

Started in 2014

Two books, 40+ speeches, 80+ blog posts

30+ frameworks and libraries

50+ fans registered

Six bloggers
e.g. pragmaticobjects.com, g4s8.wtf, amihaiemil.com

Five “Object Thinking” Meetups

https://www.elegantobjects.org →

Object-Oriented Lies
JPoint Student Day
Moscow, Russia, 22-24 April 2016

Java vs. OOP
JavaDay 2016
Minsk, Belarus, 11 June 2016

Java vs. OOP
JavaDay Kyiv 2016
Kyiv, Ukraine, 15 October 2016

What's Wrong With OOP?
RigaDevDays 2017
Riga, Latvia, 15 May 2017

EOLANG: Our New Programming Language

```
[] > calculatesFibonacciNumberWithTail
  eq. > @
    13
    fibonacci 7
  [n] > fibonacci
    if. > @
      n.less 3
      small n
      rec n 1 1
    [n] > small
      if. > @
        n.eq 2
        1
        n
    [n minus1 minus2] > rec
      if. > @
        n.eq 3
        minus1.add minus2
        rec (n.sub 1) (minus1.add minus2) minus1
```

https://www.eolang.org →

If you want to help:

EO to JavaScript/Go/Rust/Ruby compilers

REPL for EO

Static analysis of EO code

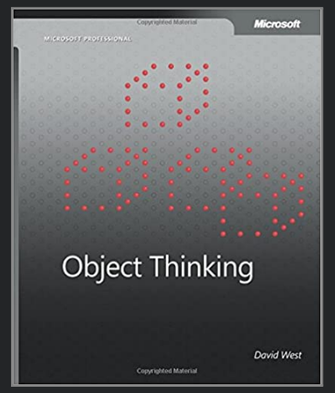
EO integration with Java/C++

Automated refactoring of EO code

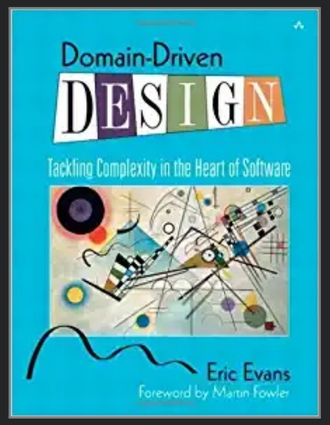
JetBrains plugin for EO

Chapter #6:

Books, Venues, Call-to-Action



"Object Thinking" by David West



"Domain-Driven Design: Tackling Complexity in the Heart of Software" by Eric Evans

"Elegant Objects, vol. 1" by Yegor Bugayenko

"Elegant Objects, vol. 2" by Yegor Bugayenko

Where to publish:

SPLASH: ACM SIGPLAN conference on Systems, Programming, Languages, and Applications

PLDI: ACM SIGPLAN Conference on Programming Language Design and Implementation

POPL: The Annual Symposium on Principles of Programming Languages

Call to Action:

Take `yegor256/hangman` repository and rewrite it in true object-oriented manner, submit a pull request with your changes.

Still unresolved issues:

- How to motivate coders for better OO practices?
- How to create better OO programming languages?
- How to catch bad OO practices automatically?
- How to prove some OO practices are bad?