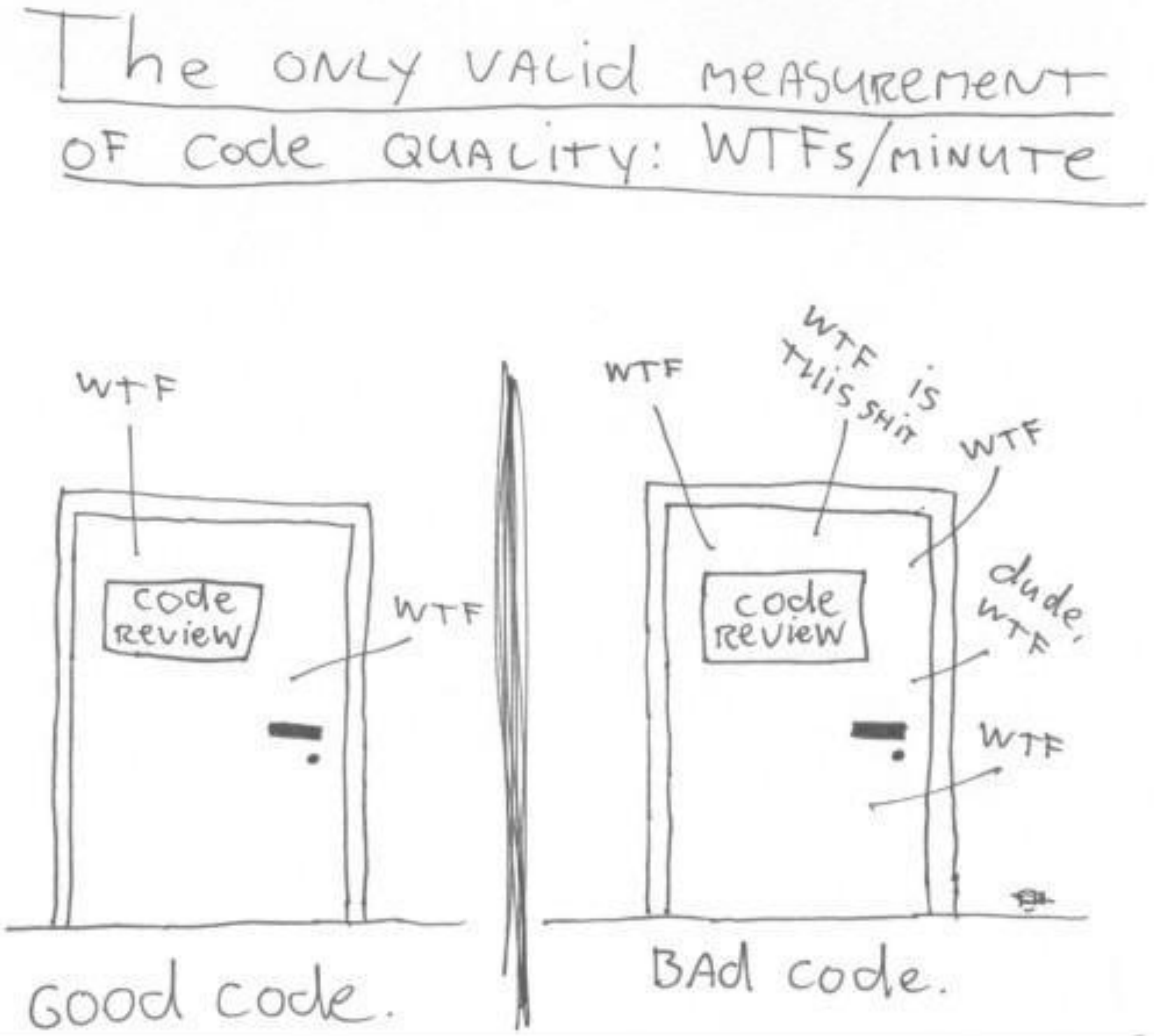
Refactoring

FIUBA

Refactoring


The only valid measurement of code quality: WTFs/minute
WTF
code review
WTF
Good code.
WTF
WTF is this shit
WTF
code review
dude, WTF
WTF
BAd code.

What is Refactoring?

A **technique** for **restructuring** an existing body of **code**, **altering** its **internal structure without changing** its **external behavior**

How to achieve it?

- Unit tests to guarantee the external behavior has not been changed

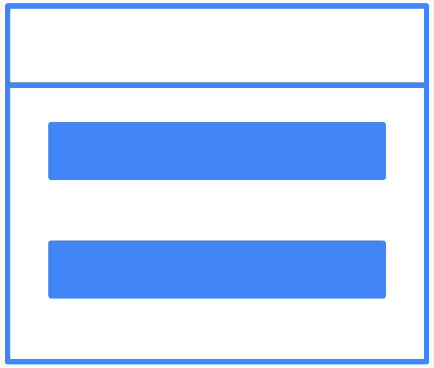
- Applying the proposed refactorings

Refactoring Flow

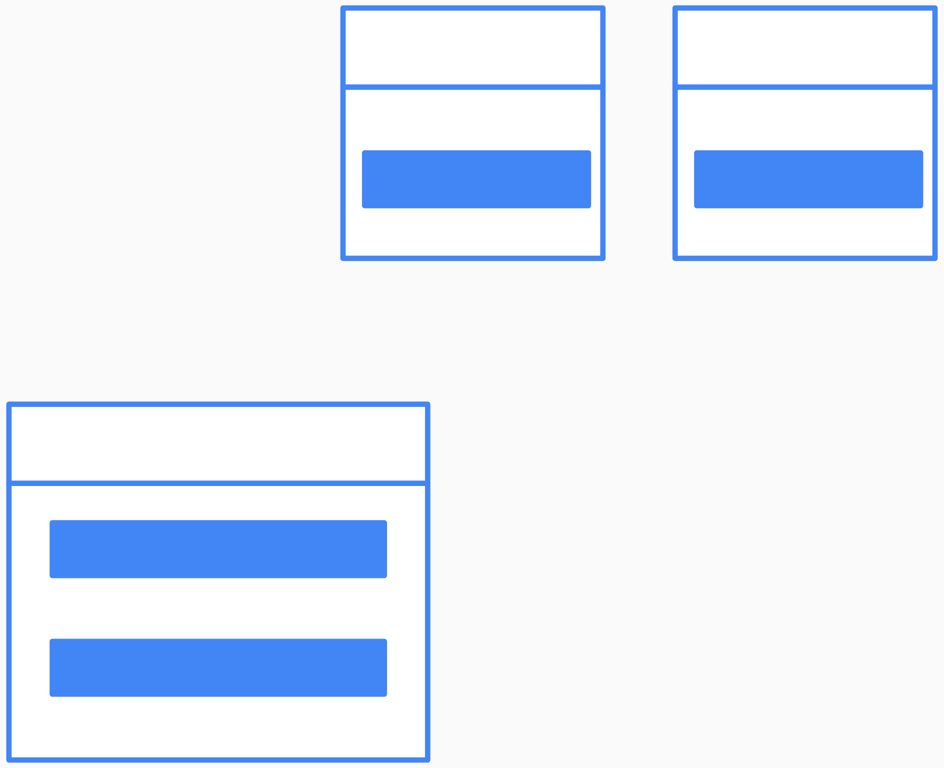
- Ensure all tests pass
- Find code that smells
- Find refactoring
- Apply refactoring

Code Smells

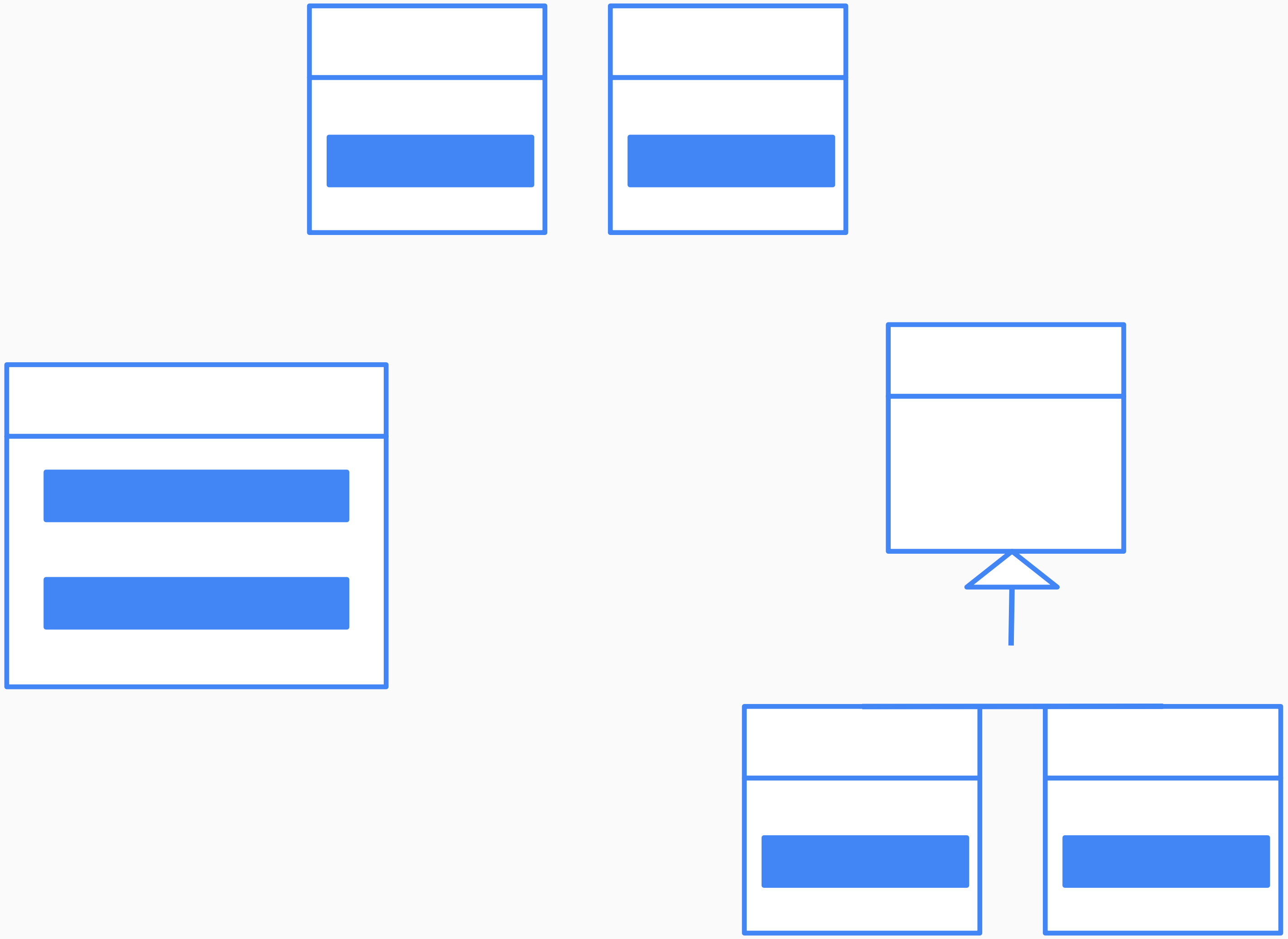
Duplicated Code



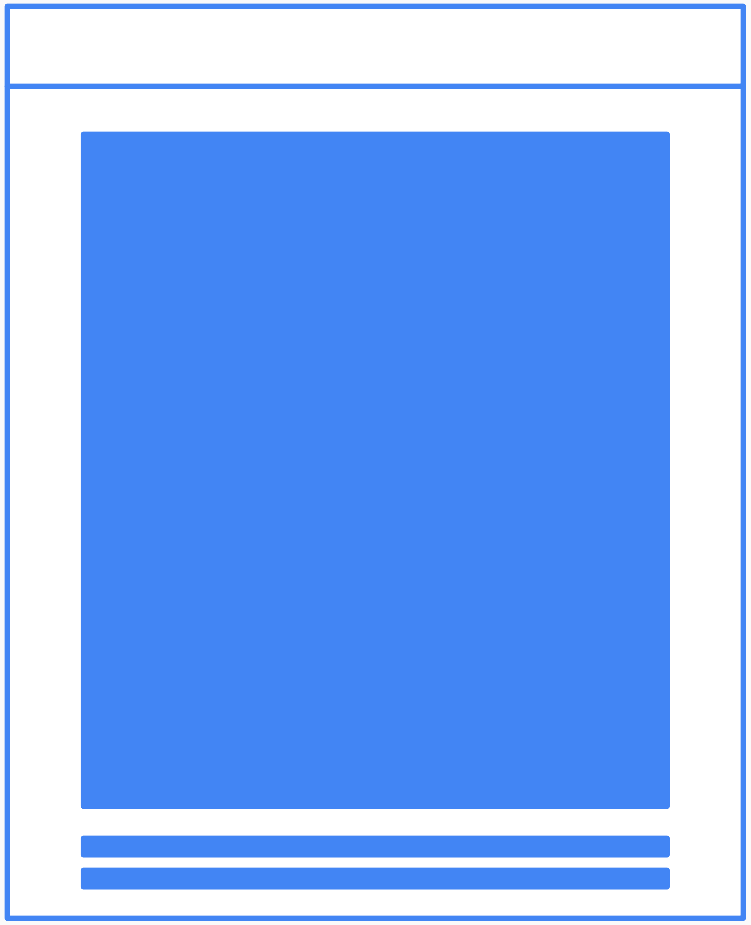
Duplicated Code



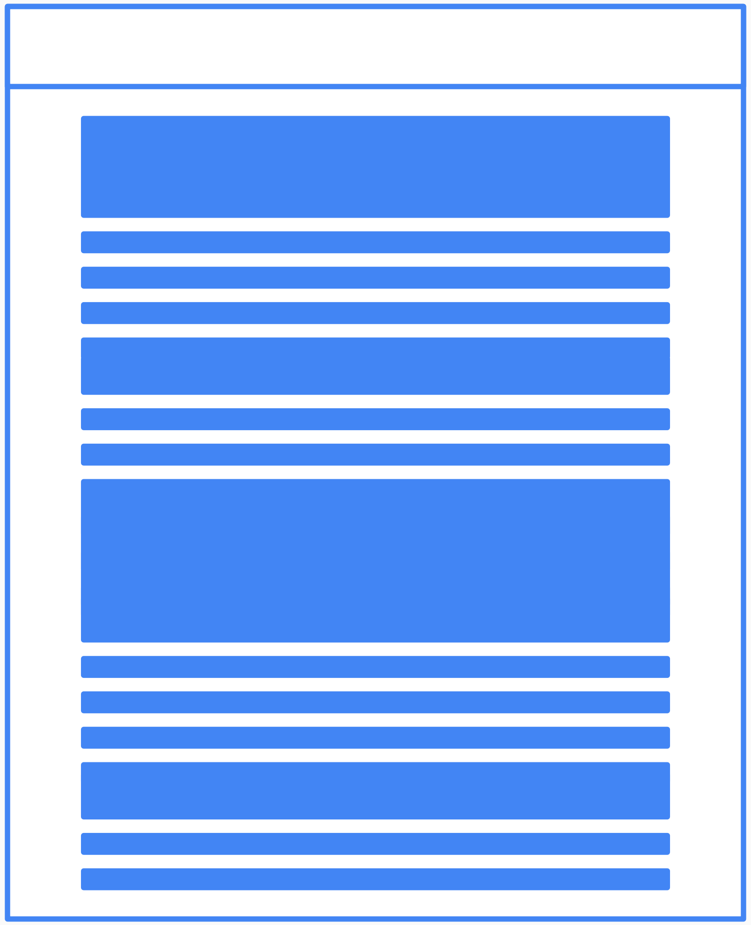
Duplicated Code



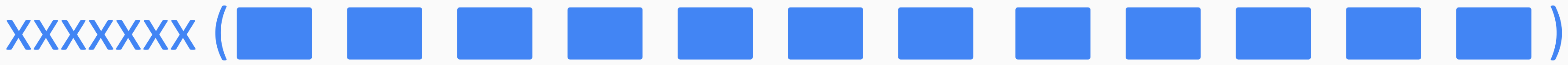
Long Method



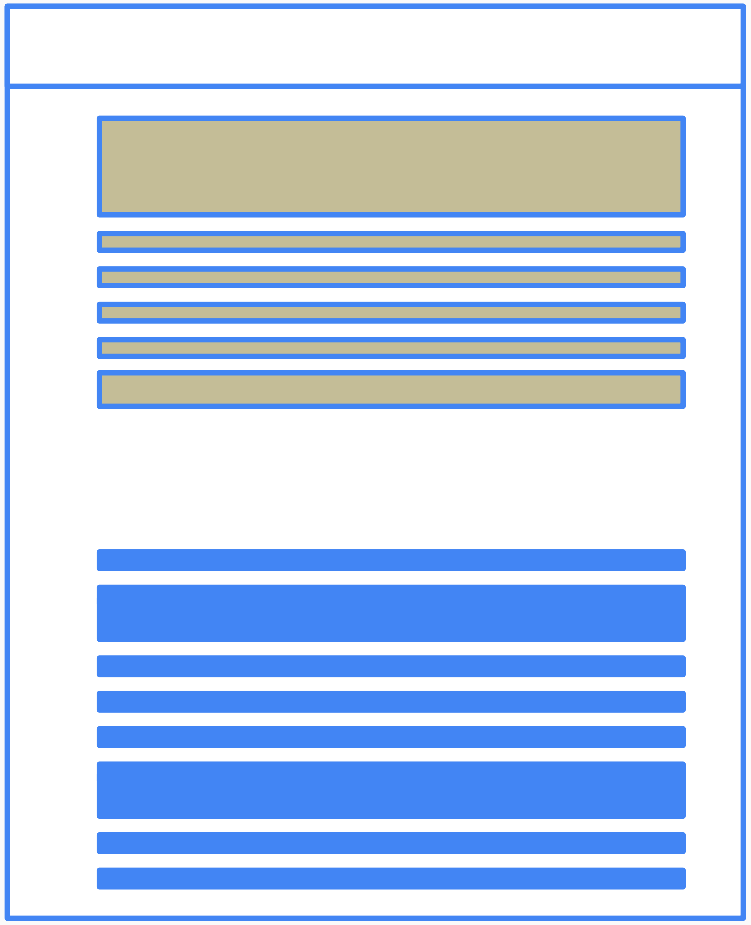
Large Class



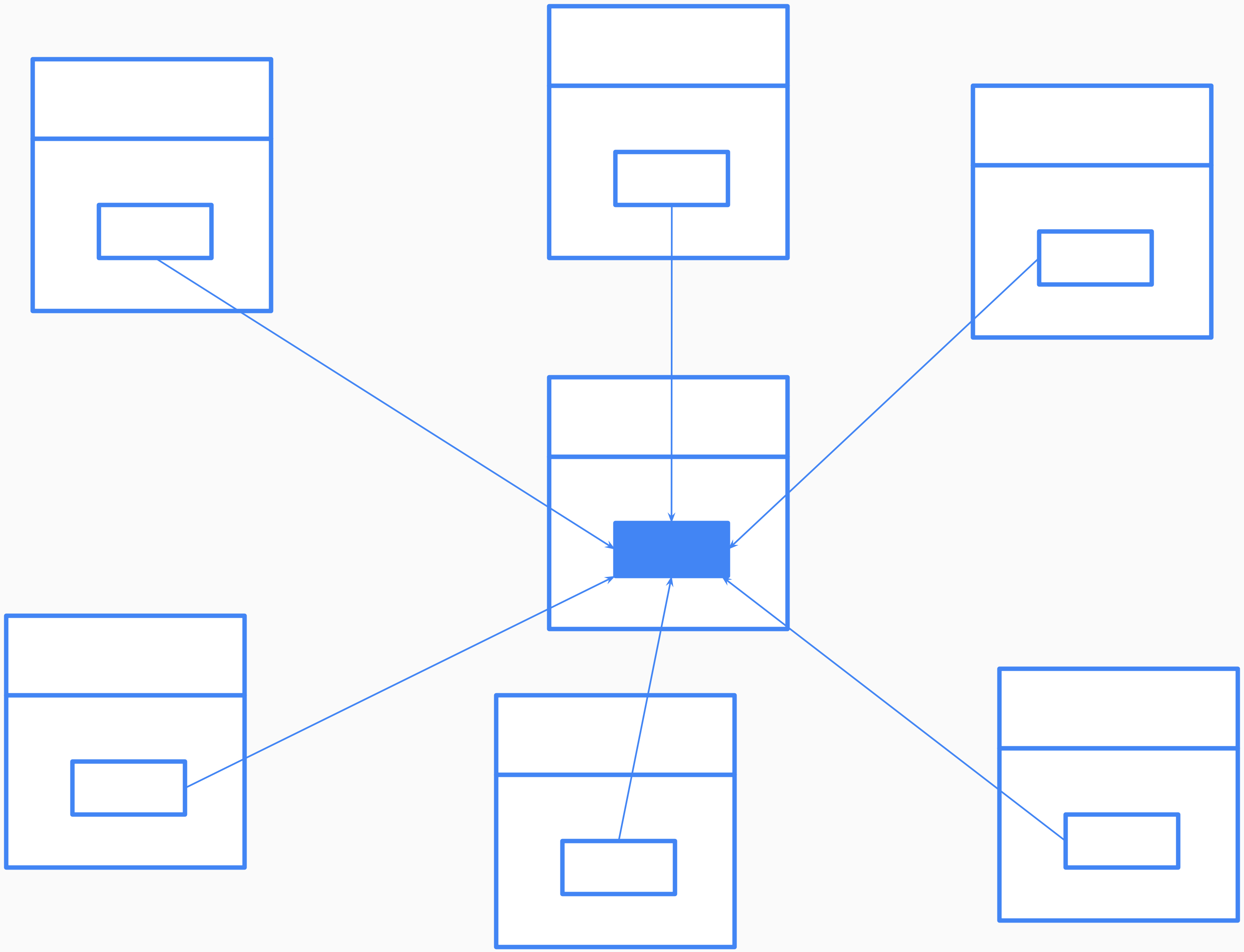
Long parameter list



Divergent Change



Shotgun Surgery



feature envy

```java
class CapitalCalculator {
 ...

 public double capital(Loan loan) {
  if (loan.getExpiry() == null && loan.getMaturity() != null)
   return loan.getCommitment()*loan.duration()*loan.riskFactor();

  if (loan.getExpiry() != null && loan.getMaturity() == null) {
    if (loan.getUnusedPercentage() != 1.0)
       return loan.getCommitment() * loan.getUnusedPercentage() *
                loan.duration() * loan.riskFactor();
    else
       return (loan.outstandingRiskAmount()*loan.duration()
                  * loan.riskFactor())
       + (loan.unusedRiskAmount() * loan.duration()
                  * loan.unusedRiskFactor());
   }

   return 0.0;
 }
 ...
}
```

```
class CapitalCalculator {
 ...

 public double capital(Loan loan) {
  if (loan.getExpiry() == null && loan.getMaturity() != null)
   return loan.getCommitment()*loan.duration()*loan.riskFactor();

  if (loan.getExpiry() != null && loan.getMaturity() == null) {
    if (loan.getUnusedPercentage() != 1.0)
       return loan.getCommitment() * loan.getUnusedPercentage() *
                 loan.duration() * loan.riskFactor();
    else
       return (loan.outstandingRiskAmount()*loan.duration()
                 * loan.riskFactor())
      + (loan.unusedRiskAmount() * loan.duration()
                 * loan.unusedRiskFactor());
   }

   return 0.0;
 }
 ...
}
```

Data Clumps

method1(■ ▲ ◆ ●)

method2(■ ▲ ◆ ●)

method3(■ ▲ ◆ ●)

method4(■ ▲ ◆ ●)

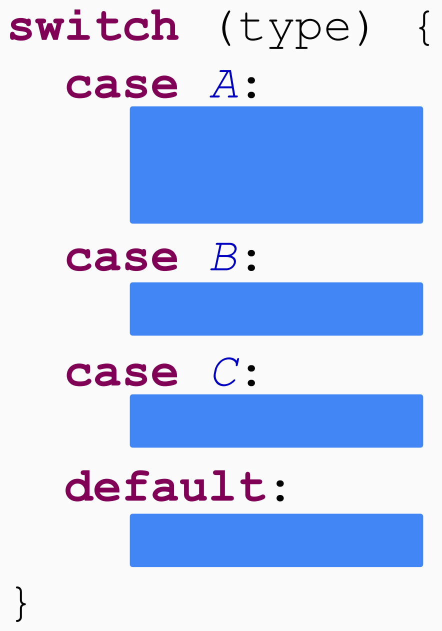
Primitive Obsession

```
double money;
String phone;
String zipCode;
String password;
```

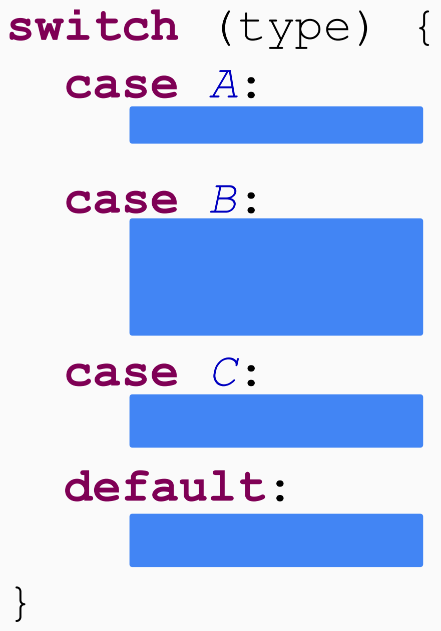
Switch Statements

```
switch (type) {
  case A:


  case B:

  case C:

  default:

}
```

Switch Statements

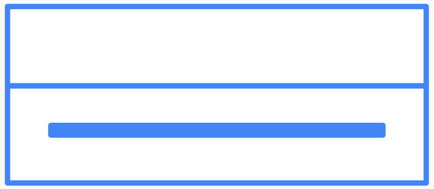


```
switch (type) {
  case A:

  case B:

  case C:

  default:

}
```

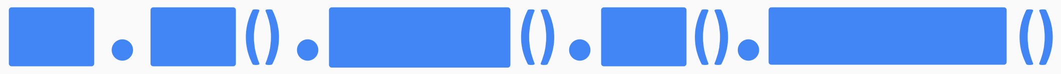


```
switch (type) {
  case A:

  case B:

  case C:

  default:

}
```

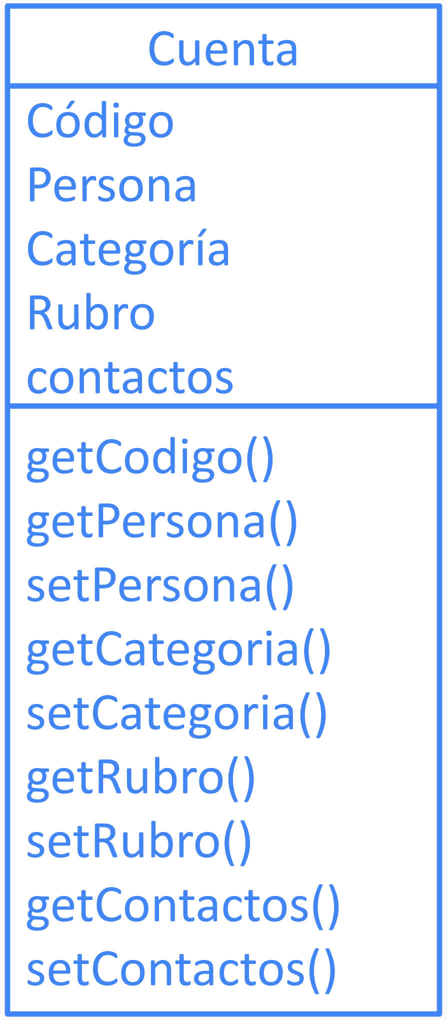
Lazy Class



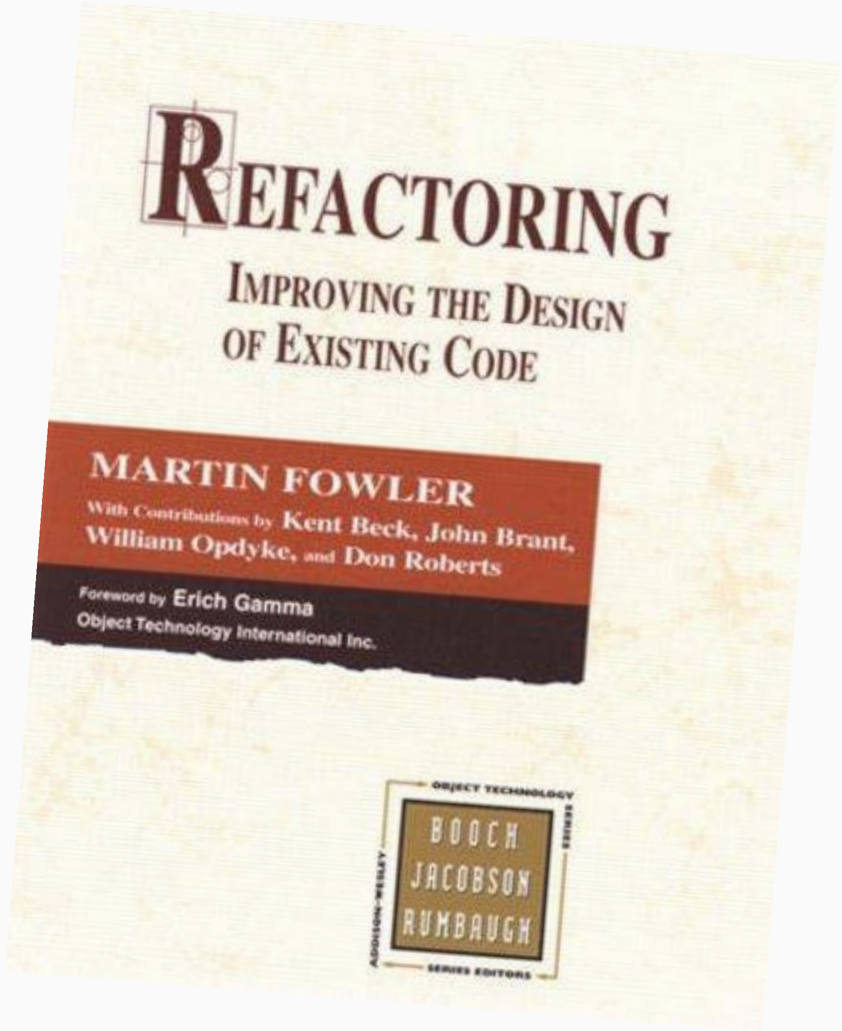
Message Chains



Data Class


Cuenta
Código
Persona
Categoría
Rubro
contactos
getCodigo()
getPersona()
setPersona()
getCategoria()
setCategoria()
getRubro()
setRubro()
getContactos()
setContactos()

Bibliografía



REFACTORING

IMPROVING THE DESIGN OF EXISTING CODE

MARTIN FOWLER

With Contributions by Kent Beck, John Brant, William Opdyke, and Don Roberts

Foreword by Erich Gamma
Object Technology International Inc.

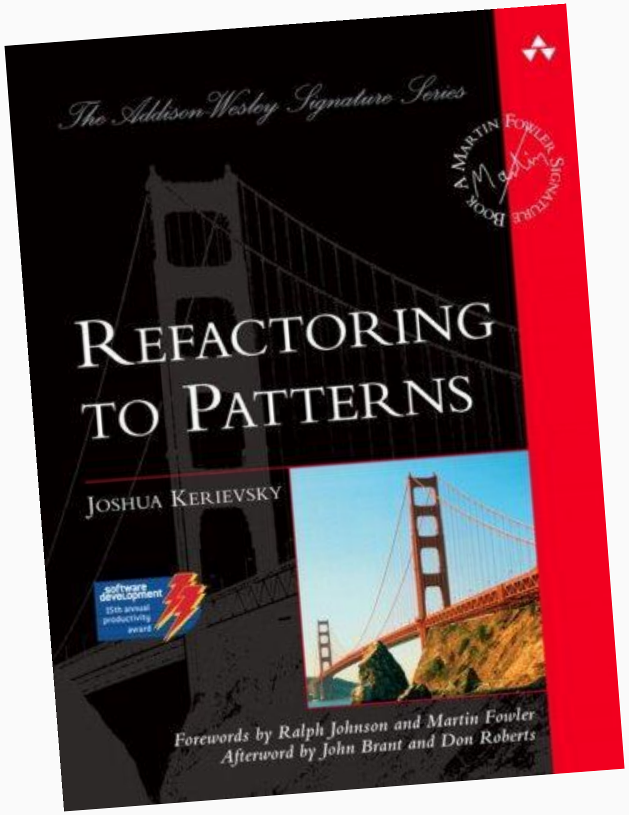
OBJECT TECHNOLOGY SERIES

BOOCH
JACOBSON
RUMBAUGH

ADDISON-WESLEY

SERIES EDITORS

Lectura Adicional


The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Refactoring to Patterns
Joshua Kerievsky
software development 15th annual productivity award
Forewords by Ralph Johnson and Martin Fowler
Afterword by John Brant and Don Roberts

?