Code Smells and Refactorings

Tuesday, October 22

Oregon State University

Announcements

Sprint 1 grades are out

Thursday interview with Mihai Codoban

Thursday graded class activity

Informal Early Feedback at the end of class.

Oregon State University

What are code smells?

"*[…] certain structures in the code that suggest (sometimes they scream for) the possibility of refactoring."* [Fowler]

They are clear signs that your **design is starting to decay.**

…Long term decay leads to "software rot"

Oregon State University

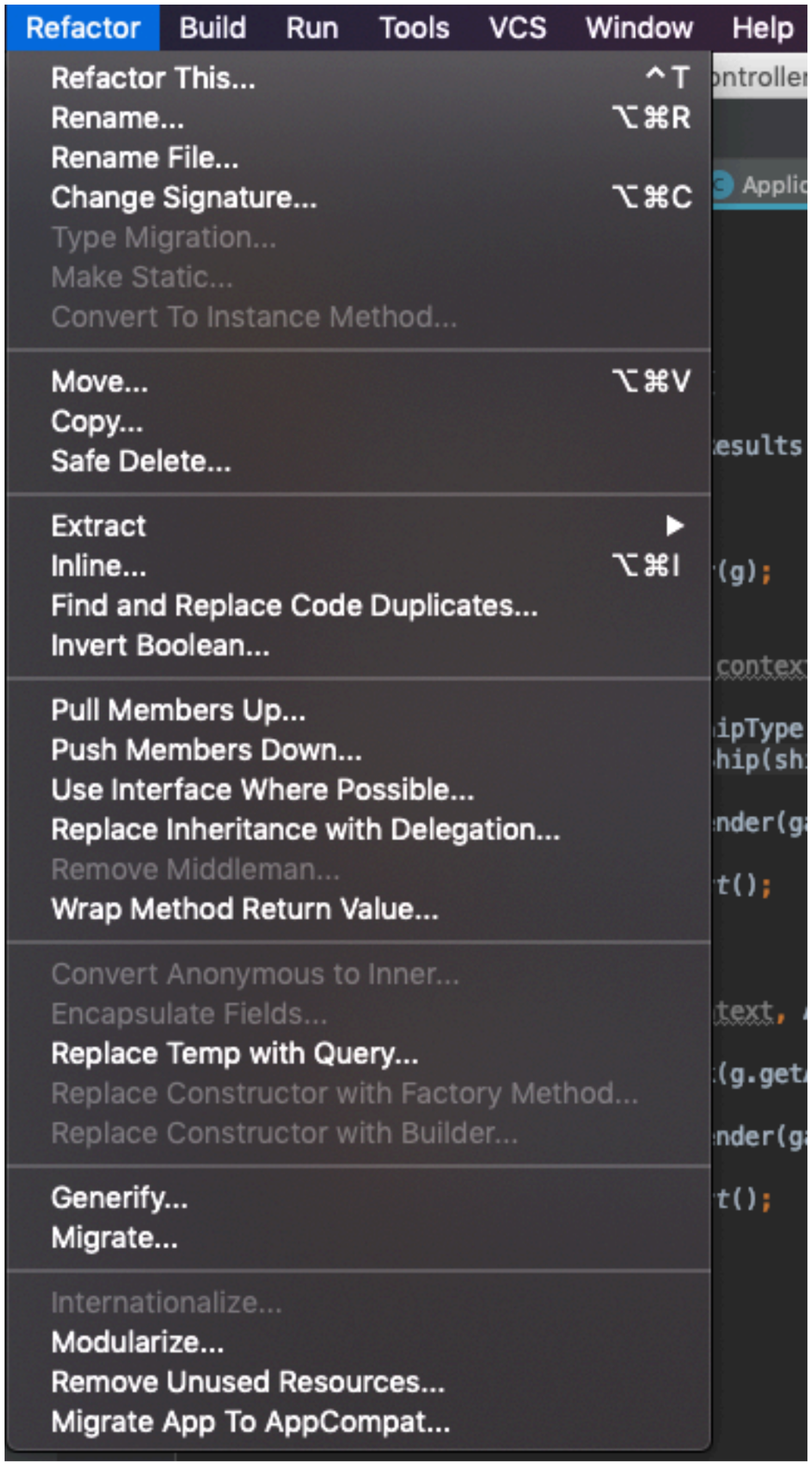
Refactorings

The main purpose of refactoring is to fight t**echnical debt.** It transforms a mess into clean code and simple design.

Refactorings will change the code but not its behavior (it still does the same thing!)

Many modern IDE's will provide automatic refactorings

Oregon State University



Oregon State University

Code Smells

Bloaters: Code, methods and classes that have increased to such gargantuan proportions that they are hard to work with;

OO Abusers: Incomplete or incorrect application of object-oriented programming principles;

Change Preventers: Any change requires you to make many changes in other places too;

Dispensables: Something pointless whose absence would make the code better;

Couplers: Excessive coupling between classes.

Oregon State University

Code Smells

Bloaters: Code, methods and classes that have increased to such gargantuan proportions that they are hard to work with;

OO Abusers: Incomplete or incorrect application of object-oriented programming principles;

Change Preventers: Any change requires you to make many changes in other places too;

Dispensables: Something pointless whose absence would make the code better;

Couplers: Excessive coupling between classes.

Oregon State University

Bloaters

Long Method

Long Class

Long Parameter List

Oregon State University

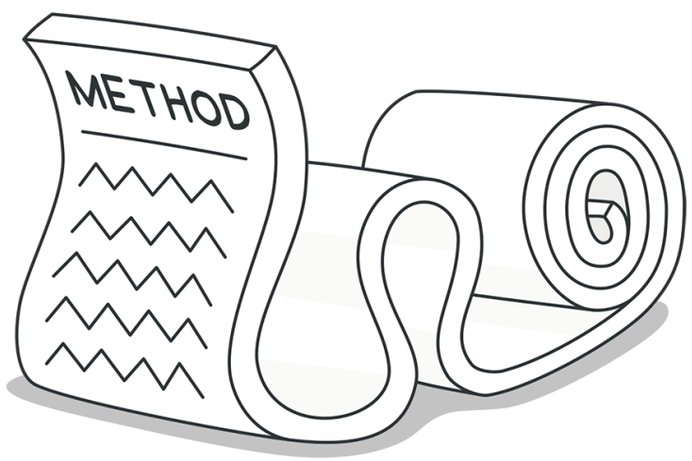
Long Method

A method containing too many lines of code.

Any line longer than 10 lines is suspicious. **More than 30 lines is problematic.**

If you have to scroll to read the whole method, it is definitely too long.

Buying a larger display is not the solution



Oregon State University

Long Method

```
98       */
99      public <S extends Sequence> MergeResult<S> merge(
100             SequenceComparator<S> cmp, S base, S ours, S theirs) {
101         List<S> sequences = new ArrayList<S>(3);
102         sequences.add(base);
103         sequences.add(ours);
104         sequences.add(theirs);
105         MergeResult<S> result = new MergeResult<S>(sequences);
106
107         if (ours.size() == 0) {
108             if (theirs.size() != 0) {
109                 EditList theirsEdits = diffAlg.diff(cmp, base, theirs);
110                 if (!theirsEdits.isEmpty()) {
111                     // we deleted, they modified -> Let their complete content
112                     // conflict with empty text
113                     result.add(1, 0, 0, ConflictState.FIRST_CONFLICTING_RANGE);
114                     result.add(2, 0, theirs.size(),
115                             ConflictState.NEXT_CONFLICTING_RANGE);
116                 } else
117                     // we deleted, they didn't modify -> Let our deletion win
118                     result.add(1, 0, 0, ConflictState.NO_CONFLICT);
119             } else
```

```
294                 current = Math.max(oursEdit.getEndA(), theirsEdit.getEndA());
295                 oursEdit = nextOursEdit;
296                 theirsEdit = nextTheirsEdit;
297             }
298         }
299         // maybe we have a common part behind the last edit: copy it to the
300         // result
301         if (current < base.size()) {
302             result.add(0, current, base.size(), ConflictState.NO_CONFLICT);
303         }
304         return result;
305     }
```


Oregon State University

Extract Method:Refactoring

Extract parts of the code, into a new method

Use this to split the long method into manageable ones.

Good opportunities:

- Code that is preceded by comments.
- Long blocks in if/else/while/for statements.
- Long conditions in if/else/while/for statements.

Always give the new methods a **meaningful name.** It should express the intent of the method (helper1 is a very very bad name).

Oregon State University

```
void printOwing() {
  printBanner();

  //print details
  System.out.println("name: " + name);
  System.out.println("amount: " + getOutstanding());
}
```

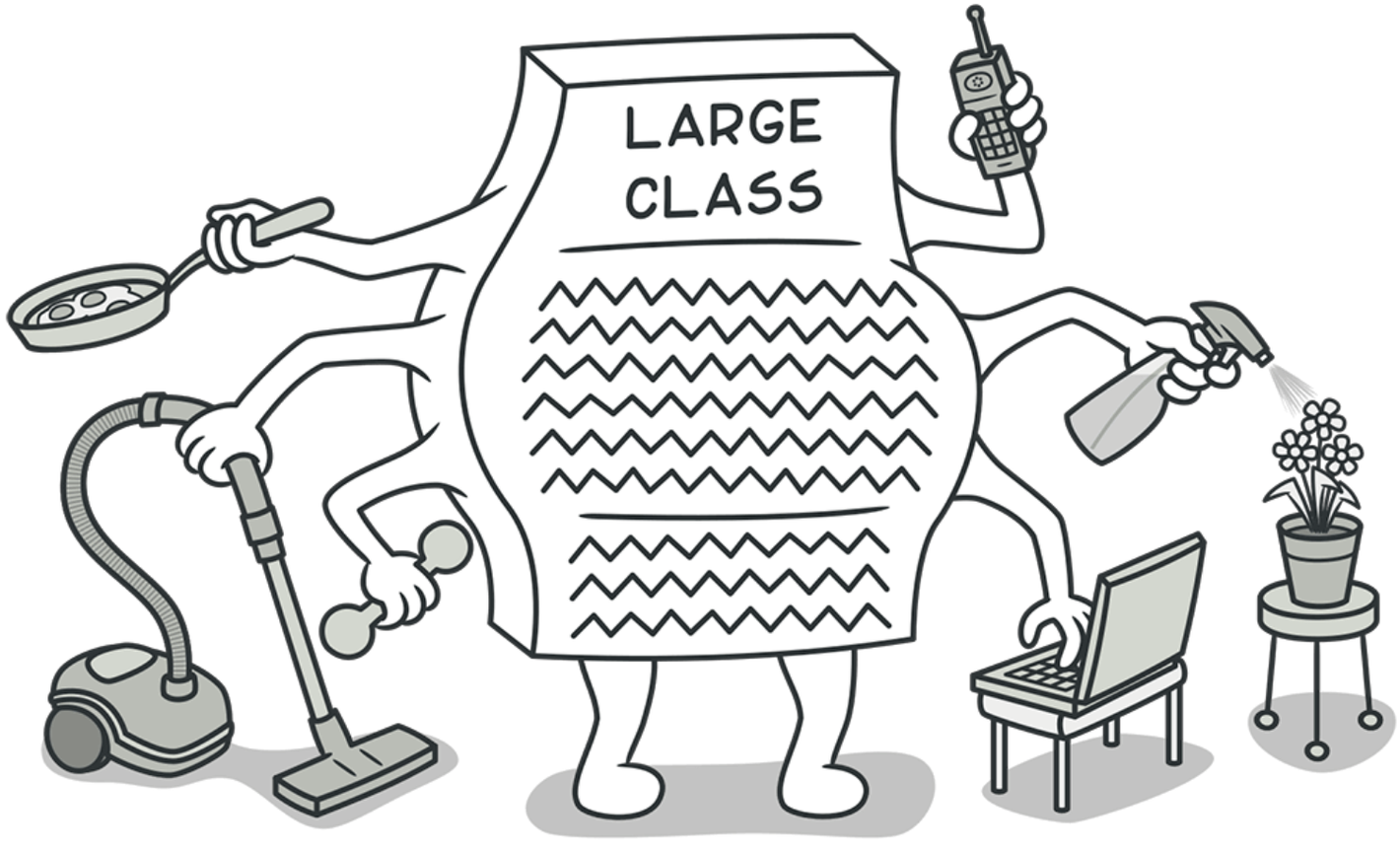
```
void printOwing() {
  printBanner();
  printDetails(getOutstanding());
}

void printDetails(double outstanding) {
  System.out.println("name: " + name);
  System.out.println("amount: " + outstanding);
}
```


Oregon State University

Long Class

A class contains many fields/methods/lines of code.

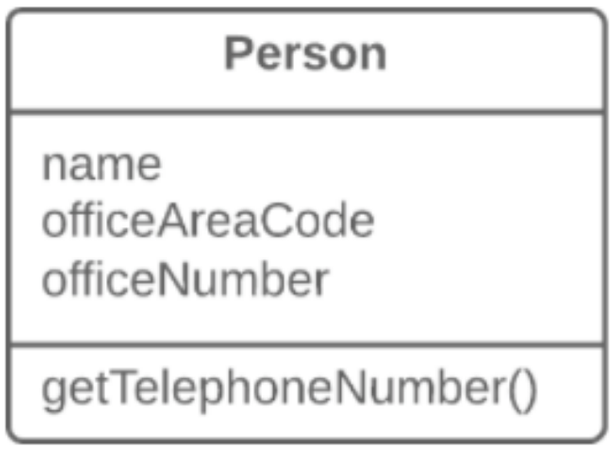
It breaks SRP

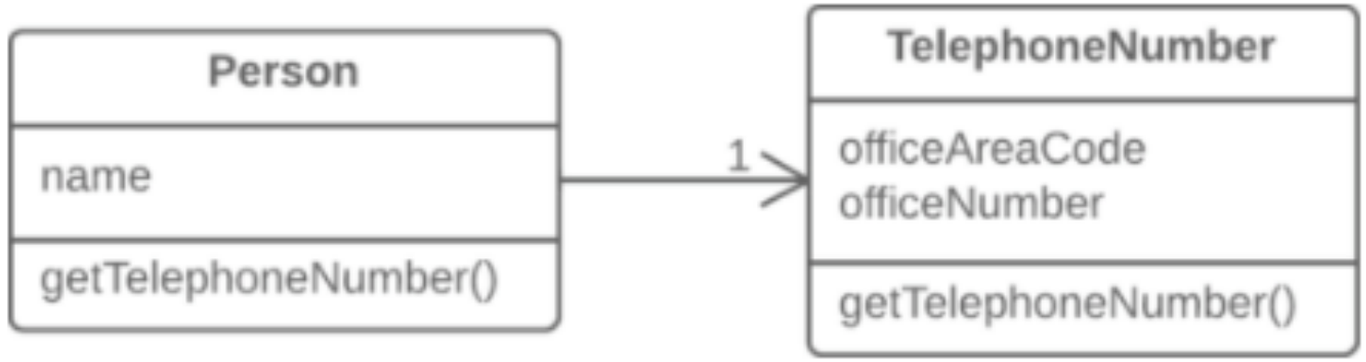


Oregon State University

Refactoring

Extract Class to split the class into multiple smaller ones



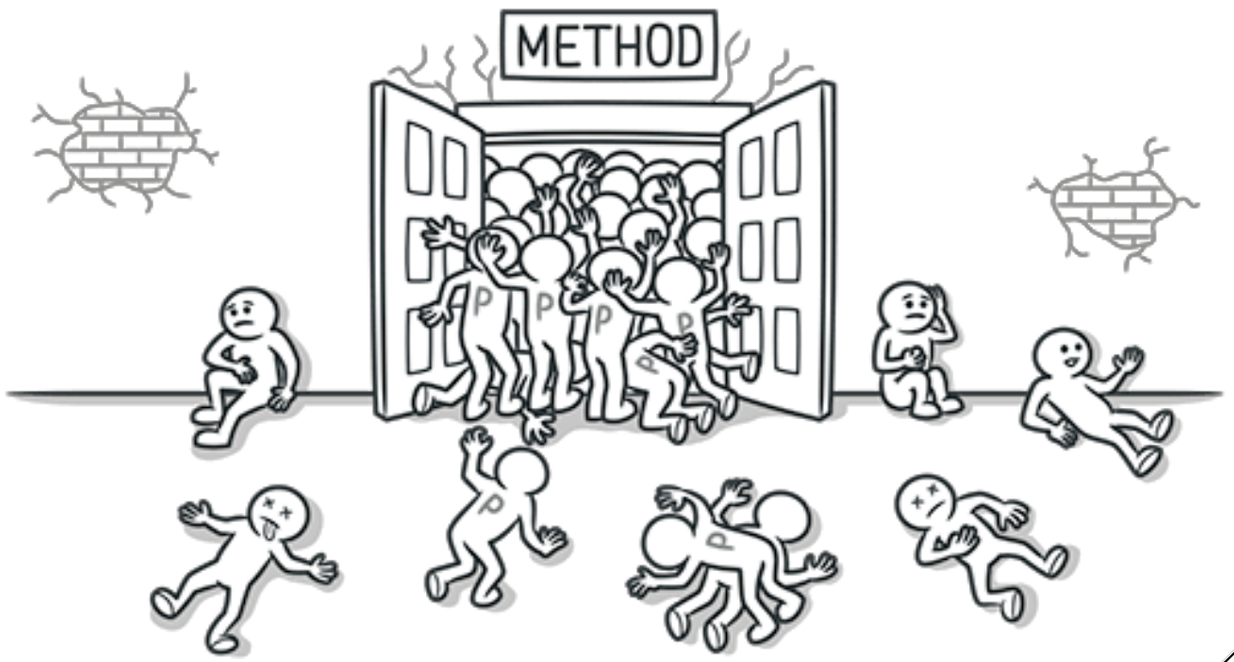


Oregon State University

Long Parameter List

Any method that has more than 4 parameters has too many

This is an indication of an inadequate abstraction level: too low



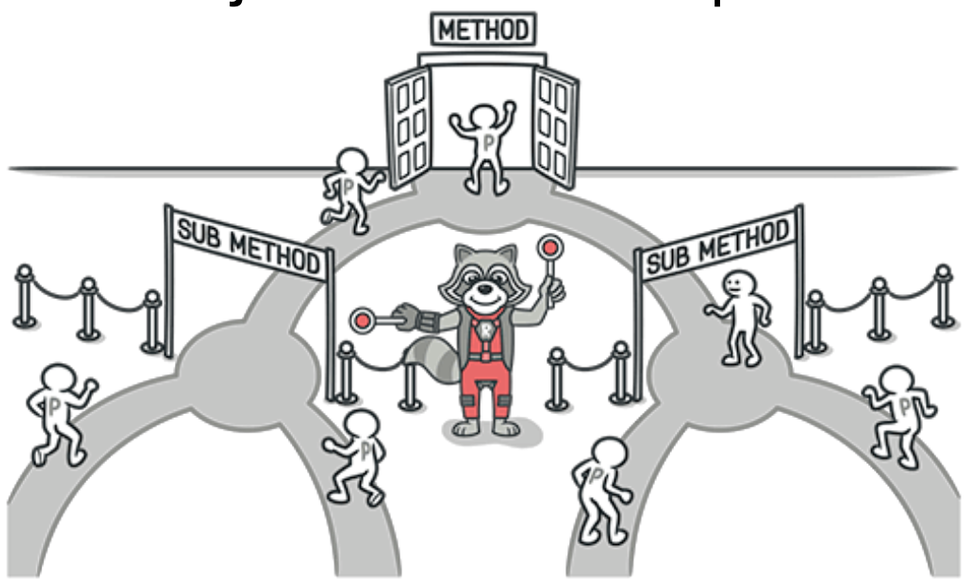
Oregon State University

Refactoring

The parameters can be encapsulated in their own objects, using introduce parameter object

If a parameter is passed repeatedly to multiple methods, it can be stored as a field

If the parameters are fields that belong to a another object, the whole object can be passed as a parameter



Oregon State University

Code Smells

Bloaters: Code, methods and classes that have increased to such gargantuan proportions that they are hard to work with;

OO Abusers: Incomplete or incorrect application of object-oriented programming principles;

Change Preventers: Any change requires you to make many changes in other places too;

Dispensables: Something pointless whose absence would make the code better;

Couplers: Excessive coupling between classes.

Oregon State University

OO Abusers

Refused Bequest

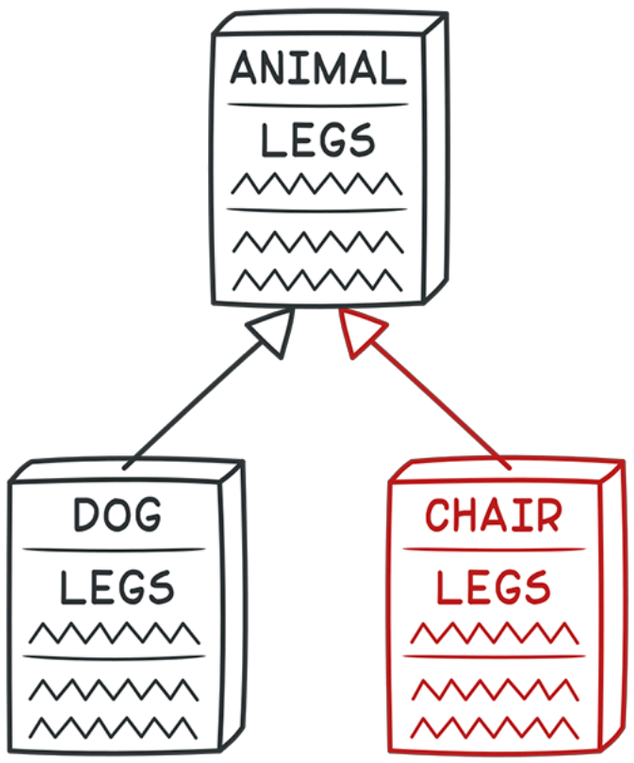
Switch Statement

Oregon State University

Refused Bequest

A subclass that uses only some of the inherited fields and method

The unneeded methods are unused or redefined to do nothing (or throw exceptions)

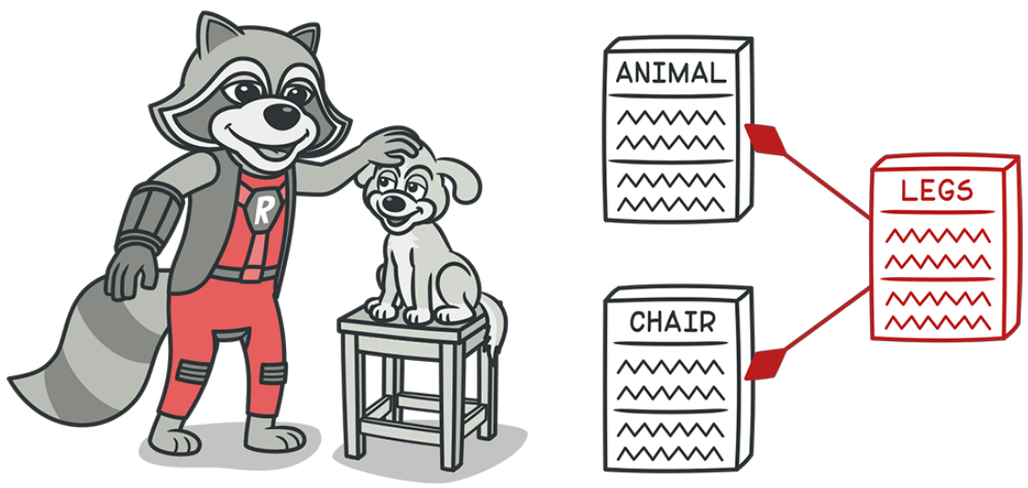


Oregon State University

Refactoring

Extract Superclass: Extract the common behavior needed by the subclass into a separate superclass, and extend from that

Replace Inheritance with Delegation: Extract the common behavior in another class, delegate methods to the super class

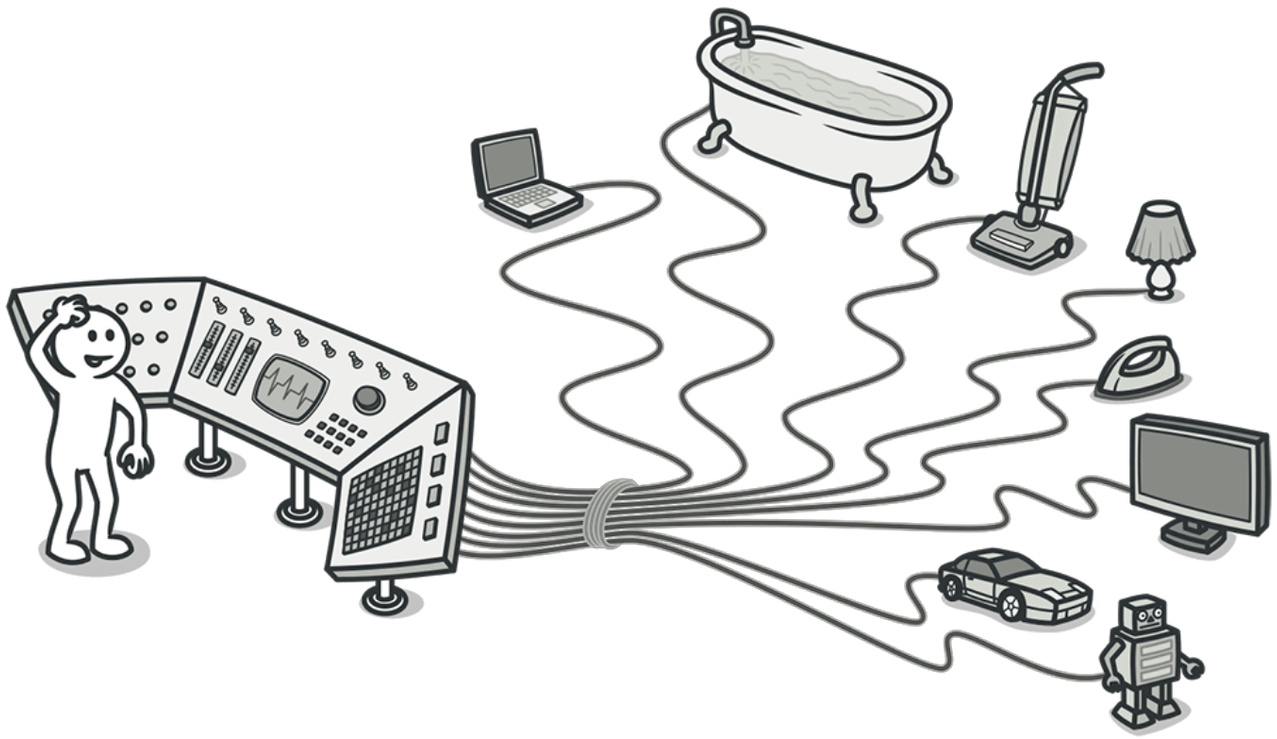


Oregon State University

Switch statements

A complex switch operator or a sequence of if statements.

It's an indication of a missing class hierarchy.



Oregon State University

Refactoring

Replace the switch/if statement with an inheritance hierarchy.

Each branch of the switch/if becomes part of subclass.

Oregon State University

Code Smells

Bloaters: Code, methods and classes that have increased to such gargantuan proportions that they are hard to work with;

OO Abusers: Incomplete or incorrect application of object-oriented programming principles;

Change Preventers: Any change requires you to make many changes in other places too;

Dispensables: Something pointless whose absence would make the code better;

Couplers: Excessive coupling between classes.

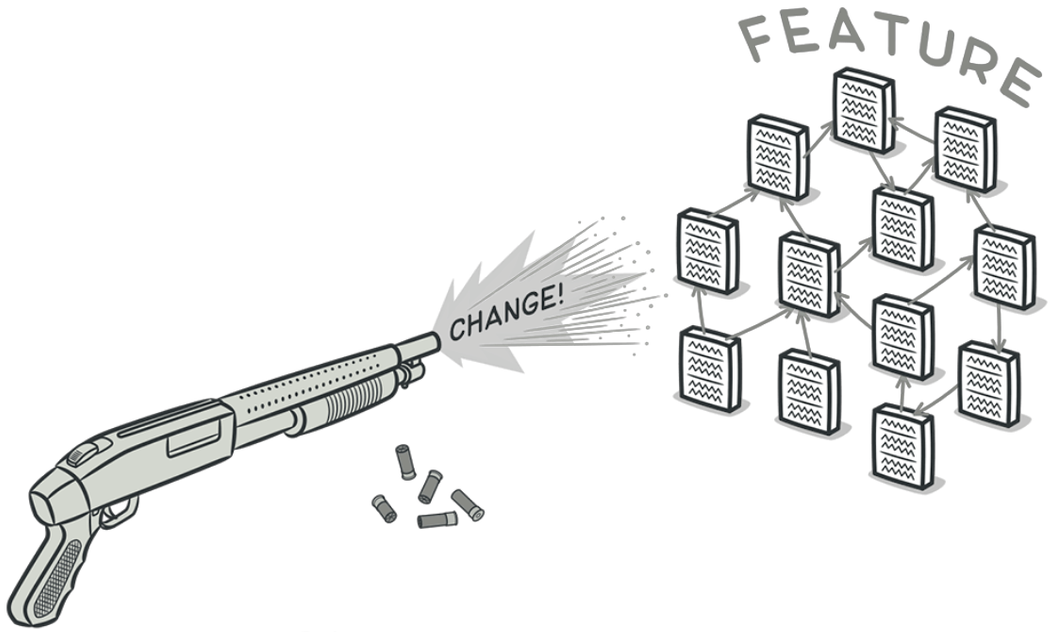
Oregon State University

Shotgun Surgery

Shotgun Surgery is a change preventer.

Making any modifications requires many small changes to many different classes

A single responsibility has been distributed among different classes.

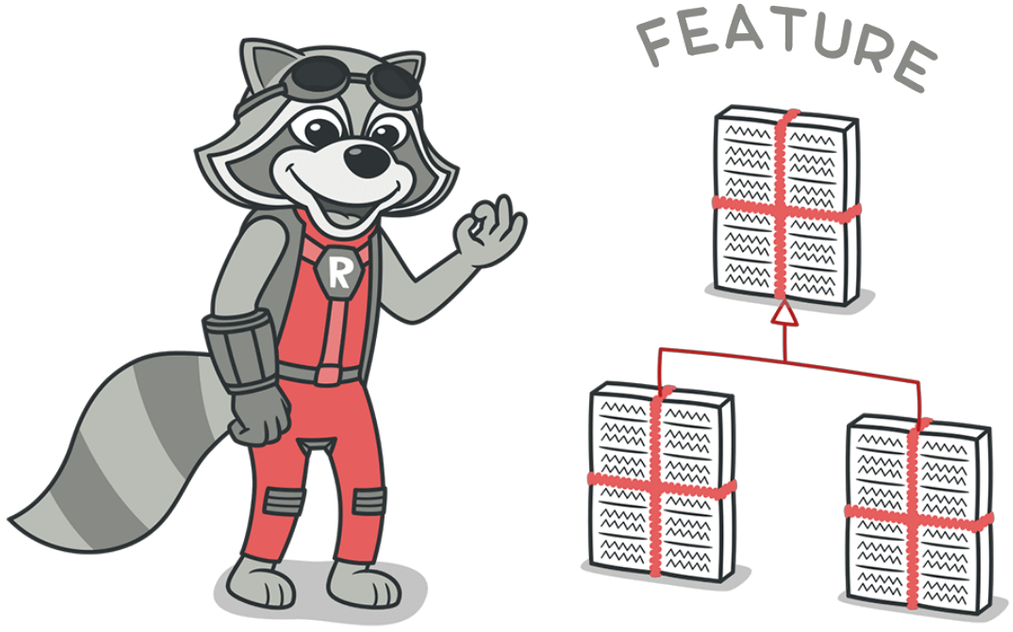


Oregon State University

Refactoring

You want to consolidate that responsibility into a single place.

Use **Move Method** and **Move Field** to move the existing behavior to the right class.



Oregon State University

Code Smells

Bloaters: Code, methods and classes that have increased to such gargantuan proportions that they are hard to work with;

OO Abusers: Incomplete or incorrect application of object-oriented programming principles;

Change Preventers: Any change requires you to make many changes in other places too;

Dispensables: Something pointless whose absence would make the code better;

Couplers: Excessive coupling between classes.

Oregon State University

Dispensables

Duplicated Code

Data Class

Speculative Generality

Oregon State University

Duplicated Code

Two code fragments look almost identical.

Changes need to be performed to both copies.

Oregon State University

Refactoring

If the duplicated code is in the same class you **Extract Method** and place calls to the new method in both places.

If the two methods are "independent," use **Extract Superclass** to extract a common superclass.

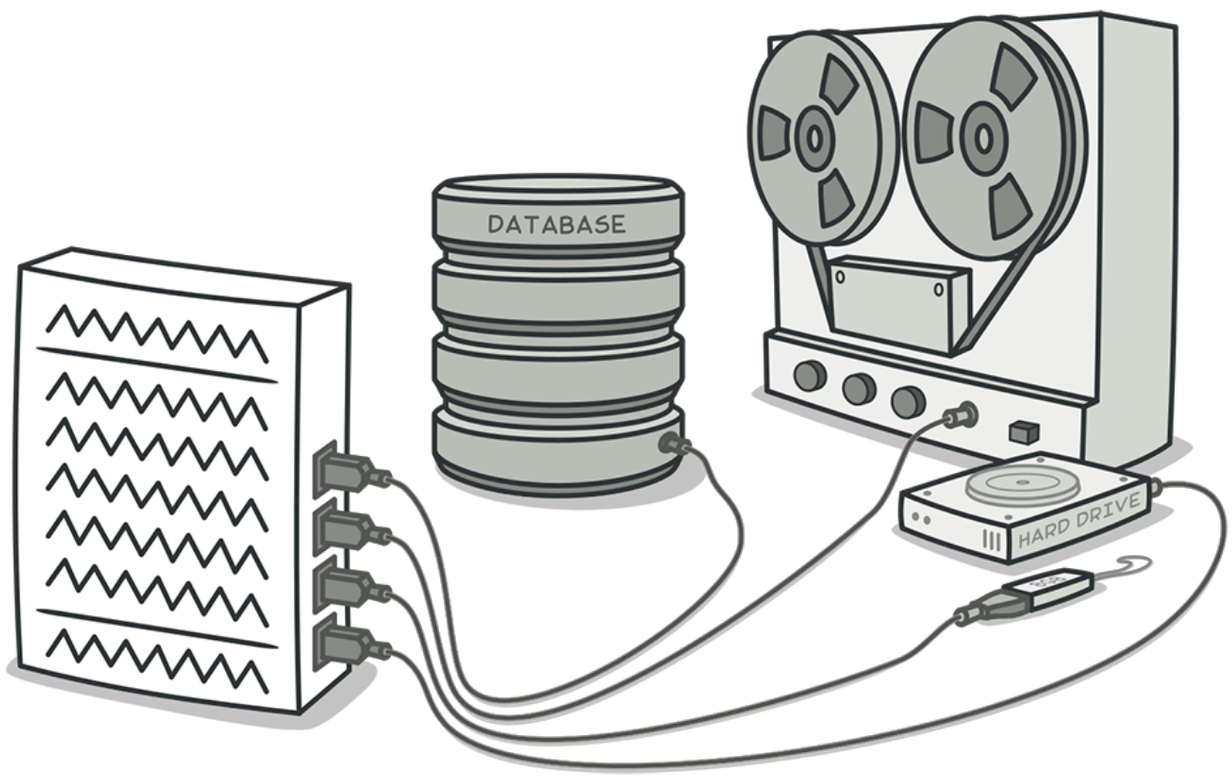
If it is on the same level of a class hierarchy, use **Extract Method** for both classes, then **Pull Up Method** to move that method to the superclass.

Oregon State University

Data Class

Contains only fields and crude methods for accessing them (getters and setters).

They are simply containers for data used by other classes.



Oregon State University

Refactoring

Look at the client code (consumer). It's very likely that the client has responsibilities that can be moved to the data class.

Move Method and **Extract Method** can be used to move functionality to the data class.

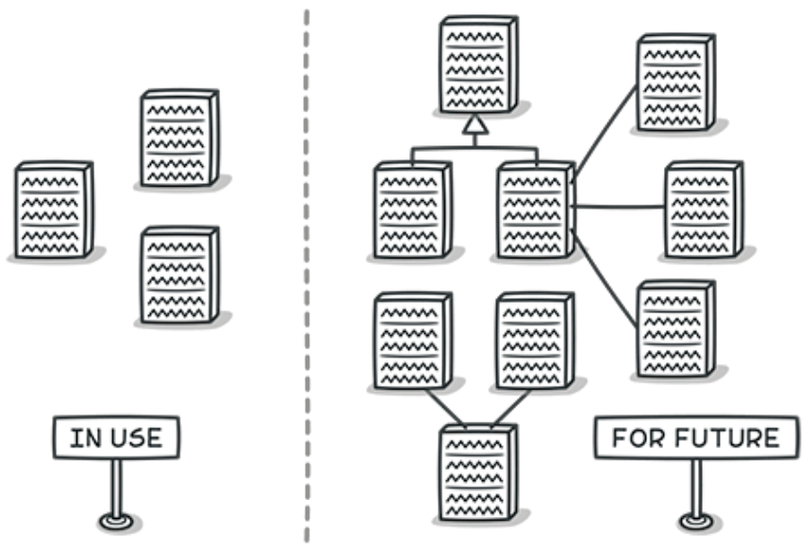
Oregon State University

Speculative Generality

Unused classes, fields or parameters

Code that is created "just in case" to support anticipated future features that never get implemented.

e.g. Abstract classes that are only implemented by one subclass



Oregon State University

Refactoring

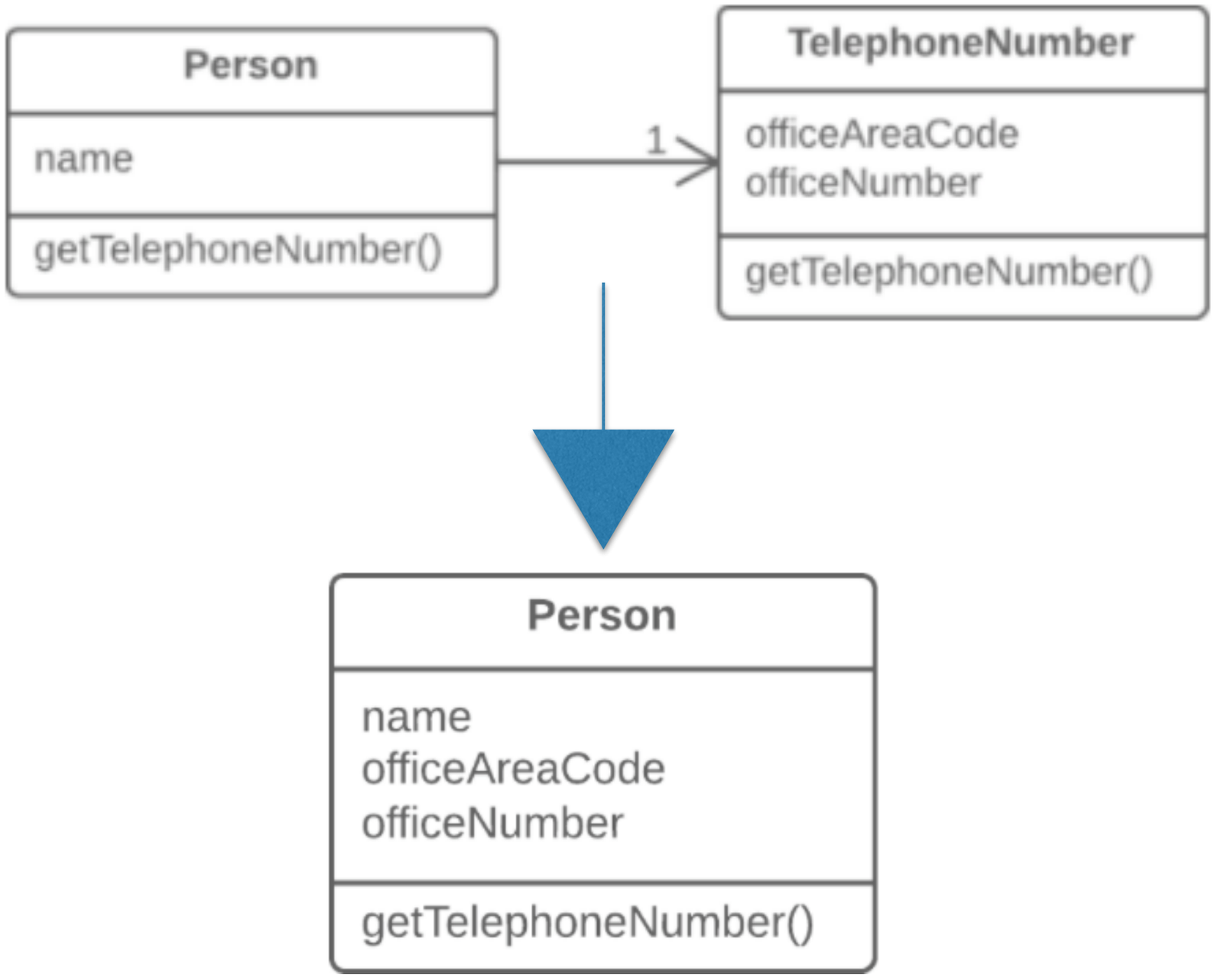
Unused abstract classes can be removed using **Collapse Hierarchy**

Unnecessary classes can be removed via **Inline Class**

Unused fields and method can be simply removed.

Oregon State University

Move to in-line class



Oregon State University

Code Smells

Bloaters: Code, methods and classes that have increased to such gargantuan proportions that they are hard to work with;

OO Abusers: Incomplete or incorrect application of object-oriented programming principles;

Change Preventers: Any change requires you to make many changes in other places too;

Dispensables: Something pointless whose absence would make the code better;

Couplers: Excessive coupling between classes.

Oregon State University

Couplers

Feature envy

Inappropriate intimacy

Oregon State University

Feature Envy

A method accesses the data of another object more than its own data.

Oregon State University

Refactoring

Use **Move Method** to move the methods to another place

If only part of a method is envious, then use **Extract Method,** together with **Move Method**

Oregon State University

Inappropriate Intimacy

One class uses the internal fields and methods of another class.

Good classes should know as little about each other as possible.

Oregon State University

Refactoring

Use **Move Method** and **Move Fields** to move parts from one class to the other.

Oregon State University

Class exercise (in Pairs)

Look at the code that you downloaded from GitHub

https://github.com/cs361fall2018/videostore

Identify the code smells in the code base

Oregon State University

Class exercise (in Pairs)

Look at the code that you downloaded from GitHub

https://github.com/cs361fall2018/videostore

Identify the code smells in the code base

Now what refactoring will you do to clean the code smell?

Oregon State University