Refactoring Fundamentals: Code Smells

Steve Smith

Ardalis.com

Twitter: @ardalis

pluralsight
hardcore developer training

In This Course

- What is Refactoring?
- Why do it?
- What's the process?
- What are some tools that can assist with it?
- What is a *Code Smell*?
- What are some examples of Code Smells?
- What are some common refactorings?
- How does one apply them correctly?

Code Smells

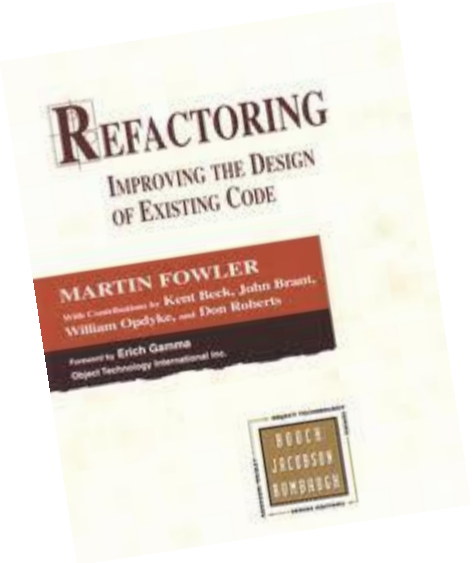
http://commons.wikimedia.org/wiki/File:Striped_Skunk.jpg

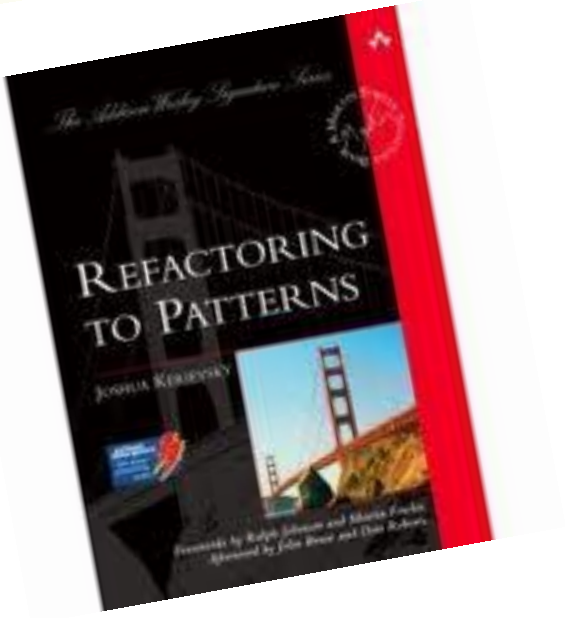
Principle of Least Surprise

- "Do what users expect"
- Design API to behave as programmers would expect
- Be simple
- Be clear
- Be consistent

http://en.wikipedia.org/wiki/Principle_of_least_astonishment

Code Smells in Literature





Organizing Code Smells

- **Taxonomy proposed by Mäntylä, M. V. and Lassenius, C.**
  - http://www.soberit.hut.fi/~mmantyla/ESE_2006.pdf

- **Organization of Code Smells into 5 Groups**
  - The Bloaters
  - The Object-Orientation Abusers
  - The Change Preventers
  - The Dispensables
  - The Couplers

- **I've added three more:**
  - The Obfuscators
  - Environment Smells
  - Test Smells

Code Smells: The Bloaters

- **Things that have grown out of control**
  - Probably little-by-little
- **Things that cause bloat within a system**

http://flic.kr/p/6UN7rp

The Bloaters: Long Method

```
4    namespace CodeSmells.Bloaters.LongMethod
5    {
6        public class ManagerUtility
7        {
8            public void DoEverythingForTheApplication()...
3176     }
```

- **Small methods are easier to understand at a glance**

- **How small?**
  - Certainly the method should fit on one screen.
  - Ideally, 10 lines or fewer

The Bloaters: Long Method

What about performance? There's overhead in calling methods…

- **Don't prematurely optimize**
- **Minimal impact – verify with performance testing (I have a course on this…)**
- **Far better performance tweaks than combining method**

Related Smells

- **Long Loops**

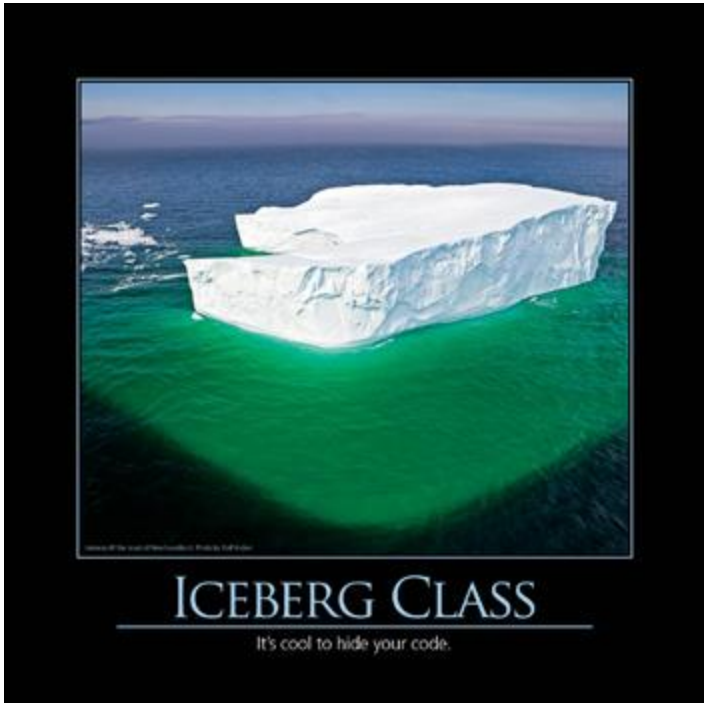
- **Functions That Do More Than One Thing**

The Bloaters: Long Method

Refactor Long Methods

- **Extract Method**
  - Introduce Parameter Object
  - Replace Temp with Query
  - Replace Method with Method Object
- **Compose Method**
- **Replace Nested Conditional with Guard Clause**
- **Replace Conditional Dispatcher with Command**
- **Move Accumulation to Visitor**
- **Replace Conditional Logic with Strategy**

The Bloaters: Large Class



- **Violating Single Responsibility Principle**
- **Too many instance variables**
- **Too many private methods**

  Iceberg Class http://deviq.com/iceberg-class

- **Lack of cohesion**
  - Some methods work with some instance variables, others with others
  - Compartmentalized Class

The Bloaters: Large Class

Refactoring Large Classes

- **Extract Method (and hopefully combine logic)**
- **Extract Class**
- **Extract Subclass / Extract Interface**
- **Replace Conditional Dispatcher with Command**
- **Replace State-Altering Conditionals with State**
- **Replace Implicit Language with Interpreter**

The Bloaters: Primitive Obsession

- **Over-use of primitives, instead of better abstractions, results in excess code to shoehorn the type**
  - Guard clauses
  - Validation
- **Less intention-revealing**
- **McConnell: Primitive Type is overloaded**

http://en.wikipedia.org/wiki/File:San_Rock_Art_-_Cederberg.jpg

The Bloaters: Primitive Obsession

Refactoring away from Primitive Obsession

- **Replace Data Value with Object**
- **Replace Type Code with Class**
- **Replace Type Code with Subclass**
- **Extract Class**
- **Introduce Parameter Object**
- **Replace Array with Object**
- **Replace State-Altering Conditionals with State**
- **Replace Conditional Logic with Strategy**

Examples of Primitive Obsession

```
// method call relying on primitives only
AddHoliday(7,4);
```

Examples of Primitive Obsession

```
// method call relying on primitives only
AddHoliday(7,4);

// use a higher level type
Date independenceDay = new Date(7,4);
AddHoliday(independenceDay);
```

Examples of Primitive Obsession

```
// method call relying on primitives only
AddHoliday(7,4);

// use a higher level type
Date independenceDay = new Date(7,4);
AddHoliday(independenceDay);

// go even further
public class July {
private const int _month = 7;
public static readonly Date Fourth {
  get { return new Date(_month, 4); } } }

AddHoliday(July.Fourth);
```

Examples of Primitive Obsession

```
// method call relying on primitives
DrawLine(5,20,25,40);
```

Examples of Primitive Obsession

```
// method call relying on primitives
DrawLine(5,20,25,40,0,0,255);
```

Examples of Primitive Obsession

```
// method call relying on primitives
DrawLine(5,20,25,40);
DrawLine(5,20,25,40,0,0,255);

// refactored to use higher level concepts
var startPoint = new Point(5,20);
var endPoint = new Point(25,40);
var color = Color.Blue;

DrawLine(startPoint, endPoint);
DrawLine(startPoint, endPoint, color);
```

Examples of Primitive Obsession

```
// method call relying on primitives
TransferFunds(23423, 23434, 48432);
```

Examples of Primitive Obsession

```
// method call relying on primitives
TransferFunds(23423, 23434, 48432);

// refactored to be more clear and use higher level
   objects
Account transferFrom = _accountRepository.Get(23423);
Account transferTo = _accountRepository.Get(23434);
var transferAmount = new Money(48432, Currency.USD);
TransferFunds(transferFrom, transferTo, transferAmount);
```

Examples of Primitive Obsession

- ZIP / Postal Codes
- Phone Numbers
- Social Security Numbers
- Telephone Numbers
- Money
- Age
- Temperature
- Address
- Credit Card Information

The Bloaters: Long Parameter List

- **Bloats code and reduces readability**
- **May indicate procedural rather than OO programming style**

- **Related Smells (from *Refactoring*)**
  - Message Chains
  - Middle Man

- **Related Smells (from *Clean Code*)**
  - No More Than Three
  - No output arguments
  - No flag arguments
  - Selector arguments

methods	# Parameters
AddSummaryRow(Int32,String,Int32,String,Int32,String,DateTime,DateTime ,Int32,String,Int32,Int32,Int32,Int32,Int32)	15
LoadContacts(InsertionOrder,Guid,User&,User&,Int32&,String&,String& ,String&,Boolean&,Int32&,String&,String&,String&,Boolean&)	14

- ***Prefer more, smaller, well-named methods to fewer methods with behavior controlled by parameters***

The Bloaters: Long Parameter List

Refactoring from Long Parameter List

- **Replace Parameter with Method**
- **Preserve Whole Object**
- **Introduce Parameter Object**
- **Extract Method**

Tools

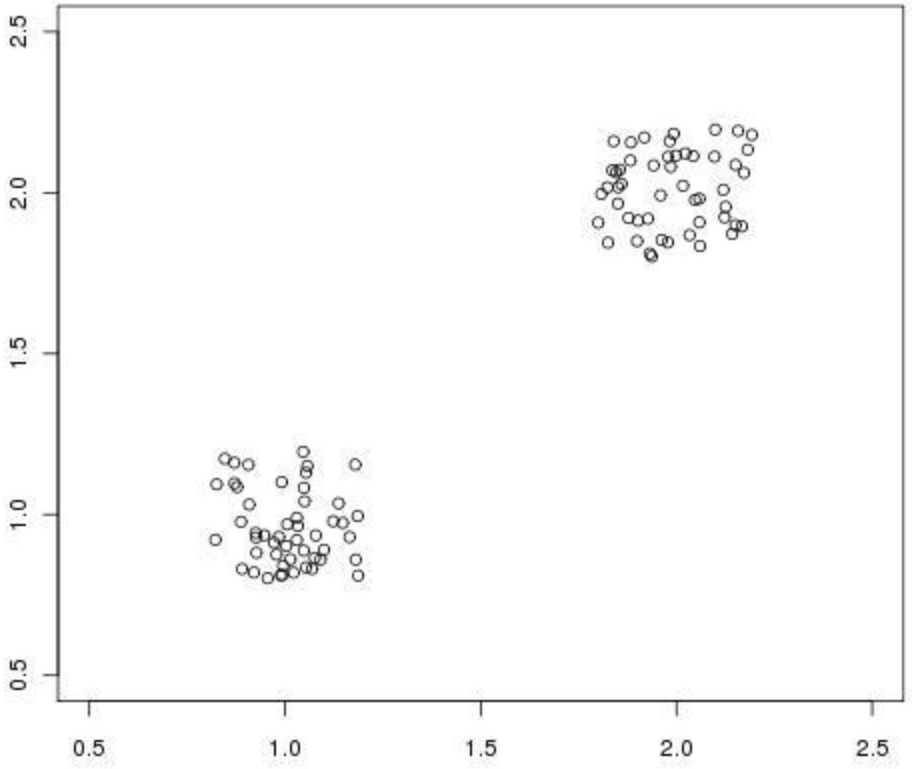
- **NDepend (Ndepend.com)**
- **Nitriq (Nitriq.com)** ***Free***

The Bloaters: Data Clumps

- A set of data items that are always used together, but are not organized together in the design

- Similar to Primitive Obsession, but different in that the smell deals with the presence of several items

- Frequently, Data Clumps give rise to Long Parameter Lists



The Bloaters: Data Clumps

Refactoring from Data Clumps

- **Extract Class**
  - Introduce Parameter Object *or*
  - Preserve Whole Object

Examples of Data Clumps

```
// data clumps on an order
Order.CreditCardName = creditCardName;
Order.CreditCardNumber = creditCardNumber;
Order.ExpiresMonth = creditCardMonth;
Order.ExpiresYear = creditCardYear;
Order.SecurityCode = creditCardSecurityCode;
```

Examples of Data Clumps

```
// data clumps on an order
Order.CreditCardName = creditCardName;
Order.CreditCardNumber = creditCardNumber;
Order.ExpiresMonth = creditCardMonth;
Order.ExpiresYear = creditCardYear;
Order.SecurityCode = creditCardSecurityCode;

// refactored by Extracting a new CreditCardInfo class
CreditCardInfo cardInfo = new CreditCardInfo(
  creditCardNme, creditCardNumber, creditCardMonth,
  creditCardYear, creditCardSecurityCode);

Order.CreditCard = cardInfo;
```

The Bloaters: Combinatorial Explosion

- Number of logical cases combine to result in a massive increase in number of methods needed to cover every possibility

The Bloaters: Combinatorial Explosion

- **Example – Data Query Methods**
  - ListCars()
  - ListCarsByManufacturer(string manufacturer);
  - ListCarsByRegion(string region);
  - ListCarsByManufacturerAndRegion(string manufacturer, string region);
  - Etc.

Refactor

- **Replace Implicit Language with Interpreter**

IQueryable / LINQ / Predicates can often be used effectively in these cases

but be careful not to leak concerns between layers in your application

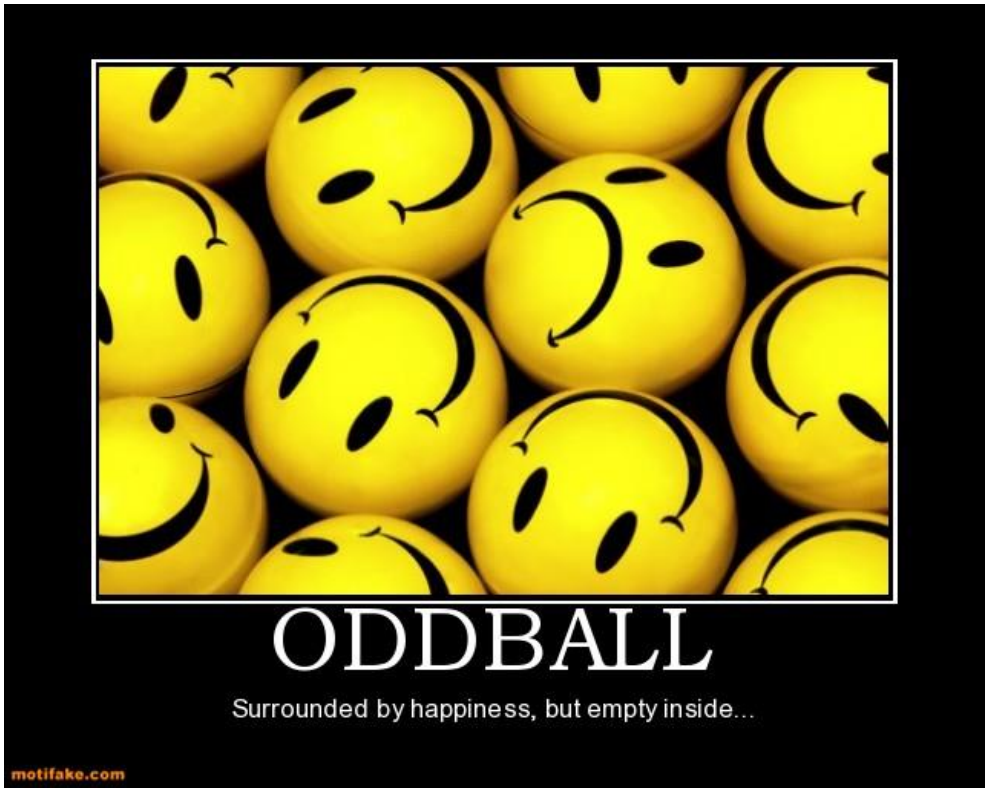
Examples of Avoiding Combinatorial Explosion

```
public IQueryable<T> FindBy(
Expression<Func<T, bool>> predicate)
{
   IQueryable<T> query =
   _entities.Set<T>().Where(predicate);
   return query;
}

// or, not exposing IQueryable
public IList<T> FindBy(Expression<Func<T, bool>>
   predicate)
{
  return _entities.Set<T>().Where(predicate).ToList();
}
```

The Bloaters: Oddball Solution

- A different way of solving a common problem within the system
- Usually indicates duplicate code
- Choose your preferred solution and attempt to use it consistently



Refactoring from Oddball Solution:

- Substitute Algorithm
- Unify Interfaces with Adapter

http://www.motifake.com/oddball-oddball-unhappy-smile-demotivational-posters-148589.html

The Bloaters: Class Doesn't Do Much

- Class's responsibilities can be moved elsewhere
- Perhaps used to be important, but no longer
- Consider if class's contents can be moved to other class(es)
- Remove the (now empty) class

http://flic.kr/p/GF6SR

The Bloaters: Required Setup/Teardown Code

- A class or method requires several lines of code before and/or after its use
- Problem increases with frequency of use of this class or method
- May indicate an improper abstraction level

Refactor away from required setup/teardown with:

- Introduce Parameter Object
- Replace Constructor with Factory Method
- Implement IDisposable

Summary

Organization of Code Smells into 5 Groups

- The Bloaters
- The Object-Orientation Abusers
- The Change Preventers
- The Dispensables
- The Couplers

I've added three more:

- The Obfuscators
- Environment Smells
- Test Smells

References

Related Pluralsight Courses

SOLID Principles of Object Oriented Design http://bit.ly/rKbR9a

Design Patterns Library http://bit.ly/SJmAX1

Books

Refactoring http://amzn.to/110tscA

Refactoring to Patterns http://amzn.to/Vq5Rj2

Working Effectively with Legacy Code http://amzn.to/VFFYbn

Code Complete http://amzn.to/Vq5YLv

Clean Code http://amzn.to/YjUDI0

Thanks!

Steve Smith

Ardalis.com

Twitter: @ardalis