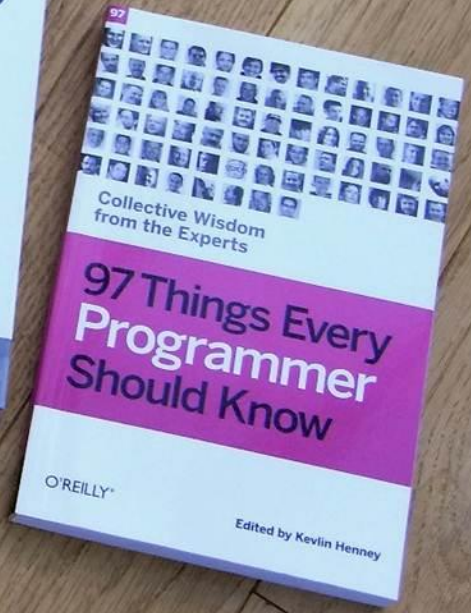
Seven Ineffective Coding Habits of Many Java Programmers

@KevlinHenney


WILEY SERIES IN
SOFTWARE DESIGN PATTERNS
PATTERN-ORIENTED SOFTWARE ARCHITECTURE
A Pattern Language for Distributed Computing
Volume 4
Frank Buschmann
Kevlin Henney
Douglas C. Schmidt


WILEY SERIES IN
SOFTWARE DESIGN PATTERNS
PATTERN-ORIENTED SOFTWARE ARCHITECTURE
On Patterns and Pattern Languages
Volume 5
Frank Buschmann
Kevlin Henney
Douglas C. Schmidt


97
Collective Wisdom from the Experts
97 Things Every Programmer Should Know
O'REILLY
Edited by Kevlin Henney


physicsworld
physicsworld.com
Volume 26 No 7 July 2013
PHYSICS OF CANCER
ew tools and fresh perspectiv
THE BEST NEW WRITING FROM THE INTERNATIONAL SALT PRIZES
The SALT ANTHOLOGY NEW WRITING 2013
Edited by
Chris Hamilton-Emery
and Jen Hamilton-Emery
SALT
Scraps
A collection of flash-fictions from
National Flash-Fiction Day 2013
Edited by Calum Kerr and Holly Howitt

It turns out that style matters in programming for the same reason that it matters in writing. It makes for better reading.

Douglas Crockford
JavaScript: The Good Parts


HOLY GOD
WILL BRING
JUDGMENT
DAY ON
MAY 21, 2011
CRY MIGHTILY UNTO
GOD FOR MERCY SEE
PSALMS - 51:
JONAH - 3:
DELICATESSEN
Exit only

Noisy Code


HOLY GOD
WILL BRING
JUDGMENT
DAY ON
MAY 21, 2011
CRY MIGHTILY UNTO
GOD FOR MERCY SEE
PSALMS - 51:
JONAH - 3:

Signal-to-noise ratio (often abbreviated SNR or S/N) is a measure used in science and engineering that compares the level of a desired signal to the level of background noise.

Signal-to-noise ratio is sometimes used informally to refer to the ratio of useful information to false or irrelevant data in a conversation or exchange.

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

To be, or not to be: that is the question:
Whether 'tis nobler in the mind to suffer
The slings and arrows of outrageous fortune,
Or to take arms against a sea of troubles,
And by opposing end them?

William Shakespeare
Hamlet

Continuing existence or cessation of existence: those are the scenarios. Is it more empowering mentally to work towards an accommodation of the downsizings and negative outcomes of adversarial circumstance, or would it be a greater enhancement of the bottom line to move forwards to a challenge to our current difficulties, and, by making a commitment to opposition, to effect their demise?

Tom Burton
Long Words Bother Me

```
public class RecentlyUsedList
{
    private List<String> items;
    public RecentlyUsedList()
    {
        items = new ArrayList<String>();
    }
    public void add(String newItem)
    {
        if (items.contains(newItem))
        {
            int position = items.indexOf(newItem);
            String existingItem = items.get(position);
            items.remove(position);
            items.add(0, existingItem);
        }
        else
        {
            items.add(0, newItem);
        }
    }
    public int size()
    {
        int result = items.size();
        return result;
    }
    public String get(int index)
    {
        int position = 0;
        for (String item : items)
        {
            if (position == index)
                return item;
            ++position;
        }
        throw new IndexOutOfBoundsException();
    }
}
```

```
public class RecentlyUsedList
{
    private List<String> items;

    public RecentlyUsedList()
    {
        items = new ArrayList<String>();
    }
    public void add(String newItem)
    {
        if (items.contains(newItem))
        {
            int position = items.indexOf(newItem);
            String existingItem = items.get(position);
            items.remove(position);
            items.add(0, existingItem);
        }
        else
        {
            items.add(0, newItem);
        }
    }
    public int size()
    {
        int result = items.size();
        return result;
    }
    public String get(int index)
    {
        int position = 0;
        for (String item : items)
        {
            if (position == index)
                return item;
            ++position;
        }
        throw new IndexOutOfBoundsException();
    }
}
```

```
public class RecentlyUsedList
{
    private List<String> items = new ArrayList<String>();

    public void add(String newItem)
    {
        items.remove(newItem);
        items.add(newItem);
    }
    public int size()
    {
        return items.size();
    }
    public String get(int index)
    {
        return items.get(size() - index – 1);
    }
}
```

```
/*
 * Copyright (c) 1995, 2008, Oracle and/or its affiliates. All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 *   - Redistributions of source code must retain the above copyright
 *     notice, this list of conditions and the following disclaimer.
 *
 *   - Redistributions in binary form must reproduce the above copyright
 *     notice, this list of conditions and the following disclaimer in the
 *     documentation and/or other materials provided with the distribution.
 *
 *   - Neither the name of Oracle or the names of its
 *     contributors may be used to endorse or promote products derived
 *     from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
 * IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
 * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

/**
 * The HelloWorldApp class implements an application that
 * simply prints "Hello World!" to standard output.
 */
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!"); // Display the string.
    }
}
```

```
/*
 * Copyright (c) 1995, 2008, Oracle and/or its affiliates. All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 *   - Redistributions of source code must retain the above copyright
 *     notice, this list of conditions and the following disclaimer.
 *
 *   - Redistributions in binary form must reproduce the above copyright
 *     notice, this list of conditions and the following disclaimer in the
 *     documentation and/or other materials provided with the distribution.
 *
 *   - Neither the name of Oracle or the names of its
 *     contributors may be used to endorse or promote products derived
 *     from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
 * IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
 * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

/**
 * The HelloWorldApp class implements an application that
 * simply prints "Hello World!" to standard output.
 */
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!"); // Display the string.
    }
}
```

Comments

A delicate matter, requiring taste and judgement. I tend to err on the side of eliminating comments, for several reasons. First, if the code is clear, and uses good type names and variable names, it should explain itself. Second, comments aren't checked by the compiler, so there is no guarantee they're right, especially after the code is modified. A misleading comment can be very confusing. Third, the issue of typography: comments clutter code.

Rob Pike, "Notes on Programming in C"

There is a famously bad comment style:

```
i=i+1;              /* Add one to i */
```

and there are worse ways to do it:

```
/**********************************
 *                                *
 *           Add one to i         *
 *                                *
 **********************************/

                  i=i+1;
```

Don't laugh now, wait until you see it in real life.

Rob Pike, "Notes on Programming in C"

A common fallacy is to assume authors of incomprehensible code will somehow be able to express themselves lucidly and clearly in comments.

Kevlin Henney

*https://twitter.com/KevlinHenney/status/38102180294190694*4


MIND THE GAP

Unsustainable Spacing

To be, or not to be: that is the question:
Whether 'tis nobler in the mind to suffer
The slings and arrows of outrageous fortune,
Or to take arms against a sea of troubles,
And by opposing end them?

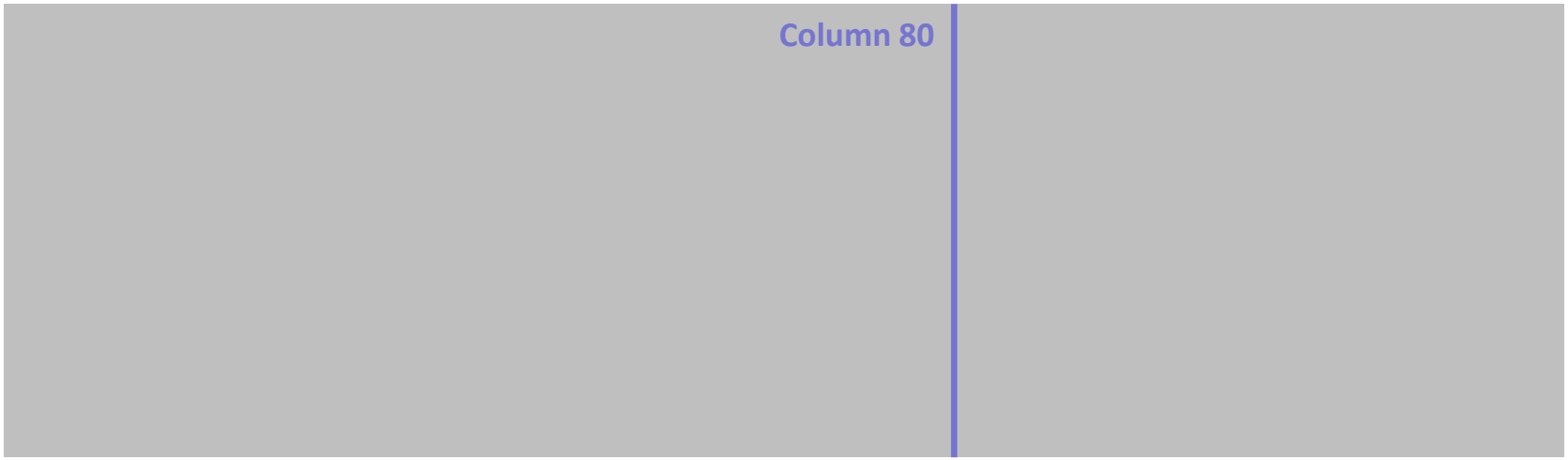
William Shakespeare
Hamlet

Continuing existence or cessation of existence: those are the scenarios. Is it more empowering mentally to work towards an accommodation of the downsizings and negative outcomes of adversarial circumstance, or would it be a greater enhancement of the bottom line to move forwards to a challenge to our current difficulties, and, by making a commitment to opposition, to effect their demise?

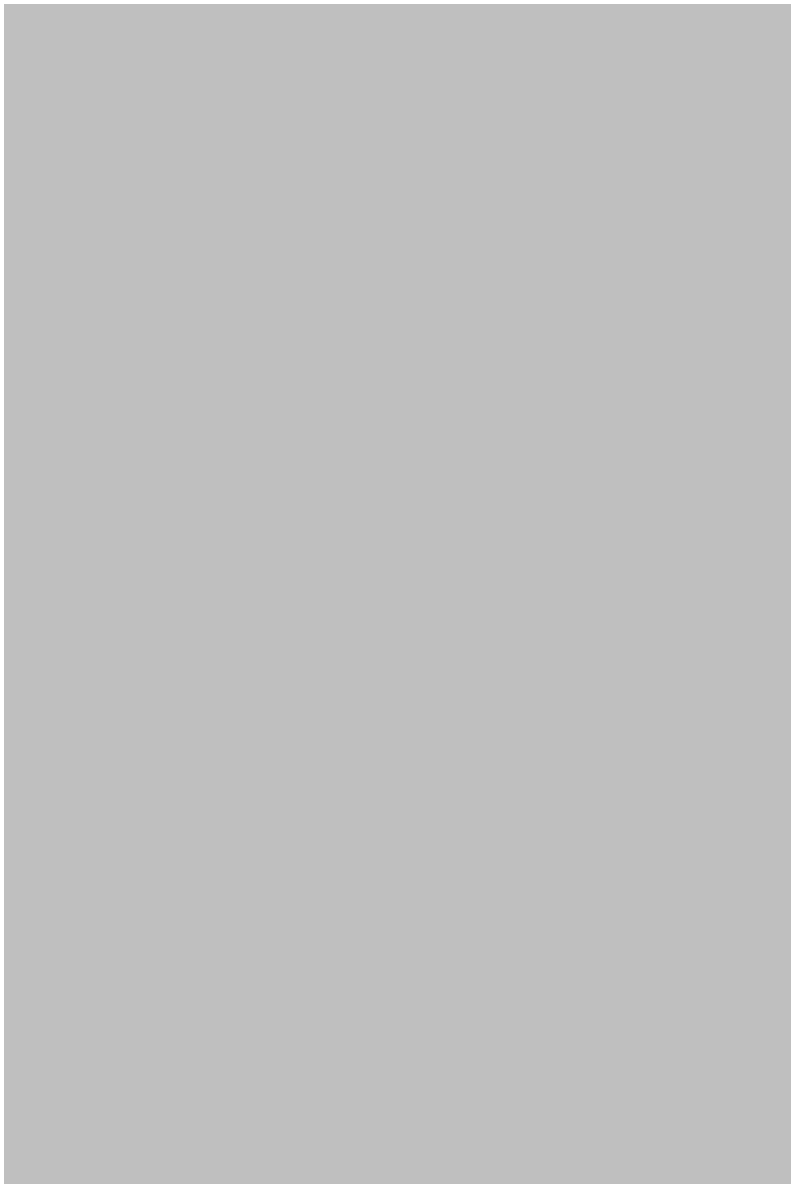
Tom Burton
Long Words Bother Me

Continuing existence or cessation of existence:

those are the
more empow
to work towa
accommodat
downsizings
outcomes of
circumstanc
a greater enh
the bottom li
forwards to a
our current d
by making a
opposition, t
demise?



The way many programmers lay out their code



The way people read

To answer the question "What is clean design?" most succinctly: a clean design is one that supports visual thinking so people can meet their informational needs with a minimum of conscious effort.

You convey information by the way you arrange a design's elements in relation to each other. This information is understood immediately, if not consciously, by the people viewing your designs.

This is great if the visual relationships are obvious and accurate, but if they're not, your audience is going to get confused. They'll have to examine your work carefully, going back and forth between the different parts to make sure they understand.

```
public int howNotToLayoutAMethodHeader(int firstArgument,
    String secondArgument)

public int ensureArgumentsAreAlignedLikeThis(
    int firstArgument,
    String secondArgument)

public int orEnsureArgumentsAreGroupedLikeThis(
    int firstArgument, String secondArgument)

public int butNotAlignedLikeThis(int firstArgument,
                                 String secondArgument)
```

```
int doNotFormat = likeThis(someArgumentOrExpression,
                           anotherArgumentOrExpression);


int insteadFormat =
    somethingLikeThis(
        someArgumentOrExpression,
        anotherArgumentOrExpression);


int orFormat = somethingLikeThis(
        someArgumentOrExpression,
        anotherArgumentOrExpression);
```

```
int asItIs = unstable(someArgumentOrExpression,
                      anotherArgumentOrExpression);


int butThisIs =
    stable(
        someArgumentOrExpression,
        anotherArgumentOrExpression);


int andThisIs = stable(
        someArgumentOrExpression,
        anotherArgumentOrExpression);
```

```
public ResultType arbitraryMethodName(FirstArgumentType firs
                                      SecondArgumentType sec
                                      ThirdArgumentType thir
    LocalVariableType localVariable = method(firstArgument,
                                             secondArgument)
    if (localVariable.isSomething(thirdArgument,
                                  SOME_SHOUTY_CONSTANT)) {
        doSomethingWith(localVariable);
    }
    return localVariable.getSomething();
}
```

```
public ResultType arbitraryMethodName(
    FirstArgumentType firstArgument,
    SecondArgumentType secondArgument,
    ThirdArgumentType thirdArgument) {
    LocalVariableType localVariable =
        method(firstArgument, secondArgument);
    if (localVariable.isSomething(
        thirdArgument, SOME_SHOUTY_CONSTANT)) {
        doSomething(localVariable);
    }
    return localVariable.getSomething();
}
```

```
public ResultType arbitraryMethodName(
        FirstArgumentType firstArgument,
        SecondArgumentType secondArgument,
        ThirdArgumentType thirdArgument) {
    LocalVariableType localVariable =
        method(firstArgument, secondArgument);
    if (localVariable.isSomething(
        thirdArgument, SOME_SHOUTY_CONSTANT)) {
        doSomething(localVariable);
    }
    return localVariable.getSomething();
}
```

```
public ResultType arbitraryMethodName(
    FirstArgumentType firstArgument,
    SecondArgumentType secondArgument,
    ThirdArgumentType thirdArgument)
{
    LocalVariableType localVariable =
        method(firstArgument, secondArgument);
    if (localVariable.isSomething(
        thirdArgument, SOME_SHOUTY_CONSTANT))
    {
        doSomething(localVariable);
    }
    return localVariable.getSomething();
}
```

```
class AttributedBody extends Body
    implements Attributed
{
    AttributedImpl attrImpl = new AttributedImpl();
    ...
}
```

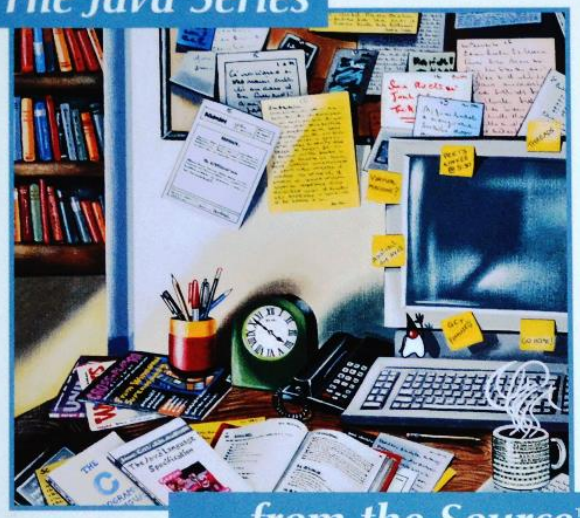
```
public static String quotedString(
    String from, char start, char end)
{
    int startPos = from.indexOf(start);
    int endPos = from.lastIndexOf(end);
    if (startPos == -1)      // no start found
        return null;
    else if (endPos == -1)   // no end found
        return from.substring(startPos);
    else                     // both start and end found
        return from.substring(startPos, endPos + 1);
}
```

```
public void replaceValue(String name, Object newValue)
    throws NoSuchAttributeException
{
    Attr attr = find(name);          // lookup the attr
    if (attr == null)                // it isn't found
        throw new NoSuchAttributeException(name, this);
    attr.valueOf(newValue);
}
```

Ken Arnold • James Gosling

The Java™ Programming Language

The Java Series



... from the Source™

JAVA


MONOGRAPHS IN COMPUTER SCIENCE
A THEORY OF OBJECTS
Springer
MARTÍN ABADI
LUCA CARDELLI

Lego Naming


MONOGRAPHS IN COMPUTER SCIENCE
A THEORY OF OBJECTS
MARTÍN ABADI
LUCA CARDELLI
Springer

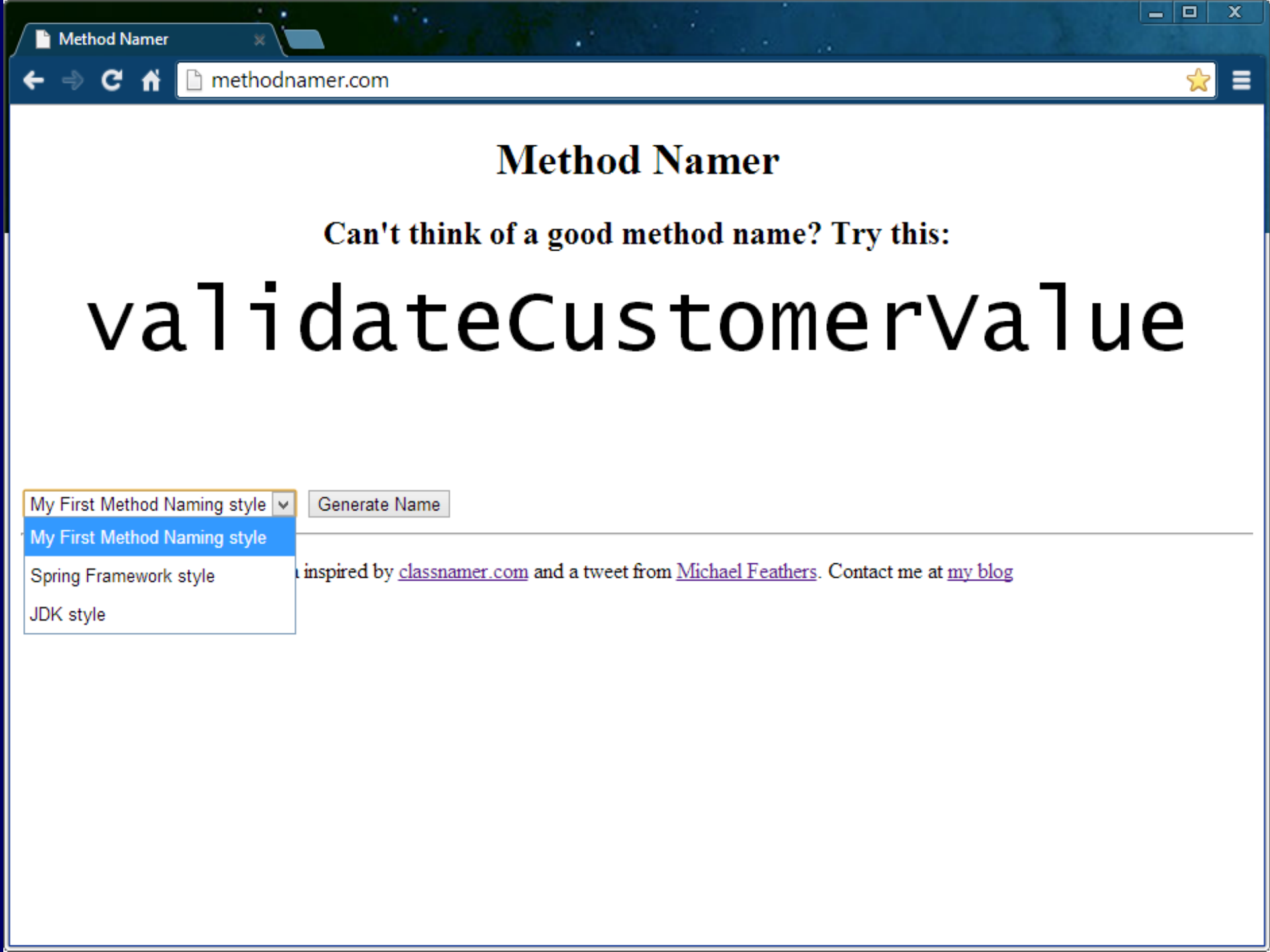
Agglutination is a process in linguistic morphology derivation in which complex words are formed by stringing together morphemes, each with a single grammatical or semantic meaning. Languages that use agglutination widely are called agglutinative languages.

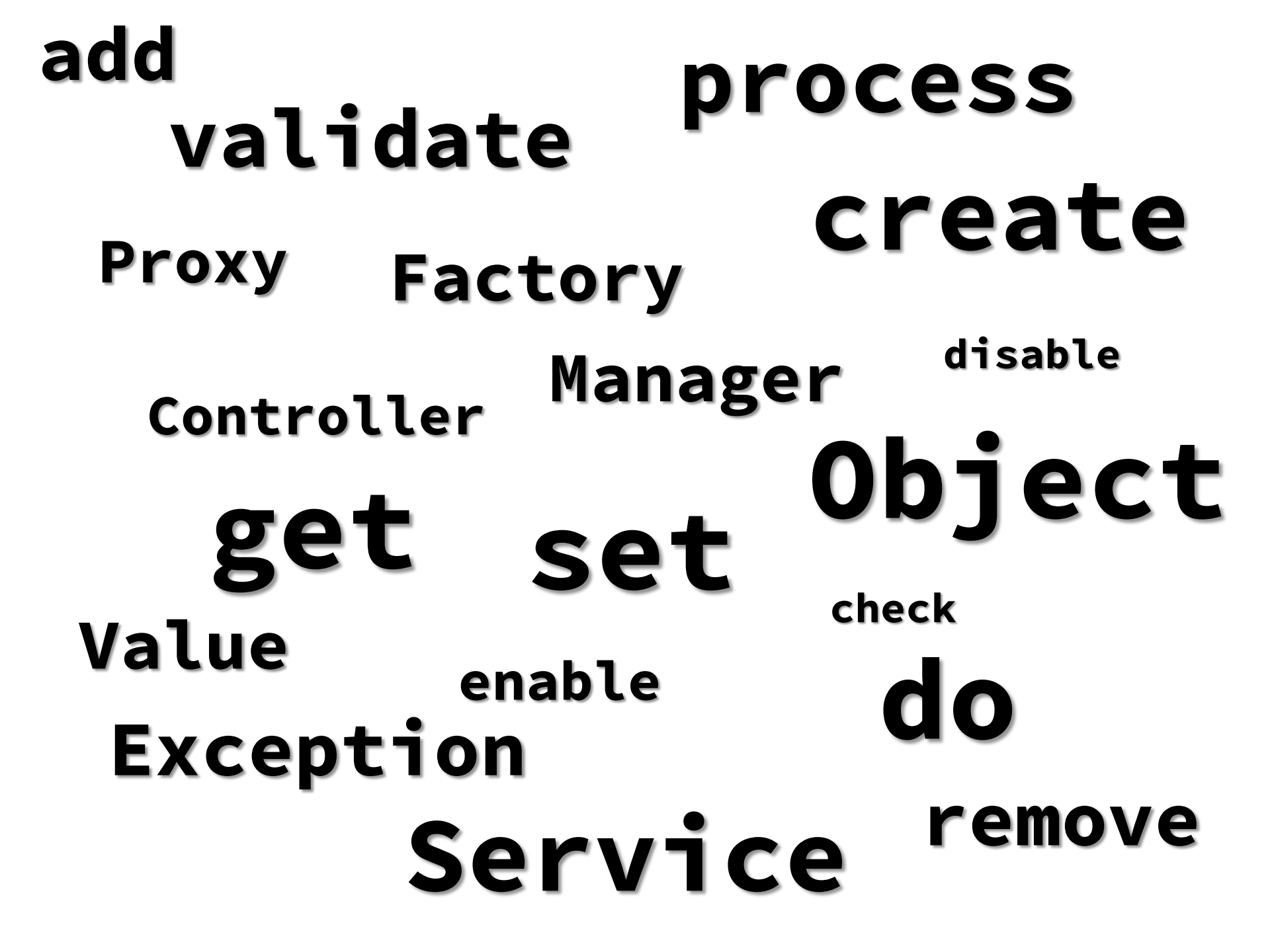
http://en.wikipedia.org/wiki/Agglutination

pneumonoultramicroscopicsilicovolcanoconiosis

fylkestrafikksikkerhetsutvalgssekretariatslederfunksjonene

Rindfleischetikettierungsüberwachungsaufgabenübertragungsgesetz


Method Namer
methodnamer.com
Method Namer
Can't think of a good method name? Try this:
validateCustomerValue
My First Method Naming style
Generate Name
My First Method Naming style
Spring Framework style
JDK style
inspired by classnamer.com and a tweet from Michael Feathers. Contact me at my blog


add
process
validate
create
Proxy
Factory
disable
Manager
Controller
Object
get
set
check
Value
enable
do
Exception
remove
Service

```
public interface BookEntity ...
public interface BookEntityFactory ...
public interface ISBNValidator ...
public interface CatalogueRepository ...
public interface CatalogueRepositoryProvider ...
public abstract class AbstractBookEntity
    implements BookEntity
public class BookEntityImpl
    extends BookEntity ...
public class BookEntityFactoryImpl
    implements BookEntityFactory ...
public class ISBNValidatorImpl
    implements ISBNValidator ...
public class CatalogueRepositoryImpl
    implements CatalogueRepository ...
public class CatalogueRepositoryProviderImpl
    implements CatalogueRepositoryProvider ...
```

```
public interface Book ...
public final class ISBN ...
public interface Catalogue ...
public class DescriptionOfCatalogueImplementation
    implements Catalogue ...
```

```
public interface ConditionChecker
{
    boolean checkCondition();
}
```

```
public interface Condition
{
    boolean isTrue();
}
```

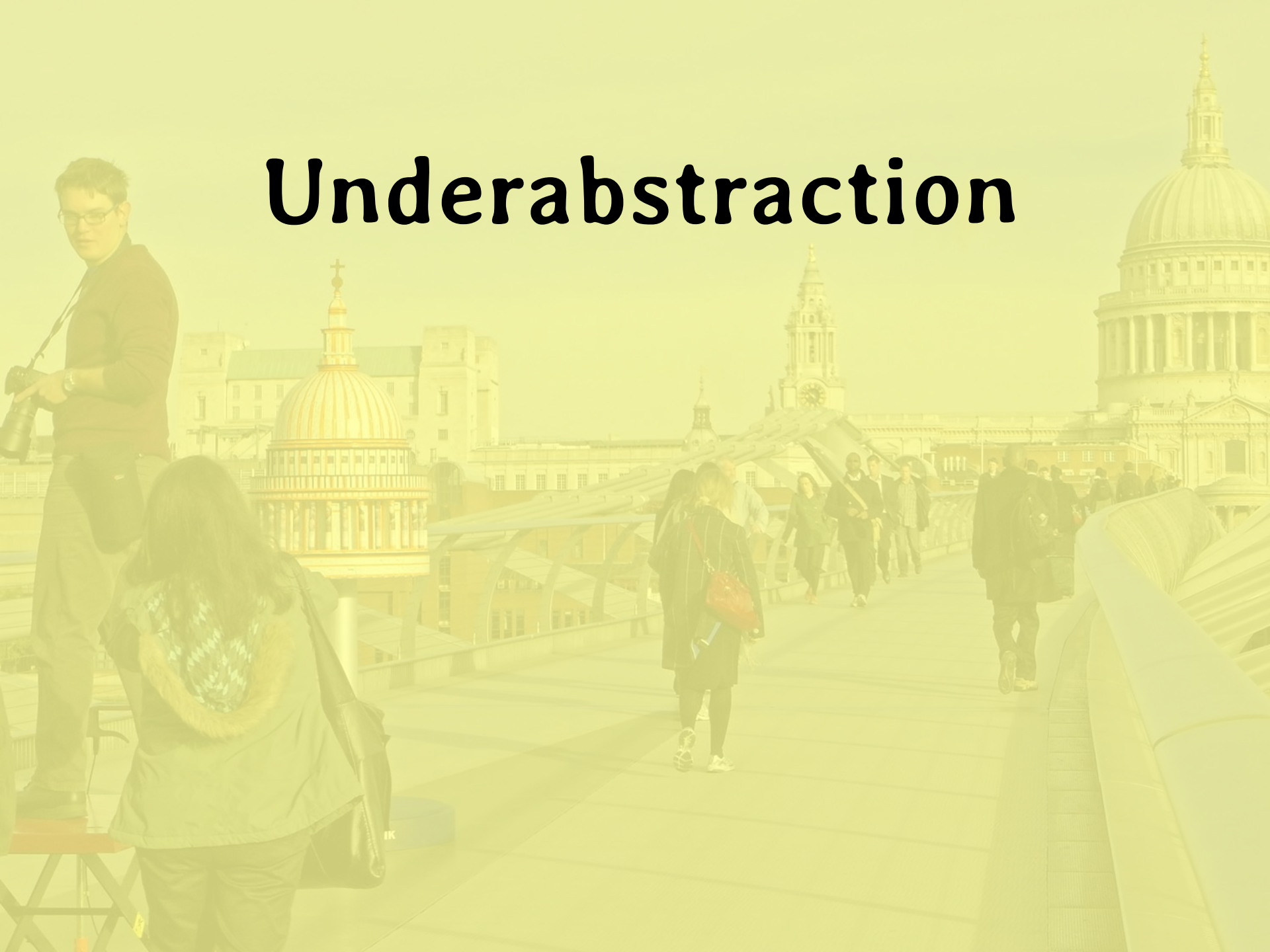
```
public Connection createConnection(Provider...)
    throws ConnectionFailureException
...
```

```
public Connection connectTo(Provider...)
    throws ConnectionFailure
...
```

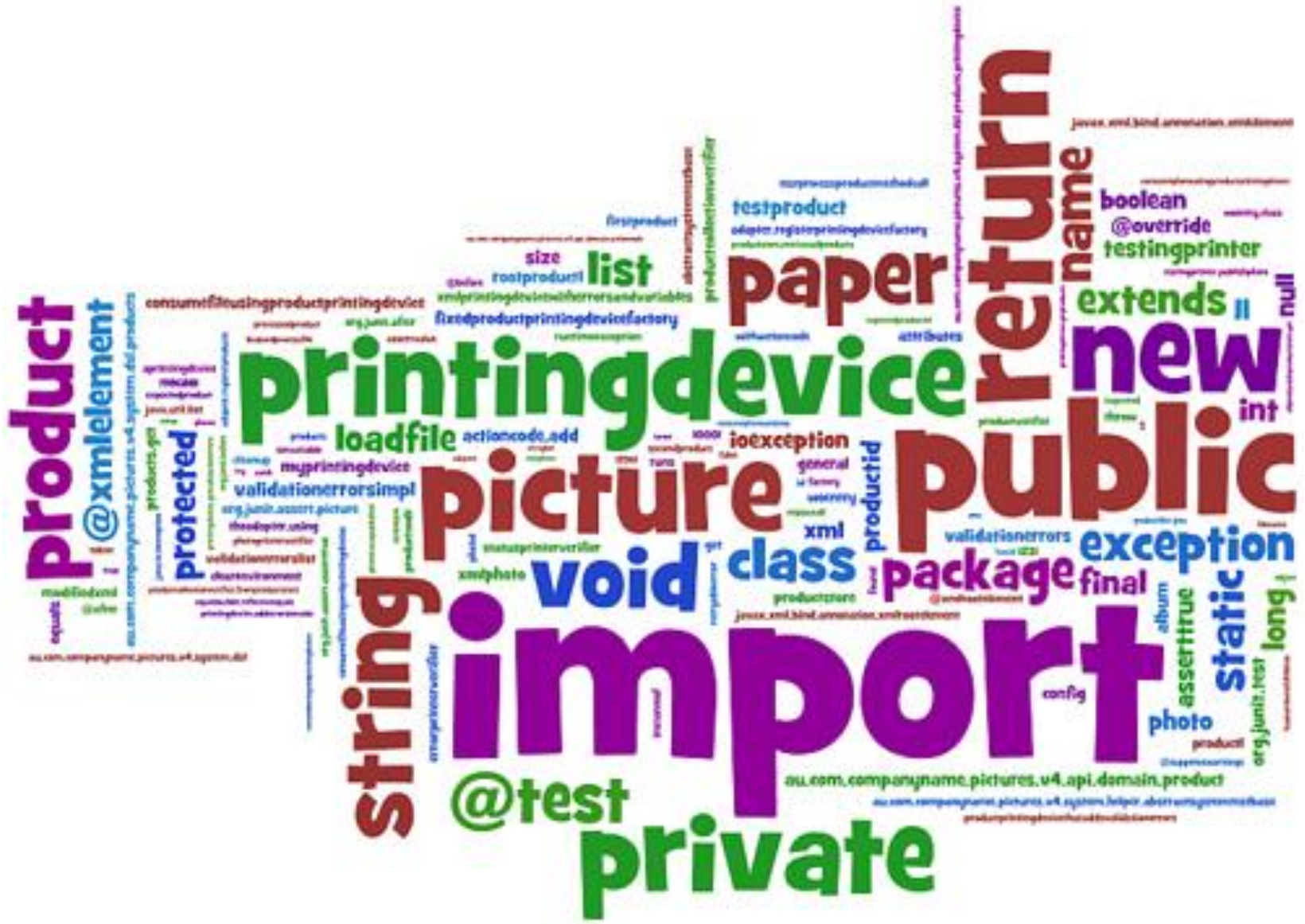
Omit needless words.

William Strunk and E B White
The Elements of Style

Underabstraction



http://fragmental.tw/2009/04/29/tag-clouds-see-how-noisy-your-code-is/



http://fragmental.tw/2009/04/29/tag-clouds-see-how-noisy-your-code-is/

```
if (portfolioIdsByTraderId.get(trader.getId())
    .containsKey(portfolio.getId()))
{
    ...
}
```

Dan North, "Code in the Language of the Domain"
97 Things Every Programmer Should Know

```
if (trader.canView(portfolio))
{
    ...
}
```

Dan North, "Code in the Language of the Domain"
97 Things Every Programmer Should Know

```
parser.processNodes(text, false);
```

Gregor Hohpe, "Convenience Is Not an -ility"
97 Things Every Programmer Should Know

If you have a procedure with ten parameters, you probably missed some.

Alan Perlis


THE SINGLETON
Single Malt Scotch Whisky
of Dufftown
DUFFTOWN
ESTD 1896
MATURED FOR
12 YEARS
12
PRODUCT OF
SCOTLAND
PERFECTLY BALANCED, NATURALLY RICH & SMOOTH
DUFFTOWN DISTILLERY, DUFFTOWN, KEITH
1litre

Unencapsulated State

encapsulate *enclose (something) in or as if in a capsule.*

- *express the essential feature of (someone or something) succinctly.*
- *enclose (a message or signal) in a set of codes which allow use by or transfer through different computer systems or networks.*
- *provide an interface for (a piece of software or hardware) to allow or simplify access for the user.*

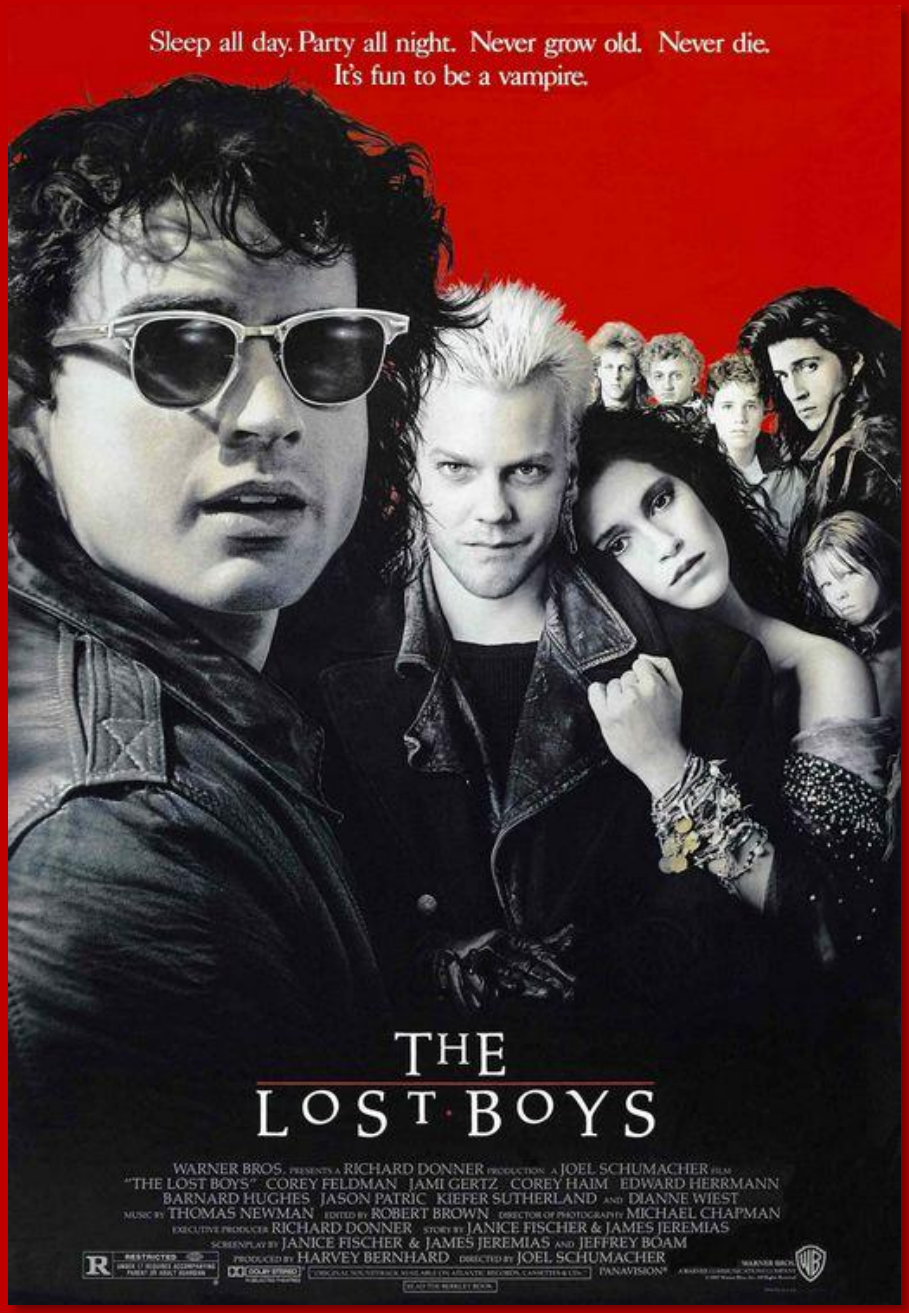
The New Oxford Dictionary of English

An *affordance* is a quality of an object, or an environment, which allows an individual to perform an action. For example, a knob affords twisting, and perhaps pushing, while a cord affords pulling.

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

```
public class RecentlyUsedList
{
    private List<String> items = new ArrayList<String>();
    public List<String> getList()
    {
        return items;
    }
    public void add(String newItem)
    {
        if(newItem == null)
            throw new IllegalArgumentException();
        items.remove(newItem);
        items.add(0, newItem);
    }
    ...
}
```

```
public class RecentlyUsedList
{
    private List<String> items = new ArrayList<String>();
    public List<String> getList()
    {
        return items;
    }
    public void add(String newItem)
    {
        if(newItem == null)
            throw new IllegalArgumentException();
        items.remove(newItem);
        items.add(0, newItem);
    }
    ...
}
```



Don't ever invite a vampire into your house, you silly boy. It renders you powerless.

```
public class RecentlyUsedList
{
    private List<String> items = new ArrayList<String>();
    public int isEmpty()
    {
        return items.isEmpty()
    }
    public int size()
    {
        return items.size();
    }
    public String get(int index)
    {
        return items.get(index);
    }
    public void add(String newItem)
    {
        if(newItem == null)
            throw new IllegalArgumentException();
        items.remove(newItem);
        items.add(0, newItem);
    }
    ...
}
```

```
public class RecentlyUsedList
{
    private List<String> items = new ArrayList<String>();
    public int isEmpty()
    {
        return items.isEmpty()
    }
    public int size()
    {
        return items.size();
    }
    public String get(int index)
    {
        return items.get(index);
    }
    public void add(String newItem)
    {
        if(newItem == null)
            throw new IllegalArgumentException();
        items.remove(newItem);
        items.add(0, newItem);
    }
    ...
}
```

```
public class RecentlyUsedList
{
    private List<String> items = new ArrayList<String>();
    public int isEmpty()
    {
        return items.isEmpty()
    }
    public int size()
    {
        return items.size();
    }
    public String get(int index)
    {
        return items.get(size() – index - 1);
    }
    public void add(String newItem)
    {
        if(newItem == null)
            throw new IllegalArgumentException();
        items.remove(newItem);
        items.add(newItem);
    }
    ...
}
```


STRICTLY
PRIVATE

Do not use *new* in constructors?

It's OK to use *new* in constructors.

Do not use *throw* in constructors?

It's OK to use *throw* in constructors.


beyond
SHARED PATH
Please consider
other path users
THIS IS A WORK
ALL USERS SHO
CYCLISTS ARE A
No liability will be accep

Getters and Setters


OED
Oxford English Dictionary
Dictionary
Advanced Search
Results
History
Bookmarks
Options
Help
Home
get
✓ PRONUNCIATION
✓ SPELLINGS
✓ ETYMOLOGY
✓ QUOTATIONS
Find
get, n.1
get, n.2
get, n.3
get, v.
geta, n. pl.
Getan, a. (and n.
get-at-able, a.
get-away, getaw
gete, n.
gete, v.
gete
getee
geten
getenly, adv.
geterne
get-go, n.
gethe
gether, adv.
gethicall
Gethsemane
get, v.
(gɛt)
Pa. tense got (arch. gat). Pa. pple. got (gotten). Pres. pple. getting. Forms: inf. 3–4 geten, (5 getyn), 3–6 gete, (4 geit, geyt, gite, Sc. gat(e, 4–5 gyte, 6 Sc. gait), 3–7 gett, (4–6 gette, 4 gitte, 5 gytt, 9 dial. git), 3– get. pa. tense 3–7 gate, (3 gait, 4 get, pl. gaten, geton, -yn, geetun, getton, 5 geten), 3–6 gatt, (4–6 gatte), 3– gat, 6– got, (6 got(t)e). pa. pple. α. 3–5 geten, (3 ȝeten, getun, 4 getin, geteyn, giten, -in, gyten, -in, 4–6 getyn, 5 geton), 3–5 getten, (4–5 gettyn, 5 getton, 6 gitten), 4–6 gete, (4 i-gete, 5 y-gete, gyte), 4–6 gette, (5 y-gette), 5–6 gett, (5 get). β. 3–4 gotin, 3– 6 goten, (4 gotyn, gote, 5 y-goten, goton, gothen), 4–6 Sc. gottin, -yn, 5–7 gotton, 6– gotten, got, (6 y-got).
[a. ON. geta (gat, gátum, getenn) to get, obtain, to beget, also, to guess (Sw. gitta, Da. gide to be able or willing, MSw. gäta, Da. gjette to guess) = OE. -ȝietan (only in the compounds a-, be-, for-, ofer-, on-, under-ȝietan: see BEGET, FORGET), OFris. (ur-, for-)jeta, OS. (bi-, far-)getan (MDu. ver-gheten, Du. ver-geten), OHG. ge‹zced›‹zced›an, ke‹zced›‹zced›an (once in pple.
✗ SORTED BY DATE
Back
Lost for words
Copy
Print
Mark
Find in entry

set

✓ PRONUNCIATION ✓ SPELLINGS ✓ ETYMOLOGY ✓ QUOTATIONS

Find

set, *v.*[1]

(sɛt)

Forms: see below. Pa. tense and pple. **set**.

[Com. Teut.: OE. *sęttan* = OFris. *setta* (mod.Fris. *sette*), OS. *settian* (MDu., MLG. *setten*, Du. *zetten*), OHG. *sezzan* beside *sazzan* (MHG. *sezzen*, G. *setzen*), ON. *setja* (Sw. *satta*, Da. *sætte*), Goth. *satjan*; causal of **setjan* (*sitjan*) to SIT.

Confusion between *set* and *sit* arose as early as the beginning of the 14th c., owing partly to the identity or close similarity of the forms of their past tenses and pa. pples., and partly to the identity of meaning in some uses, as between *to be set* (= seated) and *to sit*; cf. SIT *v.* (etym. note and A. 5 a α note). For cases of mere substitution of forms of *sit* for forms of *set*, see A. 1 γ, 2 ζ below. The spelling *sett* is still sometimes found in technical senses; cf. SET *n.*[1]]

A. Inflexional Forms.

1. a. *inf.* and *pres. stem.* (α) 1 **settan** (*Northumb.* **setta**), 2–5 (6 *arch.*) **setten**, 3–6 **sette** (2 **setton**, **seotte**, 3 *Orm.* **settenn**, *Lay.* **sætten**, 4 *Kent.* **zetten**, 5 **settyn**, **cettyn**, **satte**, 6 **seatt-**), 4–9

reset

✓ PRONUNCIATION ✓ SPELLINGS ✓ ETYMOLOGY ✓ QUOTATIONS

Find

reset, *v.*[2]

(riːˈsɛt)

Also **re-set**.

[RE- 5 a.]

trans. To set again, in various senses of the verb.

I. 1. a. To replace (*esp.* gems) in a (former or new) setting.

1655 FULLER *Ch. Hist.* v. iv. §7 Elizabeth,..finding so fair a flower..fallen out of her Crown, was careful quickly to gather it up again, and get it re-sett therein. **1684** R. WALLER *Nat. Exper.* Pref., For a time they fall out of their Collets.., and [are] worth nothing till..they are again reset in their proper places. **1830** LYTTON *P. Clifford* xix, A stray trinket or two—not of sufficient worth to be re-set. **1883** HALDANE *Worksh. Rec.* Ser. II. 371/2 The hair can be again reset as firmly as it was before [etc.].

b. *Surg.* To set (a broken limb) again.

✗ SORTED BY DATE

Back | Lost for words | Copy | Print | Mark | Find in entry

unset

✓ PRONUNCIATION ✓ SPELLINGS ✓ ETYMOLOGY ✓ QUOTATIONS

Find

un'set, *v.*

[UN-[2] 3, 7. Cf. OE. *unsettan* (once), to take down.]

1. *trans.* To put out of place or position; to undo the setting of.

1602 MARSTON *Ant. & Mel.* III. Wks. 1856 I. 37 O, you spoyle my ruffe, unset my haire. **1611** COTGR., *Desplanter*,..to vnplant, vnset, remoue. **1761** GRAY *Lett.* (1900) II. 204 The man was sent for: he unset it; it was a paste not worth 40 shillings. **1775** MRS. DELANY in *Life & Corr.* Ser. II. (1862) II. 105 There is some hazard in unsetting enamel for fear of chipping the edges. **1836** MARRYAT *Midsh. Easy* xxxii, How could he put the young men to fresh tortures by removing splints and unsetting limbs? **1884** *Law Times* 1 Nov. 8/1 On the morning in question Dawson had unset the gun.

2. *intr.* To get out of place or position.

1703 THORESBY *Let. to Ray*, *Spelk*, a wooden splinter tied on, to keep a broken bone from bending or unsetting again.

✗ SORTED BY DATE

Back | Lost for words | Copy | Print | Mark | Find in entry

```
public final class Date implements ...
{
    ...
    public int getYear() ...
    public int getMonth() ...
    public int getDayInMonth() ...
    public void setYear(int newYear) ...
    public void setMonth(int newMonth) ...
    public void setDayInMonth(int newDayInMonth) ...
    ...
}
```

```
public final class Date implements ...
{
    ...
    public int getYear() ...
    public int getMonth() ...
    public int getWeekInYear() ...
    public int getDayInYear() ...
    public int getDayInMonth() ...
    public int getDayInWeek() ...
    public void setYear(int newYear) ...
    public void setMonth(int newMonth) ...
    public void setWeekInYear(int newWeek) ...
    public void setDayInYear(int newDayInYear) ...
    public void setDayInMonth(int newDayInMonth) ...
    public void setDayInWeek(int newDayInWeek) ...
    ...
}
```

When it is not necessary to change, it is necessary not to change.

Lucius Cary

```
public final class Date implements ...
{
    ...
    public int getYear() ...
    public int getMonth() ...
    public int getWeekInYear() ...
    public int getDayInYear() ...
    public int getDayInMonth() ...
    public int getDayInWeek() ...
    ...
}
```

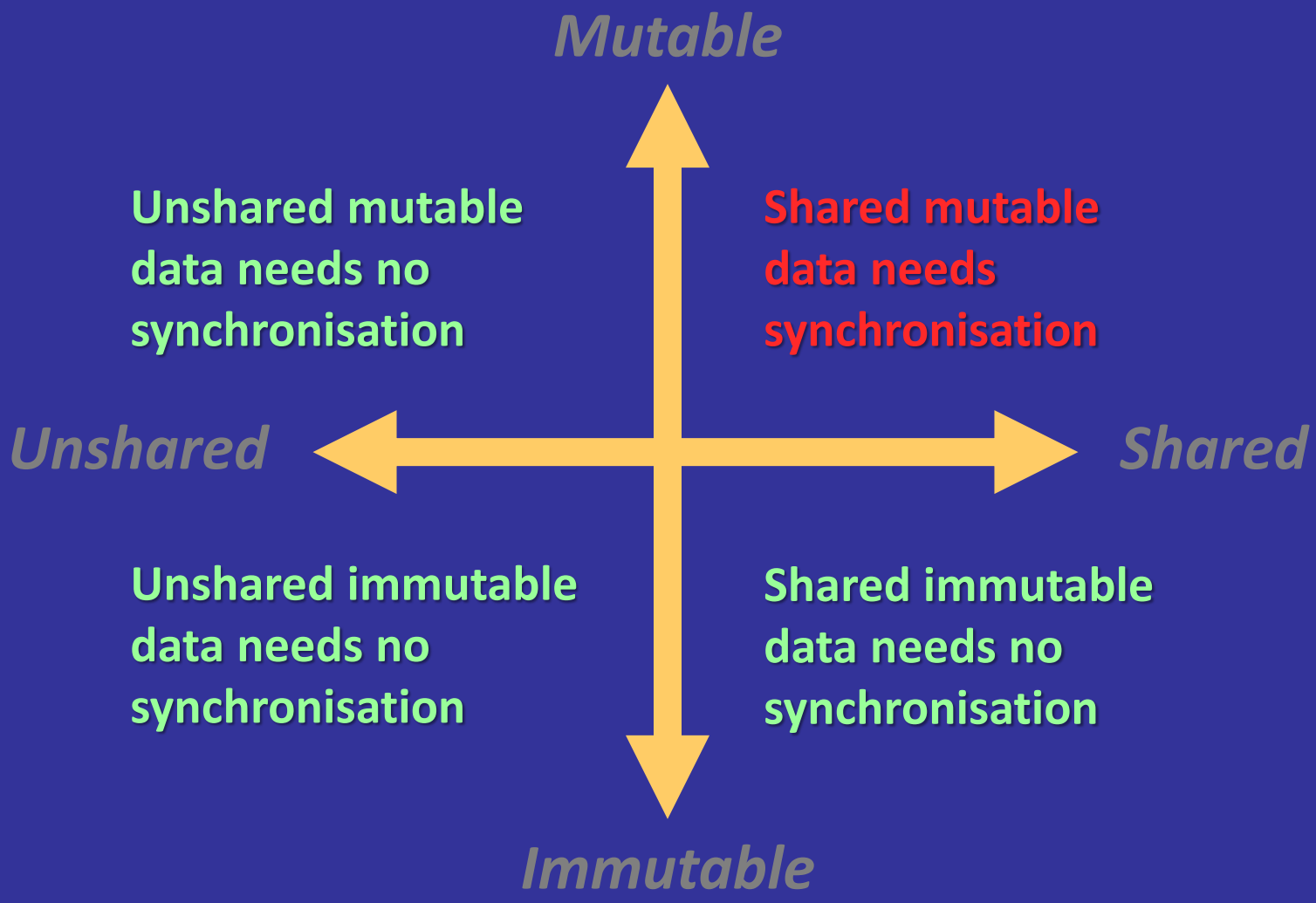
```
public final class Date implements ...
{
    ...
    public int year() ...
    public int month() ...
    public int weekInYear() ...
    public int dayInYear() ...
    public int dayInMonth() ...
    public int dayInWeek() ...
    ...
}
```

```
public final class Date implements ...
{
    ...
    public int year() ...
    public Month month() ...
    public int weekInYear() ...
    public int dayInYear() ...
    public int dayInMonth() ...
    public DayInWeek dayInWeek() ...
    ...
}
```

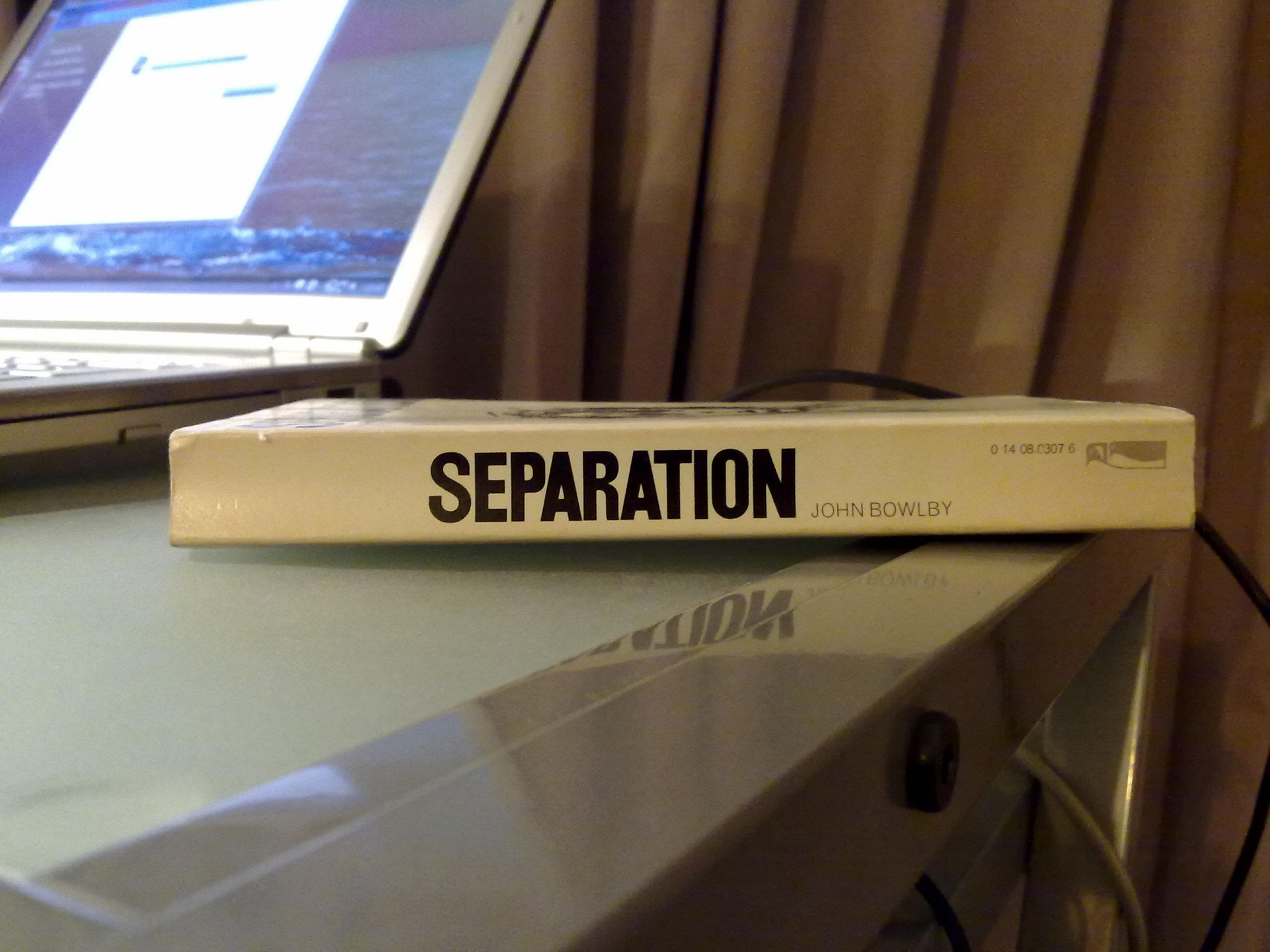
Some people, when confronted with a problem, think, "I know, I'll use threads," and then two they hav erpoblesms.

Ned Batchelder

https://twitter.com/#!/nedbat/status/19487382982532710 4



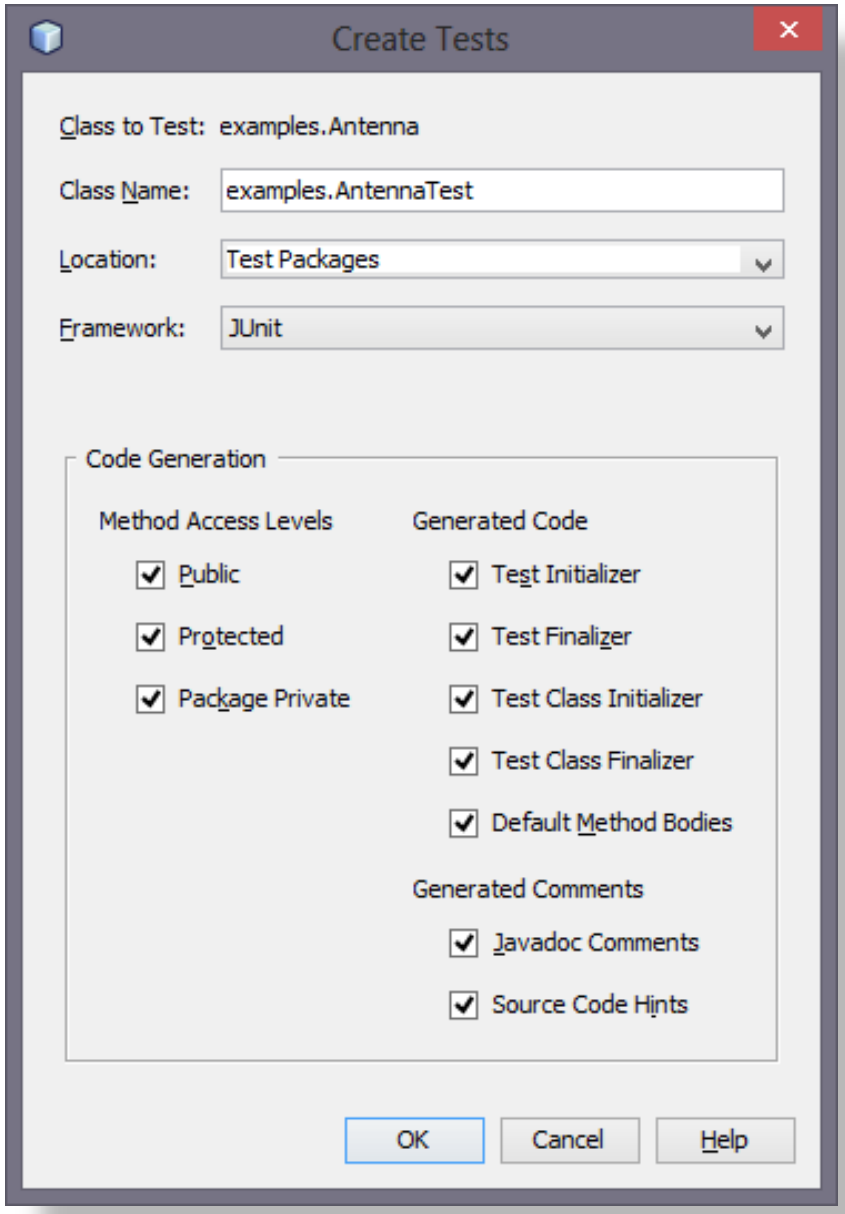
Synchronisation Quadrant


SEPARATION
JOHN BOWLBY
0 14 08 0307 6

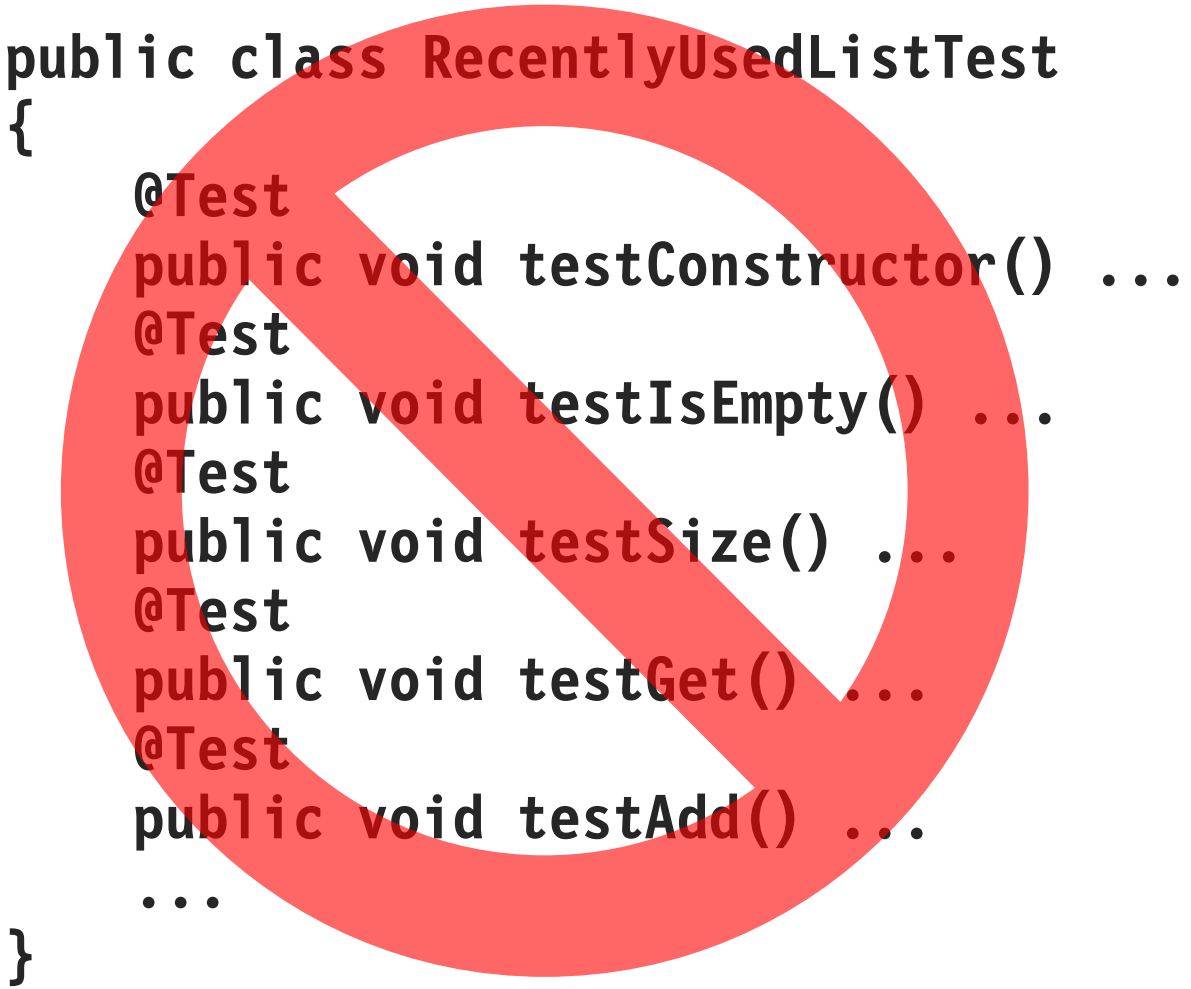
Uncohesive Tests

Everybody knows that TDD stands for Test Driven Development. However, people too often concentrate on the words "Test" and "Development" and don't consider what the word "Driven" really implies. For tests to drive development they must do more than just test that code performs its required functionality: they must clearly express that required functionality to the reader. That is, they must be clear specifications of the required functionality. Tests that are not written with their role as specifications in mind can be very confusing to read.

Nat Pryce and Steve Freeman
"Are Your Tests Really Driving Your Development?"


Create Tests
Class to Test: examples.Antenna
Class Name: examples.AntennaTest
Location: Test Packages
Framework: JUnit
Code Generation
Method Access Levels
Public
Protected
Package Private
Generated Code
Test Initializer
Test Finalizer
Test Class Initializer
Test Class Finalizer
Default Method Bodies
Generated Comments
Javadoc Comments
Source Code Hints
OK
Cancel
Help

```
public class RecentlyUsedList
{
    ...
    public RecentlyUsedList() ...
    public boolean isEmpty() ...
    public int size() ...
    public String get(int index) ...
    public void add(String newItem) ...
    ...
}
```


public class RecentlyUsedListTest
{
@Test
public void testConstructor() ...
@Test
public void testIsEmpty() ...
@Test
public void testSize() ...
@Test
public void testGet() ...
@Test
public void testAdd() ...
...
}

```
public class RecentlyUsedList_spec
{
    public static class A_new_list
    {
        @Test public void is_empty() ...
    }
    public static class An_empty_list
    {
        @Test public void retains_a_single_addition() ...
        @Test public void retains_unique_additions_in_stack_order() ...
    }
    public static class A_non_empty_list
    {
        @Test public void is_unchanged_when_head_item_is_readded() ...
        @Test public void moves_non_head_item_to_head_when_it_is_readded() ...
    }
    public static class Any_list
    {
        @Test public void rejects_addition_of_null_items() ...
        @Test public void rejects_indexing_past_its_end() ...
        @Test public void rejects_negative_indexing() ...
    }
}
```

```
public class RecentlyUsedList_spec
{
    public static class A_new_list
    {
        @Test public void is_empty() ...
    }
    public static class An_empty_list
    {
        @Test public void retains_a_single_addition() ...
        @Test public void retains_unique_additions_in_stack_order() ...
    }
    public static class A_non_empty_list
    {
        @Test public void is_unchanged_when_head_item_is_readded() ...
        @Test public void moves_non_head_item_to_head_when_it_is_readded() ...
    }
    public static class Any_list
    {
        @Test public void rejects_addition_of_null_items() ...
        @Test public void rejects_indexing_past_its_end() ...
        @Test public void rejects_negative_indexing() ...
    }
}
```

A test case should be just that: it should correspond to a single case.

Style is the art of getting yourself out of the way, not putting yourself in it.

David Hare