ThoughtWorks

the productive programmer: practice

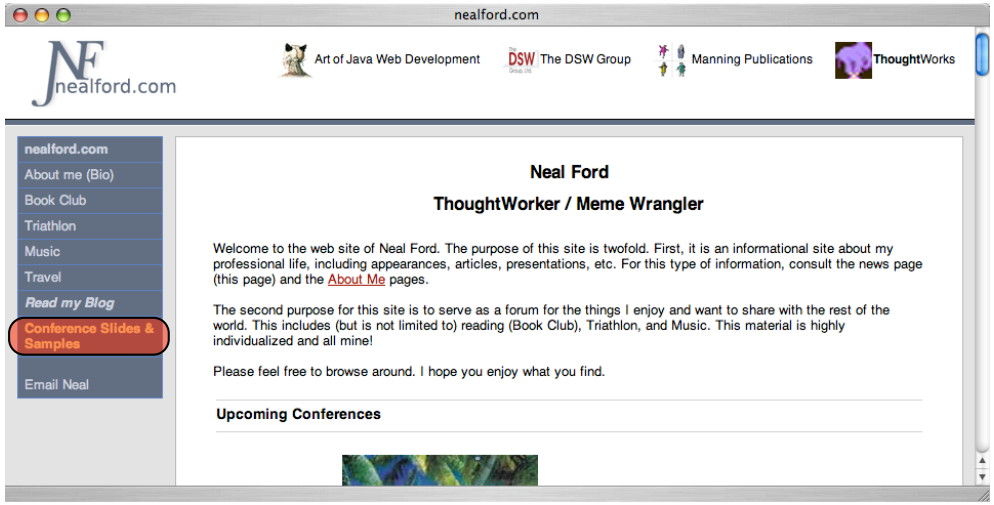
10 ways to improve your code

NEAL FORD software architect / meme wrangler

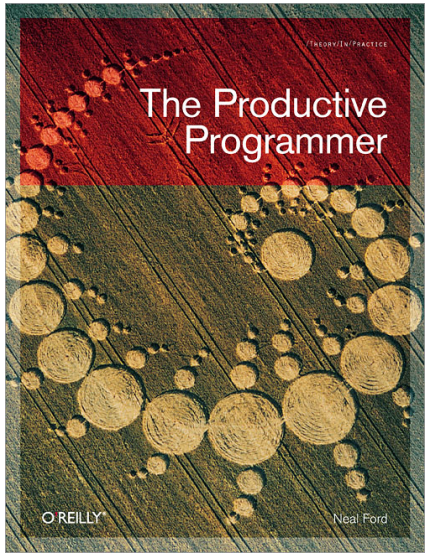
ThoughtWorks

nford@thoughtworks.com
3003 Summit Boulevard, Atlanta, GA 30319
www.nealford.com
www.thoughtworks.com
memeagora.blogspot.com

NF


nealford.com
nealford.com
Art of Java Web Development
The DSW Group
Manning Publications
ThoughtWorks
nealford.com
About me (Bio)
Book Club
Triathlon
Music
Travel
Read my Blog
Conference Slides & Samples
Email Neal
Neal Ford
ThoughtWorker / Meme Wrangler
Welcome to the web site of Neal Ford. The purpose of this site is twofold. First, it is an informational site about my professional life, including appearances, articles, presentations, etc. For this type of information, consult the news page (this page) and the About Me pages.
The second purpose for this site is to serve as a forum for the things I enjoy and want to share with the rest of the world. This includes (but is not limited to) reading (Book Club), Triathlon, and Music. This material is highly individualized and all mine!
Please feel free to browse around. I hope you enjoy what you find.
Upcoming Conferences

from whence?



2 parts:

mechanics

practices

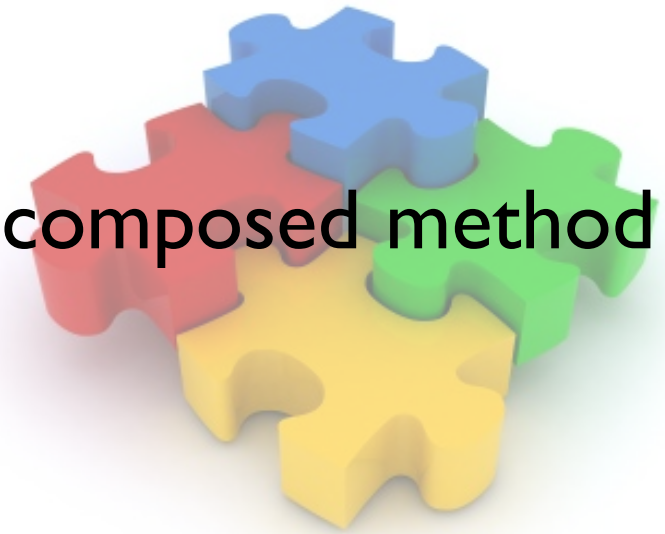
the unexamined life
is not worth living

socrates

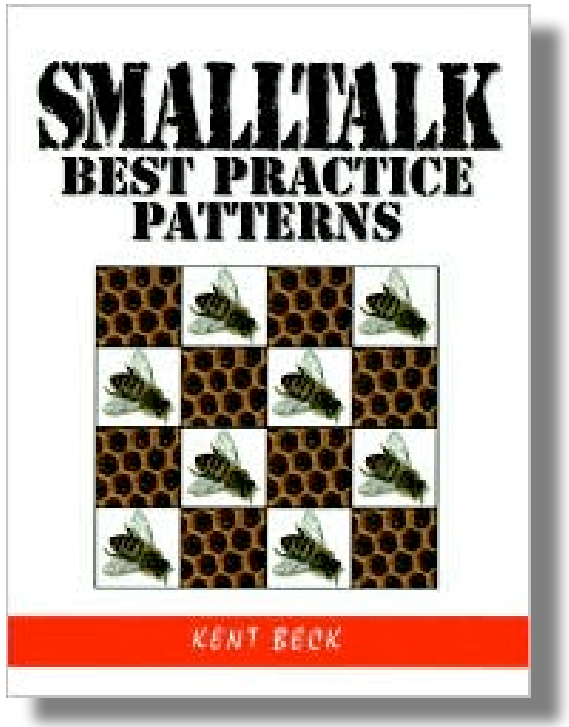
unexamined code isn't worth
executing

neal, stealing from socrates

1



composed method


SMALLTALK
BEST PRACTICE
PATTERNS
KENT BECK

composed method

Divide your program into methods that perform one identifiable task.

Keep all of the operations in a method at the same level of abstraction.

This will naturally result in programs with many small methods, each a few lines long.


refactoring to
composed method

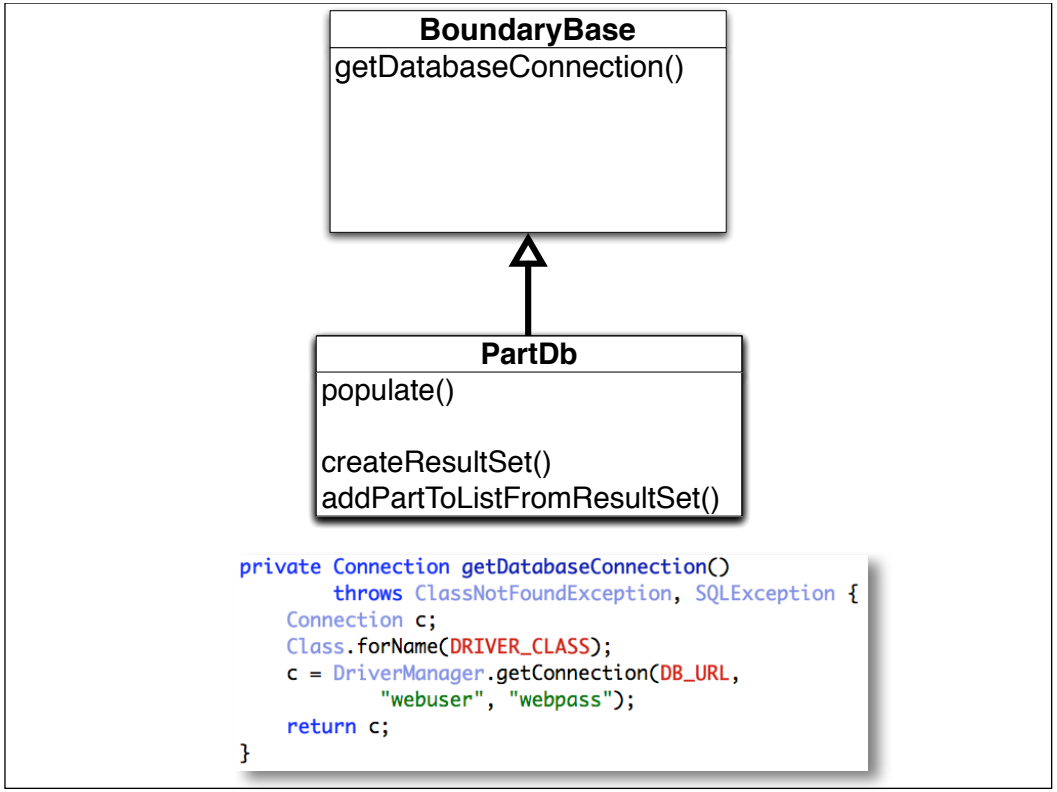
```
public void populate() throws Exception  {
    Connection c = null;
    try {
        Class.forName(DRIVER_CLASS);
        c = DriverManager.getConnection(DB_URL, USER, PASSWORD);
        Statement stmt = c.createStatement();
        ResultSet rs = stmt.executeQuery(SQL_SELECT_PARTS);
        while (rs.next()) {
            Part p = new Part();
            p.setName(rs.getString("name"));
            p.setBrand(rs.getString("brand"));
            p.setRetailPrice(rs.getDouble("retail_price"));
            partList.add(p);
        }
    } finally {
        c.close();
    }
}
```

```
private void addPartToListFromResultSet(ResultSet rs)
        throws SQLException {
    Part p = new Part();
    p.setName(rs.getString("name"));
    p.setBrand(rs.getString("brand"));
    p.setRetailPrice(rs.getDouble("retail_price"));
    partList.add(p);
}
```

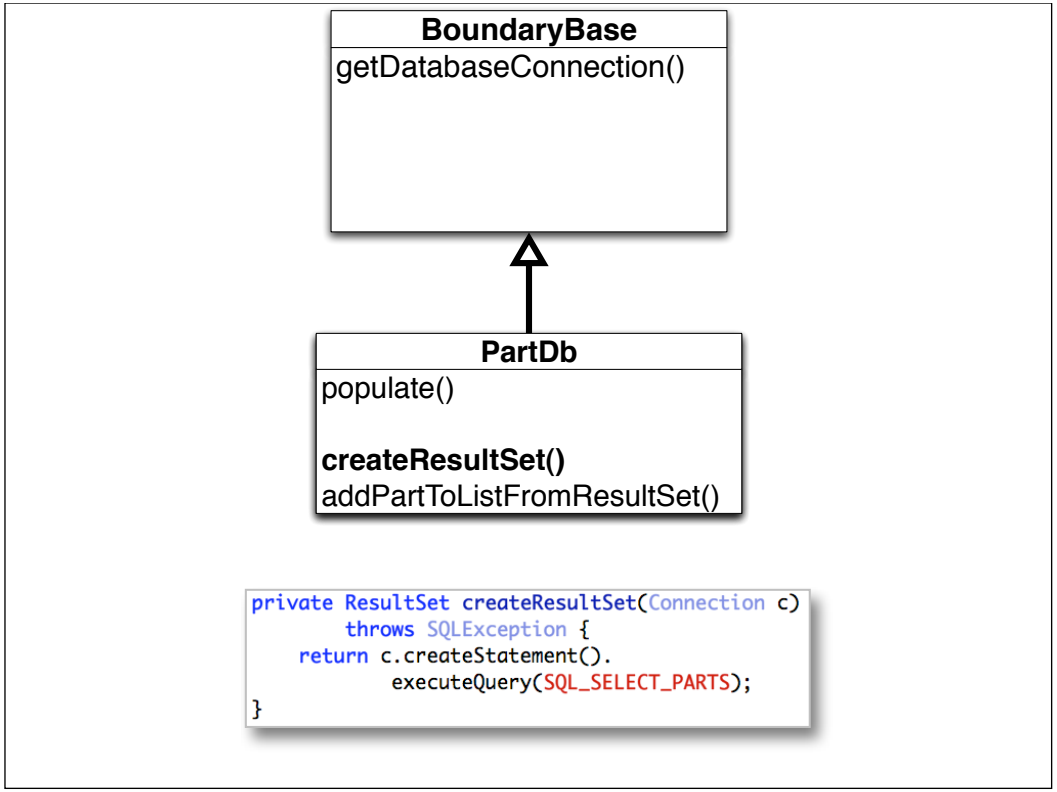
```
public void populate() throws Exception {
    Connection c = null;
    try {
        c = getDatabaseConnection();
        ResultSet rs = createResultSet(c);
        while (rs.next())
            addPartToListFromResultSet(rs);
    } finally {
        c.close();
    }
}
```

```
private ResultSet createResultSet(Connection c)
        throws SQLException {
    return c.createStatement().
            executeQuery(SQL_SELECT_PARTS);
}
```

```
private Connection getDatabaseConnection()
        throws ClassNotFoundException, SQLException {
    Connection c;
    Class.forName(DRIVER_CLASS);
    c = DriverManager.getConnection(DB_URL,
            "webuser", "webpass");
    return c;
}
```




```
private Connection getDatabaseConnection()
        throws ClassNotFoundException, SQLException {
    Connection c;
    Class.forName(DRIVER_CLASS);
    c = DriverManager.getConnection(DB_URL,
            "webuser", "webpass");
    return c;
}
```


BoundaryBase
getDatabaseConnection()
PartDb
populate()
createResultSet()
addPartToListFromResultSet()
private ResultSet createResultSet(Connection c)
throws SQLException {
return c.createStatement().
executeQuery(SQL_SELECT_PARTS);
}

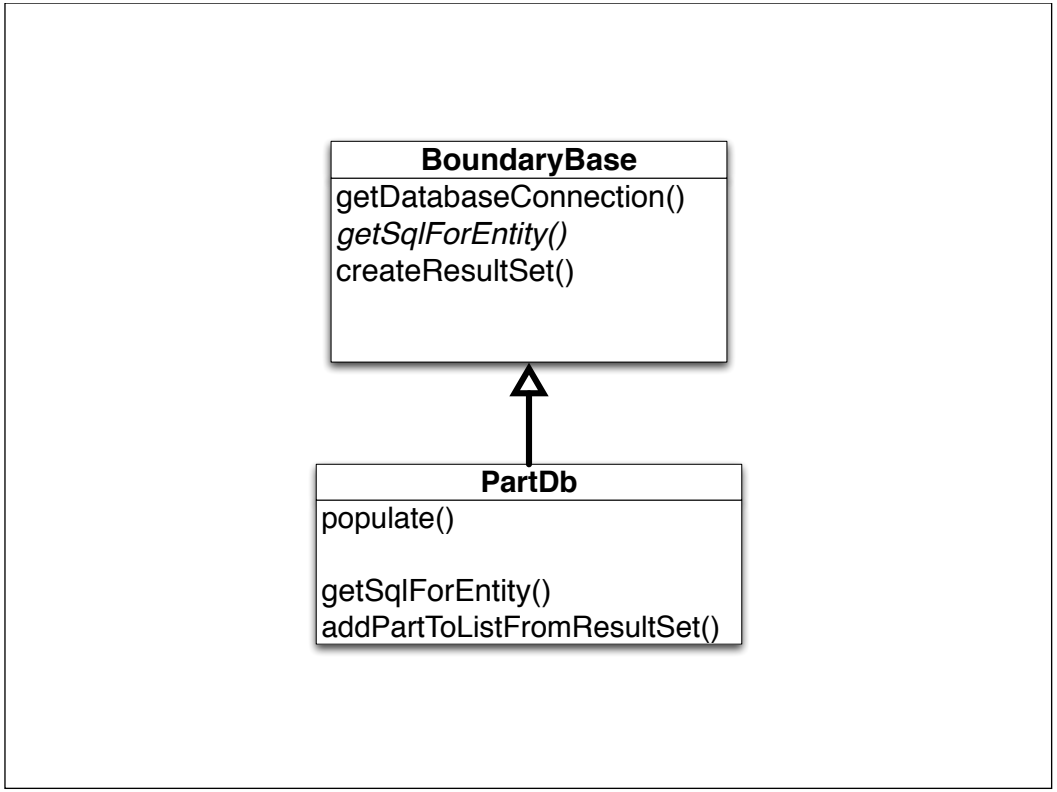
BoundaryBase

```
abstract protected String getSqlForEntity();

protected ResultSet createResultSet(Connection c) throws SQLException {
    Statement stmt = c.createStatement();
    return stmt.executeQuery(getSqlForEntity());
}
```

```
private ResultSet createResultSet(Connection c)
        throws SQLException {
    return c.createStatement().
            executeQuery(SQL_SELECT_PARTS);
}
```

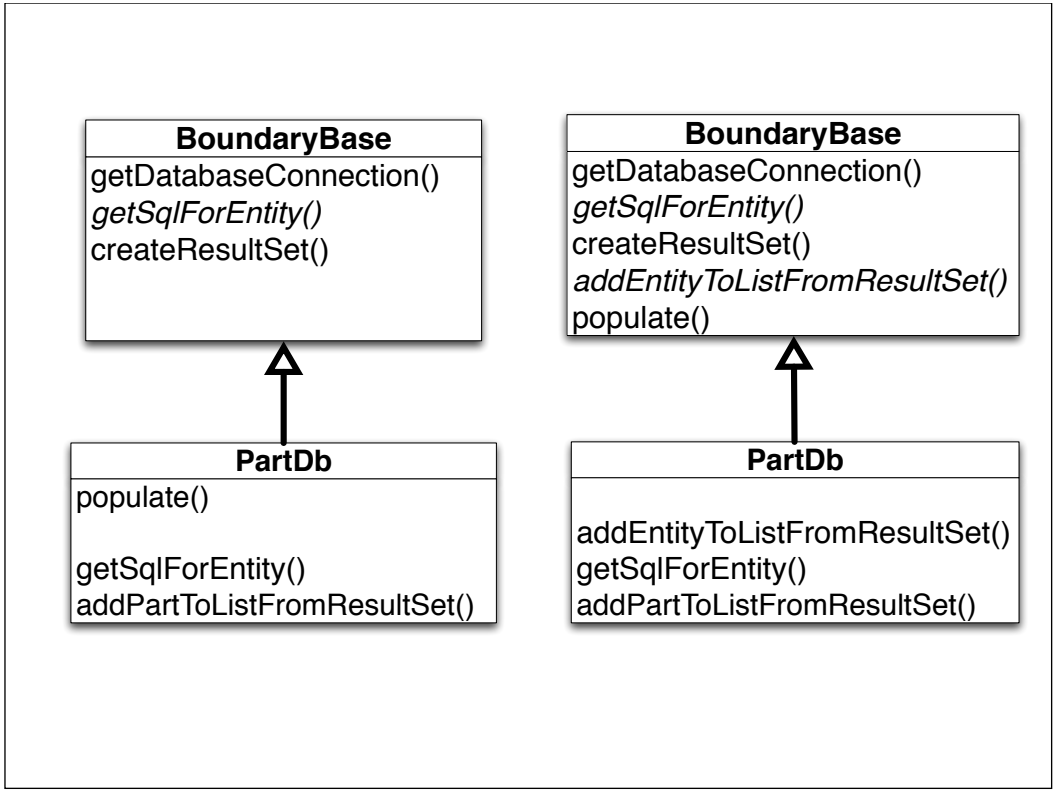
PartDb

```
protected String getSqlForEntity() {
    return SQL_SELECT_PARTS;
}
```


BoundaryBase
getDatabaseConnection()
getSqlForEntity()
createResultSet()
PartDb
populate()
getSqlForEntity()
addPartToListFromResultSet()

```
public void populate() throws Exception {
    Connection c = null;
    try {
        c = getDatabaseConnection();
        ResultSet rs = createResultSet(c);
        while (rs.next())
            addPartToListFromResultSet(rs);
    } finally {
        c.close();
    }
}
```

```
abstract protected void addEntityToListFromResultSet(ResultSet rs)
        throws SQLException;

public void populate() throws Exception {
    Connection c = null;
    try {
        c = getDatabaseConnection();
        ResultSet rs = createResultSet(c);
        while (rs.next())
            addEntityToListFromResultSet(rs);
    } finally {
        c.close();
    }
}
```


BoundaryBase
getDatabaseConnection()
getSqlForEntity()
createResultSet()
PartDb
populate()
getSqlForEntity()
addPartToListFromResultSet()
BoundaryBase
getDatabaseConnection()
getSqlForEntity()
createResultSet()
addEntityToListFromResultSet()
populate()
PartDb
addEntityToListFromResultSet()
getSqlForEntity()
addPartToListFromResultSet()

```
protected Connection getDatabaseConnection() throws ClassNotFoundException,
        SQLException {
    Connection c;
    Class.forName(DRIVER_CLASS);
    c = DriverManager.getConnection(DB_URL, "webuser", "webpass");
    return c;
}

abstract protected String getSqlForEntity();

protected ResultSet createResultSet(Connection c) throws SQLException {
    Statement stmt = c.createStatement();
    return stmt.executeQuery(getSqlForEntity());
}

abstract protected void addEntityToListFromResultSet(ResultSet rs)
        throws SQLException;

public void populate() throws Exception {
    Connection c = null;
    try {
        c = getDatabaseConnection();
        ResultSet rs = createResultSet(c);
        while (rs.next())
            addEntityToListFromResultSet(rs);
    } finally {
        c.close();
    }
}
```

BoundaryBase

PartDb

```
public Part[] getParts() {
    return (Part[]) partList.toArray(TEMPLATE);
}

protected String getSqlForEntity() {
    return SQL_SELECT_PARTS;
}

protected void addEntityToListFromResultSet(ResultSet rs) throws SQLException {
    Part p = new Part();
    p.setName(rs.getString("name"));
    p.setBrand(rs.getString("brand"));
    p.setRetailPrice(rs.getDouble("retail_price"));
    partList.add(p);
}
```

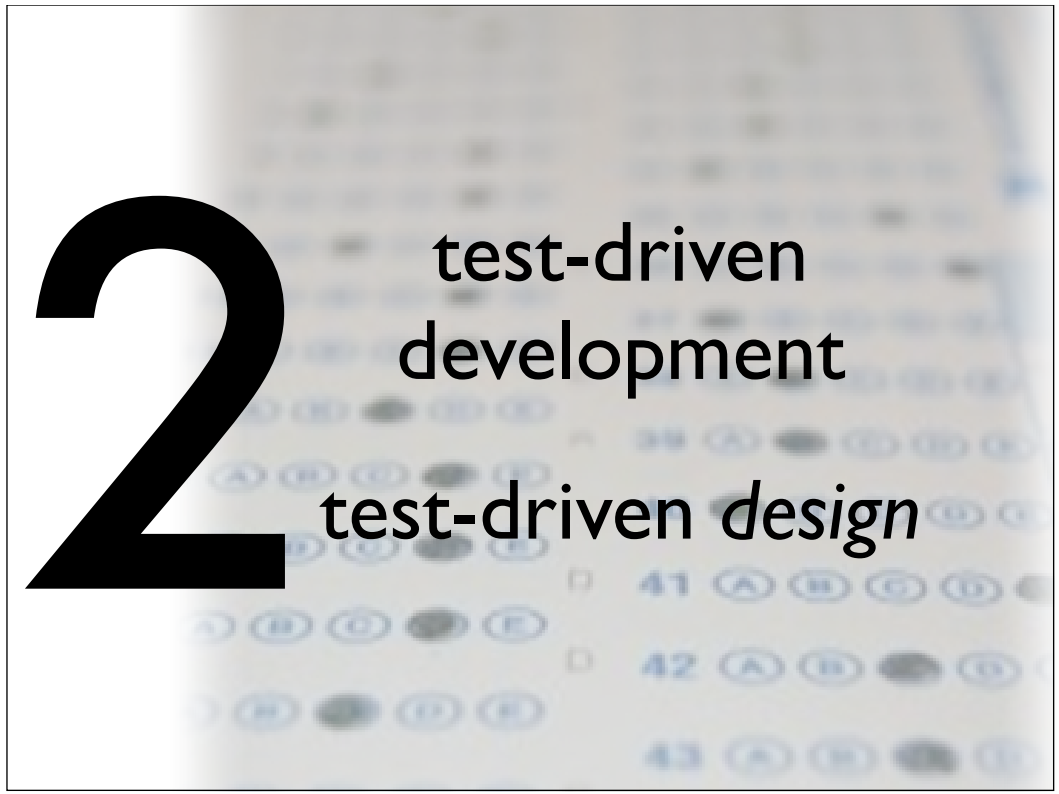
benefits of composed method

shorter methods easier to test

method names become documentation

large number of very cohesive methods

discover reusable assets that you didn't know were there


2
test-driven
development
test-driven *design*

design benefits of tdd

first consumer

think about how the rest of the world uses this class

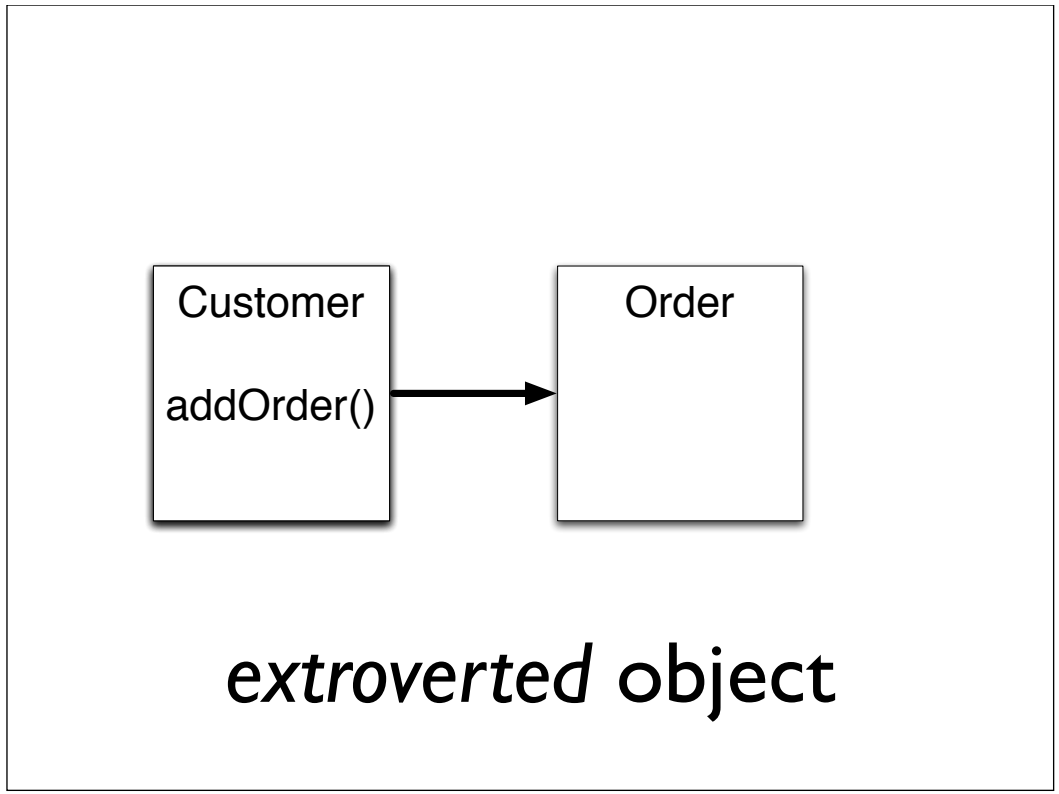
creates *consumption awareness*

design benefits of tdd

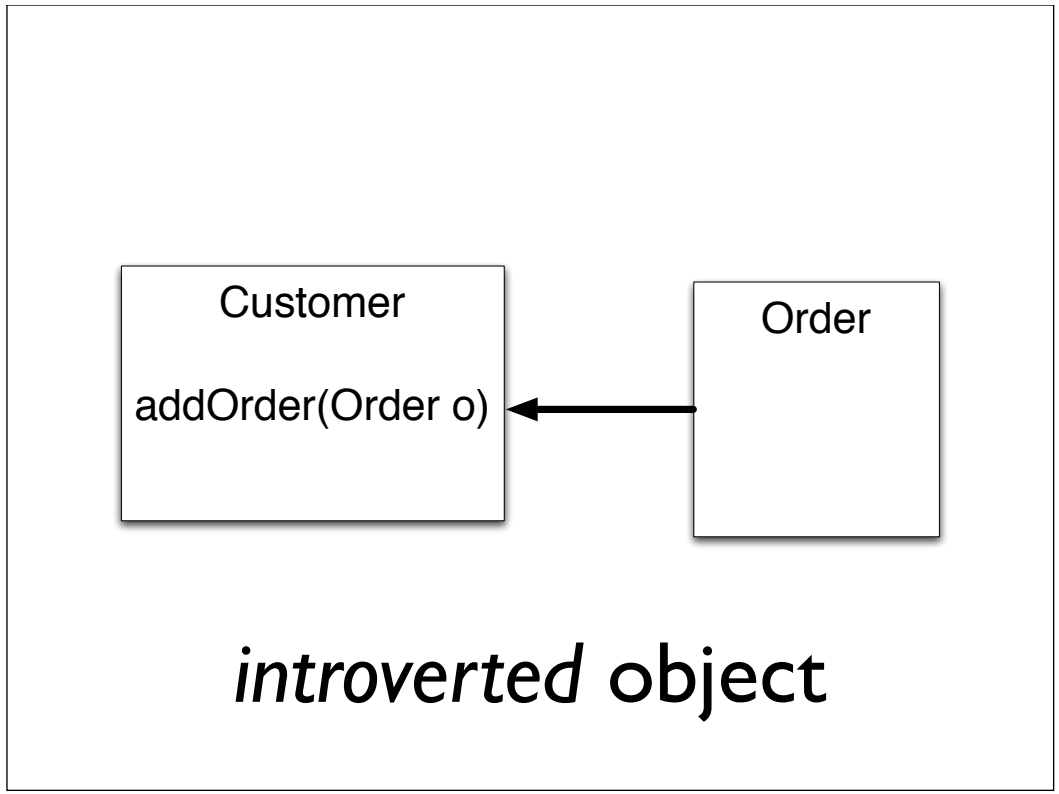
forces mocking of dependent objects

naturally creates composed method

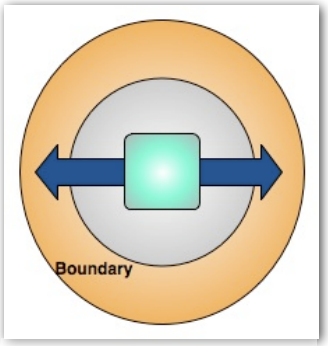
cleaner metrics

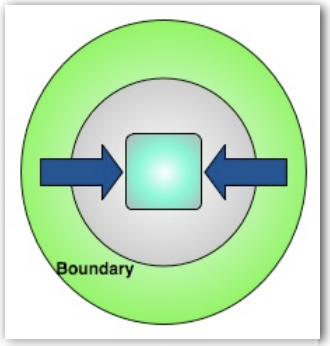


extroverted object



introverted object





extroverted objects

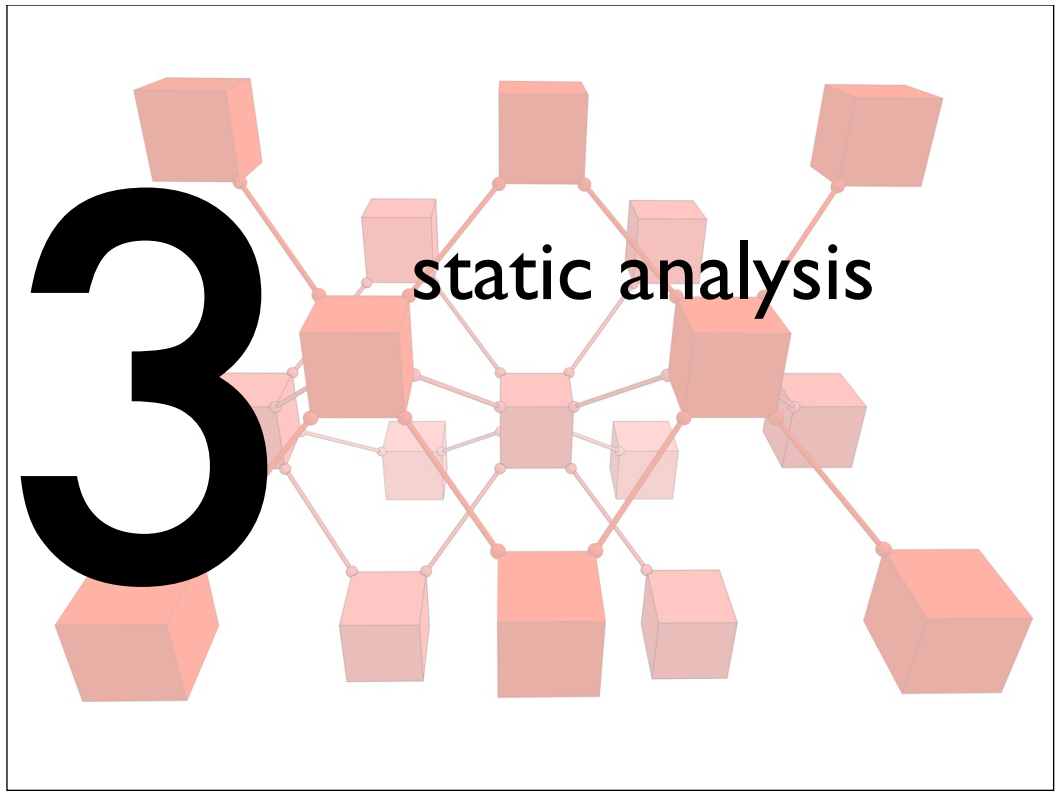
introverted objects

“reach out” to create objects

cleaner dependencies

ad hoc creation

moves object construction to a few simple places


3
static analysis

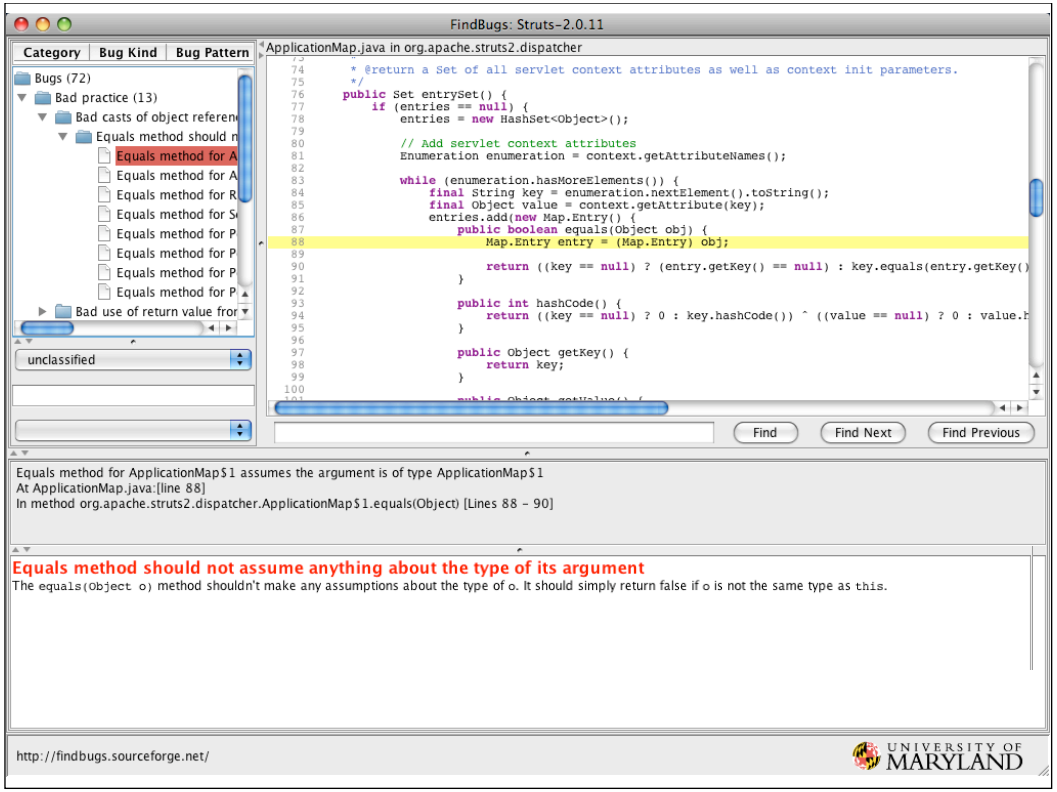
byte-code analysis: findbugs

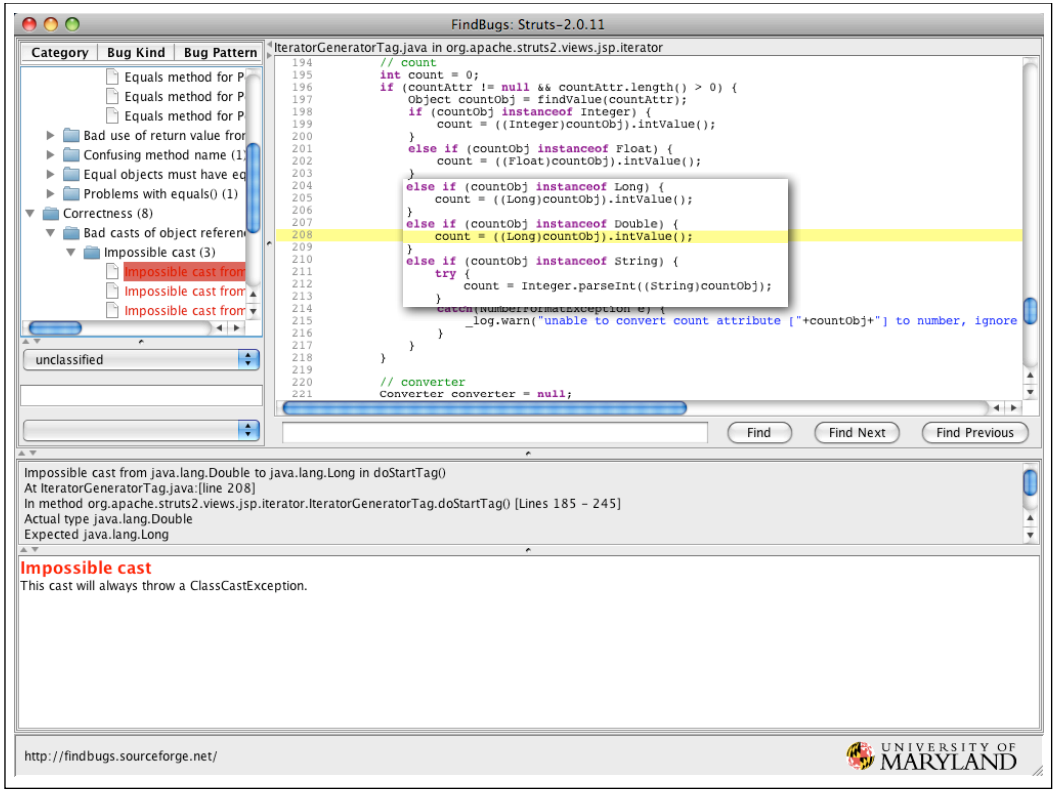
bug categories

correctness
- probable bug

bad practice
- violation of recommended & essential coding practice

dodgy
- confusing, anomalous, written poorly


FindBugs: Struts-2.0.11
Category
Bug Kind
Bug Pattern
Bugs (72)
Bad practice (13)
Bad casts of object referen
Equals method should n
Equals method for A
Equals method for A
Equals method for R
Equals method for S
Equals method for P
Equals method for P
Equals method for P
Equals method for P
Bad use of return value fror
unclassified
ApplicationMap.java in org.apache.struts2.dispatcher
* @return a Set of all servlet context attributes as well as context init parameters.
*/
public Set entrySet() {
if (entries == null) {
entries = new HashSet<Object>();
// Add servlet context attributes
Enumeration enumeration = context.getAttributeNames();
while (enumeration.hasMoreElements()) {
final String key = enumeration.nextElement().toString();
final Object value = context.getAttribute(key);
entries.add(new Map.Entry() {
public boolean equals(Object obj) {
Map.Entry entry = (Map.Entry) obj;
return ((key == null) ? (entry.getKey() == null) : key.equals(entry.getKey()
}
public int hashCode() {
return ((key == null) ? 0 : key.hashCode()) ^ ((value == null) ? 0 : value.h
}
public Object getKey() {
return key;
}
Find
Find Next
Find Previous
Equals method for ApplicationMap$1 assumes the argument is of type ApplicationMap$1
At ApplicationMap.java:[line 88]
In method org.apache.struts2.dispatcher.ApplicationMap$1.equals(Object) [Lines 88 - 90]
Equals method should not assume anything about the type of its argument
The equals(Object o) method shouldn't make any assumptions about the type of o. It should simply return false if o is not the same type as this.
http://findbugs.sourceforge.net/
UNIVERSITY OF MARYLAND


FindBugs: Struts-2.0.11
Category
Bug Kind
Bug Pattern
Equals method for P
Equals method for P
Equals method for P
Bad use of return value fror
Confusing method name (1
Equal objects must have eq
Problems with equals() (1)
Correctness (8)
Bad casts of object referen
Impossible cast (3)
Impossible cast from
Impossible cast from
Impossible cast from
unclassified
IteratorGeneratorTag.java in org.apache.struts2.views.jsp.iterator
194 // count
195 int count = 0;
196 if (countAttr != null && countAttr.length() > 0) {
197 Object countObj = findValue(countAttr);
198 if (countObj instanceof Integer) {
199 count = ((Integer)countObj).intValue();
200 }
201 else if (countObj instanceof Float) {
202 count = ((Float)countObj).intValue();
203 }
204 else if (countObj instanceof Long) {
205 count = ((Long)countObj).intValue();
206 }
207 else if (countObj instanceof Double) {
208 count = ((Long)countObj).intValue();
209 }
210 else if (countObj instanceof String) {
211 try {
212 count = Integer.parseInt((String)countObj);
213 }
214 catch(NumberFormatException e) {
215 _log.warn("unable to convert count attribute ["+countObj+"] to number, ignore
216 }
217 }
218 }
219
220 // converter
221 Converter converter = null;
Find
Find Next
Find Previous
Impossible cast from java.lang.Double to java.lang.Long in doStartTag()
At IteratorGeneratorTag.java:[line 208]
In method org.apache.struts2.views.jsp.iterator.IteratorGeneratorTag.doStartTag() [Lines 185 - 245]
Actual type java.lang.Double
Expected java.lang.Long
Impossible cast
This cast will always throw a ClassCastException.
http://findbugs.sourceforge.net/
UNIVERSITY OF MARYLAND

source analysis & pmd

pmd targets

possible bugs — empty try/catch blocks

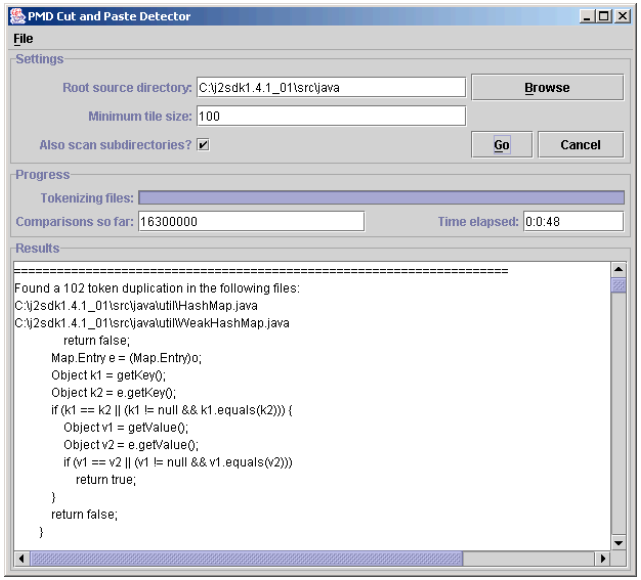
dead code — unused local variables, parameters, private variables

suboptimal code — wasteful string usage

overcomplicated expressions

31	hibernate/query/QueryTranslator.java	401	Avoid unused private fields such as 'NO_INTS'
32	hibernate/query/WhereParser.java	134	Avoid unused private fields such as 'quoted'
33	hibernate/query/WhereParser.java	136	Avoid unused private fields such as 'bracketsSinceFunction'
34	hibernate/test/Child.java	38	Avoid unused formal parameters such as 'id'
35	hibernate/test/Child.java	38	Avoid unused private methods such as 'setId(long)'
36	hibernate/test/FooComponent.java	76	Avoid unused private methods such as 'getNull()'
37	hibernate/test/FooComponent.java	79	Avoid unused private methods such as 'setNull(String)'
38	hibernate/test/Fum.java	46	Avoid unused private methods such as 'setId(FumCompositeID)'
39	hibernate/test/Qux.java	78	Avoid unused private methods such as 'setCreated(boolean)'
40	hibernate/test/Qux.java	86	Avoid unused private methods such as 'setDeleted(boolean)'
41	hibernate/test/Qux.java	93	Avoid unused private methods such as 'setLoaded(boolean)'
42	hibernate/test/Qux.java	100	Avoid unused private methods such as 'setStored(boolean)'
43	hibernate/test/Qux.java	108	Avoid unused private methods such as 'setKey(long)'
44	hibernate/test/Qux.java	149	Avoid unused private methods such as 'getChildKey()'
45	hibernate/test/Qux.java	153	Avoid unused private methods such as 'setChildKey(Long)'
46	hibernate/tools/SchemaExport.java	191	Avoid unused private methods such as 'format(String)'
47	hibernate/tools/SchemaExportTask.java	56	Avoid unused private fields such as 'formatSQL'
48	hibernate/tools/SchemaExportTask.java	59	Avoid unused private fields such as 'delimiter'
49	hibernate/tools/codegen/ClassMapping.java	399	Avoid unused private methods such as 'addImport(String)'
50	hibernate/tools/reflect/ReflectedClass.java	263	Avoid unused private methods such as 'capitalize(String)'
51	hibernate/tools/reverse/MapGui.java	548	Avoid unused formal parameters such as 'evt'
52	hibernate/tools/reverse/MapGui.java	560	Avoid unused formal parameters such as 'evt'
53	hibernate/tools/reverse/MapGui.java	570	Avoid unused formal parameters such as 'evt'
54	hibernate/tools/reverse/MapGui.java	606	Avoid unused formal parameters such as 'evt'
55	hibernate/tools/reverse/MapGui.java	612	Avoid unused formal parameters such as 'evt'
56	hibernate/tools/reverse/MapGui.java	617	Avoid unused formal parameters such as 'evt'
57	hibernate/tools/reverse/MapGui.java	641	Avoid unused formal parameters such as 'evt'
58	hibernate/tools/reverse/MapGui.java	651	Avoid unused formal parameters such as 'evt'
59	hibernate/tools/reverse/MapGui.java	709	Avoid unused formal parameters such as 'evt'
60	hibernate/tools/reverse/MapGui.java	737	Avoid unused private fields such as 'buttonGroup1'
61	hibernate/tools/reverse/ParamsPanel.java	142	Avoid unused formal parameters such as 'evt'
62	hibernate/tools/reverse/ParamsPanel.java	149	Avoid unused formal parameters such as 'evt'
63	hibernate/tools/reverse/ParamsPanel.java	158	Avoid unused formal parameters such as 'evt'
64	hibernate/tools/reverse/ParamsPanel.java	167	Avoid unused formal parameters such as 'evt'
65	hibernate/type/ArrayType.java	19	Avoid unused private fields such as 'elementClass'
66	hibernate/type/ComponentType.java	31	Avoid unused private fields such as 'parentProperty'

cpd


PMD Cut and Paste Detector
File
Settings
Root source directory:
C:\j2sdk1.4.1_01\src\java
Browse
Minimum tile size:
100
Also scan subdirectories?
Go
Cancel
Progress
Tokenizing files:
Comparisons so far:
16300000
Time elapsed:
0:0:48
Results
==========
Found a 102 token duplication in the following files:
C:\j2sdk1.4.1_01\src\java\util\HashMap.java
C:\j2sdk1.4.1_01\src\java\util\WeakHashMap.java
return false;
Map.Entry e = (Map.Entry)o;
Object k1 = getKey();
Object k2 = e.getKey();
if (k1 == k2 || (k1 != null && k1.equals(k2))) {
Object v1 = getValue();
Object v2 = e.getValue();
if (v1 == v2 || (v1 != null && v1.equals(v2)))
return true;
}
return false;
}


4
good citizenship

getters & setters !=
encapsulation

accessors/mutators

knee-jerk creating getters/setters voids encapsulation

create atomic mutators for dependent fields

“should I unit test my getters & setters?”

the new strategy

only create getters & setters when you need them for other methods

testing:

- shouldn't have to tdd them
- they will get code coverage automatically

as easy upon use as upon creation

constructors

specific contract for how to create valid objects

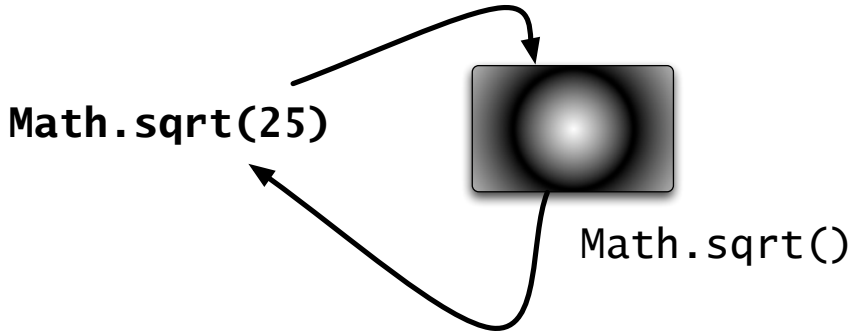
how often is a blank object valid?

never!

don't provide default constructors for domain objects

push back on frameworks that require this

static methods



mixing static + state

singleton

singleton is bad because:

- mixes responsibilities
- untestable
- the object version of global variables

avoiding singletons

1. create a pojo for the business behavior

   simple

   testable!

2. create a factory to create the pojo

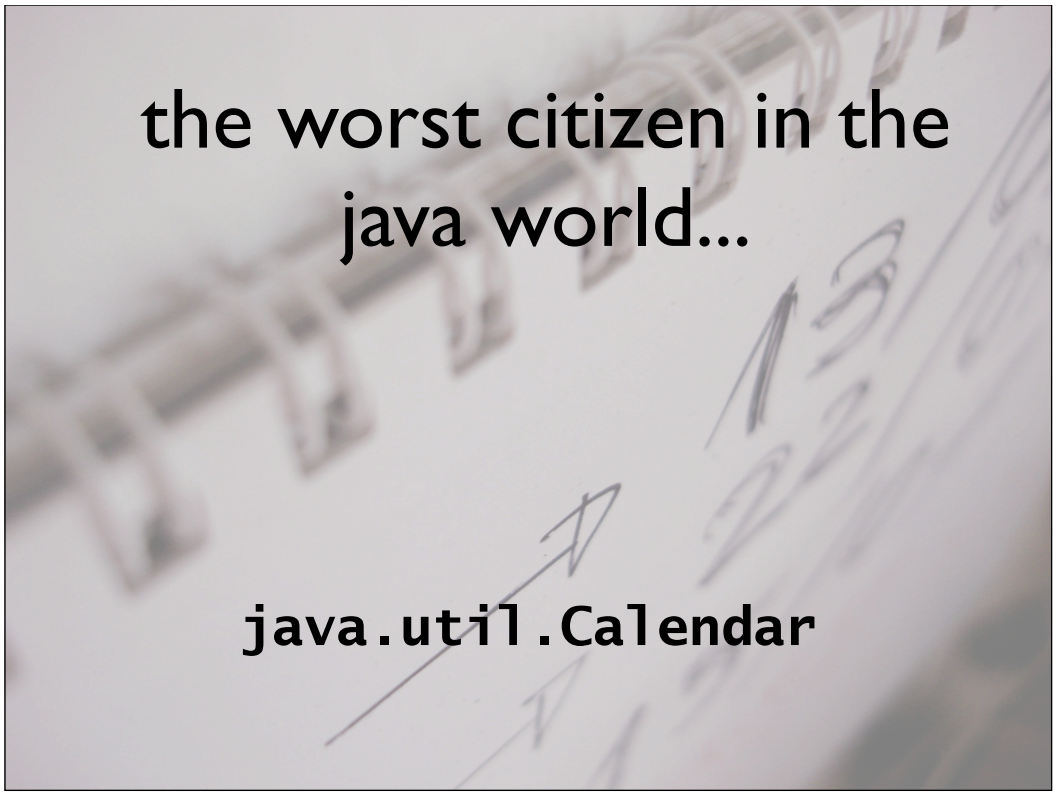
   also testable

```
public class ConfigSingleton {
    private static ConfigSingleton myInstance;
    private Point _initialPosition;

    public Point getInitialPosition() {
        return _initialPosition;
    }

    private ConfigSingleton() {
        Dimension screenSize =
                Toolkit.getDefaultToolkit().getScreenSize();
        _initialPosition = new Point();
        _initialPosition.x = (int) screenSize.getWidth() / 2;
        _initialPosition.y = (int) screenSize.getHeight() / 2;
    }

    public static ConfigSingleton getInstance() {
        if (myInstance == null)
            myInstance = new ConfigSingleton();
        return myInstance;
    }
}
```

```
public class Configuration {
    private Point _initialPosition;

    private Configuration(Dimension screenSize) {
        _initialPosition = new Point();
        _initialPosition.x = (int) screenSize.getWidth() / 2;
        _initialPosition.y = (int) screenSize.getHeight() / 2;
    }

    public int getInitialX() {
        return _initialPosition.x;
    }

    public int getInitialY() {
        return _initialPosition.y;
    }
}
```

```
public class ConfigurationFactory {
    private static Configuration myConfig;

    public static Configuration getConfiguration() {
        if (myConfig == null) {
            try {
                Constructor cxtor[] =
                    Configuration.class.getDeclaredConstructors();
                cxtor[0].setAccessible(true);
                myConfig = (Configuration) cxtor[0].newInstance(
                        Toolkit.getDefaultToolkit().getScreenSize());
            } catch (Throwable e) {
                throw new RuntimeException("can't construct Configuration");
            }
        }
        return myConfig;
    }
}
```

```
public class TestConfigurationFactory extends TestCase {

    public void test_Creation_creates_a_single_instance() {
        Configuration config1 = ConfigurationFactory.getConfiguration();
        assertNotNull(config1);
        Configuration config2 = ConfigurationFactory.getConfiguration();
        assertNotNull(config2);
        assertSame(config1, config2);
    }
}
```


the worst citizen in the java world...
java.util.Calendar

5

yagni

you ain't gonna need it

discourages gold plating

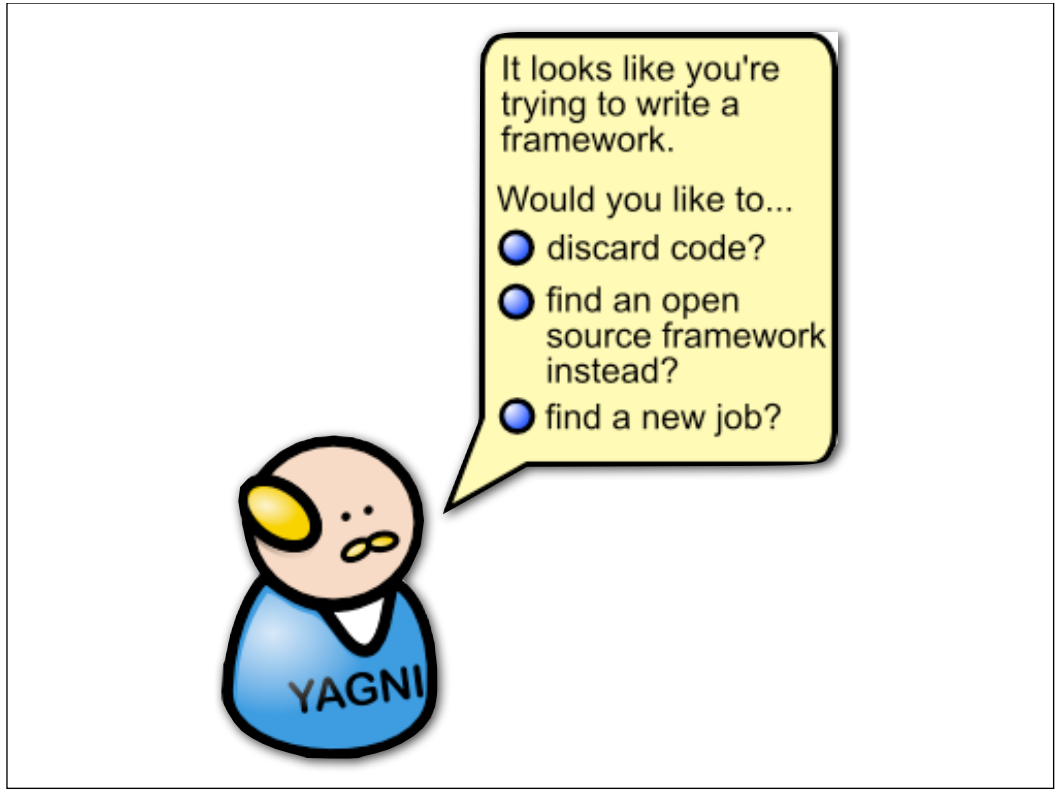
build the simplest thing that we need *right now*

don't indulge in speculative development

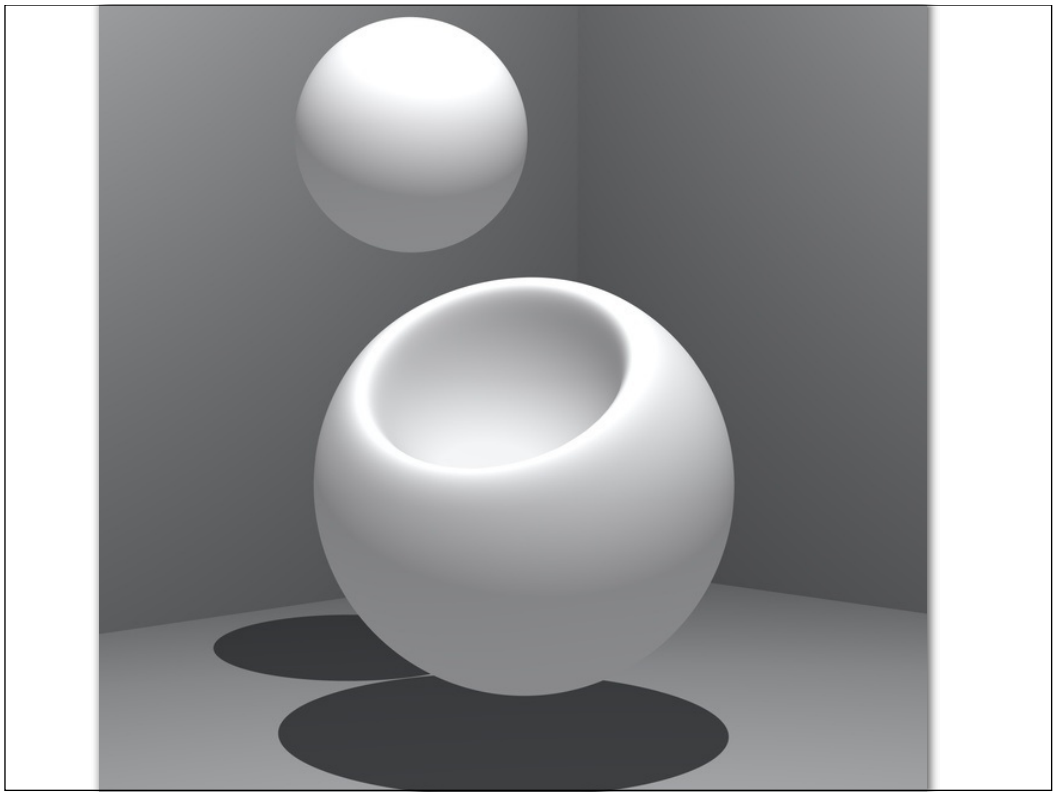
- increases *software entropy*
- only saves time if you can guarantee you won't have to change it later
- leads to *frameworks*

a public plea to the java community:

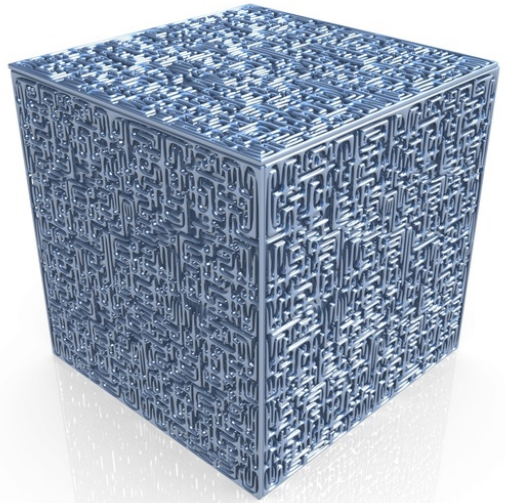
please stop building frameworks!


It looks like you're trying to write a framework.
Would you like to...
discard code?
find an open source framework instead?
find a new job?
YAGNI


This is just
what they need!

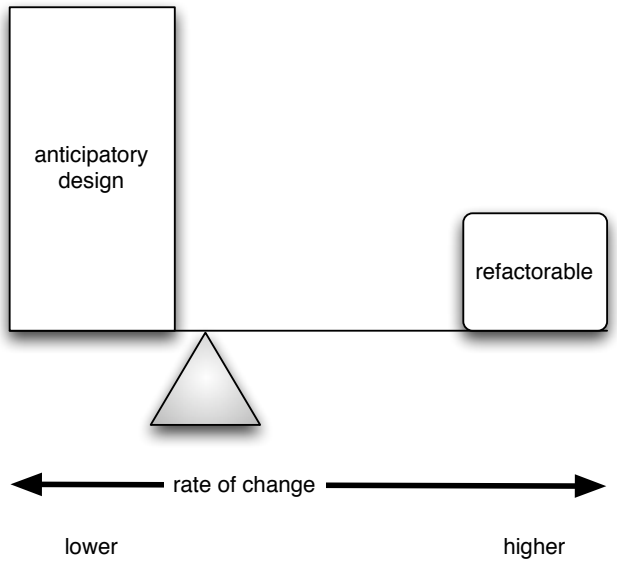


a cautionary tale



building a simple framework

changeability



corporate code smells

6. We have an Architect who reviews all code pre-checkin and decides whether or not to allow it into version control.

7. We can't use any open source code because our lawyers say we can't.

8. We use WebSphere because...(I always stop listening at this point)

9. We bought the entire tool suite (even though we only needed about 10% of it) because it was cheaper than buying the individual tools.

10. We invented our own web/persistence/ messaging/caching framework because none of the existing ones was good enough.

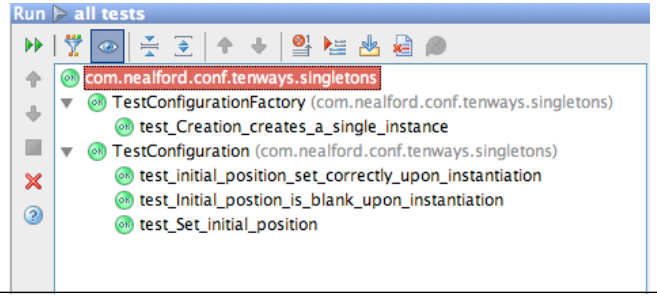
1. There is a reason that WSAD isn't called WHAPPY.
2. The initial estimate must be within 15% of the final cost, the post-analysis estimate must be within 10%, and the post-design estimate must be with 5%
3. We don't have time to write unit tests (we're spending too much time debugging)
4. We keep all of our business logic in stored procedures...for performance reasons.
5. The only JavaDoc is the Eclipse message explaining how to change your default JavaDoc template.


6
question authority


angry monkeys &
christmas roasts

test names

```
testUpdateCacheAndVerifyThatItemExists() {

}

test_Update_cache_and_verify_that_item_exists() {

}
```



api's

```
Car car = new CarImpl();
MarketingDescription desc = new
    MarketingDescriptionImpl();
desc.setType("Box");
desc.setSubType("Insulated");
desc.setAttribute("length", "50.5");
desc.setAttribute("ladder", "yes");
desc.setAttribute("lining type", "cork");
car.setDescription(desc);
```

fluent interfaces

```
Car car = Car.describedAs()
          .box()
          .length(12)
          .includes(Equipment.LADDER)
          .has(Lining.CORK);
```

what stands in the way?

the javabean specification!

what's bad about beans?

forces you to create default constructors

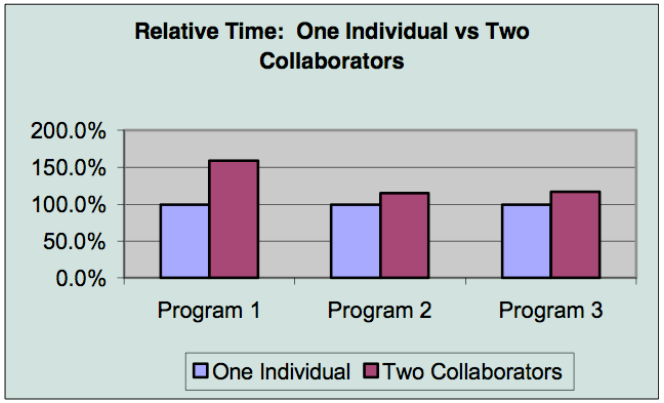
- creates bad citizens
- harms constructor as specification

setXXX() methods return void

can't use beans for fluent interfaces

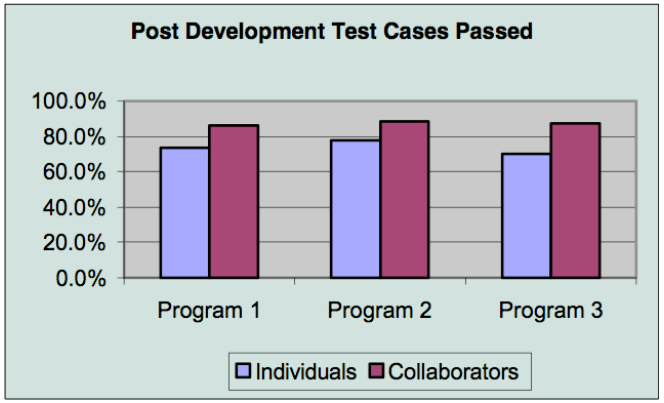
non-intuitive

pair programming studies



after adjusting, pairs produced code 15% more slowly than individuals...

pair programming studies



...with 15% fewer defects

7

slap

single level of
abstraction principle

s l a p

keep all lines of code in a method at the same level of abstraction

jumping abstraction layers makes code hard to understand

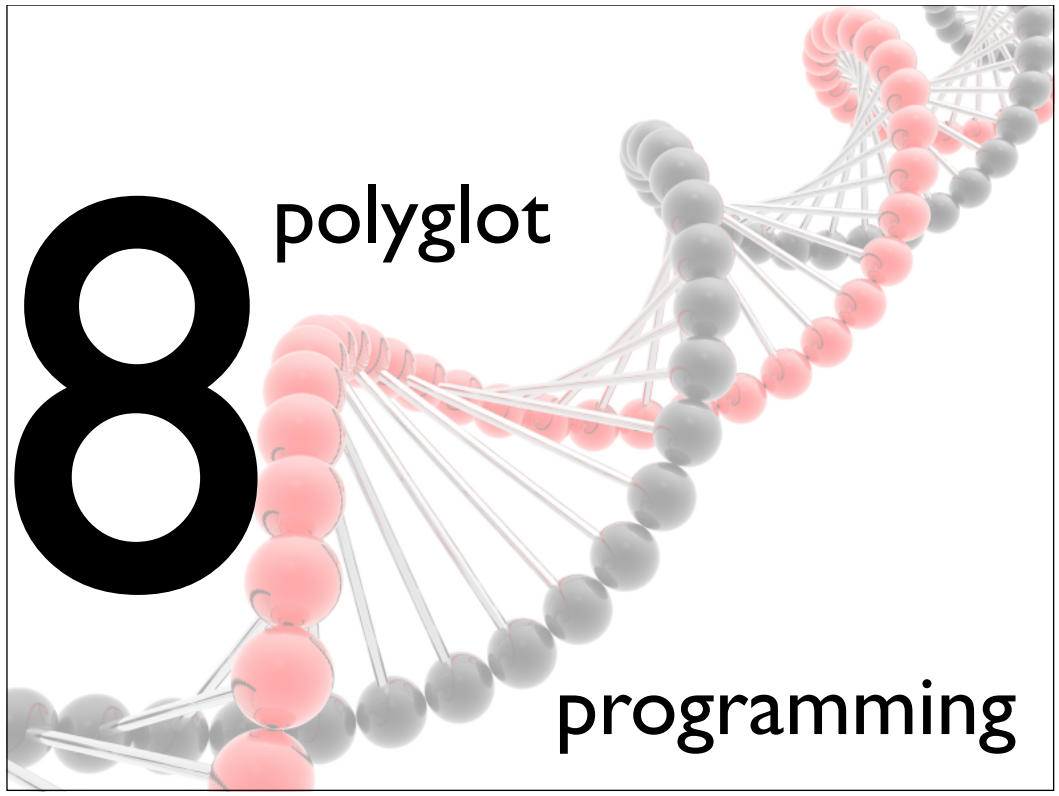
composed method => slap

refactor to slap

even if it means single-line methods

```
public void addOrder(final ShoppingCart cart, String userName,
                     Order order) throws SQLException {
    Connection c = null; PreparedStatement ps = null;
    Statement s = null; ResultSet rs = null;
    boolean transactionState = false;
    try {
        c = dbPool.getConnection();
        s = c.createStatement();
        transactionState = c.getAutoCommit();
        int userKey = getUserKey(userName, c, ps, rs);
        c.setAutoCommit(false);
        addSingleOrder(order, c, ps, userKey);
        int orderKey = getOrderKey(s, rs);
        addLineItems(cart, c, orderKey);
        c.commit();
        order.setOrderKey(orderKey);
    } catch (SQLException sqlx) {
        s = c.createStatement();
        c.rollback();
        throw sqlx;
    } finally {
        try {
            c.setAutoCommit(transactionState);
            dbPool.release(c);
            if (s != null)  s.close();
            if (ps != null) ps.close();
            if (rs != null) rs.close();
        } catch (SQLException ignored) {
        }
    }
}
```

```
public void addOrder(final ShoppingCart cart, String userName,
                     Order order) throws SQLException {
    Connection connection = null; PreparedStatement ps = null;
    Statement statement = null; ResultSet rs = null;
    boolean transactionState = false;
    try {
        connection = dbPool.getConnection();
        statement = connection.createStatement();
        transactionState = setupTransactionStateFor(connection, transactionState);
        addSingleOrder(order, connection, ps, userKeyFor(userName, connection));
        order.setOrderKey(generateOrderKey(statement, rs));
        addLineItems(cart, connection, order.getOrderKey());
        completeTransaction(connection);
    } catch (SQLException sqlx) {
        rollbackTransactionFor(connection);
        throw sqlx;
    } finally {
        cleanUpDatabaseResources(connection, transactionState, statement, ps, rs);
    }
}
```


8
polyglot
programming

leveraging existing
platforms with *languages*
targeted at specific
problems and
applications

why do this?

looming problems/ opportunities

massively parallel threading

- use a functional language: jaskell, scala

schedule pressure

- jruby on rails, grails

looming problems/ opportunities

everyday coding

groovy, ruby

stop banging rocks together & get some work done!

face it:

EBXL™

looming problems/ opportunities

writing more declarative code via **dsl**s

build fluent interfaces

complexity

doesn't polyglot programming add complexity?

In the past, language == platform

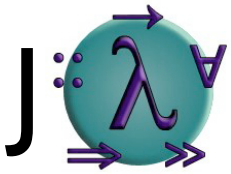
now, language != platform

```
public class LineNumbers {
    public LineNumbers(String path) {
        File file = new File(path);
        LineNumberReader reader = null;
        try {
            reader = new LineNumberReader(new FileReader(file));
            while (reader.ready()) {
                out.println(reader.getLineNumber() + ":"
                        + reader.readLine());
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                reader.close();
            } catch (IOException ignored) {
            }
        }
    }

    public static void main(String[] args) {
        new LineNumbers(args[0]);
    }
}
```

```
def number=0
new File (args[0]).eachLine { line ->
    number++
    println "$number: $line"
}
```

```
class SafeArray{
    private final Object[] _arr;
    private final int _begin;
    private final int _len;

    public SafeArray(Object[] arr, int len){
        _arr = arr;
        _begin = begin;
        _len = len;
    }

    public Object at(int i){
        if(i < 0 || i >= _len){
            throw new ArrayIndexOutOfBoundsException(i);
        }
        return _arr[_begin + i];
    }

    public int getLength(){
        return _len;
    }
}
```

```
newSafeArray arr begin len = {
  length = len;
  at i = if i < begin || i >= len then
           throw $ ArrayIndexOutOfBoundsException.new[i]
         else
           arr[begin + i];
}
```




9
every nuance

java's back alleys

reflection

"reflection is slow"

no longer true

elegant solutions to problems

```
public class Configuration {
    private Point _initialPosition;

    private Configuration(Dimension screenSize) {
        _initialPosition = new Point();
        _initialPosition.x = (int) screenSize.getWidth() / 2;
        _initialPosition.y = (int) screenSize.getHeight() / 2;
    }

    public int getInitialX() {
        return _initialPosition.x;
    }

    public int getInitialY() {
        return _initialPosition.y;
    }
}
```

```
@Before public void setUp() {
    try {
        Constructor cxtor[] =
                Configuration.class.getDeclaredConstructors();
        cxtor[0].setAccessible(true);
        c = (Configuration) cxtor[0].newInstance(
                Toolkit.getDefaultToolkit().getScreenSize());
    } catch (Throwable e) {
        fail();
    }
}
```

```
@Test
public void initial_position_set_correctly_upon_instantiation() {
    Configuration specialConfig = null;
    Dimension screenSize = null;
    try {
        Constructor cxtor[] =
                Configuration.class.getDeclaredConstructors();
        cxtor[0].setAccessible(true);
        screenSize = new Dimension(26, 26);
        specialConfig = (Configuration) cxtor[0].newInstance(screenSize);
    } catch (Throwable e) {
        fail();
    }

    Point expected = new Point();
    expected.x = (int) screenSize.getWidth() / 2;
    expected.y = (int) screenSize.getHeight() / 2;
    assertEquals(expected.x, specialConfig.getInitialX());
    assertEquals(expected.y, specialConfig.getInitialY());
}
```

regular expressions &

learn the nuances of
java...

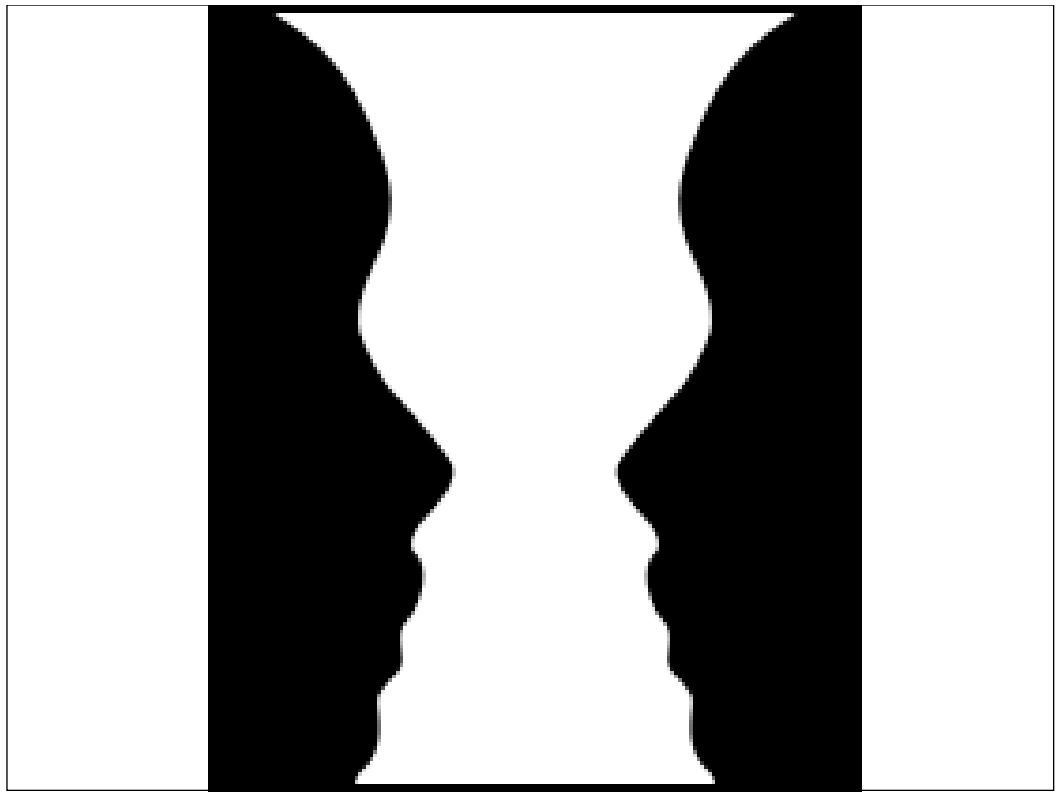
...then tell the other
people on your project

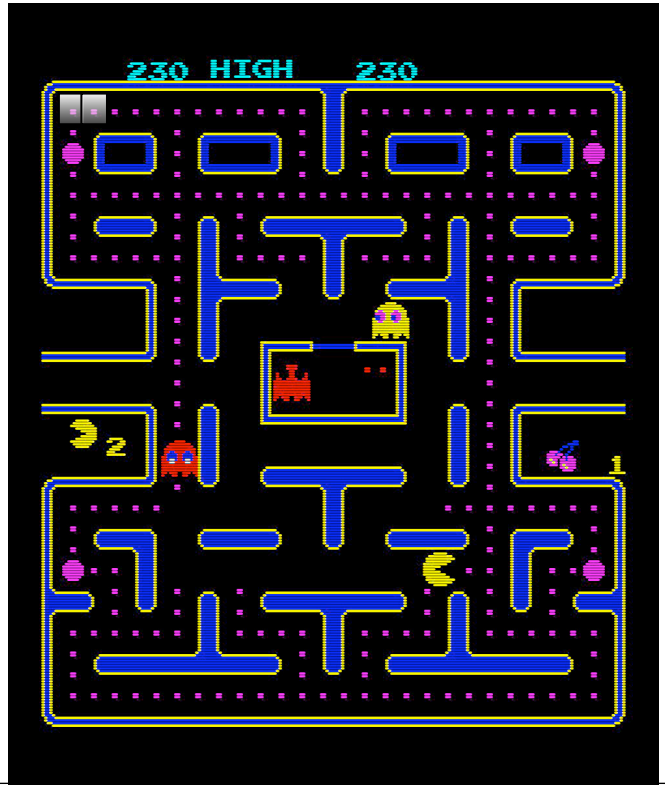

10
anti-objects

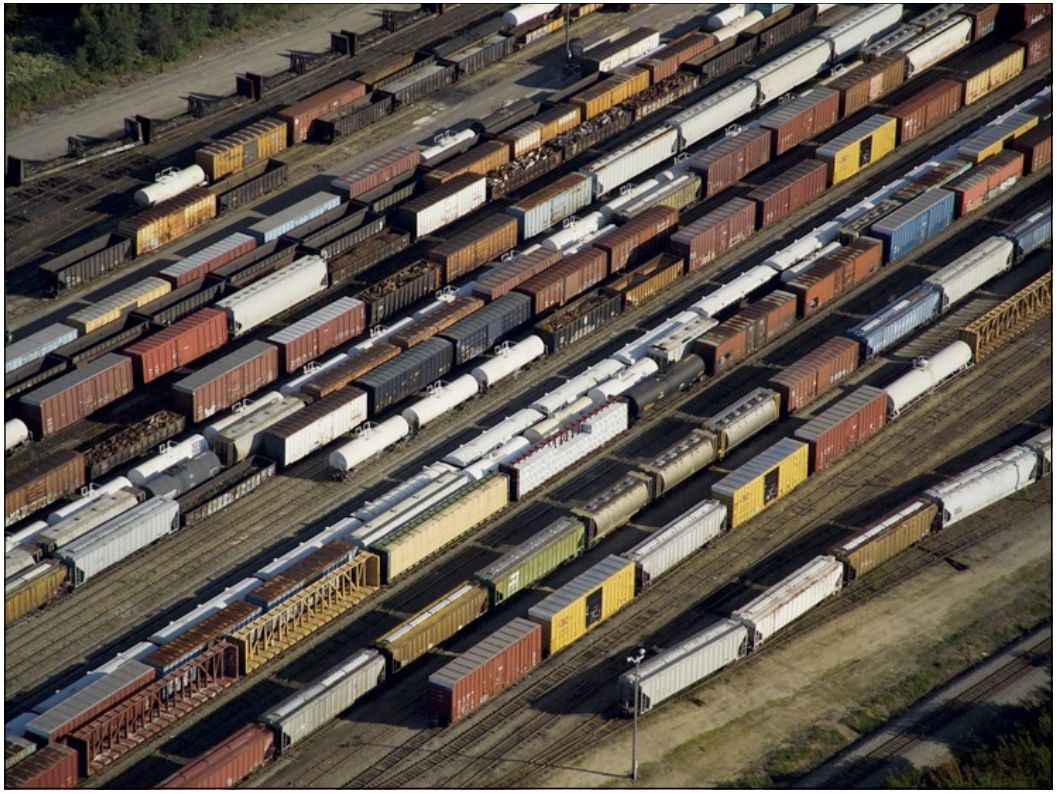
collaborative diffusion

"The metaphor of objects can go too far by making us try to create objects that are too much inspired by the real world."

"...an antiobject is a kind of object that appears to essentially do the opposite of what we generally think the object should be doing."




230 HIGH 230
2
1



to improve is to change; to be perfect is to change often.

winston churchill

without continual growth and progress, such words as improvement, achievement, and success have no meaning.

benjamin franklin

questions?

please fill out the session evaluations
slides & samples available at nealford.com

NEAL FORD software architect / meme wrangler

ThoughtWorks

nford@thoughtworks.com
3003 Summit Boulevard, Atlanta, GA 30319
www.nealford.com
www.thoughtworks.com
memeagora.blogspot.com

resources

An Initial Investigation of Test Driven Development in Industry -
Laurie Williams, Boby George
http://collaboration.csc.ncsu.edu/laurie/Papers/TDDpaperv8.pdf

findbugs
http://findbugs.sourceforge.net/

pmd/cpd
http://pmd.sourceforge.net/

The legend of the leaning tower
http://physicsworld.com/cws/article/print/16806

AntiPatterns Catalog
http://c2.com/cgi/wiki?AntiPatternsCatalog

resources

Smalltalk Best Practice Patterns Kent Beck
Prentice Hall PTR (October 13, 1996)
ISBN-10: 013476904X

Polyglot Programming
http://memeagora.blogspot.com/2006/12/polyglot-programming.html

Optical Illusions
http://en.wikipedia.org/wiki/Optical_illusion

Collaborative Diffusion: Programming Anti-objects - A Repenning
http://www.cs.colorado.edu/~ralex/papers/PDF/OOPSLA06antiobjects.pdf

resources
pair programming

http://c2.com/cgi/wiki?PairProgramming

http://www.xprogramming.com/Practices/PracPairs.html

http://collaboration.csc.ncsu.edu/laurie/Papers/XPSardinia.PDF

http://www.cs.utah.edu/~lwilliam/Papers/ieeeSoftware.PDF