JavaOne℠

Java is a trademark of Sun Microsystems, Inc.

Don't Do This! How Not To Write Java™ Technology-Based Software

Dean Wampler
Object Mentor, Inc.
dean@objectmentor.com @deanwampler

Object Mentor
Industry experts in software best practices

- Mentor, Trainer, Consultant at Object Mentor, Inc.

  objectmentor.com

  polyglotprogramming.com

Search | Blog | Contact

Object Mentor
Industry experts in software best practices

Home | Company | Team | Practices | Training | Resources | Podcasts

Agile Transition Strategies
Agile and XP Training and Certification
Advanced Object Oriented Design Training
Project Mentoring and Support

Featured Stories

Is your enterprise thinking of migrating to Agile?
Object Mentor has been introducing enterprises to XP and Agile since 1999. Find out how we can help your organization.

A Tale of Two Methodologies
Escrow.com was lucky enough to be able to run two projects side-by-side, one using XP and one not. Find out how it went.

Symantec Goes Extreme
This article appeared in Software Development magazine and describes the progress of several teams at Symantec as they transitioned to XP.

From Our Blogs...

Why the sea is boiling hot.
Uncle Bob
11 May, 2009
A response to @dhh's blog on artistry and engineering.

The Scatology of Agile Architecture
Uncle Bob

Public Courses

Certified Scrum Master
Ron Jeffries
September 9-10, Chicago
September 16-17, Phoenix
October 20-21, Phoenix
December 2-3, Chicago

Advanced Object-Oriented Design with Design Patterns
Michael Feathers
May 13-15, Chicago

Test-Driven Development "Soup to Nuts"
Bob Martin
May 18-22, Chicago

FitNesse - Acceptance Testing Tool
Bob Martin
May 28-29, Chicago

Test-Driven Development and Refactoring in C#
Bob Martin
May 28-29, Chicago
June 1-3, Chicago

Test-Driven Development and Refactoring in Java
Bob Martin

I'm speaking at Agile2009 Conference
Chicago, USA
August 24-28, 2009

New Articles

Clean Code Tip of the Week #6: Avoid Poorly Written Comments
Uncle Bob
In this sixth tip in the series, the crewmen try to interpret a poorly worded comment.
27 Feb, 2009

Clean Code Tip of the Week #5: Avoid Redundant Comments
Uncle Bob
The programmers discuss redundant comments, which describe something that adequately describes itself.
18 Feb, 2009

Clean Code Tip of the Week #4: Avoid Obsolete Comments
Uncle Bob

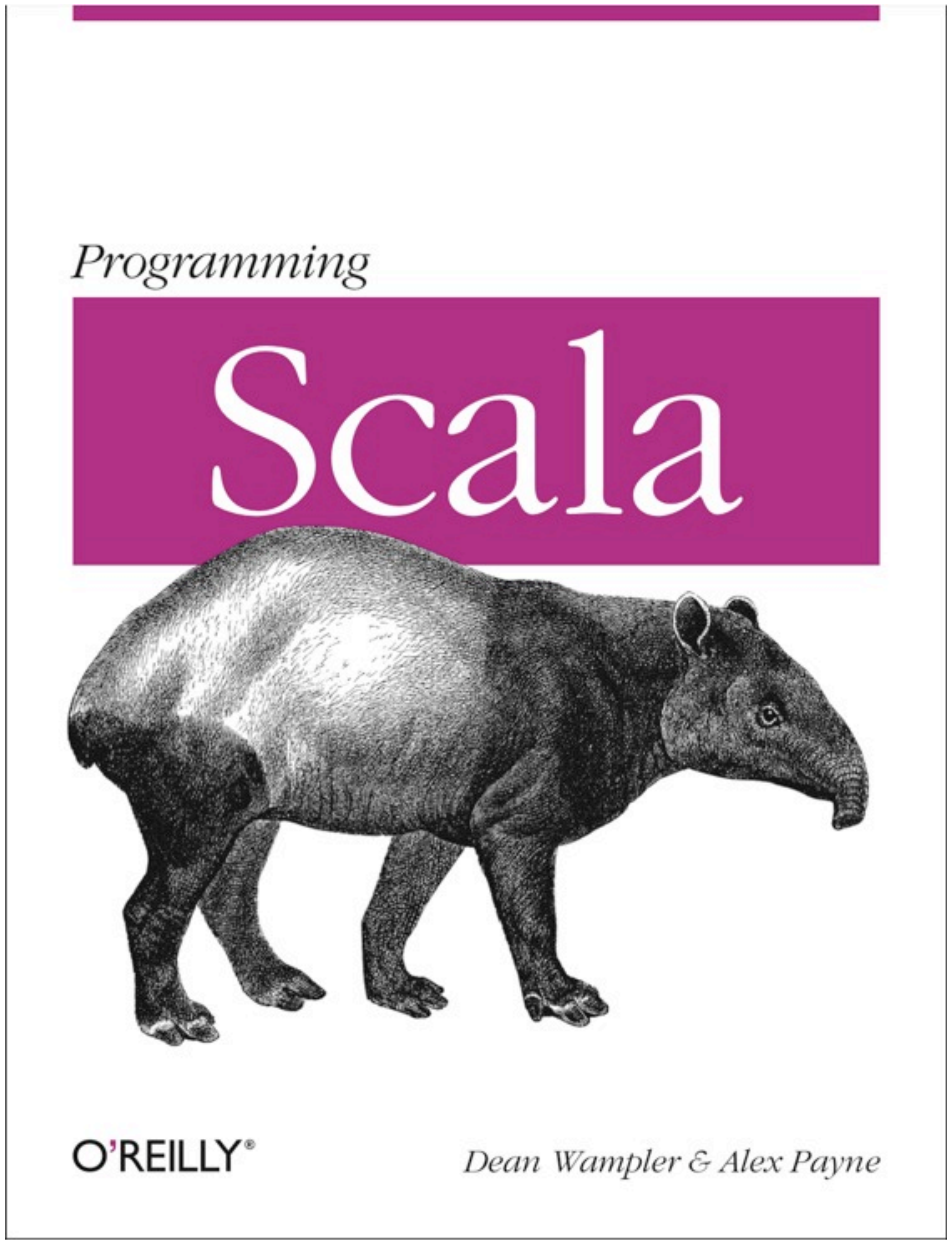
Sign up today!
Enter your email address below to receive automatic notification about articles, events, courses and special discounts!
Submit

Receive our tweets!
Keep up with our mentor's latest rants and raves! Now you can subscribe to our tweets.
Uncle Bob's Tweets
Michael Feathers Tweets
Bob Koss Tweets
Brett Schuchert Tweets
Dean Wampler Tweets

Upcoming Events
Dean Wampler
Don't Do This! How Not to Write Java Software
JavaOne
3 June, 2009

Dean Wampler

- September 2009
  oreilly.com/catalog/9780596157746/
- Read it now:
  programmingscala.com



Lessons from the trenches...

http://everystockphoto.com/photo.php?imageId=4027778

10 mistakes and how to avoid them.

http://everystockphoto.com/photo.php?imageId=4027778

Mistake #1: Comment everything!

```
public class Account { …
  /**
   * Withdraw money from account.
   * @param amount to withdraw (double)
   * @return new balance (double).
   */
  public double withdraw(double amount) {
    balance -= amount;
    return balance;
  }
}
```

Version 1

```java
public class Account { …
  /**
   * Withdraw money from account.
   * @param amount to withdraw (double)
   * @return new balance (double).
   */
  public double withdraw(double amount) {
    balance -= amount;
    return balance;
  }
}
```

Command-query separation?

Version 1

```
public class Account { …
  /**
   * Withdraw money from account.
   * @param amount to withdraw (double)
   * @return new balance (double).
   */
  public void withdraw(double amount) {
    balance -= amount;

  }
}
```

Version 2

```java
public class Account { …
  /**
   * Withdraw money from account.
   * @param amount to withdraw (double)
   * @return new balance (double).
   */
  public void withdraw(double amount) {
    balance -= amount;

  }
}
```

What about overdrafts?

Version 2

```
public class Account { …
  /**
   * Withdraw money from account.
   * @param amount to withdraw (double)
   * @return new balance (double).
   */
  public void withdraw(double amount)
      throws OverdraftException {
    if (balance < amount)
      throw new OverdraftException(
        balance, amount);
    balance -= amount;
  }}
```

Version 3

```
public class Account { …
  /**
   * Withdraw money from account.
   * @param amount to withdraw (double)
   * @return new balance (double).
   */
  public void withdraw(double amount)
      throws OverdraftException {
    if (balance < amount)
      throw new OverdraftException(
        balance, amount);
    balance -= amount;
}}
```

Version 3

```
public class Account { …
  /**
   * Withdraw money from account.
   * @param amount to withdraw (double)
   * @return new balance (double).
   */
  public void withdraw(double amount)
      throws OverdraftException {
    if (balance < amount)
      throw new OverdraftException(
        balance, amount);
    balance -= amount;
}}
```

Doubles???

```
public class Account { …
  /**
   * Withdraw money from account.
   * @param amount to withdraw (double)
   * @return new balance (double).
   */
  public void withdraw(Currency amount)
      throws OverdraftException {
    if (balance.lessThan(amount))
      throw new OverdraftException(
        balance, amount);
    balance = balance.minus(amount);
}}
```

Version 4

```
public class Account { …
  /**
   * Withdraw money from account.
   * @param amount to withdraw (double)
   * @return new balance (double).
   */
  public void withdraw(Currency amount)
      throws OverdraftException {
    if (balance.lessThan(amount))
      throw new OverdraftException(
        balance, amount);
   balance = balance.minus(amount);
}}
```

Version 4

```
public class Account { …
  /**
   * Withdraw money from account.
   * @param amount to withdraw (double)
   * @return new balance (double).
   */
  public void withdraw(Currency amount)
      throws Ove                ption {
    if (balance.                mount))
      throw new OverdraftException(
        balance, amount);
    balance = balance.minus(amount);
}}
```

Still Accurate??

Version 4

How do you test-drive comments?

Why comments?

To communicate.

Communicate with literate code and tests.

```
class AccountTest { …
  @Test(expected=OverdraftException.class)
  public void overdraftThrowsException() {
    Currency c1 = new Currency(1000.00,…);
    Currency c2 = new Currency(1000.01,…);
    Account account = new Account(c1);
    account.withdraw(c2);
  }
}
```

Tests as documentation

#2: Here, have an exception!

http://www.damnit.org/2008-02/29osqid.jpg/view

#2: Here, have an exception!

"Use *checked* exceptions."

http://www.damnit.org/2008-02/29osqid.jpg/view

```java
import java.io.*
public class FileFilter {

  public static interface Filter {
    String filterLine(String line);
  }
  …
```

```
…
public void filter(File src, File dest,
    Filter filter) {
  String lineSeparator = …;
  BufferedReader in = new BufferedReader(
    new FileReader(src));
  BufferedWriter out= new BufferedWriter(
    new FileWriter(dest));
  …
```

```
...
public void filter(File src, File dest,
    Filter filter) {
```

```
FileFilter.java:10: unreported exception
java.io.FileNotFoundException; must be caught ...
        BufferedReader in  = new BufferedReader(new ...
```

```
  BufferedWriter out= new BufferedWriter(
    new FileWriter(destination));

  ...
```

```
…
public void filter(File src, File dest,
    Filter filter)
  throws FileNotFoundException,
         IOException {
  String lineSeparator = “…”;
  BufferedReader in = new BufferedReader(
    new FileReader(source));
  BufferedWriter out= new BufferedWriter(
    new FileWriter(destination));
  …
```

How are the exceptions handled?

```
… main(String[] args) {
  … workFlowProcess(…) {
    … stuffInTheMiddle(…) {
      … manipulateFiles(…) {
        FileFilter fileFilter = new …;
        fileFilter.filter(…);
        …
```

Who handles the `FileNotFoundException` and IOException?

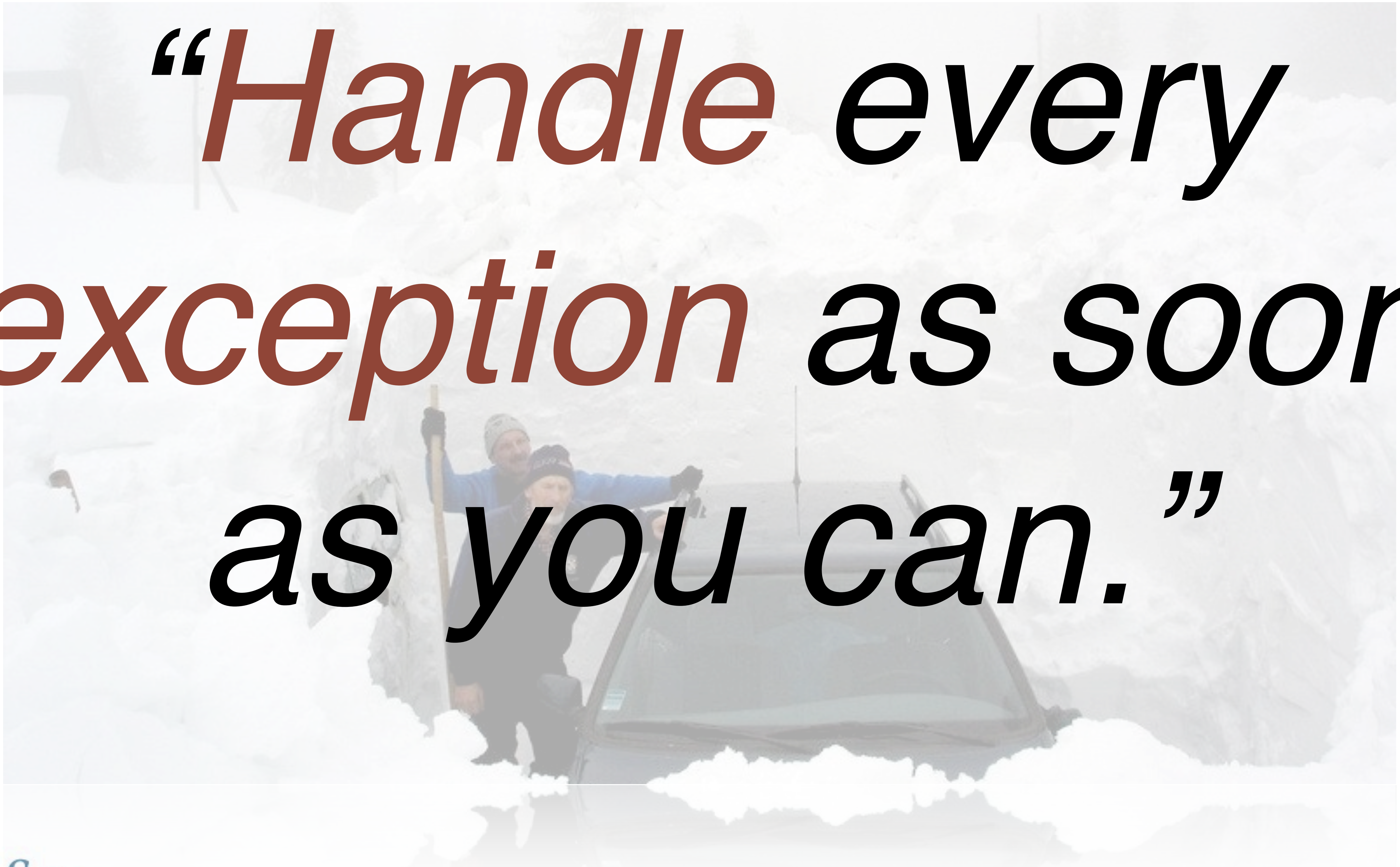
Could add throws at every level of the stack...

Namespace pollution

Could eat the exception immediately...

Do you *really* know how to recover??

#2: Here, have an exception!



“*Handle every exception as soon as you can.*”

http://www.damnit.org/2008-02/29osqid.jpg/view

```
… main(String[] args) {
  … workFlowProcess(…) {
    … stuffInTheMiddle(…) {
      … manipulateFiles(…) {
        try {
          FileFilter fileFilter = new …;
          fileFilter.filter(…);
        } catch (Throwable th) {
          log(th);
          // Now what!!
        }
        …
```

Eat it…?

```
… main(String[] args) {
  … workFlowProcess(…) {
    … stuffInTheMiddle(…) {
      … manipulateFiles(…) {
        FileFilter fileFilter = new …;
        fileFilter.filter(…);
        …
```

One of these methods knows what to do.

Use
unchecked
exceptions.

Handle exceptions strategically.

```
… main(String[] args) {
  … workFlowProcess(…) {
    … stuffInTheMiddle(…) {
      … manipulateFiles(…) {
        FileFilter fileFilter = new …;
        fileFilter.filter(…);

        …
```

Maybe you catch file IO exceptions and attempt recovery...

#3: Just because you're paranoid doesn't mean you shouldn't check for **nulls**...

http://www.flickr.com/photos/brewedfreshdaily89/2909152638/in/photostream/

```
…
public void filter(File src, File dest,
    Filter filter)
  throws FileNotFoundException,
         IOException {
  if (src == null || dest == null ||
      filter == null)
    panic("…");
  …
```

```
…
public void filter(File src, File dest,
    Filter filter)
  throws FileNotFoundException,
         IOException {
  if (src == null || dest == null ||
      filter == null)
    panic("…");
  …
```

Null checks obscure code.

Null checks have to be test driven.

But, isn't defensive programming good?

Use strategic data validation.

Check at module boundaries.

Weed out nulls with automated tests.

#4: We can build a better X in house.

http://picturethis.channel4.com/photo/9075

NIH syndrome.

Examples: message queues.

Examples: rules engines.

Examples: web template engines.

What’s the cost of development?

What's the cost of *long-term* maintenance?

In-house tools become a maintenance burden.

Porting to a 3rd-party tool is painful.

#5: I’ll grab my **own** JDBC connection, thank you very much!

```
public void transfer(
    Account src, Account dest,
    Currency amount) {
  try {
    src.withdraw(amount);
    dest.deposit(amount);
    Class.forName("sun.jdbc...");
    Connection con =
      DriverManager.getConnection(…);
    Statement stmt = con.createStatement();

    …
```

```
public void transfer(
    Account src, Account dest,
    Currency amount) {
  try {
    src.withdraw(amount);
    dest.deposit(amount);
    Class.forName("sun.jdbc...");
    Connection con =
      DriverManager.getConnection(…);
    Statement stmt = con.createStatement();
    …
```

… or any other “hard” dependency.

How do you unit test transfer?

Hide dependencies behind abstractions.

Inject dependencies: inversion of control.

```
public void transfer(
    Account src, Account dest,
    Currency amount) {
  try {
    src.withdraw(amount);
    dest.deposit(amount);
    accountPersister.persist(src);
    accountPersister.persist(dest);
    …
```

```
public void transfer(
    Account src, Account dest,
    Currency amount) {
  try {
    src.withdraw(amount);
    dest.deposit(amount);
    accountPersister.persist(src);
    accountPersister.persist(dest);
    …
```

accountPersister set through constructor or setter.

For testing, set accountPersister to a test double.

For production, set accountPersister using Spring.

You can remove the persistence code completely...

E.g., using *Aspects*.

#6: Why retest when you can copy and paste?

“*Manual testing hurts.*

So don't edit, retest and redeploy code.

Copy, paste, and tweak it instead!”

⇒ Massive duplication!

Automated testing eliminates the pain.

#7: This code doesn't need to be thread safe.

http://www.flickr.com/photos/billward/359774589/

Folk definition of insanity:

Do the same thing over and over again and expect the results to be different.

“The Problem with Threads”, IEEE Computer, May 2006

That's multithreaded programming in a nutshell.

Code should tell its story.

```
public class Account { …
  public void withdraw(Currency amount)
        throws OverdraftException {
    if (balance.lessThan(amount))
      throw new OverdraftException(
        balance, amount);
    balance = balance.minus(amount);
  }
}
```

```
public class Account { …
  public void withdraw(Currency amount)
      throws OverdraftException {
    if (balance.lessThan(amount))
      throw new OverdraftException(
        balance, amount);
    balance = balance.minus(amount);
  }
}
```

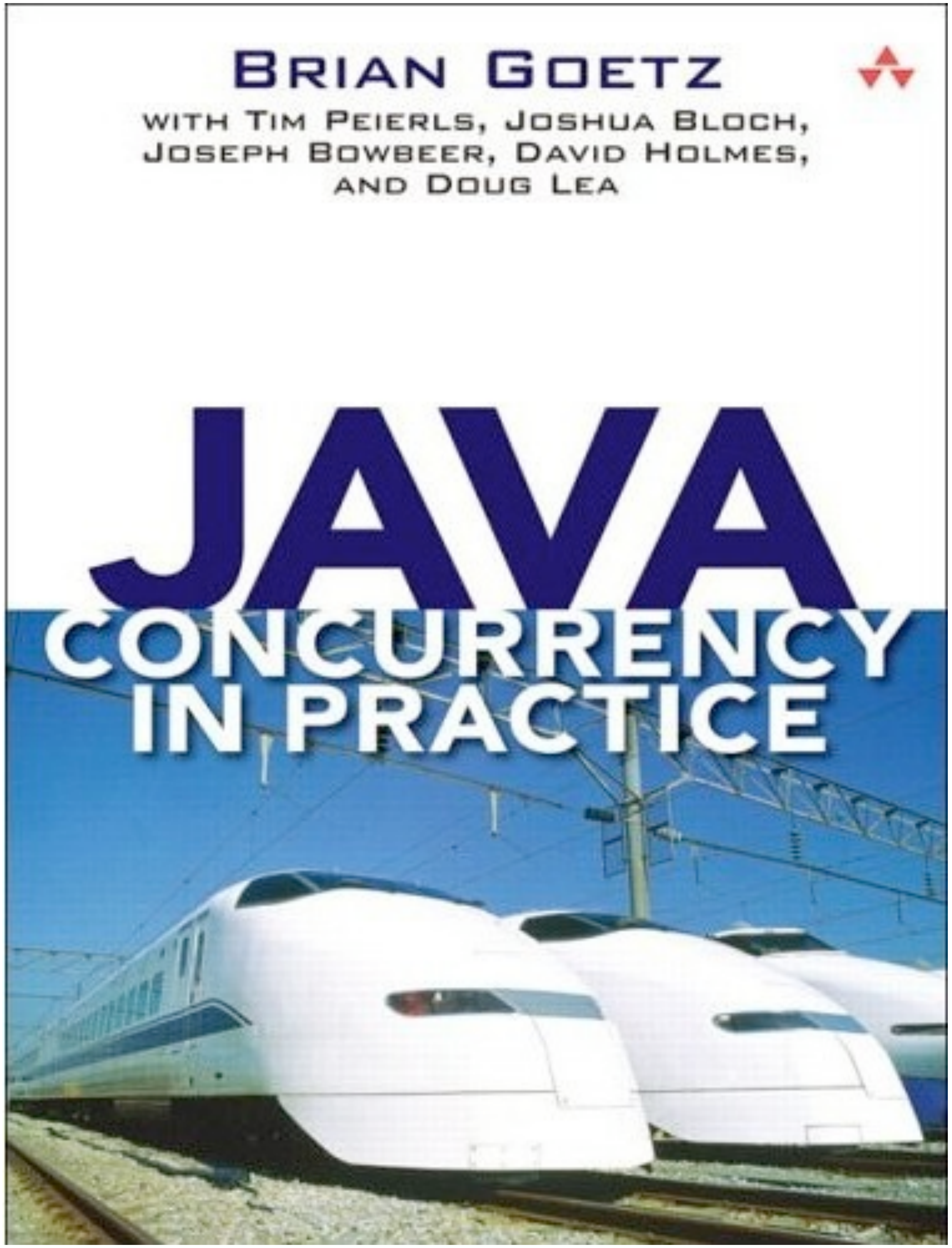
With threads, this code isn't telling me the whole story.

```
public class Account { …
  public void withdraw(Currency amount)
      throws OverdraftException {
    if (balance.lessThan(amount))
      throw new OverdraftException(
        balance, amount);
    balance = balance.minus(amount);
  }
}
```

These two operations must be *atomic!*

2 ways to fix this code:

#1: Use thread synchronization primitives.


BRIAN GOETZ
WITH TIM PEIERLS, JOSHUA BLOCH, JOSEPH BOWBEER, DAVID HOLMES, AND DOUG LEA
JAVA CONCURRENCY IN PRACTICE

#2: Write concurrent code *without* threads.

Use Actors.

Go to Jonas Bonér’s talk tomorrow for other options...

Actors

Message passing between autonomous Actors.

Actors

No shared, mutable state.

Actors

Made famous by Erlang.

Also supported in Scala.

Google: Java actors

#8: Sophisticated code needs sophisticated API's.

http://www.flickr.com/photos/randomurl/440190706/

“Enterprise apps require EJBs.”

Accidental *vs.* essential complexity.

“Do the *simplest thing* that could *possibly work*!”

2 ways to stay focused:

#1: Use Test-Driven Development (TDD).

#2: Use Domain-Specific Languages (DSLs).

```
Vacation vacation = vacation()
   .starting("10/09/2007")
   .ending("10/17/2007")
   .city("Paris")
   .hotel("Hilton")
   .airline("United")
   .flight("UA-6886");
```

http://www.infoq.com/articles/internal-dsls-java

```
Vacation vacation = vacation()
    .starting("10/09/2007")
    .ending("10/17/2007")
    .city("Paris")
    .hotel("Hilton")
    .airline("United")
    .flight("UA-6886");
```

Expresses business logic.

http://www.infoq.com/articles/internal-dsls-java

```
Vacation vacation = vacation()
    .starting("10/09/2007")
    .ending("10/17/2007")
    .city("Paris")
    .hotel("Hilton")
    .airline("United")
    .flight("UA-6886");
```

Hides implementation.

http://www.infoq.com/articles/internal-dsls-java

What are the appropriate details at *this* level of abstraction?

#9: Everything is an object.

© Dean Wampler

Most apps are CRUD.

Do you *really* need ORM and OO middleware?

Business rules:

objects

or

functions?

Why are map/reduce and key-value DBs so hot?

Embrace other paradigms:

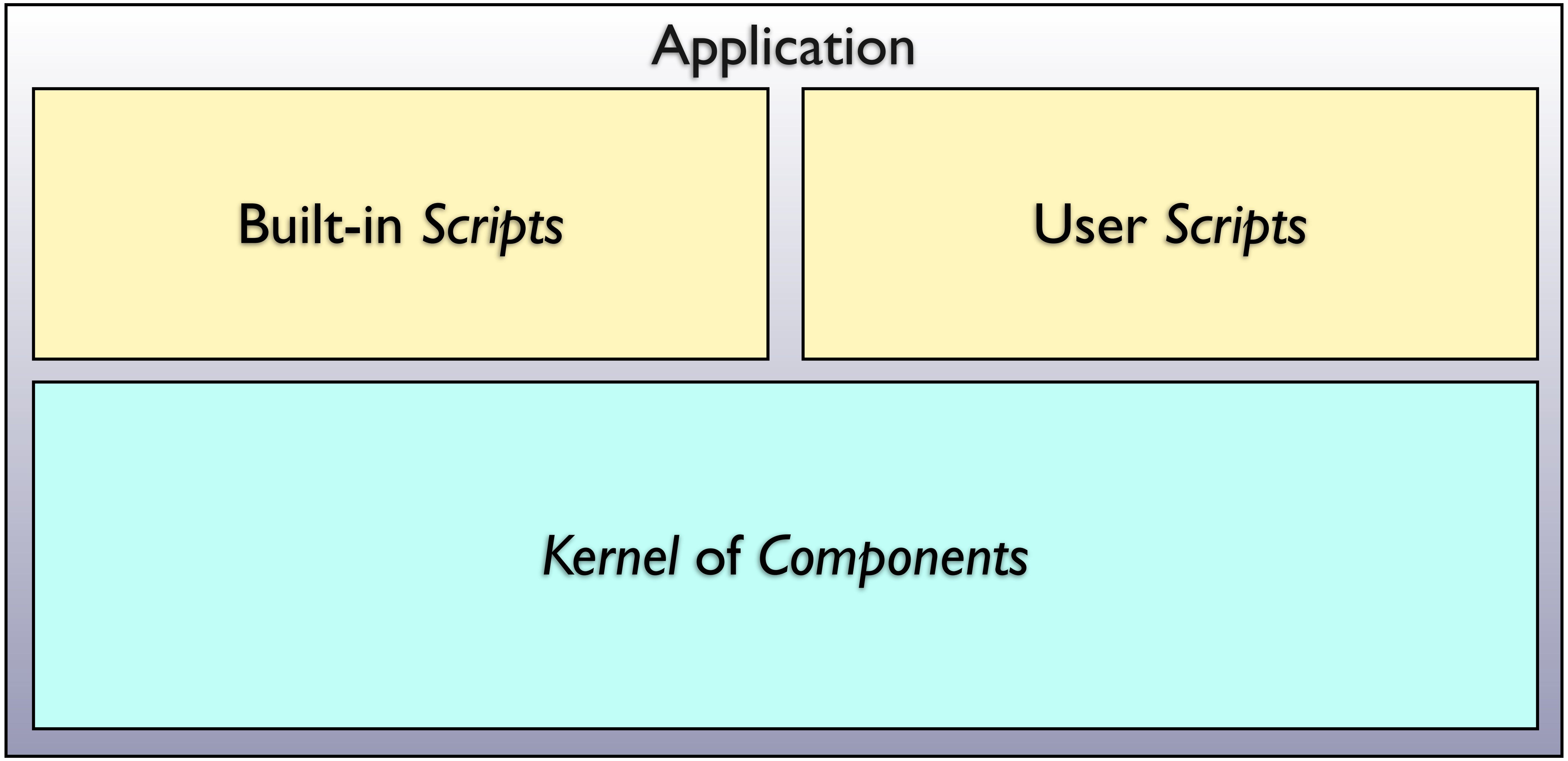
functional, aspects, logic, ...

#10: Java and XML are all we really need.

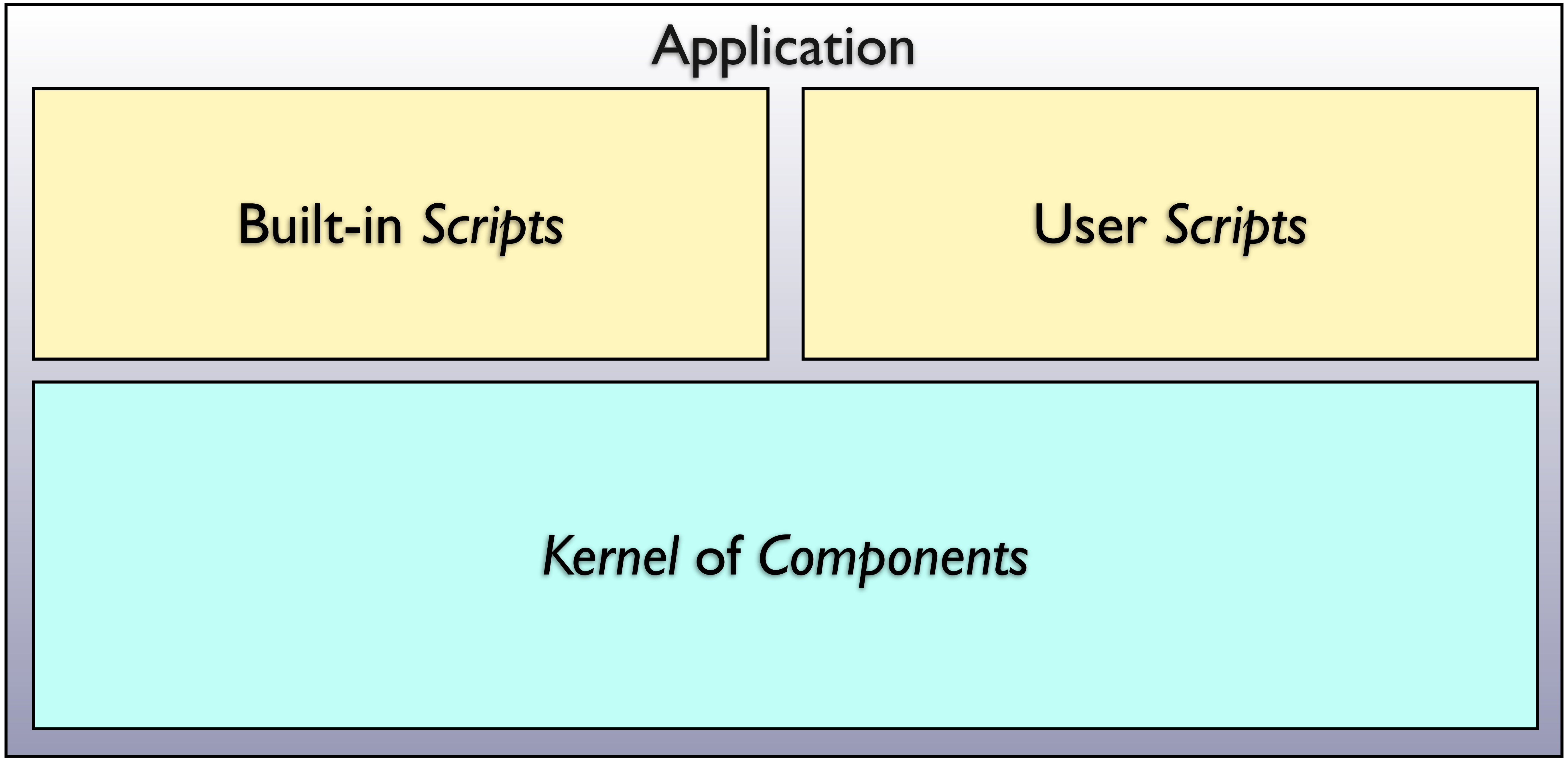
http://www.flickr.com/photos/randomurl/440190706/

Why did we enter XML Hell?

XML is for data, not scripting.



(Java Components) +
(Groovy/JRuby/Jython/... Scripts)
= Applications!



Components + Scripts = Applications

Why is Emacs still relevant?

C + ELisp = Emacs

A Way Forward...

© Dean Wampler

#1: Comments

Communicate thru code and tests.

#2: Exceptions

Handle them strategically.

#3: Paranoid?

Validate data at boundaries.

#4: Dependencies

Use inversion of control.

#5: NIH Syndrome

What is central to *your* business?

#6: Copy & Paste

Avoid duplication.
Automate testing.

#7: Thread Safety

Avoid shared, mutable state.

#8: Complexity

Focus using TDD.
Use DSLs.

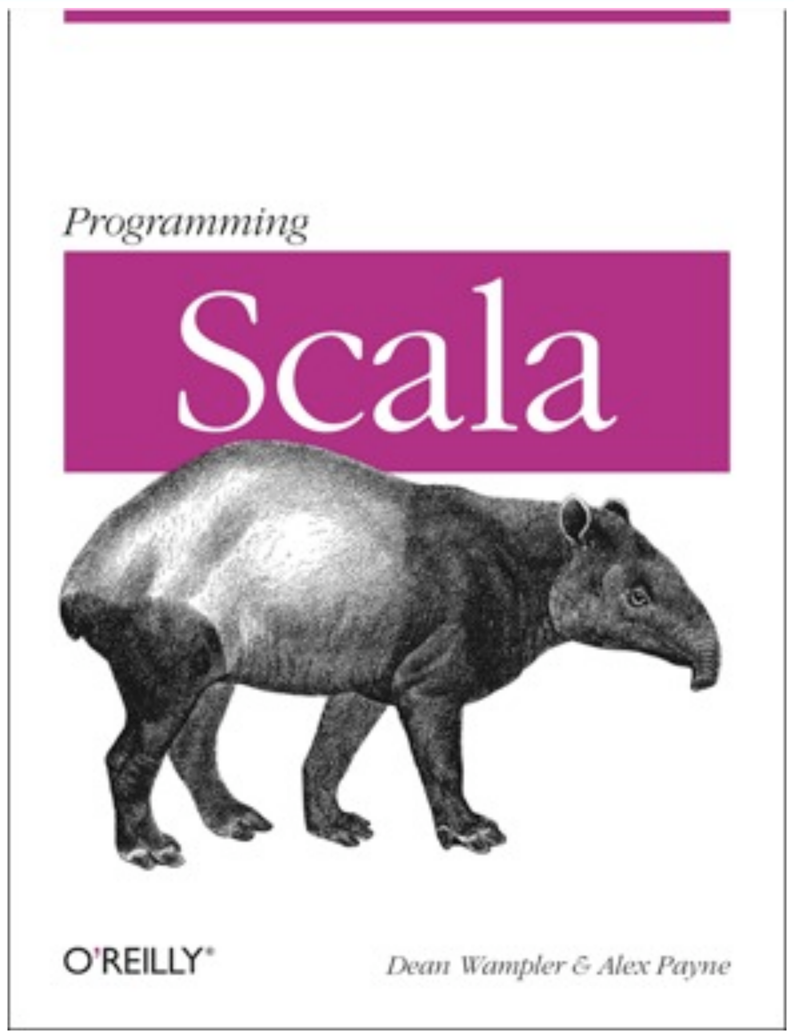
#9: Objects Only?

Use FP, AOP, Relational, Logic...

#10: Java Only?

Components + Scripts = Apps

JavaOne Thank You



Dean Wampler
Object Mentor, Inc.

dean@objectmentor.com
@deanwampler
blog.objectmentor.com
polyglotprogramming.com/papers
programmingscala.com