~~Test-Driven Development vs. Test-After Development~~
Doing TDD Well

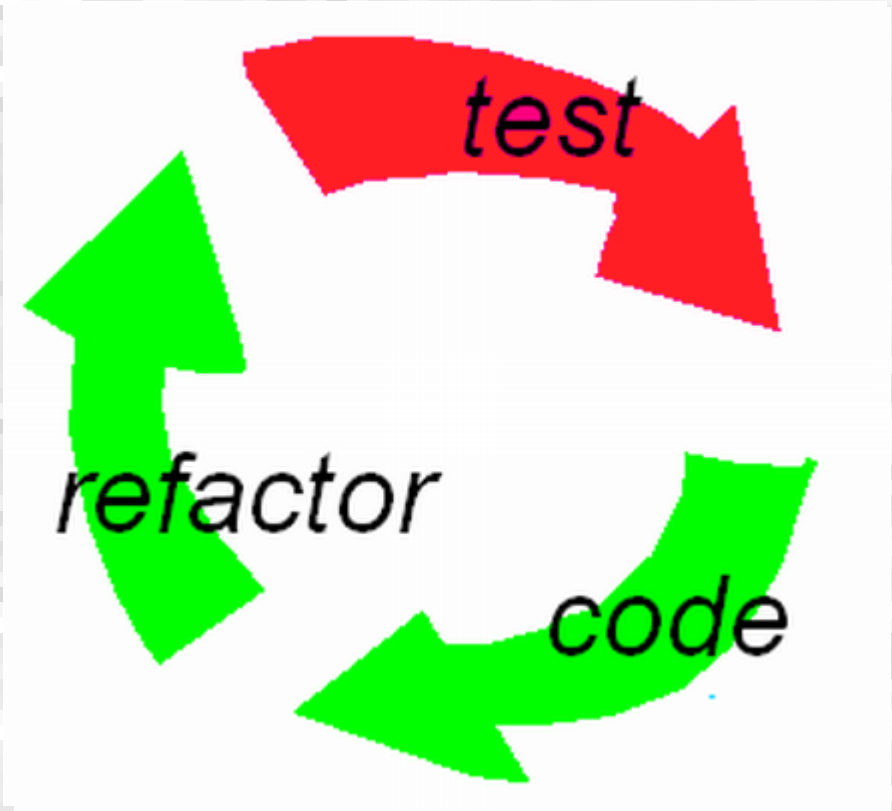
Jeff Langr

Langr Software Solutions

http://langrsoft.com

Copyright © 2007, Langr Software Solutions. All Rights Reserved.

Test-Driven Development



A design technique

Copyright © 2007, Langr Software Solutions. All Rights Reserved.

Unit Testing
Test-after (TAD) vs. Test-first (TDD)

- Allows some refactoring
- Coverage levels up to ~75%
- No direct design impact
- Can reduce defects
- Can be treated as separate task

Unit testing is:
- expensive
- never the whole picture

- Enables continual refactoring
- Coverage approaching 100%
- Drives the design
- Significantly reduced defects, debugging cycles
- Part of the coding process
- Clarifies, documents understanding of requirements
- Continual progress, consistent pacing
- Continual feedback and learning
- Sustainable

Copyright © 2007, Langr Software Solutions. All Rights Reserved.

Doing TDD Well: Some Simple Suggestions

There is no "advanced" TDD

Copyright © 2007, Langr Software Solutions. All Rights Reserved.

Doing TDD Well—Think About

It's just code!

- Spec by example
- Testability and design
- Incrementalism

- Keeping it simple

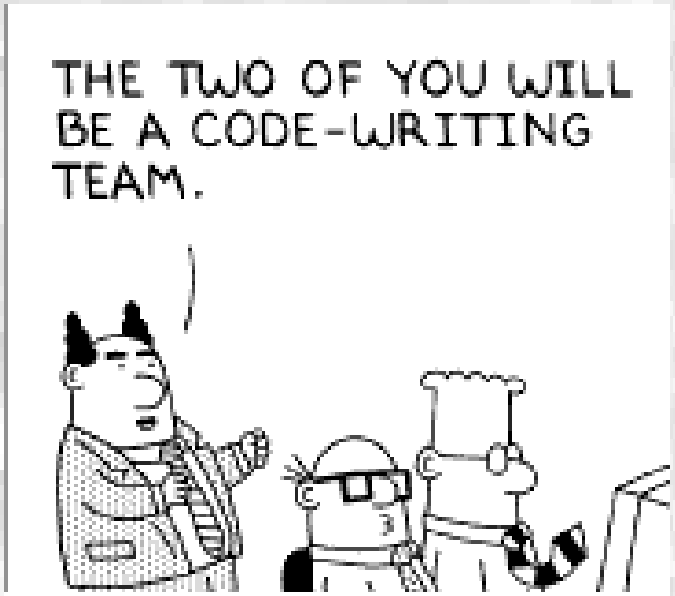
Copyright © 2007, Langr Software Solutions. All Rights Reserved.

Doing TDD Well

Practice

Copyright © 2007, Langr Software Solutions. All Rights Reserved.

Doing TDD Well



Pair

Copyright © 2007, Langr Software Solutions. All Rights Reserved.

Doing TDD Well

Paraphrase

Copyright © 2007, Langr Software Solutions. All Rights Reserved.

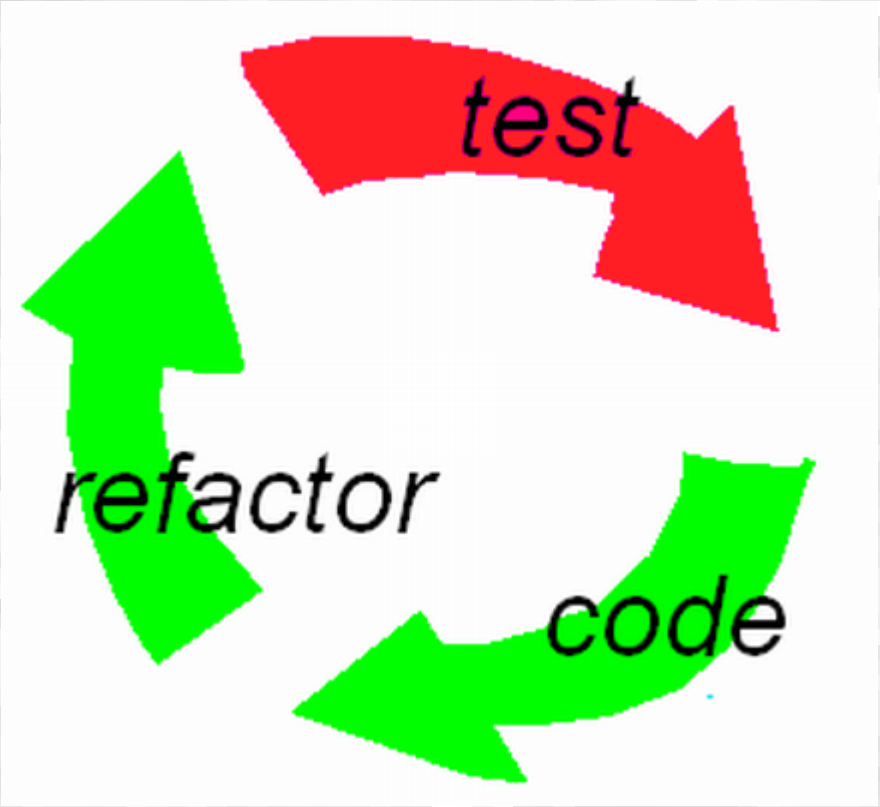
Test everything

- Use integration tests if necessary, but minimize
- Don't avoid tests for difficult challenges
- Decompose tougher problems; isolate complexity

Copyright © 2007, Langr Software Solutions. All Rights Reserved.

Fail first

Take *smaller* steps than you are now



Copyright © 2007, Langr Software Solutions. All Rights Reserved.

Run all the tests

10-minute rule

- Discard code
- Requires fast tests
  - Unit vs. integration isn't as important as fast vs. slow

Copyright © 2007, Langr Software Solutions. All Rights Reserved.

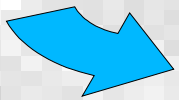
Keep the build green

Copyright © 2007, Langr Software Solutions. All Rights Reserved.

Refactor zealously

- Little things matter
- Tests too
- Consider "2nd time" instead of "3rd time" refactoring


```
writer.write(headerText);
writer.newLine();
writer.write(detailText);
writer.newLine();
```



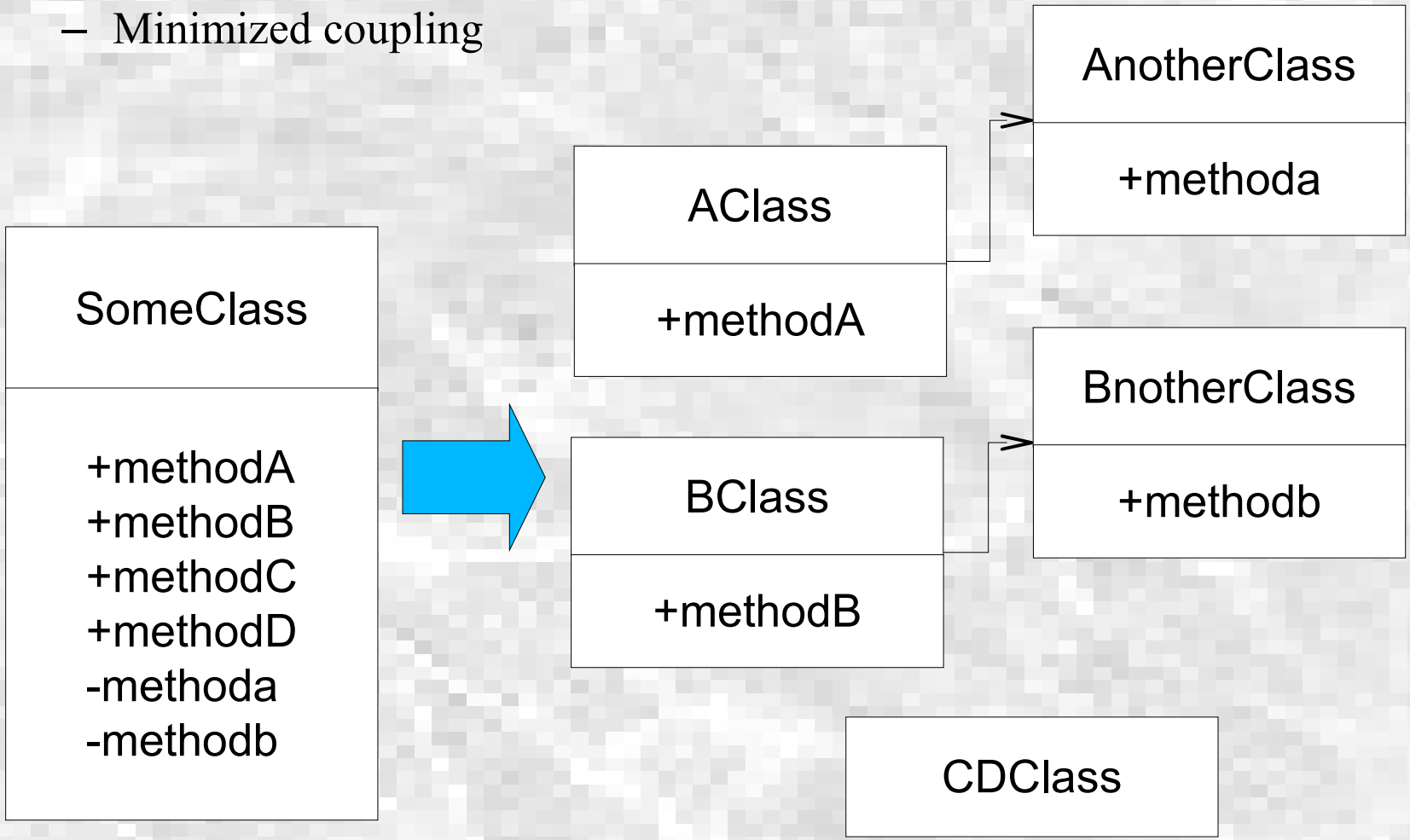
```
writeLine(writer, headerText);
writeLine(writer, detailText);

private void writeLine(
    BufferedWriter writer, String text)
        throws IOException {
    writer.write(text);
    writer.newLine();
}
```

Copyright © 2007, Langr Software Solutions. All Rights Reserved.

Don't forget OO 101

- Very small single-responsibility classes
- Minimized coupling



Copyright © 2007, Langr Software Solutions. All Rights Reserved.

Heed cohesion in tests

- Decrease asserts per test
  - But don't insist on “always one”
- Build tests around behavior/cases, not methods
- Build fixtures around common setup

Copyright © 2007, Langr Software Solutions. All Rights Reserved.

Rename continually

```
@Test public void something()
@Test public void create()
@Test public void defaultCreate()
@Test public void isEmptyOnDefaultCreation()
```

Copyright © 2007, Langr Software Solutions. All Rights Reserved.

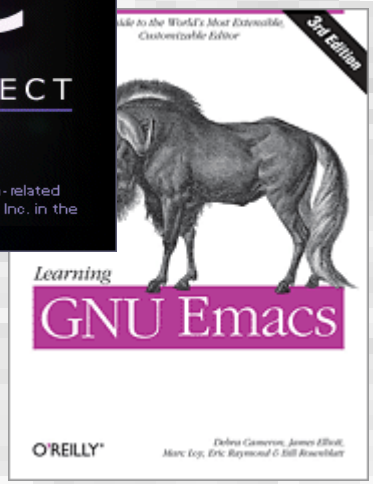
Don't overuse mocks

- But don't refuse to use them
- Mocks tightly couple tests to production code
  - Can violate encapsulation
  - Can inhibit refactoring

Copyright © 2007, Langr Software Solutions. All Rights Reserved.

Use good tools

Master them



Copyright © 2007, Langr Software Solutions. All Rights Reserved.

Don't code for the future

Think and act hard about the present

Keep a to-do list

- Scrap by check-in, task, day, iteration end

Copyright © 2007, Langr Software Solutions. All Rights Reserved.

Never be blocked! (Uncle Bob's “prime directive”)

- Abstract away volatility
- Decouple from others
- Don't wait for definition
  - Start the feedback loop
  - Start the process

Copyright © 2007, Langr Software Solutions. All Rights Reserved.

Read

- Books
  - **Test Driven: TDD and Acceptance TDD for Java Developers**, Lasse Koskela
  - **xUnit Test Patterns**, Gerard Meszaros
  - **Implementation Patterns**, Kent Beck
- Articles
- Yahoo! groups
  - testdrivendevelopment, extremeprogramming, JUnit, etc.

Copyright © 2007, Langr Software Solutions. All Rights Reserved.

Work as part of a team

- Look for standards
- Welcome opportunities to review
- Talk frequently

Copyright © 2007, Langr Software Solutions. All Rights Reserved.

Remain humble

- Keep an open mind
- Be willing to back up and take a different approach
- Be willing to change how you develop
- Revisit failures, to learn
- Get some rest

Copyright © 2007, Langr Software Solutions. All Rights Reserved.

Doing TDD Well

TDD is a skill.

Practice,
practice,
practice.

Copyright © 2007, Langr Software Solutions. All Rights Reserved.