

Test Driven Development in Java

Live and Uncensored



So who's this guy?

- Principal Consultant at Improving Enterprises
- Creator of Infinitest
- Automated Testing Trainer and Mentor

email	benrady@gmail.com
twitter	twitter.com/benrady
web	FeedbackJunkies.com

Why Test Driven?

- Using our code as we write it ensures that it is easy to use
- Testing our code as we write it ensures that it is well tested



The Three Laws



The Three Laws

- I. Do not write production code until you have written a failing test.



The Three Laws

1. Do not write production code until you have written a failing test.
2. Do not write more of a test than is sufficient to fail.



The Three Laws

1. Do not write production code until you have written a failing test.
2. Do not write more of a test than is sufficient to fail.
3. Do not write more production code than is sufficient to pass the current failing test.



Good tests are...

- Fast
- Informative
- Reliable
- Exhaustive



Good tests are...

- Fast
- Informative
- Reliable
- Exhaustive

100s of tests per second



Good tests are...

- Fast *100s of tests per second*
- Informative *Act as executable specifications*
- Reliable
- Exhaustive



Good tests are...

- Fast *100s of tests per second*
- Informative *Act as executable specifications*
- Reliable *Consistently pass or fail*
- Exhaustive



Good tests are...

- Fast *100s of tests per second*
- Informative *Act as executable specifications*
- Reliable *Consistently pass or fail*
- Exhaustive *All valuable behavior is tested*



Good tests are...

- **F**ast *100s of tests per second*
- **I**nformative *Act as executable specifications*
- **R**eliable *Consistently pass or fail*
- **E**xhaustive *All valuable behavior is tested*



Questions?

Concerns?

Doubts?



Meanwhile, in the
“real world”...



“Learning TDD”
is like...



“Learning TDD” is like...

Winning At Poker

Playing Golf

Investing Wisely

Flossing Your Teeth

Making Your Bed

The real path to TDD



The real path to TDD

Write tests, like, whenever



The real path to TDD

Write tests, like, whenever

Write tests after writing code



The real path to TDD

Write tests, like, whenever

Write tests after writing code

Write tests before writing code

The real path to TDD

Write tests, like, whenever

Write tests after writing code

Write tests before writing code

Write expressive tests before writing code

The real path to TDD

Write tests, like, whenever

Write tests after writing code

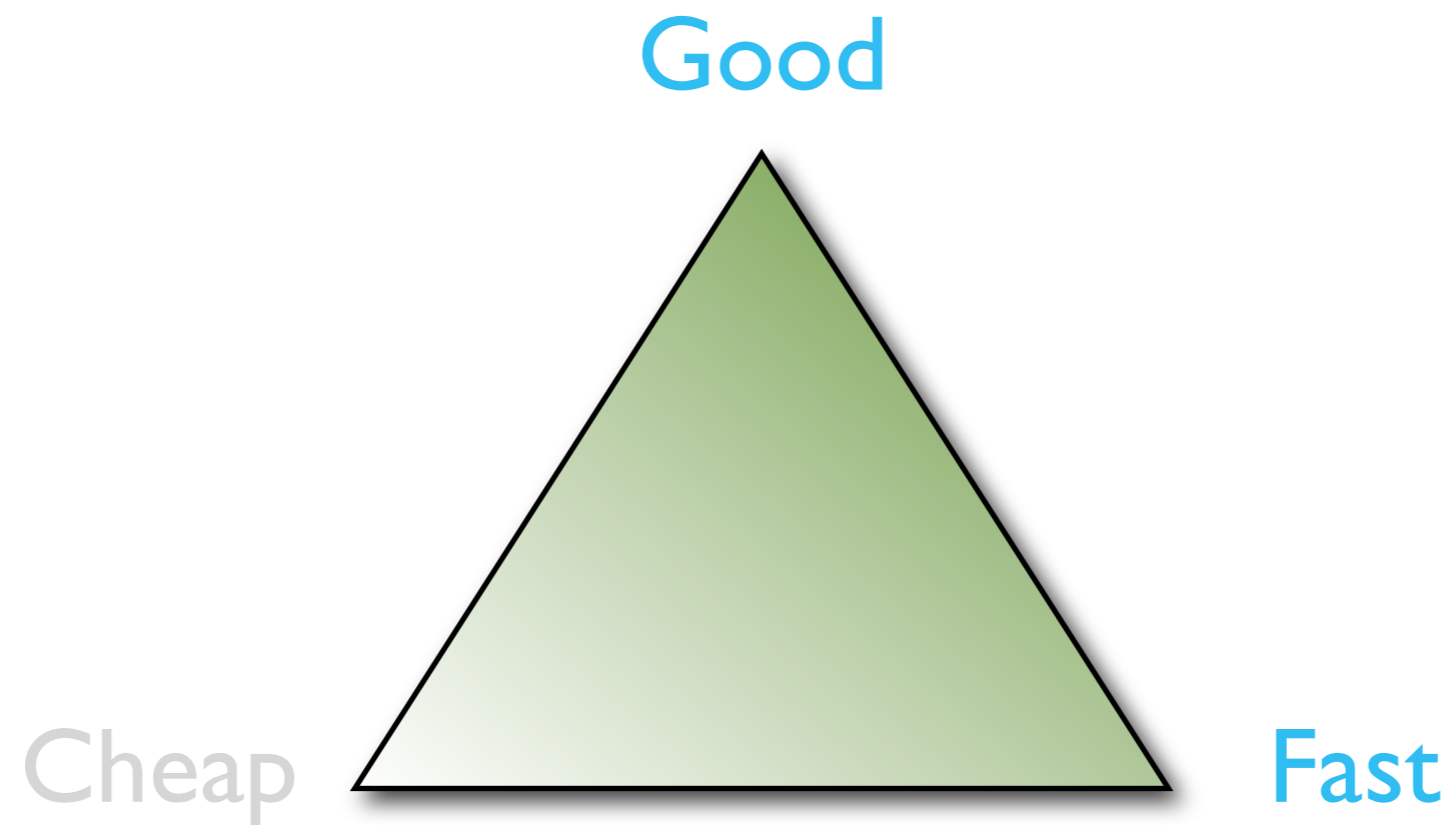
Write tests before writing code

Write expressive tests before writing code

Write the minimum amount of expressive test before writing just enough code to make it pass

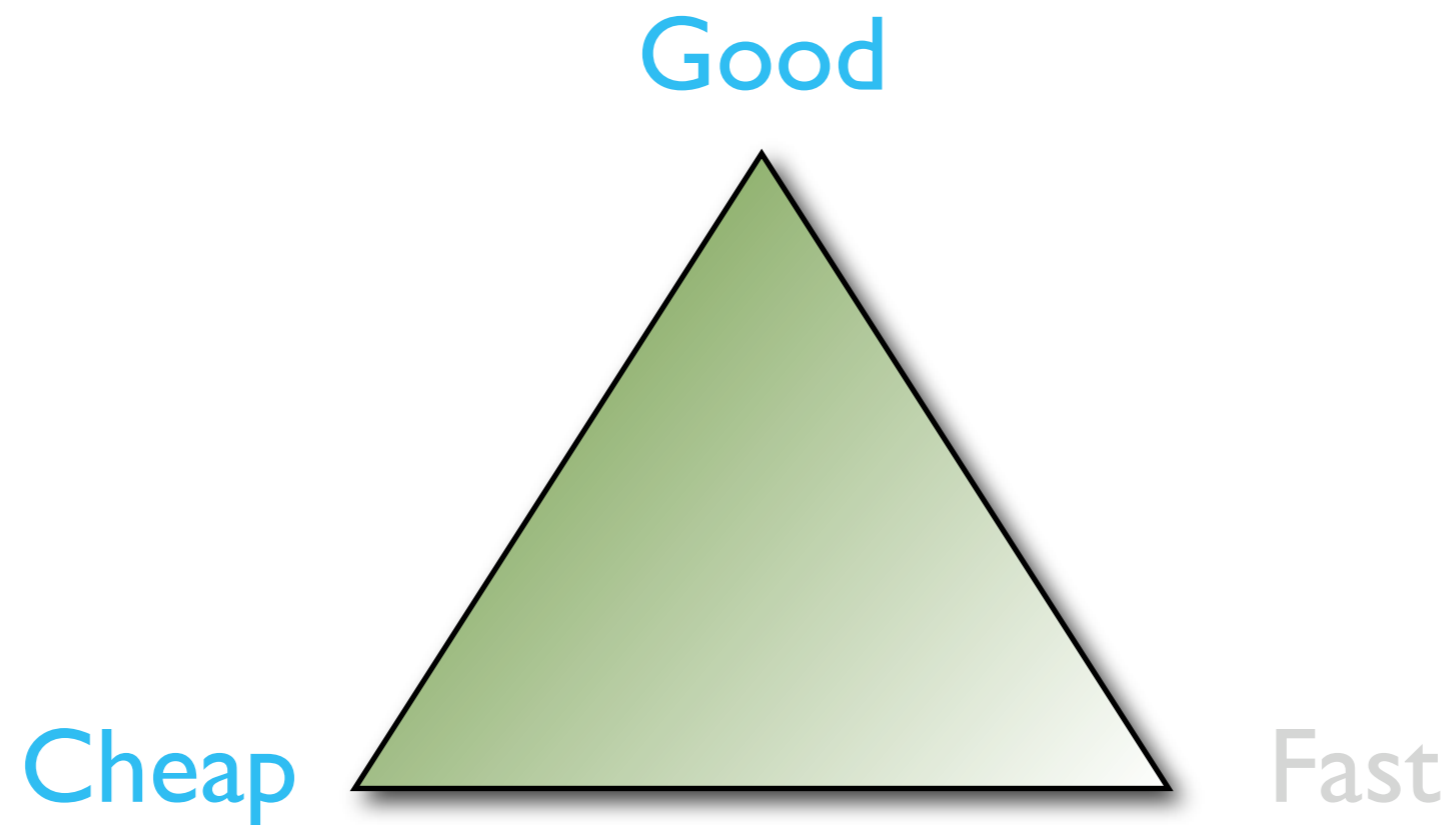
Option #1

Mentoring from an experienced practitioner



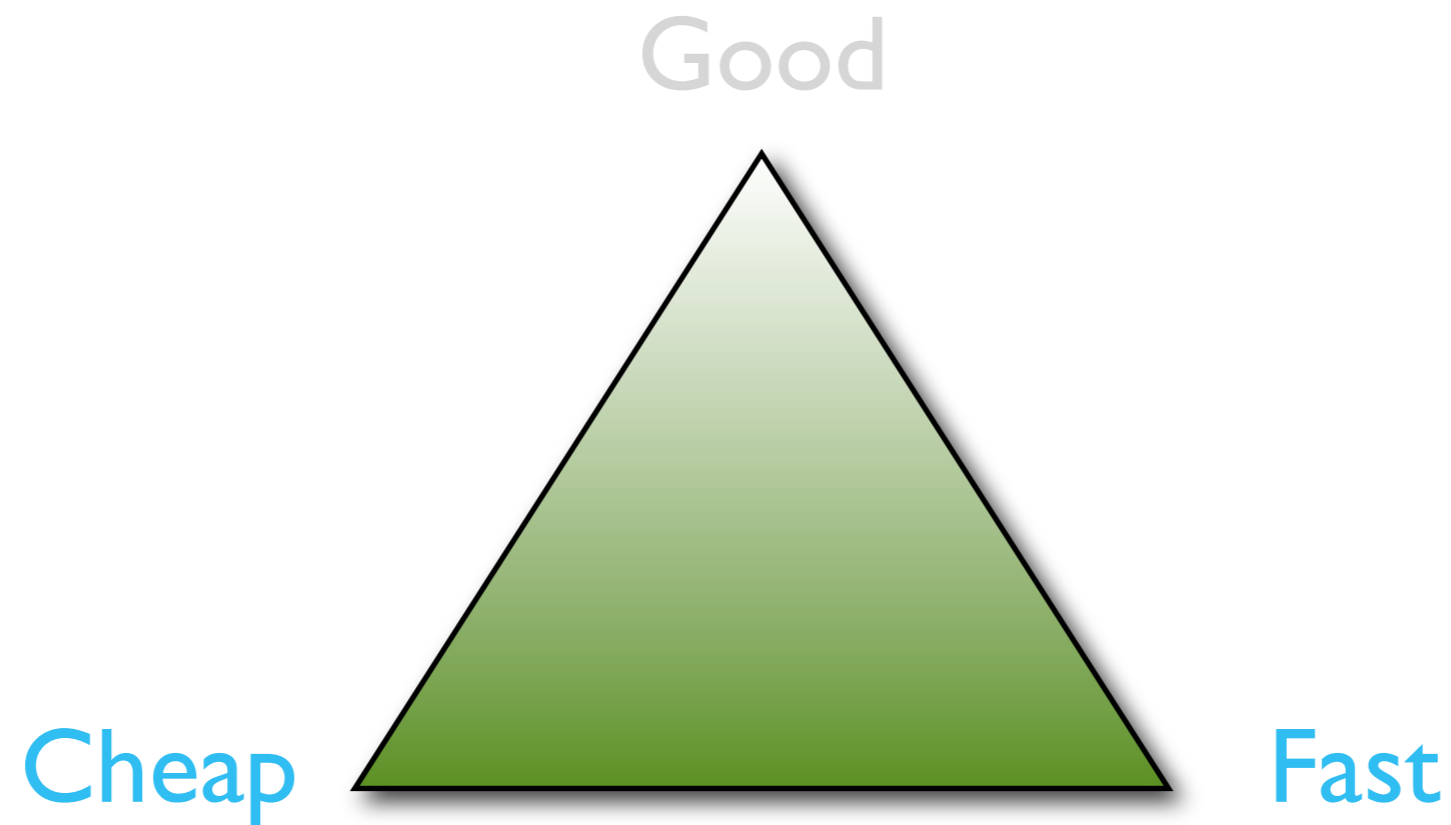
Option #2

Learning on your own



Option #3

Conference Presentations



Rules to learn by



Rules to learn by

I. Cycle frequently



Rules to learn by

1. Cycle frequently
2. Run tests (more than one) on each change



Rules to learn by

1. Cycle frequently
2. Run tests (more than one) on each change
3. If it hurts, you're doing it wrong



Rules to learn by

1. Cycle frequently
2. Run tests (more than one) on each change
3. If it hurts, you're doing it wrong
4. Code Code Code Code Code!



Today's Rules



Today's Rules

- I. Whoever holds the pig decides what we do next.



Today's Rules

1. Whoever holds the pig decides what we do next.
2. After you decide, pass the pig.



Today's Rules

1. Whoever holds the pig decides what we do next.
2. After you decide, pass the pig.
3. If you don't know what do, throw the pig to someone else (even me).



Today's Rules

1. Whoever holds the pig decides what we do next.
2. After you decide, pass the pig.
3. If you don't know what do, throw the pig to someone else (even me).
- 4. Don't be afraid to backtrack!**



Porting Java



Refactor Me!



Dao-ism



Message Dispatcher



Creative Commons Attributions

- “Test Drive” - Campechano
- “Coffee Love” - Po Yang

