Learning Test-Driven Development

Jeff Langr
Sabre Holdings

You can find “starter” test code at:
Hands-On?

• How fast do I code?

Try 'n' keep up!
Hands-On?

Uncomment test code
Hands-On?

Watch & enjoy!
Agile

Continually delivering quality software that responds to changing business needs

How the $#@! do we do that?!??
Test-Driven Development

A design technique
TDD Basics – Stock Portfolio

- Records purchases (symbol, # shares, timestamp)
- Answers whether or not it's empty
- Answers # of symbols
- Answers # of shares for a given symbol
- Throws exception on insert of null symbol
- (Returns purchases between a date range)
Demo—Key Concepts

• Test behavior
• Fail first
• Hardcoding
• Small, small steps
• Incrementalism
• Continual refactoring
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

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

Unit testing is:
- expensive
- never the whole picture
Doing TDD Well

• Practice
• Pair
• Paraphrase
Think About

- Spec by example
- Testability and design
- Incrementalism

*It's just code!*
“We switched our development style practically overnight. That's not to say that we had it all figured out right away, but that we could tell there was no going back.”

—Jerry Jackson
Testing Challenge

• Return the value of a portfolio's holdings
  – Testing goal: prove Portfolio value method logic

• External lookup provides stock values
  – Unfortunately this is volatile

<table>
<thead>
<tr>
<th>Portfolio</th>
<th>StockLookupService</th>
</tr>
</thead>
<tbody>
<tr>
<td>-holdings: List&lt;Holding&gt;</td>
<td></td>
</tr>
<tr>
<td>value(): int</td>
<td>price(symbol: String): int</td>
</tr>
</tbody>
</table>
Faking For Test Purposes

- Use an interface to break the dependency
- The fake emulates behavior that's already tested

Portfolio
- holdings: List<Holding>
  value(): int

<<interface>>
StockLookupService
  price(symbol: String): int

PortfolioTest

MockLookupSvc

ProductionLookupSvc