Slow down to go faster
Slow Down to Go Faster

If you want to move faster, you have to slow down.

3 ways to slow down to go faster

Jerod Santo
Sept 29
http://fuelyourcoding.com/slow-down-to-go-faster/
Testing

Automated test first thing dropped during time crunches

But software has to be tested

Manual testing is slow

Writing tests makes you think about problem which reduces development time
Naming Things

Only 2 hard problems in CS
  cache invalidation
  naming things
  off-by-one errors

Good names make it easier to understand code

Makes it faster to modify/maintain code
Documentation

Makes it easier to modify/maintain code

Don't write novels - be brief

Document intent not implementation
An Opposing View
Unit testing is teh suck, Urr.
Wil Shipley

When modify code test it your self - try to break the code

Use people to Beta test programs
Another View

bbum
http://www.friday.com/bbum/2005/09/24/unit-testing/

Mac/iOS Core Data library

Unit testing done during development

Unit testing made library solid & stable

Wil Shipley agrees with bbum
Some things are hard to test

GUI
Network connections
Databases
Testing
What to Test

Everything that could possibly break

Test values
   Inside valid range
   Outside valid range
   On the boundary between valid/invalid

GUIs are very hard to test
   Keep GUI layer very thin
   Unit test program behind the GUI, not the GUI
Common Things Programs Handle Incorrectly

Adapted with permission from “A Short Catalog of Test Ideas” by Brian Marick

Any Object
nil pointer

Strings
Empty String

Collections
Empty Collection
Collection with one element
Collection with duplicate elements
Collections with maximum possible size

Numbers
Zero
The smallest number
Just below the smallest number
The largest number
Just above the largest number

struct -2
Some Heuristics

2.8 A class should capture one and only one key abstraction

2.9 Keep related data and behavior in one place
3.3 Beware of classes that have many accessor method defined in their public interface

Related data & operations are may not be in same place
Node Class - Just Accessor methods

Instance Variables
next
previous
data

Methods
next
next:
previous
previous:
data
data:
Another Test

Write 1-3 sentences describing the class

Are there any actions in the description?

If not operations and data may not be in same place
Linked List Example

Operations
  Add elements
  Test if list contains an element
printOn:
  size
includes:

LinkedList>>includes: anObject
    | current |
    head isNil ifTrue: [^false];
    current := head.
    [current notNil] whileTrue: [
        current data = anObject ifTrue: [^true].
        current := current next].
^return false
includes: with Node

LinkedList>>includes: anObject
  head isNil ifTrue: [^false].
  ^head includes: anObject

Node>>includes: anObject
  data = anObject ifTrue: [^true].
  next ifNil: [^false].
  ^next includes: anObject

includes: F
includes: with Node

**LinkedList>>includes: anObject**
- head isNil ifTrue:[^false].
- ^head includes: anObject

**Node>>includes: anObject**
- data = anObject ifTrue:[^true].
- next ifNil:[^false].
- ^next includes: anObject
includes: with Node

LinkedList>>includes: anObject
head isNil ifTrue: [^false].
^head includes: anObject

Node>>includes: anObject
data = anObject ifTrue: [^true].
next ifNil: [^false].
^next includes: anObject
includes: with Node

**LinkedList>>includes: anObject**
- `head isNil ifTrue: [^false].`
- `^head includes: anObject`

**Node>>includes: anObject**
- `data = anObject ifTrue: [^true].`
- `next ifNil: [^false].`
- `^next includes: anObject`
includes: with Node

LinkedList>>includes: anObject
head isNil ifTrue: [^false].
^head includes: anObject

Node>>includes: anObject
data = anObject ifTrue: [^true].
next ifNil: [^false].
^next includes: anObject
includes: with Node

LinkedList>>includes: anObject
   head isNil ifTrue:[^false].
   ^head includes: anObject

Node>>includes: anObject
   next ifNil:[^false].
   ^next includes: anObject
includes: & Nil node

**LinkedList>>includes: anObject**

^head includes: anObject

**Node>>includes: anObject**

data = anObject ifTrue: [^true].
^next includes: anObject

**NilNode>>includes: anObject**

^false

```
A -> C -> F -> null
```