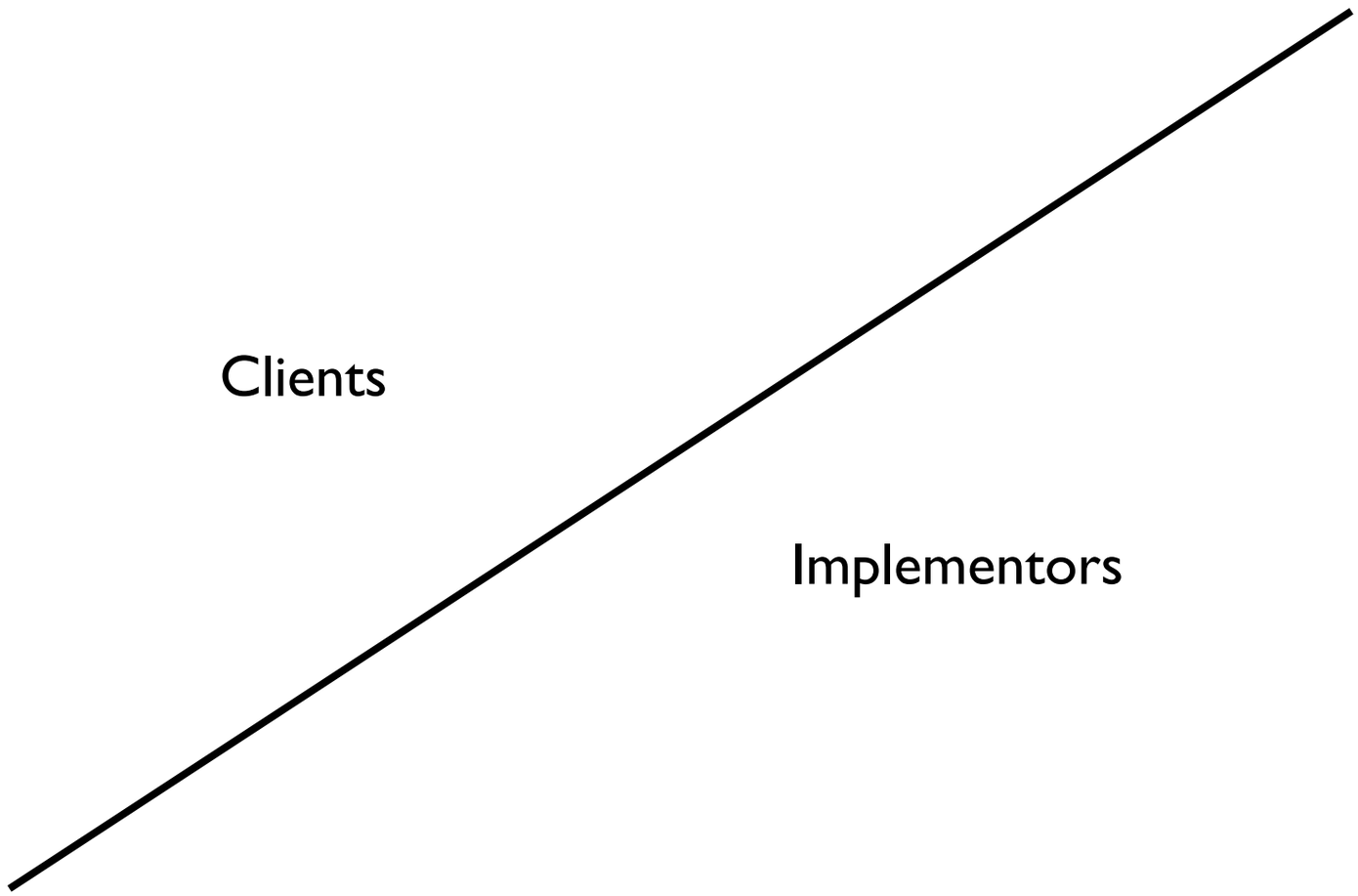
CS 5500

Spring 2013

Testing

```
Midterm Exam.
CS 5500.  Software Development
```

Abstraction Barrier



Interchangeable Parts

Thomas Gainsborough (1727-1788) (public domain)
http://en.wikipedia.org/wiki/File:Gainsborough-Andrews.jpg


D2
BENCHMADE


BENCHMADE

Interchangeable Parts

clear specifications
precise tolerances

Interchangeable Parts

standardized components
abstract data types

Black-Box Testing

test against specification
every test is client code
independent of tested implementation

Glass-Box Testing

may test things unmentioned by specification
may rely on details of the tested code
must be revised when tested code changes

The Purpose of Testing

The Purpose of Testing

to find bugs
testing is a debugging technique
testing can't establish correctness

Example: the Pentium FDIV Bug

4195835.0/3145727.0 = 1.333 820 449 136 241 000 (Correct value)

4195835.0/3145727.0 = 1.333 739 068 902 037 589 (Pentium)

Floating Point Bugs Are Not Uncommon

Code compiled with the /Og option can cause incorrect calculation for doubles. In some cases, it loads only half of a double's bytes into memory causing incorrect results in later calculations.

This bug was corrected in Visual Studio 6.0 Service Pack 3.

http://support.microsoft.com/kb/217033

Why Is Floating Point Arithmetic So Hard To Test?

4195835.0 :

00000000 00000000 00000000 11000000 01111110 00000001 01010000 01000001

3145727.0 :

00000000 00000000 00000000 10000000 11111111 11111111 01000111 01000001

Why Is Floating Point Arithmetic So Hard To Test?

$$2^{128} \approx 3 \cdot 10^{38}$$

Why Is Floating Point Arithmetic So Hard To Test?

$3 \cdot 10^{38}$ flo / (10^9 flops)
$\approx 3 \cdot 10^{29}$ s
$\approx 1 \cdot 10^{26}$ hours
$\approx 4 \cdot 10^{24}$ days
$\approx 1 \cdot 10^{22}$ years
$\approx 1 \cdot 10^{12}$ universes

Computer-Generated Proofs of Correctness

Advanced Micro Devices
J Strother Moore
ACL2 theorem prover
AMD5k86 (FDIV)
AMD K7 (floating point kernel)

What is a Good Test?

What is a Good Test?

Effective C++: 50 Specific Ways to Improve Your Programs and Design, 2nd Edition.

More Effective C++: 35 New Ways to Improve Your Programs and Designs.

How Much Is Enough?

equation coverage
condition coverage
trivial cases
boundary cases
statement coverage

Bad Tests

Main Ideas

abstraction barrier
interchangeable parts
clear specifications
standardized components
abstract data types
black-box testing
glass-box testing
purpose of testing
test coverage
proofs of correctness