PURDUE UNIVERSITY

CS18000: Problem Solving And Object-Oriented Programming

Debugging

2 March 2011

Prof. Chris Clifton

COMPUTER SCIENCES

Finding Errors in Your Code (and that of others)

- Don't make mistakes!
  - Seriously
  - Clean program design
  - Well-separated abstractions
  - Small components
- But what if this isn't enough?

Common Errors

- Loops
  - Off-by-one
  - Infinite
  - Zero
- Arrays
  - Out-of-bounds
  - Uninitialized objects
- Scope

```
int i=0;
void func() {
    for (int i=0; i<5; i++) i+=1;
    // What is i?
}
```

- Equivalence Testing
  - == tests if same object
  - .equals() tests if objects same
- Failure to initialize
  - Null pointer exception
- Numeric Errors
  - Precision
  - Overflow/Underflow
  - Casting

So your code isn't working right. What now?

- Print statements
  - Watch program flow
- Assertions
  - Ensure what you expect is true
- Step through execution
  - One statement at a time
- Breakpoints
  - Inspect at specific statement

Print Statements

- Gather data at various stages of program execution
  - Variable values
  - Was this code reached?
- Can give too much data
  - Particularly in loops
- How to remove?
  - if (debuglevel > 1) System.err.println("test");
- What about embedded devices?

Assertions

- Test if things are as they should be
  - You know something is true
  - But is it?
- assert <condition> [: <value>];
- Use to test things that should *always* hold
  - Internal consistency
  - Proper parameters for private/protected methods
- Do *not* use to check **public** method input
  - Test input and throw exception

Breakpoints

- Interrupt program when it reaches a point
- Stop and take a look
  - Variable values
  - Possibly modify
- Continue
- DrJava:
  - Debugger / Toggle Breakpoint
  - Debugger / Debug Mode
  - Run

Step through program

- Programs execute too fast to follow
  - Slow it down
- First, set a breakpoint
  - Please you want to start stepping
- When it stops, can
  - Step Into (execute next basic statement)
  - Step Over (Treat method calls as statements)
  - Step Out (Finish this method then stop)
  - Resume (to next breakpoint)

Best method: *Use your Brain!*

- Think about what your program is supposed to do
  - Compare with what it does
- Have someone else look at your code
  - *Explain* to them how your code works
  - You'll figure out why it doesn't
  - *Don't expect them to solve the problem…*