Group 5:
David Bangerter
Matt Laroche
Melissa Ludowise
Ben McCann

Overview

- Two parts:
  - antic – checks syntax
  - jlint – checks semantics
- Binaries available for Windows, source provided
  - Didn't compile initially on OS X
  - It's not a commercial product

Scope

- antic can be run on C, C++, Objective C, and Java
  - Suspicious use of operator priorities
    - `x && y & z`
  - No break in switch code
    - ```
      switch (action) {
        case op_remove: do_remove();
        case op_insert: do_insert();
        case op_edit: do_edit();
      }
      ```
  - Lower case l at the end of a long constant
    - `long l = 0x111111l;`
  - And more things that make code hard to read but aren't language violations

Scope cont.

- jlint is run on Java only
  - Bounds checking
  - Deadlock detection
  - Race conditions
    - (Variables not declared volatile when accessed by multiple threads)
  - Catches redundant and suspicious calculations
    - ```
      public boolean foo(int x, int y) {
        return ((x & 1) == y*2);
        // will be true only for x=y=0
      }
      ```

Running

- `$ antic –java “path to source dir”`
  - Can also do `antic –java *.java`
- `$ jlint “path to source dir”`
  - Can also do `jlint +verbose *.class`

Errors Caught

- antic caught no errors in our code base
- jlint caught one error:
  - `if(currentLine == null || currentLine == "")`
  - Should have been:

    `if(currentLine == null || currentLine.equals(“”))`

Errors Caught Cont.

- jlint also caught two errors in the java.lang package when run on our code
  - `java\lang\Double.java:1: hashCode() was overridden but not equals()`
  - `java\lang\Integer.java:1: hashCode() was overridden but not equals()`
- These probably should have been suppressed, as it is very unlikely that either of these classes have errors with their equals() or hashCode() methods

Benefits

- Very fast
- Low learning curve
- Do not have to do any configuration
- Do not have to tell it anything about your code
  - Don't even need the source code
    - But error messages are more descriptive with it
- Will help you write better code
  - if `(x == y & 1)` – there should probably be another set of parentheses for clarity

Drawbacks

- Does not cover a lot
  - Only caught one error in our code
- For synchronization it may produce too many warnings to be useful
  - They actually recommend disabling much of the synchronization warnings!

Side notes

- There's lint like tools for other languages
  - splint for C
  - PC-Lint for C/C++
  - Matlab
- Ran the tool on some Sun code
  - They don't mark some shared variables as `volatile`
- Ran the tool on a Hibernate class file
  - They don't check for `null` sometimes

Conclusion

- jlint's fairly helpful
- It does not catch many bugs, but will still save you time especially given the low overhead in learning and using it as a tool
- It would be nice to have integrated into Eclipse so it's run on the fly and not later