



# CS1632, LECTURE 21: STATIC ANALYSIS, PART 2

Bill Laboon



# Bug Finders

- Static tools that focus on finding defects without executing code
- Full of false positives!
- Can be pointers to unclear or incorrect code

# Example

```
public void doStuff(int x) {  
    if (x == 0) {  
        x = 1;  
    } else {  
        x = 3;  
    }  
    x = 6;  
}
```

# Useless method

- Has no return value
- Has no side effects
- Does nothing except take up space on stack

# Example

```
public static void main(String[] args) {  
    double x = 0.1;  
    double y = 0.2;  
    double z = x + y;  
    if (z == 0.3) {  
        System.out.println("math works!");  
    } else {  
        System.out.println("math is arbitrary!");  
    }  
}
```

# Direct Equality Comparison of Floating-Point Values

- Floating-point values are approximations
- Always check to see if values are within epsilon of each other, e.g.
  - `if (Math.abs(z - 3.0) < 0.01) { ... }`
- Or use `BigDecimal`, `Rational`, etc.

# Example

```
public double calculate() {  
    double x = Math.sqrt(90);  
    return x;  
}
```

# X will always be the same value

- Just put the calculated value instead of calculating each time



# Example

```
public class Quux {  
    public int numBaz = 0;  
  
    public Quux(int x) {  
        numBaz = x;  
    }  
  
    public boolean equals(Object o) {  
        if (o.getClass() == Quux.class) {  
            return ((Quux) o).numBaz == this.numBaz;  
        } else {  
            return false;  
        }  
    }  
}
```

# equals() will stop working if you subclass this!

- Explicitly checking class in an equals() method
- Use `this.getClass()` instead

# All of these issues can be found without running code

- Some are performance defects
- Some are functional defects (causing incorrect behavior)
- Some are just confusing code – which can cause even more problems!

# Linters

- Poorly written code can cause problems
- Multiple people writing code in different styles can cause issues

# Imagine reading this (VALID!) code...

```
public int DOSOMETHING(int num) {  
    int nUmScHnIrPs = num * 2;  
    int NumNirps = nUmScHnIrPs - 1;  
if (NumNirps >  
6) {  
    if (NumNirps < 10)  
    {  
        return 1;  
    } else  
    {  
        return 4;  
    }  
}  
return 5;  
}
```

Linters allow an entire team to use consistent spacing, tabs, variable naming, etc.

- Very, very common!
- I can't remember ever working on professional code that did not have a style guide
- It's been over a decade since I worked on code which did not have an automated tool to check it

# Let's see some in action...

- Findbugs – bug-finding static analysis software
- checkstyle – Java linter