

Objectives

- Metrics Plugin
- Liskov Substitution Principle
- Code Critique
 - Identifying smells
- Refactoring

Oct 27, 2008

Sprenkle - CS209

1

Review

- What goal are we designing to?
- What is the typical fix for code smells?
 - What is a limitation of those fixes?

Oct 27, 2008

Sprenkle - CS209

2

Metrics Plugin

- Provides information about your classes
 - # of classes
 - # of lines of code per method
 - # of attributes
 - Coupling (to/from package)
 - ...
- <http://metrics.sourceforge.net>

Oct 27, 2008

Sprenkle - CS209

3

Example: Lack of Cohesion of Methods (LCOM)

- A measure for a class's cohesiveness
- Calculated with the Henderson-Sellers method:
 - If $m(A)$ is the number of methods accessing an attribute A , calculate the average of $m(A)$ for all attributes, subtract the number of methods m and divide the result by $(1-m)$
- Low value \rightarrow a cohesive class
- Value close to 1 \rightarrow a lack of cohesion
 - Suggests class might better be split into a number of (sub)classes

Oct 27, 2008

Sprenkle - CS209

4

Liskov Substitution Principle (LSP)

- Named after Barbara Liskov
 - Professor of Engineering at MIT
- The substitution principle:

If for each object o_1 of type S there is an object o_2 of type T such that for all programs P defined in terms of T , the behavior of P is unchanged when o_1 is substituted for o_2 , then S is a subtype of T .
- In other words...

Functions that use pointers or references to base classes must be able to use objects of derived classes without knowing it.

Oct 27, 2008

Sprenkle - CS209

5

Code Smell: Using instanceof

```
public void drawShape( Shape shape ) {
    if ( shape instanceof Square ) {
        drawSquare(shape);
    }
    else if( shape instanceof Circle ) {
        drawCircle(shape);
    }
}
```

- Why isn't this good code?
- How could we write this in a better way?

Oct 27, 2008

Sprenkle - CS209

6

Code Smell: Using instanceof

- Previous example: had to know all of the Shape classes
 - Update whenever a Shape is added or removed
- Better code:

```
public void drawShape( Shape shape ) {
    shape.draw();
}
```

Oct 27, 2008

Sprenkle - CS209

7

Another Example: Rectangle Class

```
public class Rectangle {
    private int myHeight;
    private int myWidth;

    public void setWidth( int w ) {
        myWidth = w;
    }

    public void setHeight( int h ) {
        myHeight = h;
    }

    // getters...
}
```

Oct 27,

8

Square Class

- A square is a rectangle
 - But a rectangle is not a square
- In the interest of code reuse


```
public class Square extends Rectangle
```
- Any problems with this implementation?
 - Inherits:

```
private int myHeight;
private int myWidth;
public void setWidth( int w );
public void setHeight( int h );
```

Oct 27, 2008

Sprenkle - CS209

9

To Keep Square Consistent...

```
public void setWidth( int w ) {
    super.setWidth(w);
    super.setHeight(w);
}

public void setHeight( int h ) {
    super.setWidth(h);
    super.setHeight(h);
}
```

Oct 27, 2008

Sprenkle - CS209

10

But What About Users of Classes?

- Consider the function:


```
public void testFunction( Rectangle r ) {
    r.setWidth(5);
    r.setHeight(4);
    assertEquals(20, r.getWidth()*r.getHeight());
}
```
- What happens if it's called with a Square?

Oct 27, 2008

Sprenkle - CS209

11

The Problem

- A Square object is **not** a Rectangle object
- Behaviors are different
 - Clients depend on behaviors
- All derivatives of class must have the same behavior

Oct 27, 2008

Sprenkle - CS209

12

Design by Contract

- Methods of classes declare preconditions and postconditions
 - Preconditions must be met for method to execute
 - After executing, postconditions must be true
 - Example for Rectangle's `setWidth`:
 - `myWidth == newWidth && myHeight == oldHeight`
- For derivatives
 - Preconditions can only be weakened
 - Postconditions can only be strengthened
 - Derivatives must adhere to constraints for base class

Oct 27, 2008

Sprenkle - CS209

13

Summary of LSP

- Open-closed principle (OCP) is fundamental to Object-oriented Design
- Liskov Substitution Principle (a.k.a. design by contract) is an important feature of programs that conform to OCP
 - Derived types must be completely substitutable for their base types
 - Derived types can then be modified without consequence

Oct 27, 2008

Sprenkle - CS209

14

CODE CRITIQUE

Oct 27, 2008

Sprenkle - CS209

15

Discussion of Bins Solution

- What does the code do?
 - What is the purpose/responsibility of each class?
- What are the good parts of the code?
- What are some of the code smells?

Oct 27, 2008

Sprenkle - CS209

16

Assignment 10: Code Critique & Refactoring

- Given: a problem specification and a solution to the problem
 - You refactoring your own code is emotional
 - More objective with someone else's solution
- Goals
 - Read and understand someone else's code
 - Haven't done much of this in Java
 - Critique code (do you smell something?)
 - Identify, articulate problems
 - Refactor code to solve problems identified
 - Write tests to verify the code

Oct 27, 2008

Sprenkle - CS209

17